NASA Technical Memorandum 100646

THE COMPUTATIONAL STRUCTURAL MECHANICS TESTBED PROCEDURES MANUAL

Caroline B. Stewart, Compiler

December 1991

NASA
National Aeronautics and Space Administration

Langley Research Center
Hampton, Virginia 23665

Preface

The purpose of this manual is to document standard procedures of the Computational Structural Mechanics (CSM) Testbed software system. A description of each procedure including its function, commands, data interface, and use is presented.

Periodically, updates to this manual will be released which describe new procedures or changes to existing procedures.

The contents of this manual were compiled by Caroline B. Stewart of Analytical Services and Materials, Inc. Contributors include:

Lockheed Palo Alto Research Laboratory

David S. Kang
Bahram Nour-Omid
Shahram Nour-Omid
Charles R. Rankin
Marc E. Regelbrugge
Gary M. Stanley
Phillip Underwood
Mary A. Wright

NASA Langley

D. Dale Davis, Jr.
William H. Greene
Norman F. Knight, Jr.
Jonathan B. Ransom

Lockheed Engineering and Scences Company

Christine G. Lotts
Steven C. Macy
Lise D. Maring
Susan L. McCleary

Analytical Services and Materials, Inc.

Mohammad A. Aminpour
T. Krishnamurthy

Awesome Computing, Inc.

Eugene L. Poole

Update Log	Date
Initial draft	June, 1989
Revised draft	May, 1990

CSM Testbed Procedures Manual

Table of Contents

1.0 Introduction to CSM Testbed Procedures Manual

1.1 CLAMP Directives
1.2 Executing Processors
1.3 Runstream Organization
1.4 Creating and Using Procedures
1.5 The Testbed Procedures Manual
1.6 Examples
1.7 References

2.0 Preprocessing Procedures

2.1 `GEN_BEAM` (Beams Modeled with Beam Elements)
2.2 `GEN_CANTILEVER` (Cantilevered Beams Modeled with Shell Elements)
2.3 `GEN_CURVED_BM` (Curved Beams Modeled with Shell Elements)
2.4 `GEN_PLATE` (General Quadrilateral Plates)
2.5 `GEN_SHELL` (General Shells and Curved Surfaces)

3.0 Solution Procedures

3.1 `L_DYNAMIC_0` (Linear Transient Dynamics using Modal Analysis)
3.2 `L_DYNAMIC_1` (Linear Transient Dynamics using Newmark Algorithm)
3.3 `L_STABIL_1` (Linear Stability (Buckling) Analysis; Prescribed Prestress)
3.4 `L_STABIL_2` (Linear Stability (Buckling) Analysis; Linearly-Computed Prestress)
3.5 `L_STATIC` (Linear Static Analysis)
3.6 `L_VIBRAT_0` (Linear Vibration Analysis about an Unstressed State)
3.7 `L_VIBRAT_1` (Linear Vibration Analysis about a Prescribed Prestressed State)
3.8 `L_VIBRAT_2` (Linear Vibration Analysis about a Linearly-Computed Prestressed State)
3.9 `NL_STATIC_1` (Nonlinear Static Analysis with Arc-length Control)
3.10 `NL_STATIC_2` (Advanced Riks Method)
3.11 `NL_DYNAMIC_1` (Nonlinear Dynamic Analysis)

4.0 Application Procedures

4.1 `CLAMPED_BEAM` (Transient Response of Clamped Beam)
4.2 `COMPRESSED_CYL` (Postbuckling of Compressed Cylindrical Shell)
4.3 `COOK_MEM` (Inplane Bending of Trapezoidal Membrane)

4.4 ELASTICA (Large Rotations of Cantilevered Beam)
4.5 EULER_COLUMN (Euler Column Buckling Problem)
4.6 FOCUS_PANEL
4.7 FREE_EDGE (Free Edge Stress Analysis of Composite Laminate)
4.8 GEN_STF_PANEL (Buckling of Flat Stiffened Panels)
4.9 HINGED_CYL (Snap-Through of Hinged Cylindrical Shell)
4.10 PEAR_CYL (Buckling of Pear-Shaped Cylindrical Shell)
4.11 PINCHED_CYL (Bending of Pinched Cylindrical Shell)
4.12 PWHOLE (Isotropic Membrane with a Circular Hole)
4.13 RECT_PLATE (Rectangular Plate Problems)
4.14 RHOMBIC_PLATE
4.15 TRUNCATED_CONE (Impulsively-Loaded Truncated Conical Shell)
4.16 VIB_2D (Vibration of Beams and Arch Using 2-D Elements)

5.0 Element Assessment Procedures

5.1 DISTORTED_PC (Distorted Pinched Cylinder)
5.2 DISTORTED_PC_3D (3-D Distorted Pinched Cylinder)
5.3 MH_BEAMS
5.4 MH_CYL (Thick-Walled Cylinder)
5.5 MH_PATCH (Patch Tests)
5.6 MH_PLATE
5.7 MH_SPHERE
5.8 MH_ROOF
5.9 SKEWED_GRID

6.0 Postprocessing Procedures

6.1 HISTORY (Tabulate Response History in Database)
6.2 POST (Tabulate Selected Results in Database)
6.3 STRESS (Compute Stresses and/or Strains from Displacements)
6.4 TOTAL_LOAD (Sum Total Load for Applied Displacement Problems)

7.0 Utility Procedures

7.1 CONSTRAIN (Impose Scaled Applied Displacements)
7.2 COPY_DS (Copy a Dataset and Rename)
7.3 EIGEN (Perform Eigenvalue Analysis)
7.4 ES (Generic Element Processor Control)
7.5 FACTOR (Factor (Decompose) System Stiffness Matrix)
7.6 FORCE (Form Force Vectors)
7.7 IMPERFECTION (Superpose Initial Geometric Imperfection)

7.8 INITIALIZE (Model Initialization)
7.9 MASS (Form Mass Matrix)
7.10 MODEL_SUMMARY (Model Summary Information)
7.11 PRINT_EFIL (Print Selected Segments of EFIL Dataset)
7.12 RESEQUENCE (Resequence Nodal Equations)
7.13 SOLVE (Solve System of Equations)
7.14 STIFFNESS (Form Stiffness Matrix)
7.15 SWITCH_DS (Switch Two Datasets)

CSM TESTBED DOCUMENTATION SET

1. Introduction to the Computational Structural Mechanics Testbed
   NASA TM 89096, 1987
2. Utilities for Master Source Code Distribution: MAX and Friends
   NASA CR 178383, 1988
3. The Computational Structural Mechanics Testbed Architecture:
   Volume I - The Language
   NASA CR 178384, 1988
4. The Computational Structural Mechanics Testbed Architecture:
   Volume II - Directives
   NASA CR 178385, 1988
5. The Computational Structural Mechanics Testbed Architecture:
   Volume III - The Interface
   NASA CR 178386, 1988
6. The Computational Structural Mechanics Testbed Architecture:
   Volume IV - The Global-Database Manager GAL-DBM
   NASA CR 178387, 1988
7. The Computational Structural Mechanics Testbed Architecture:
   Volume V - The Input-Output Manager DMGASP
   NASA CR 178388, 1989
8. The Computational Structural Mechanics Testbed User's Manual
   NASA TM 100644, 1989
9. The Computational Structural Mechanics Testbed Data Library Description
   NASA TM 100645, 1988
10. The Computational Structural Mechanics Testbed Generic
    Structural-Element Processor Manual
    NASA CR 181728, 1989
11. The Computational Structural Mechanics Testbed Procedures Manual
    NASA CR 100646, 1990
12. The Computational Structural Mechanics Testbed Utility Manual
    NASA CR XXXXXX, 1990

1.0 Introduction to CSM Testbed Procedures Manual

This manual is designed to assist users in defining and using command procedures to perform structural analyses. It is expected that the user has read Chapters 1 and 2 of the CSM Testbed User's Manual (reference 1-1). It is assumed that the user is familiar with terms such as CLIP, macrosymbol, processor, and dataset.

Runstreams are the vehicle used to perform structural analyses with the CSM Testbed. The term "runstream" most commonly refers to the file (or files) used to perform a specific analysis, although it may also refer to input at an interactive session. A runstream will typically contain CLAMP directives and processor commands.

Directives, recognized and processed by CLIP, provide the user with, among other things, a means of defining command procedures. These command procedures, defined using the `*PROCEDURE` directive, bear some resemblance to FORTRAN subroutines. They may contain branching and looping constructs (implemented using the `*DO`, `*IF`, and `*WHILE` directives) as well as other directives and processor and macroprocessor commands. Command procedures may be given arguments which, unlike FORTRAN subroutine arguments, may be assigned default values. When a command procedure is called (using the `*CALL` directive) execution control shifts to the command procedure until the last directive (an `*END` directive) in the procedure is encountered. Once the `*END` directive is encountered, control returns to the input line in the calling procedure or runstream immediately following the call.

Command procedures, while extremely useful, are not a requirement for performing many types of simple analyses. A command procedure is only required if using the looping or branching constructs (*i.e.*, the `*DO`, `*IF`, and `*WHILE` directives). Procedures should not be used to carry out the computationally intensive activities that are better performed by processors.

This chapter begins with a discussion of CLAMP directives and continues with a discussion of the mechanics of processor execution. A template for linear, static analyses is provided in Section 1.3. Section 1.4 offers some suggestions for creating and using procedures and the CSM Testbed Procedures Manual is described in Section 1.5. Some examples of command procedures are given in Section 1.6.

THIS PAGE LEFT BLANK INTENTIONALLY.

1.1 CLAMP Directives

Directives are special commands that are recognized and processed by CLIP and are not transported to the processor. A directive is to CLIP like ordinary input is to the processor. A directive is distinguished from ordinary input by beginning with a keyword prefixed by an asterisk. The keyword (directive verb) may be followed by a verb modifier, qualifiers, and parameters, as required by the syntax of the specific directive. See references 1-2 and 1-3 for a complete description of the command language. An interactive help facility, accessed by the *HELP directive, is built in to explain directives. For a complete list with full descriptions, the user is directed to reference 1-3.

A summary of the most useful directives, grouped according to their function in the Testbed execution environment is provided here for easy reference. Detailed descriptions of all directives are provided in reference 1-3.

Table 1.1-1 CLAMP Directive Summary

	Global Data Manager Interface
*OPEN	Open data library
*CLOSE	Close data library
*TOC	Print table of contents of library
*PRINT	Print table of contents, dataset record contents, or record access table of dataset
*PACK	Pack a data library, deleting disabled datasets
*COPY	Copy datasets or dataset records
*DELETE	Delete (*i.e.*, disable) dataset or record
*ENABLE	Enable previously deleted or disabled datasets or records
*FIND	Returns information on libraries, datasets, or records
*RENAME	Renames dataset or record
	Command Procedure Management
*SET PLIB	Set procedure library for residence of command procedures
*PROCEDURE	Initiates definition of command procedure
*CALL	Redirects input to a callable procedure ("calls" a procedure with optional argument replacement)
	Nonsequential Command Processing
*IF	Conditional branching construct
*ELSE	
*ELSEIF	
*ENDIF	
*DO	Looping construct
*ENDDO	

*WHILE	While-looping construct
*ENDWHILE	
*JUMP	Transfer control to specified label
*RETURN	Force exit from command procedure
*END	Terminate definition of command procedure

Macrosymbol Directives

*DEFINE	Define a macrosymbol or macrosymbol array
*UNDEFINE	Delete macrosymbol(s)
*SHOW MACRO	Show macrosymbols
*G2M	Define a macrosymbol from a database entity
*M2G	Create a database entity from the value of a macrosymbol
Built-in macrosymbols	Common constants, mathematical functions, generic functions, reserved variables, boolean functions, logical functions, string catenator, string matchers, and status macros

SuperClip Directives

*RUN	Start execution of another program
*STOP	Stops RUN-initiated execution and restarts the parent processor

General Directives

*HELP	Lists information from NICE HELP file
*SET	Sets specified NICE control parameters
*SHOW	Shows specified NICE control parameters
*ADD	Redirects input to a text file
*REMARK	Print remark line
*UNLOAD	Unload contents of GAL library to an ASCII file
*LOAD	Load contents of GAL library from an ASCII file

1.2 Executing Processors

There are two types of analysis modules, or processors, associated with the Testbed: internal processors, which have been installed as part of the macroprocessor, and external processors. Either type of processor may be executed using the macroprocessor execute command, `[XQT`. The user merely appends the processor name to the `[XQT` and the named processor will begin execution. For example,

```
[XQT INV
```

will start the execution of the processor INV. In order to use this method of execution for external processors, the executable version of the processor must reside in the default directory being used by the macroprocessor or in other pre-defined directories depending on computer system syntax. In addition, the name of an external processor cannot be the same as the name of any internal processor.

External processors may also be executed using the `*RUN` directive. When this directive is used, a full pathname may be given so that external processors may be kept anywhere. For example, under the VMS operating system,

```
*RUN dua0:[testbed.extp]INVX
```

will begin execution of processor INVX, located in `dua0:[testbed.extp]`.

Once a processor (internal or external) is running, it will begin to accept input according to the requirements of the individual processor as described in Chapters 4 through 14 of the CSM Testbed User's Manual (Ref. 1-1). The processor will continue accepting input until either another `[XQT`, a `STOP`, or a `*STOP` is encountered. If a `STOP` occurs, execution will proceed to completion of the processor's assigned task after which the processor named on the next `[XQT` command begins execution. If an `[XQT` occurs, execution will proceed to completion of the processor's assigned task after which the processor named on that last `[XQT` begins execution. A `*STOP` terminates processor execution immediately.

The following runstream provides an example of processor, macroprocessor, and CLIP interaction. The linear, static analysis, a very simple example, with one procedure definition, has been taken from the demonstration problem set library. The procedure is defined so that the `*DO` directive may be used in defining joint locations (recall that the `*DO` directive may only be used within a command procedure). Note also that the procedure is completely defined *before* it has been called. This is an absolute requirement – **procedures must be defined before they are called.**

```
*procedure demo1                          . Directives
*open 1 demo1.101 /new                    .
[xqt TAB                                  . Macroprocessor command to execute TAB
  START 5                                 . Processor TAB input
  JOINT LOCATIONS
  *do $i = 1,5                            . Directive to generate TAB input
.   node     x    y    z                  . Comment
    <$i>  0.   0.   <10.*<$i>>            . TAB Input
.
  *enddo                                  . Directive to end input loop
  MATERIAL CONSTANTS                      . Direct TAB input
  1 10.E+6 .3 .101 .1E-4
  BEAM ORIENTATIONS
  1 1 1 1 1.
  E21 SECTION PROPERTIES
  TUBE 1 2. 2.25
  CONSTRAINT DEFINITION 1
  ZERO 1 2 3 4 5 6
   1
[xqt ELD                                  . Macroprocessor command to execute ELD
  E21                                     . Direct ELD Input
  1 2
  2 3
  3 4
  4 5
[xqt TOPO                                 . Macroprocessor command to execute TOPO
[xqt E                                    . Macroprocessor command to execute E
[xqt EKS                                  . Macroprocessor command to execute EKS
[xqt K                                    . Macroprocessor command to execute K
[xqt INV                                  . Macroprocessor command to execute INV
 ALPHA                                    . Direct AUS input
    CASE TITLES
    1'TRANSVERSE LOAD
    2'AXIAL LOAD
    SYSVEC
    APPLIED FORCES
      CASE 1
    I=2
    J=5
    1000.
      CASE 2
    I=3
    J=5
    10000.
[xqt SSOL                                 . Macroprocessor command to execute SSOL
[xqt GSF                                  . Macroprocessor command to execute GSF
[xqt PSF                                  . Macroprocessor command to execute PSF
     stop                                 . Macroprocessor command to exit
*end                                      . Directive denoting procedure end
.
*call demo1                               . Directive calling procedure demo1
[xqt exit                                 . Macroprocessor command to end execution
```

It is important to note that while directives may be used to generate input data, the directives themselves do not provide input to the processors. The `*DO` directive, used in the `JOINT LOCATIONS` subprocessor of processor TAB, provides the user with a means of automatically generating TAB input; the line containing the `*do $i = 1,5` provides no information to processor TAB but is meaningful to CLIP. The result of executing this `*DO` directive is to produce five input lines for consumption by processor TAB.

THIS PAGE LEFT BLANK INTENTIONALLY.

1.3 Runstream Organization

While the Testbed is highly modular, certain processors do depend on information generated by other processors, thus there is some degree of interdependence among the installed processors. In many cases, the order in which processors must be executed is the same as the order in which they appear in the CSM Testbed User's manual (ref. 1-1); this is not entirely true when using one or more of the independent element processors. The following section provides a template for performing a linear static analysis using one or more independent element processors (Section 1.3.1).

1.3.1 RUNSTREAM FOR INDEPENDENT ELEMENT PROCESSORS

The generic element processor template was developed to provide greater flexibility to element developers. It does however, add a level of complexity to the analysis, even to a simple linear, static analysis. This complexity is kept hidden to the average user by the use of a "cover procedure." The procedure name is ES and details of its use may be found in reference 1-4. Essentially, procedure ES manages the execution of the independent element processors, ES*i*.

Listed below is the order of processor and procedure execution for a linear static analysis using one or more of the independent element processors. Following the list is an example (the same example used in the previous section) which calls procedure ES.

1. Procedure ES. Call procedure ES to define element parameters and several global macrosymbols which may be used to automate the definition of joint locations and element connectivity. This call must be made for each different element type in the model, regardless of the number of element processors used.
2. Processor TAB. Define joint locations, constraints, reference frames.
3. Processor AUS. Build tables of material and section properties.
4. Processor LAU. Form constitutive matrix.
5. Processor ELD. Define elements. Element definitions include element connectivity, element material reference frame number, element section type number.
6. Processor E. Initialize element datasets; create the dataset which will contain all important element information (*e.g.*, intrinsic coordinates, element-to-global transformations, intrinsic stiffness matrices).
7. Procedure ES. Initialize element matrices.
8. Procedure ES. Calculate element intrinsic stiffness matrices.
9. Processor RSEQ or PFM. Resequence nodes for minimum total execution time.
10. Processor TOPO. Form maps which guide the assembly and factorization of system matrices.

11. Processor K. Assemble system stiffness matrix.
12. Processor INV. Factor system stiffness matrix.
13. Processor AUS. Create applied nodal loading.
14. Processor SSOL. Solve for static displacements.
15. Procedure ES. Calculate element stress resultants.
16. Post-process using any of the following processors: VPRT, PRTE, PLOT, CONT, T2PT.

1.3.2 EXAMPLE RUNSTREAM

The following runstream provides an example of a very simple linear static analysis. The problem is to calculate the stress in an isotropic flat plate subjected to a uniform end-shortening. One-fourth of the plate is modeled and symmetry boundary conditions are applied.

```
*set echo=off                              . Do not echo input
*add [testbed.proclib]GENUTIL.PRC          . ADD file containing procedure ES
*open/new 1, flat_plate                    . Open data library
.
*def/a es_name == 'EX91'    . Element name
*def/a es_proc == 'ES1'     . Element processor name
.
*call ES ( function = 'DEFINE ELEMENTS'; es_proc = <es_proc/p>; --
          es_name = <es_name/p>)
[xqt TAB
.
    START 25 6                             . Twenty five nodes total, dof 6 zero
.
    JOINT LOCATIONS                        . Enter joint locations
.
    1      0.0   0.0   0.0  2.5   0.0   0.0  5 1 5
    5      0.0   2.5   0.0  2.5   2.5   0.0
.
    CONSTRAINT DEFINITION 1                . Constraints:
      symm plane=1                         . Plane 2,3 plane of symmetry
      symm plane=2                         . Plane 1,3 plane of symmetry
      zero 3: 1                            . Constrain center w
      nonzero 1 : 5,25,5                   . Apply displacement at x=lx edge
.
[xqt AUS                                   . Material and Section properties
.
*def/e g = 3.84615e+6
     TABLE(NI=16,NJ=3): OMB DATA 1 1       . Table of material properties
       I=1,2,3,4,5,6,7,8,9
       J=1: 10.0E+6 .30 10.0E+6 <g> <g> <g> 0.0 0.0 .1
```

```
.                                          . Table of section properties
     TABLE (NI=3,NJ=1,itype=0): LAM OMB 1 1
       J=1 : 2 .1 0.00
.
[xqt LAU                                   . Generate constitutive matrix
.
.      NOTE -- The macrosymbols es_nip, es_nstr, es_nen, and es_opt
.                were all globally defined by procedure ES the first time
.                the procedure was called.
.
[xqt ELD                                   . Define elements
.
       *def/i nst = <<es_nip>*<es_nstr>>
       EXPE <es_name> <es_nen> <es_opt> <es_nen> 6 <nst> 1 101 2
       NSECT = 1
       1  3 13 11  2  8  12  6  7    1 2 2
.
[xqt E                                     . Initialize element datasets
     stop
     *open 1
*call ES (function='INITIALIZE')           . Initialize element matrices
*call ES (function='FORM STIFFNESS/MATL')  . Form intrinsic stiffness matrices
[xqt RSEQ                                  . Resequence
[xqt TOPO                                  . Create maps
[xqt K                                     . Assemble global stiffness matrix
[xqt INV
     online=2
[xqt AUS                                   . Form applied loading
     sysvec : appl moti
     i=1:   J=5,25,5:    -0.001
[xqt SSOL                                  . Solve for static displacements
     stop
     *open 1
*call ES (function = 'FORM STRESS'; --     . Calculate element stresses
          es_dis_ds = STAT.DISP.1.1 )
[xqt VPRT                                  . Print static displacements
     format=4
     print STAT DISP
[xqt PRTE                                  . Print element stresses
     reset seg1=7,seg2=7
[xqt exit
```

THIS PAGE LEFT BLANK INTENTIONALLY.

1.4 Creating and Using Procedures

1.4.1 CREATING A PROCEDURE

Most directives are so simple that they may easily be entered from a keyboard terminal. One could try to define simple procedures in exactly that manner. There are two problems with this approach:

1. A keyed-in sequence of directives and commands is volatile and is **not saved** unless a log file has been opened, although a procedure definition is compiled and saved.
2. Post-facto editing is impossible; once the return key it pressed, the line is gone.

These disadvantages become increasingly serious in long or involved procedures. The most practical way to create most procedures is to use a text editor. Once the procedure source text is ready on a data file, it can be inserted into the command source stream with the `*ADD` directive.

1.4.2 RESIDENCE OF CALLABLE PROCEDURE ELEMENTS

When CLIP encounters a `*PROCEDURE` directive, it enters directive mode and does not exit until the `*END` directive is detected. The result of this process is an "object" version of the procedure, known as a *callable procedure element.* CLIP can store a callable procedure element in one of two residence media:

1. An *ordinary direct-access formatted file* created through a FORTRAN 77 OPEN statement. All records of such a file have the same length (namely 80 characters) and contain one data line. The file name is the same as the procedure name except on a VMS VAX where a `.DAT` is appended to the procedure name to create the file name. For example, a procedure named `GEN_PLATE` will generate a file named `GEN_PLATE.DAT` on a VMS VAX and a file named `GEN_PLATE` on other machines. If the `*SET PLIB` directive has not been used prior to the `*PROCEDURE` directive, direct-access files will be created automatically.
2. A *data library* managed through the global data manager. A callable procedure is stored as a text group. In order to store procedures in a data library, the `*SET PLIB` directive must be used.

The text of a callable procedure element is basically a copy of the source procedure body, prefaced by three linkage tables. These tables store argument names, argument default text, labels (explicit or generated) and their locations within the body of the procedure. **NEVER** tamper with a callable procedure element. If the procedure must be changed, change the source and reprocess the file.

1.4.3 USING A PROCEDURE

Callable procedure elements are accessed through the *CALL directive. Text substitution is controlled by the argument specification mechanism. In a command procedure reference, text is passed instead of addresses to data. The text supplied in the *CALL directive is *replaced before the command is interpreted.* In addition, arguments not supplied in the *CALL, assume the default values given in the *PROCEDURE definition. A procedure body may include calls to other procedures, or may even call itself, with the ensuing call tree extending down several levels.

1.5 The CSM Testbed Procedures Manual

A GAL library which contains the callable procedure elements for all of the procedures described in the following sections (2-9) of this manual is read-accessible to all Testbed users. Separate subdirectories under the prc directory contain solution procedures (see Chapter 3), model generation procedures (see Chapter 2), utility procedures (see Chapter 6), and postprocessing procedures (see Chapter 7). This directory structure is the same across various computer systems with differences only in the description of the path name for each subdirectory. On a VMS VAX computer, this file is referred to by the name `CSM_PRC:PROCLIB.GAL`; on UNIX-type computers, it is referred to by the name `'$CSM_PRC/proclib.gal'`. If the user does not need to define any new procedures for use in a particular Testbed runstream, this file can be used as the procedure library by including the following commands in the runstream:

```
*set plib = 28
*open 28 CSM_PRC:PROCLIB.GAL /READ                    (on VMS)
   or
*open 28 '$CSM_PRC/proclib.gal' /READ                 (on UNIX)
```

The source code for the procedures resides in subdirectories under the one which contains `proclib.gal`. These procedures are also read-accessible to all Testbed users. They can be included in the user's private procedure library by placing commands like the following in a Testbed runstream:

```
*set plib=28
*open 28 proclib.gal /NEW
*add GEN_UTIL:ES.CLP                                  (on VMS)
   or
*add '$GEN_UTIL:es.clp'                               (on UNIX)
```

or alternatively; on UNIX:

```
cp $CSM_PRC/proclib.gal .
chmod 755 proclib.gal
testbed
*set plib=28
*open 28 proclib.gal
*add local.prc
.
.
.
```

or on VMS:

```
$COPY CSM_PRC:PROCLIB.GAL []
$Testbed
*set plib=28
*open 28 proclib.gal
*add local.prc
.
.
.
```

where `local.prc` is the CLAMP source file for personal procedure(s).

THIS PAGE LEFT BLANK INTENTIONALLY.

1.6 Examples

This section provides several examples of the use of procedures. For the sake of consistency, where files are discussed, VAX/VMS filenames have been used. The filename convention used is that a file with a *.CLP extension contains a single procedure, while a file with a *.PRC extension contains multiple procedures. The *ADD and the driving *CALL directives typically appear in files with the *.COM extension.

1.6.1 A SIMPLE EXAMPLE

As an introductory example, an annotated procedure is presented which may be used to run a variety of elements through the same flat plate problem. In most applications, this procedure would be kept in a file by itself and that file would be added (using the *ADD directive) to a much shorter runstream located in an execution control file. The procedure and a VAX/VMS execution control file are listed in the following subsections.

1.6.1.1 The Procedure File

The following procedure is kept in a file named FLAT_PLATE.CLP.

```
*procedure FLAT_PLATE ( es_proc ; es_name )
.
.  ARGUMENTS:
.
.       es_proc:  Independent element processor name
.       es_name:  Element name
.
*if <ifeqs([es_name];E43)> /then
   *def/i es_nen = 4
*else
   *call ES ( function = 'DEFINE ELEMENTS'; es_proc = [es_proc]; --
              es_name = [es_name])
*endif
.
.
[xqt TAB
.
     START 25 6                          . Twenty-five joints; dof 6 zero
.
     JOINT LOCATIONS                     . Define joint locations
.
     1      0.0   0.0   0.0  2.5   0.0   0.0  5 1 5
     5      0.0   2.5   0.0  2.5   2.5   0.0
.
     CONSTRAINT DEFINITION 1             . Constraints:
       symm plane=1                      . Plane 2,3 plane of symmetry
       symm plane=2                      . Plane 1,3 plane of symmetry
       zero 3: 1                         . Constrain center w
       nonzero 1 : 5,25,5                . Apply displacement at x=lx edge
```

```
.
[xqt AUS
.
*def/e g = 3.84615e+6
     TABLE(NI=16,NJ=3): OMB DATA 1 1      . Define material properties
       I=1,2,3,4,5,6,7,8,9
       J=1: 10.0E+6 .30 10.0E+6 <g> <g> <g> 0.0 0.0 .1
                                          . Define Section properties
.
     TABLE (NI=3,NJ=1,itype=0): LAM OMB 1 1
          J=1 : 2 .1 0.00
[xqt LAU                                  . Form constitutive matrix
     *if <ifeqs([es_name];E43)> /then
       reset SPAR=-1
     *endif
.
[xqt ELD                                  . Define elements
     *if <ifeqs([es_name];E43)> /then
       E43
     *else
.
       NOTE -- The macrosymbols es_nip, es_nstr, es_nen, and es_opt
.                were globally defined by procedure ES the first time
.                the procedure was called.
.
       *def/i nst = <<es_nip>*<es_nstr>>    . Number of stress resultants
       EXPE [es_name] <es_nen> <es_opt> <es_nen> 6 <nst> 1 101 2
     *endif
       NSECT = 1
     *if < <es_nen> /eq 4 > /then
.
       1 2 7 6   1  4  4                  . Element connectivity for 4-node elts
.
     *elseif < <es_nen> /eq 9 > /then
.
       1  3 13 11  2  8  12  6  7 1 2 2   . Element connectivity for 9-node elts
     *endif
.
[xqt E                                    . Initialize all element datasets
     stop
     *open 1
.
*if <ifeqs([es_name];E43)> /then        . Form intrinsic stiffness matrices:
    [xqt EKS                                      .   for E43
*else
    *call ES (function='INITIALIZE')
    *call ES (function='FORM STIFFNESS/MATL')  .   for other elements
*endif
[xqt RSEQ                                 . Resequence
[xqt TOPO                                 . Create maps
[xqt K                                    . Assemble system stiffness matrix
[xqt AUS                                  . Form applied loading
     sysvec : appl moti
     i=1: J=5,25,5:    -0.001
```

```
[xqt INV                                    . Factor stiffness matrix
     online=2
[xqt SSOL                                   . Solve for static displacements
     stop
     *open 1
                                            . Calculate stresses
*if <ifeqs([es_name];E43)> /then            .   for E43
   [xqt GSF
   [xqt PSF
        reset display=2
*else                                       .   for other elements
   *call ES ( function = 'FORM STRESS'; es_dis_ds = STAT.DISP.1.1 )
*endif
[xqt VPRT                                   . Print displacements
     format=4
     print STAT DISP
*end
```

1.6.1.2 The Execution Control File

The file FLATPLATE.COM, listed below, contains no procedures, only the *ADD and the *CALL to the procedure defined in the previous section.

```
$ testbed                          ! Execute Testbed macroprocessor
*set echo off
*open 1 flat_plate.101             . Open data library
.
*add flat_plate.clp                . Add procedure file BEFORE call
.
*call FLAT_PLATE ( es_proc=ES1; es_name=Ex97 )
.
[xqt EXIT                          . Exit macroprocessor
```

1.6.2 MACROSYMBOL USAGE EXAMPLE

The runstream described in this section still contains only one procedure; that procedure is somewhat more complicated than the procedure of the first section although the problem to be solved is the same. The number of elements along x and y have been parameterized to allow for mesh convergence studies for the various elements. The logic of the procedure remains the same; there are simply more macrosymbol definitions. The procedure and the execution control file are listed in the following subsections.

1.6.2.1 The Procedure File

The following procedure is kept in a file named FLAT_PLATE.CLP.

```
*procedure FLAT_PLATE ( es_proc=ES1; es_name=Ex97;-- )
```

```
                        NEL_x ; NEL_y )
.
.  ARGUMENTS:
.
.      es_proc:  Independent element processor name
.      es_name:  Element name
.      NEL_x:     Number of elements in the x direction
.      NEL_y:     Number of elements in the y direction
.
*if <ifeqs([es_name];E43)> /then
   *def/i es_nen = 4
*else
   *call ES ( function = 'DEFINE ELEMENTS'; es_proc = [es_proc]; --
              es_name = [es_name])
*endif
.
.    Define necessary macrosymbols
.
*if << es_nen > /eq 4 > /then              . If 4-node elements are used:
     *def/i nn_x = <<[NEL_x]> + 1 >         .  Num. nodes in x-direction
     *def/i nn_y = <<[NEL_y]> + 1 >         .  Num. nodes in y-direction
     *def/i nn_total = <<nn_x>*<nn_y>>     . Num. nodes total
*elseif << es_nen > /eq 9 > /then          . If 9-node elements are used:
     *def/i nn_x = <2*<[NEL_x]> + 1 >       .  Num. nodes in x-direction
     *def/i nn_y = <2*<[NEL_y]> + 1 >       .  Num. nodes in y-direction
     *def/i nn_total = <<nn_x>*<nn_y>>     .  Num. nodes total
*endif
.
.
[xqt TAB
.
     START <nn_total> 6                    . Twenty-five joints; dof 6 zero
.
     JOINT LOCATIONS                       . Define joint locations
.
     1       0.0   0.0   0.0  2.5   0.0   0.0  <nn_x> 1 <nn_y>
   <nn_x>   0.0   2.5   0.0  2.5   2.5   0.0
.
     CONSTRAINT DEFINITION 1               . Constraints
       symm plane=1                        . Plane 2,3 plane of symmetry
       symm plane=2                        . Plane 1,3 plane of symmetry
       zero 3: 1                           . Constrain center w
       nonzero 1 : <nn_x>,<nn_total>,<nn_x>  . Apply displacement at x=lx edge
.
[xqt AUS
.
*def/e g = 3.84615e+6
     TABLE(NI=16,NJ=3): OMB DATA 1 1       . Define material properties
       I=1,2,3,4,5,6,7,8,9
       J=1: 10.0E+6 .30 10.0E+6 <g> <g> <g> 0.0 0.0 .1
                                           . Define Section properties
.
     TABLE (NI=3,NJ=1,itype=0): LAM OMB 1 1
          J=1 : 2 .1 0.00
```

```
[xqt LAU                                    . Form constitutive matrix
     *if <ifeqs([es_name]; E43)>/ then
       reset SPAR=-1
     *endif
.
[xqt ELD                                    . Define elements
     *if <ifeqs([es_name]; E43)>/ then
       E43                                  .   E43 elements
     *else
.
.      NOTE -- The macrosymbols es_nip, es_nstr, es_nen, and es_opt
.                were globally defined by procedure ES the first time
.                the procedure was called.
.
       *def/i nst = <<es_nip>*<es_nstr>>    . Number of stress resultants
       EXPE [es_name] <es_nen> <es_opt> <es_nen> 6 <nst> 1 101 2
     *endif
       NSECT = 1
.
     *if < <es_nen> /eq 4 > /then
                                            . Use 4-node element mesh generator
       *def/i j1 = 1
       *def/i j2 = 2
       *def/i j3 = <<j2>+<nn_x>>
       *def/i j4 = <<j1>+<nn_x>>
                                            . Element connectivity for 4-node elts
       <j1> <j2> <j3> <j4>  1  <[NEL_x]> <[NEL_y]>
.
     *elseif < <es_nen> /eq 9 > /then
                                            . Use 9-node element mesh generator
       *def/i j1 = 1
       *def/i j5 = 2
       *def/i j2 = 3
       *def/i j8 = <<j1> + <nn_x>>
       *def/i j9 = <<j5> + <nn_x>>
       *def/i j6 = <<j2> + <nn_x>>
       *def/i j4 = <<j8> + <nn_x>>
       *def/i j7 = <<j9> + <nn_x>>
       *def/i j3 = <<j6> + <nn_x>>
                                            . Element connectivity for 9-node elts
.
       <j1> <j2> <j3> <j4> <j5> <j6> <j7> <j8> <j9> 1 <[NEL_x]> <[NEL_y]>
.
     *endif
.
[xqt E                                      . Initialize all element datasets
     stop
     *open 1
.
*if <ifeqs([es_name]; E43)>/ then      . Form intrinsic stiffness matrices
    [xqt EKS                                    .   for E43
*else
    *call ES (function='INITIALIZE')
```

```
   *call ES (function='FORM STIFFNESS/MATL')  .   for other elements
*endif
[xqt RSEQ                                    . Resequence
[xqt TOPO                                    . Create maps
[xqt K                                       . Assemble system stiffness matrix
[xqt AUS                                     . Form applied loading
     sysvec : appl moti
     i=1: J=<nn_x>,<nn_total>,<nn_x>:   -0.001
[xqt INV                                     . Factor stiffness matrix
     online=2
[xqt SSOL                                    . Solve for static displacements
     stop
     *open 1
                                             . Calculate stresses
*if <ifeqs([es_name]; E43)>/ then       .     for E43
   [xqt GSF
   [xqt PSF
        reset display=2
*else                                        .   for other elements
   *call ES ( function = 'FORM STRESS'; es_dis_ds = STAT.DISP.1.1 )
*endif
[xqt VPRT                                    . Print displacements
     format=4
     print STAT DISP
*end
```

1.6.2.2 The Execution Control File

The file, **FLATPLATE.COM**, listed below contains no procedures, only the ***ADD** and the ***CALL** to the procedure defined in the previous section.

```
$ testbed                          ! Execute Testbed macroprocessor
*set echo off
*open 1 flat_plate.101             . Open data library
.
*add flat_plate.clp                . Add procedure file BEFORE call
.
*call FLAT_PLATE ( es_proc=ES1; es_name=Ex97; --
                   Nel_x=4; Nel_y=4 )
.
[xqt EXIT                          . Exit macroprocessor
```

One may notice that, except for the two extra arguments (**NEL_x** and **NEL_y**) in the ***CALL** directive, this file is the same as the **FLATPLATE.COM** file of the last section.

1.6.3 A MULTIPLE PROCEDURE EXAMPLE

In many cases, it may be to the user's advantage to build and maintain a procedure library which may be used for classes of problems. For example, in a solution library, one may keep procedures for providing linear static solutions, buckling eigenvalues, and nonlinear static solutions. In this section, the procedure of the previous section is split up into three procedures – PLATE_MODEL, PLATE_BC, and L_STATIC – which generate the model, generate the boundary conditions, and perform the linear, static solution respectively. The three procedures are kept in two files: L_STATIC.CLP (contains only procedure L_STATIC) and FLATPLATE.PRC (contains PLATE_MODEL and PLATE_BC). Finally, the file FLATPLATE.COM uses the *ADD directive to add the two files and the *CALL directive to call the procedures.

1.6.3.1 The Model Generation Procedures

The following two procedures, PLATE_MODEL and PLATE_BC, are, for the sake of the example to be kept in a file named FLATPLATE.PRC. Note that the boundary conditions and applied loads are both in the procedure PLATE_BC and that if other boundary conditions were desired, this procedure could be decoupled from the model generation procedure and stored in a separate file. In that case, the procedure name could be passed as an argument to PLATE_MODEL which would then call the passed name instead of PLATE_BC.

```
  *procedure PLATE_MODEL ( es_proc ; es_name ; --
                           NEL_x ; NEL_y )
.
.  ARGUMENTS:
.
.       es_proc:  Independent element processor name
.       es_name:  Element name
.       NEL_x:     Number of elements in the x direction
.       NEL_y:     Number of elements in the y direction
.
*if <ifeqs([es_name];E43)> /then
   *def/i es_nen = 4
*else
   *call ES ( function = 'DEFINE ELEMENTS'; es_proc = [es_proc]; --
              es_name = [es_name])
*endif
.
.    Define necessary macrosymbols
.
*if << es_nen > /eq 4 > /then              . If 4-node elements are used:
     *def/i nn_x = <<[NEL_x]> + 1 >         .  Num. nodes in x-direction
     *def/i nn_y = <<[NEL_y]> + 1 >         .  Num. nodes in y-direction
     *def/i nn_total = <<nn_x>*<nn_y>>      . Num. nodes total
*elseif << es_nen > /eq 9 > /then          . If 9-node elements are used:
     *def/i nn_x = <2*<[NEL_x]> + 1 >       .  Num. nodes in x-direction
     *def/i nn_y = <2*<[NEL_y]> + 1 >       .  Num. nodes in y-direction
     *def/i nn_total = <<nn_x>*<nn_y>>      .  Num. nodes total
*endif
.
```

```
.
[xqt TAB
.
     START <nn_total> 6                    . Twenty-five joints; dof 6 zero
.
     JOINT LOCATIONS                       . Define joint locations
.
     1       0.0   0.0   0.0  2.5   0.0   0.0  <nn_x> 1 <nn_y>
   <nn_x>    0.0   2.5   0.0  2.5   2.5   0.0
.
                                           . Call boundary condition procedure
*call PLATE_BC ( nn_x = <nn_x>; --         .  to set up loads and b.c.'s
                 nn_total = <nn_total> )
.
[xqt AUS
.
*def/e g = 3.84615e+6
     TABLE(NI=16,NJ=3): OMB DATA 1 1       . Define material properties
       I=1,2,3,4,5,6,7,8,9
       J=1: 10.0E+6 .30 10.0E+6 <g> <g> <g> 0.0 0.0 .1
.                                          . Define Section properties
     TABLE (NI=3,NJ=1,itype=0): LAM OMB 1 1
          J=1 : 2 .1 0.00
[xqt LAU                                   . Form constitutive matrix
     *if <ifeqs([es_name]; E43)> /then
       reset SPAR=-1
     *endif
.
[xqt ELD                                   . Define elements
     *if <ifeqs([es_name]; E43)> /then
       E43                                 .   E43 elements
     *else
.
.      NOTE -- The macrosymbols es_nip, es_nstr, es_nen, and es_opt
.                were globally defined by procedure ES the first time
.                the procedure was called.
.
       *def/i nst = <<es_nip>*<es_nstr>>   . Number of stress resultants
       EXPE [es_name] <es_nen> <es_opt> <es_nen> 6 <nst> 1 101 2
     *endif
       NSECT = 1
.
     *if < <es_nen> /eq 4 > /then
                                           . Use 4-node element mesh generator
       *def/i j1 = 1
       *def/i j2 = 2
       *def/i j3 = <<j2>+<nn_x>>
       *def/i j4 = <<j1>+<nn_x>>
                                           . Element connectivity for 4-node elts
       <j1> <j2> <j3> <j4>  1  <[NEL_x]> <[NEL_y]>
.
     *elseif < <es_nen> /eq 9 > /then
                                           . Use 9-node element mesh generator
```

```
      *def/i j1 = 1
      *def/i j5 = 2
      *def/i j2 = 3
      *def/i j8 = <<j1> + <nn_x>>
      *def/i j9 = <<j5> + <nn_x>>
      *def/i j6 = <<j2> + <nn_x>>
      *def/i j4 = <<j8> + <nn_x>>
      *def/i j7 = <<j9> + <nn_x>>
      *def/i j3 = <<j6> + <nn_x>>
                                        . Element connectivity for 9-node elts
.
      <j1> <j2> <j3> <j4> <j5> <j6> <j7> <j8> <j9> 1 <[NEL_x]> <[NEL_y]>
.
    *endif
.
*end
```

```
*procedure PLATE_BC ( nn_x ; nn_total )
.
[xqt TAB
    CONSTRAINT DEFINITION 1             . Constraints
      symm plane=1                      . Plane 2,3 plane of symmetry
      symm plane=2                      . Plane 1,3 plane of symmetry
      zero 3: 1                         . Constrain center w
      nonzero 1 : [nn_x],[nn_total],[nn_x]  . Apply displacement at x=lx edge
.
[xqt AUS                                . Form applied loading
    sysvec : appl moti
    i=1: J=[nn_x],[nn_total],[nn_x]:   -0.001
*end
```

1.6.3.2 The Linear Static Analysis Procedure

The following procedure performs the linear static analysis for models using either SPAR E43 elements or elements implemented using the generic element processors. The procedure will be kept in a file named L_STATIC.CLP.

```
*procedure L_STATIC (es_name)
.
[xqt E                                  . Initialize all element datasets
    stop
    *open 1
```

```
.
*if <ifeqs([es_name]; E43)> /then      . Form intrinsic stiffness matrices
    [xqt EKS                                    .   for E43
*else
    *call ES (function='INITIALIZE')
    *call ES (function='FORM STIFFNESS/MATL')  .   for other elements
*endif
[xqt RSEQ                                  . Resequence
[xqt TOPO                                  . Create maps
[xqt K                                     . Assemble system stiffness matrix
[xqt INV                                   . Factor stiffness matrix
     online=2
[xqt SSOL                                  . Solve for static displacements
     stop
     *open 1
                                           . Calculate stresses
*if <ifeqs([es_name]; E43)> /then      .   for E43
   [xqt GSF
   [xqt PSF
        reset display=2
*else                                      .   for other elements
   *call ES ( function = 'FORM STRESS'; es_dis_ds = STAT.DISP.1.1 )
*endif
[xqt VPRT                                  . Print displacements
     format=4
     print STAT DISP
*end
```

1.6.3.3 The Execution Control File

The following file, **FLATPLATE.COM**, contains no procedures; it adds the two procedure files and calls the model generation and analysis procedures, **PLATE_MODEL** and **L_STATIC**.

```
$ testbed                          ! Execute Testbed macroprocessor
*set echo off
*open 1 flat_plate.101             . Open data library
.
*add flatplate.prc                 . Add procedure files BEFORE calls
*add l_static.clp
                                   . Generate model
*call PLATE_MODEL ( es_proc=ES1; es_name=Ex97;--
                    nel_x=4; nel_y=4 )
                                   . Solve for static solution
*call L_STATIC ( es_name=Ex97 )
.
[xqt EXIT                          . Exit macroprocessor
```

It should be emphasized that the procedure **L_STATIC** may be used for any linear, static analysis using either the original SPAR elements or elements implemented using one or more of the Independent Element Processors. The procedure is not limited to SPAR E43

elements as no element specific operations are being performed; element specific operations are performed in the model definition procedure(s).

By splitting the analysis into procedures, the model generation and solution have been decoupled allowing the solution procedure to be used for many different models. The advantages of this approach include the fact that a solution procedure need only be written once rather than once for each problem. It is highly recommended that the user organize procedures in this fashion.

THIS PAGE LEFT BLANK INTENTIONALLY.

1.7 References

1-1 Stewart, Caroline B.: *The Computational Structural Mechanics Testbed User's Manual.* NASA TM 100644, October 1989.

1-2 Felippa, Carlos A.: *The Computational Structural Mechanics Testbed Architecture: Volume I - The Language.* NASA CR 178384, December 1988.

1-3 Felippa, Carlos A.: *The Computational Structural Mechanics Testbed Architecture: Volume II - Directives.* NASA CR 178385, February 1989.

1-4 Stanley, Gary and Nour-Omid, Shahram: *The Computational Structural Mechanics Testbed Generic Structural-Element Processor Manual.* NASA CR 181728, March 1990.

THIS PAGE LEFT BLANK INTENTIONALLY.

2.0 Preprocessing Procedures

The five procedures documented in this chapter are general modeling procedures for specific structural geometries.

Table 2.0-1 Summary of Preprocessing Procedures

Procedure Name	*Preprocessing Function*
GEN_BEAM	Generate 1-D models of straight beams using beam elements
GEN_CANTILEVER	Generate 2-D models of a straight cantilever beam using plate/shell elements. Using the default values for the procedure arguments, the straight cantilever beam problem from the MacNeal-Harder test cases is generated.
GEN_CURVED_BM	Generate 2-D models of a curved (circular) beam using plate/shell elements. Using the default values for the procedure arguments, the curved beam problem from the MacNeal-Harder test cases is generated.
GEN_PLATE	Generates 2-D models for general quadrilateral plates.
GEN_SHELL	Generates 2-D models for general shells and curved surfaces.

THIS PAGE LEFT BLANK INTENTIONALLY.

2.1 Procedure GEN_BEAM

2.1.1 GENERAL DESCRIPTION

This section describes a procedure which generates models of a straight beam using one-dimensional beam elements.

2.1.2 PROCEDURE USAGE

Procedure **GEN_BEAM** may be used by preceding the procedure name by the ***call** directive, and following it by a list of arguments enclosed in parentheses. Procedure arguments are order-independent, and most have default values thus making them optional. The formal syntax is as follows:

```
*call GEN_BEAM ( arg1 = val1 ; arg2 = val2 ; ...)
```

where **arg1** and **arg2** represent argument names, and **val1** and **val2** represent their corresponding values. Note that semi-colons are required between arguments, and a double dash (--) may be used to continue the argument list on the next line.

The allowable arguments for procedure **GEN_BEAM** are summarized in the following table, along with their default values (if they exist). Note that arguments without defaults are generally *mandatory*, while those with defaults are generally optional. Exceptions to this rule are noted in the following section under detailed argument descriptions.

Argument	Default Value	Meaning
ES_PROC	ES6	Select element processor
ES_NAME	E210	Select element within ELT_PROC
NODES_X	3	Number of nodes in x-direction
LENGTH_X	10.	Length of beam
E	120.	Young's elastic modulus
NU	0.	Poisson's ratio
AREA	1.0	Cross-sectional area
INERT_1	1.	Principal moment of inertia, I_1
INERT_2	10.	Principal moment of inertia, I_2
INERT_TORSIONAL	1.	Uniform torsion constant
BC_PROCEDURE	BEAM_BC	Procedure for boundary conditions

2.1.3 ARGUMENT DESCRIPTIONS

2.1.3.1 AREA

Cross-sectional area of beam (default: 1.0).

2.1.3.2 BC_PROCEDURE

Boundary condition procedure name (default: CC_BC for specified forces; CCD_BC for specified displacements). The term "boundary conditions" refers both to displacement constraints and applied loading. Procedures CC_BC and CCD_BC both have the same zero displacement constraints. The only difference is that the former procedure applies axial forces to the simply supported edge, while the latter procedure prescribes non-zero axial displacements on that edge. The argument BC_PROCEDURE permits you to to supply your own boundary condition procedure, but keep in mind that this may drastically change the problem definition, and hence invalidate most of the discussion under Section 2.1.1.

2.1.3.3 E

Young's elastic modulus (default: 120.0).

2.1.3.4 ES_NAME

Element name (default: E210). This is the name of the specific beam-element type you wish to select, within the element processor defined by argument ES_PROC. The default element type, E210, is a 2-noded beam element implemented in processor ES6, and described in The Computational Structural Mechanics Testbed User's Manual (see ref. 2.1-1).

2.1.3.5 ES_PROC

Element Processor (default: ES6) This is the name of the structural element (ES) processor that contains the shell element type you wish to employ in the model. The default shell-element, processor ES6, is described in The Computational Structural Mechanics Testbed User's Manual.

2.1.3.6 INERT_1

Principal moment of inertia (default: 1.0).

2.1.3.7 INERT_2

Principal moment of inertia (default: 10.0).

2.1.3.8 INERT_TORSION

Torsional constant (default: 1.0).

2.1.3.9 LENGTH_X

Length of the beam in the x-direction (default: 10.0).

2.1.3.10 NODES_X

Number of nodes along beam length (default: 3). Note that this number should be consistent with the number of nodes per element. For example, NODES_X can be any number greater than 1 for 2-node beam elements, whereas it must be an odd number greater than 1 for 3-node beam elements.

2.1.3.11 NNODES_C

Number of circumferential nodes (default: 7). This is the number of nodes you wish to have along the circumferential direction of the cylindrical shell model, *i.e.*, along 15 degrees of circular arclength. Note that this number should be consistent with the number of nodes per element. For example, NNODES_C can be any number greater than 1 for 4-node quadrilateral elements, whereas it must be an odd number greater than 1 for 9-node quadrilateral elements.

2.1.3.12 NU

Poisson's ratio (default: 0.0).

2.1.4 USAGE GUIDELINES AND EXAMPLES

Procedure **GEN_BEAM** may be used by preceding the procedure name by the *call directive. Procedure arguments may be changed from their default values by including any or all of the arguments and their new values when the procedure is called. A space or blank is required between the end of the procedure name and the left parenthesis. If the default values of the procedure arguments are to be used, then only the procedure name is required.

```
*procedure GEN_BEAM  ( es_proc = ES6 ; es_name  = E210 ; --
                       nodes_x       = 3   ;--
                       length_x      =10.  ;--
                       E=120.;     PR=0.   ; area      = 1.0  ; --
                       inert_1=1.  ; inert_2=10. ; inert_torsion=1. ;--
                       BC_PROCEDURE = BEAM_BC )
```

(E1) To perform an entire analysis using the default options, simply invoke the procedure without any arguments, *i.e.*,

```
*call GEN_BEAM
```

2.1.5 LIMITATIONS

2.1.6 ERROR MESSAGES AND WARNINGS

None.

2.1.7 PROCEDURE FLOWCHART

2.1.8 PROCEDURE LISTING

```
*procedure GEN_BEAM  ( es_proc = ES6 ; es_name  = E210 ; --
                       nodes_x       = 3   ;--
                       length_x      =10.  ;--
                       E=120.;     PR=0.   ; area      = 1.0  ; --
                       inert_1=1.  ; inert_2=10. ; inert_torsion=1. ;--
                       BC_PROCEDURE = BEAM_BC )
*call ES_DEFN ( es_proc=[es_proc]; es_name=[es_name]
[XQT TAB
*def nodes_tot = < [nodes_x] >
```

```
START <nodes_tot>
JLOC
.
. DEFINE NODAL COORDINATES
.
*def/e dx = < [length_x] / ([nodes_x]-1) >
*def/i node = 0
   *def/e x  = 0.
   *do   $i  = 1,[nodes_x]
      *def node = < <node> + 1 >
       <node>  <x>, 0., 0.                . NODE DEFINITION
      *def x = < <x> + <dx> >
   *enddo
.
. DEFINE FICTITIOUS ELASTIC MATERIAL PROPERTIES
.
 MATC
 1 [E] [PR]
.
. BEAM FACE ORIENTATION AND PROPERTIES
.
  MREF
  FORMAT=2
  1  1  0.   1.0   0.
.
  BA
  GIVN 1  [inert_1]  0. [inert_2] 0. [area] [inert_torsion]
.
. DEFINE LOADS AND BOUNDARY CONDITIONS
.
 *call [BC_PROCEDURE] ( nnx = [nodes_x] ; --
                        nen = <es_nen> )
.
.
. GENERATE ELEMENTS
.
.       [XQT ELD
.
   <es_expe_cmd>
   NSECT = 1
.
.     Define element nodal connectivity
.
     *call BM_ELT_CONN (nnx=[nodes_x]; nen=<es_nen>)
.
*end
. =DECK BM_ELT_CONN
*procedure BM_ELT_CONN ( nnx; nen )
.
.        ============================
. Define Element Connectivity Record for ELD Processor
.        ============================
.
```

```
   *if < [nen] /eq 2 > /then
         *do $ix = 1, <[nnx]-1>
            *def/i n1 =  < $ix >
            *def/i n2 =  < <n1> +   1   >
.
.             ==========
              <n1>  <n2>
.             ==========
.
         *enddo
   *elseif < [nen] /eq 3 > /then
         *do $ix = 1, <[nnx]-2>, 2
            *def/i n1 =  < $ix >
            *def/i n2 =  < <n1> + 2 >
            *def/i n3 =  < <n2> - 1 >
.
.           ===============
            <n1> <n2> <n3>
.           ===============
.
         *enddo
   *endif
*end
```

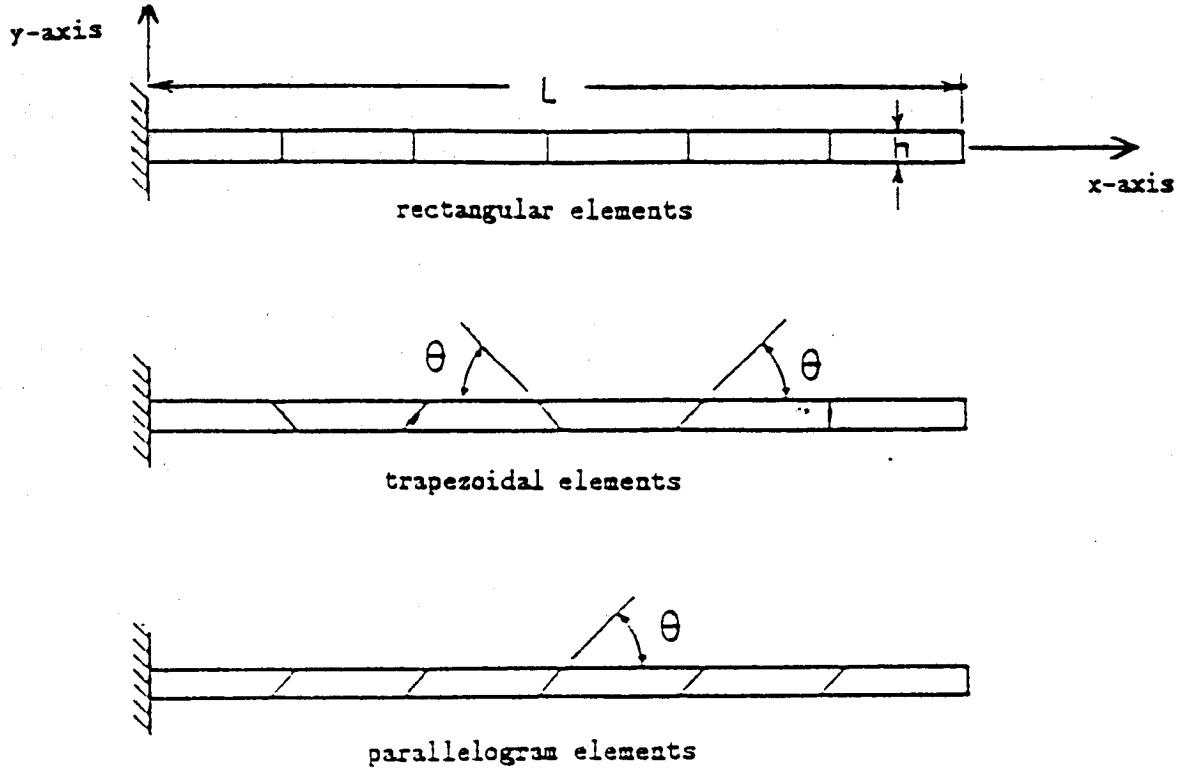
2.1.9 REFERENCES

2.1-1 Stewart, Caroline B.: *The Computational Structural Mechanics Testbed User's Manual.* NASA TM-100644, October 1989.

2.2 Procedure GEN_CANTILEVER

2.2.1 GENERAL DESCRIPTION

This section describes a procedure which generates models of a straight cantilever beam using two-dimensional plate/shell elements. Using the default values for the procedure arguments, the straight cantilever beam problem from the MacNeal-Harder test cases (see ref. 2.2-1) is generated. The model used for the MacNeal-Harder cantilever beam test cases, is shown in figure 2.2-1.



DIMENSIONS : h = .2 , L = 6 , Thickness = .1

MATERIAL PROPERTIES : $E = 1.0 \times 10^7$, $\nu = .30$

Figure 2.2-1 Generic 2-D Cantilever Beam Finite Element Models.

2.2.2 PROCEDURE USAGE

Procedure **GEN_CANTILEVER** may be used by preceding the procedure name by the ***call** directive, and following it by a list of arguments enclosed in parentheses. Procedure arguments are order-independent, and most have default values thus making them optional. The formal syntax is as follows:

```
*call GEN_CANTILEVER ( arg1 = val1 ; arg2 = val2 ; ...)
```

where **arg1** and **arg2** represent argument names, and **val1** and **val2** represent their corresponding values. Note that semi-colons are required between arguments, and a double dash (--) may be used to continue the argument list on the next line.

The allowable arguments for procedure **GEN_CANTILEVER** are summarized in the following table, along with their default values (if they exist). Note that arguments without defaults are generally *mandatory*, while those with defaults are generally optional. Exceptions to this rule are noted in the following section under detailed argument descriptions.

Argument	Default Value	Meaning
ES_PROC	ES2	Select element processor
ES_NAME	EX41	Select element with ELT_PROC
ES_PARS	0.0	Set element-research parameters
AUTO_DOF_SUP	true	Automatic d.o.f. suppression
DRILLING_DOF	false	
NODES_X	7	Number of nodes in x-direction
NODES_Y	2	Number of nodes in y-direction
LENGTH_X	6.	Beam length (x-direction)
LENGTH_Y	.2	Beam width (y-direction)
E	1.E7	Young's elastic modulus
NU	.3	Poisson's ratio
THICKESS	.1	Thickness
DISTORT	.07071	
BC_PROCEDURE	CANTILEVER_BC	Procedure for boundary conditions

2.2.3 ARGUMENT DESCRIPTIONS

2.2.3.1 AUTO_DOF_SUP

Automatic degree of freedom suppression flag (default: <true>). This option provides a convenient way of suppressing any freedoms that do not have any (or adequate) stiffness associated with them — for example, at nodes used to prescribe geometry only; or drilling freedoms in fine meshes composed of elements without normal rotational stiffnesses (see argument **DRILLING_DOF**).

2.2.3.2 BC_PROCEDURE

Boundary condition procedure name (default: CANTILEVER_BC). The term "boundary conditions" refers both to displacement constraints and applied loading. The argument BC_PROCEDURE permits the users to supply their own boundary condition procedure, but keep in mind that this may drastically change the problem definition.

2.2.3.3 DISTORT

Distorted mesh parameter (default: 0.07071).

2.2.3.4 DRILLING_DOF

Drilling degree of freedom flag (default: <false>). Drilling freedoms are defined as rotations normal to the surface of the shell. Leaving this flag off forces all drilling freedoms in the model to be suppressed. Turning it on forces all drilling freedoms to be active — unless they are automatically suppressed using use of the AUTO_DOF_SUP argument. Note that while many shell elements do not have any rotational stiffness associated with their own surface-normal directions (at nodes), when shell elements are assembled as facets approximating an arbitrary shell surface, there is usually some misalignment between the element normal and the actual shell normal. This is especially true of "flat" (*e.g.*, 4-node) elements. Hence, *some* rotational stiffness about the *shell* normal is usually present in the model. (A clear exception to this is a flat plate, where element and shell normals are identical.) For a cylindrical shell, the misalignment diminishes only as the number of elements is increased. Most shell elements in the Testbed have their own misalignment tolerance parameter, which determines when the AUTO_DOF_SUP argument will automatically suppress the drilling freedom. Note that for elements which *have* drilling stiffness, the DRILLING_DOF argument should be set to <true> regardless of how AUTO_DOF_SUP is set.

2.2.3.5 E

Young's elastic modulus (default: 1.0×10^7).

2.2.3.6 ES_NAME

Element name (default: EX41). This is the name of the specific shell-element type you wish to select, within the element processor defined by argument ES_PROC. The default shell-element type, EX41, is a 4-noded quadrilateral element implemented in Processor ES2, and described in The Computational Structural Mechanics Testbed User's Manual (see ref. 2.2-1).

2.2.3.7 ES_PARS

Element research parameters (default: 0., ...). This argument allows an optional list of element-dependent parameters that some elements provide, primarily when the element is still undergoing research and refinement.

2.2.3.8 ES_PROC

Element processor (default: ES2) This is the name of the structural element (ES) processor that contains the shell element type you wish to employ in the model. The default shell-element, processor ES2, is described in The Computational Structural Mechanics Testbed User's Manual.

2.2.3.9 NODES_X

Number of nodes along x-direction (default: 7). This is the number of nodes you wish to have along the axial direction of the beam shell model. Note that this number should be consistent with the number of nodes per element. For example, NODES_X can be any number greater than 1 for 4-node quadrilateral elements, whereas it must be an odd number greater than 1 for 9-node quadrilateral elements.

2.2.3.10 NODES_Y

Number of nodes along y-direction (default: 2). This is the number of nodes you wish to have along the depth direction of the beam shell model. Note that this number should be consistent with the number of nodes per element. For example, NODES_Y can be any number greater than 1 for 4-node quadrilateral elements, whereas it must be an odd number greater than 1 for 9-node quadrilateral elements.

2.2.3.11 NU

Poisson's ratio (default: 0.3).

2.2.3.12 THICKNESS

Beam thickness (default: 0.1).

2.2.4 USAGE GUIDELINES AND EXAMPLES

Procedure GEN_CANTILEVER may be used by preceding the procedure name by the *call directive. Procedure arguments may be changed from their default values by including any or all of the arguments and their new values when the procedure is called. A space or blank is required between the end of the procedure name and the left parenthesis. If the default values of the procedure arguments are to be used, then only the procedure name is required.

```
*call GEN_CANTILEVER ( es_proc       = ES2 ; es_name   = EX41 ; --
                       es_pars       = 0.0 ; --
                       auto_dof_sup  = <true> ; --
                       drilling_dof  = <false> ; --
                       nodes_x       = 7   ; nodes_y   = 2    ; --
                       length_x      = 6.  ; length_y  = .2   ; --
                       E=1.E7 ;      NU=.3   ; thickness  = .1  ; --
                       distort        =.07071; --
                       BC_PROCEDURE = CANTILEVERLEVER_BC )
```

(E1) To perform an entire analysis using the default options, simply invoke the procedure without any arguments, *i.e.*,

```
*call GEN_CANTILEVER
```

2.2.5 LIMITATIONS

2.2.6 ERROR MESSAGES AND WARNINGS

None.

2.2.7 PROCEDURE FLOWCHART

2.2.8 PROCEDURE LISTING

```
*procedure GEN_CANTI ( es_proc      = ES2 ; es_name  = EX41 ; --
                       es_pars      = 0.0 ; --
                       auto_dof_sup  = <true> ; --
                       drilling_dof  = <false> ; --
                       nodes_x       = 7   ; nodes_y   = 2    ; --
                       length_x      = 6.  ; length_y  = .2   ; --
                       E=1.E7 ;    NU=.3   ; thickness     = .1  ; --
                       distort       =.07071; --
                       BC_PROCEDURE = CANTILEVER_BC )
.   ------------------------------------------------------------------
.   Register Element and Define Macros: ELT_NEN, ELT_NIP, ELT_NSTR, etc.
.   ------------------------------------------------------------------
    *call ES ( function = 'DEFINE ELEMENTS' ; es_proc = [es_proc]; --
               es_name =  [es_name]       ; es_pars = [es_pars] )
[XQT TAB
*def nodes_tot = < [nodes_x] * [nodes_y] >
 START <nodes_tot>
 JLOC
.
. DEFINE NODAL COORDINATES
.
*def/e dx = < [length_x] / ([nodes_x]-1) >
*def/e dy = < [length_y] / ([nodes_y]-1) >
*def/i node = 0
*def/e y  = 0.
*def/e skew = < -1. * [distort] >
*def/e dskew = < 2.0 * [distort] / ([nodes_y]-1) >
.
*do   $j  = 1,[nodes_y]
      *def/e x  = 0.
```

```
   *do   $i  = 1,[nodes_x]
      *def node = < <node> + 1 >
       <node>  <x>, <y>, 0.                . NODE DEFINITION
      *def x = < <x> + <dx> >
         *if < <$i> /eq 1 > /then
            *def x = < <x> + <skew> >
         *endif
         *if < <$i> /eq <[nodes_x]-1> > /then
            *def x = < [length_x] >
         *endif
   *enddo
   *def y = < <y> + <dy> >
   *def skew = < <skew> + <dskew> >
*enddo
.
. DEFINE FICTITIOUS ELASTIC MATERIAL PROPERTIES
.
 MATC
 1 1.0 .3
.
. DEFINE LOADS AND BOUNDARY CONDITIONS
.
 *call [BC_PROCEDURE] ( nnx = [nodes_x] ; nny = [nodes_y] ; --
                        nen = <es_nen> ; drilling_dof = [drilling_dof] )
.
. DEFINE REAL MATERIAL/SECTION PROPERTIES
.
[XQT AUS
.
. Build Table of Material Data
TABLE(NI=16,NJ=1): OMB DATA 1 1
.
*def/e12.4 G = < [E] / (2.*(1.+[NU])) >
.
I=1,2,3,4,5,6
J=1
[E] [NU] [E]  <G> <G> <G>
.
. Build Laminate Data Tables
TABLE(NI=3,NJ=1,ITYPE=0): LAM OMB 1 1
I=1,2,3 . (material_type, layer_thickness, angle(deg.)
J=1:          1             [THICKNESS]          0.0
.
[XQT LAU
.
. GENERATE ELEMENTS
.
      [XQT ELD
.
.     Define number of integration (stress) points based on element type
.
     *def/i nst  = < <es_nip>*<es_nstr> >
.
```

```
.      Define element attributes
.
    <ES_EXPE_CMD>
.
     NSECT = 1
.
.     Define element nodal connectivity
.
     *call CANTI_ELT_CONN (nnx=[nodes_x]; nny=[nodes_y]; nen=<es_nen>)
      ---------------------------------------
.     Suppress DOFs not supported by elements
.     ---------------------------------------
      *if < [AUTO_DOF_SUP] > /then
         *call ES ( function = 'DEFINE FREEDOMS' )
      *endif
.
*end
*procedure CANTI_ELT_CONN ( nnx; nny; nen )
.
.        ============================
. Define Element Connectivity Record for ELD Processor
.        ============================
.
   *if < [nen] /eq 4 > /then
      *do $iy = 1, <[nny]-1>
         *do $ix = 1, <[nnx]-1>
            *def/i n1 =  < (<$iy>-1)*[nnx] + <$ix> >
            *def/i n2 =  < <n1> +   1   >
            *def/i n3 =  < <n2> + [nnx] >
            *def/i n4 =  < <n3> -   1   >
.
.            ======================
             <n1>  <n2>  <n3>  <n4>
.            ======================
.
         *enddo
      *enddo
   *elseif < [nen] /eq 9 > /then
      *do $iy = 1, <[nny]-2>, 2
         *do $ix = 1, <[nnx]-2>, 2
            *def/i n1 =  < (<$iy>-1)*[nnx] + <$ix> >
            *def/i n2 =  < <n1> + 2 >
            *def/i n3 =  < <n2> + (2*[nnx]) >
            *def/i n4 =  < <n3> - 2 >
            *def/i n5 =  < <n1> + 1 >
            *def/i n6 =  < <n2> + [nnx] >
            *def/i n7 =  < <n4> + 1 >
            *def/i n8 =  < <n6> - 2 >
            *def/i n9 =  < <n8> + 1 >
.
.            ==========================================
             <n1> <n2> <n3> <n4> <n5> <n6> <n7> <n8> <n9>
.            ==========================================
```

```
.
        *enddo
      *enddo
   *endif
*end
```

2.2.9 REFERENCES

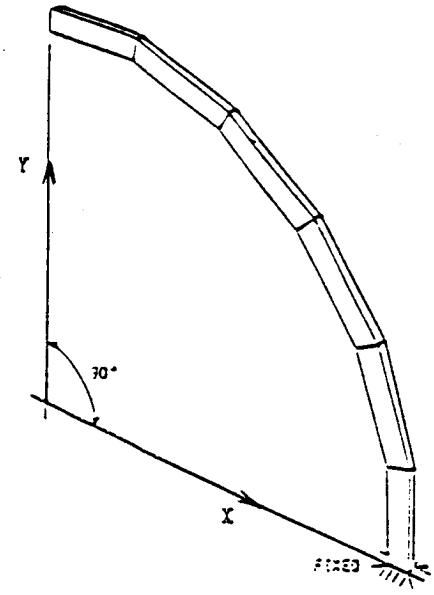
2.2-1 MacNeal, R. H.; and Harder, R. L.: "A Proposed Set of Problems to Test Finite Element Accuracy," *Finite Elements in Analysis and Design*, Vol. 1, 1985, pp. 3-20.

2.2-2 Stewart, Caroline B.: *The Computational Structural Mechanics Testbed User's Manual.* NASA TM-100644, October 1989.

2.3 Procedure GEN_CURVED_BM

2.3.1 GENERAL DESCRIPTION

This section describes a procedure which generates models of a curved (circular) beam using two-dimensional plate/shell elements. Using the default values for the procedure arguments, the curved beam problem from the MacNeal-Harder test cases (see ref. 2.3-1) is generated. The MacNeal-Harder curved beam test case is shown in figure 2.3-1.



PROBLEM : Curved Beam

DIMENSIONS : Inner radius = 4.12 Outer radius = 4.32
Thickness = .1

MATERIAL PROPERTIES : $E = 1 \times 10^{7}$ $\nu = .25$

BOUNDARY CONDITIONS : Cantelever beam fixed at y = 0

LOADING : Unit forces applied at free end ;
1) in-plane (vertical) -- y-direction (case 1)
2) out-of-plane -- z-direction (case 2)

Figure 2.3-1 Generic 2-D Curved Beam Problem.

2.3.2 PROCEDURE USAGE

Procedure GEN_CURVED_BM may be used by preceding the procedure name by the *call directive, and following it by a list of arguments enclosed in parentheses. Procedure arguments are order-independent, and most have default values thus making them optional. The formal syntax is as follows:

```
*call GEN_CURVED_BM ( arg1 = val1 ; arg2 = val2 ; ...)
```

where arg1 and arg2 represent argument names, and val1 and val2 represent their corresponding values. Note that semi-colons are required between arguments, and a double dash (--) may be used to continue the argument list on the next line.

The allowable arguments for procedure GEN_CURVED_BM are summarized in the following table, along with their default values (if they exist). Note that arguments without defaults are generally *mandatory*, while those with defaults are generally optional. Exceptions to this rule are noted in the following section under detailed argument descriptions.

Argument	Default Value	Meaning
ES_PROC	ES2	Select element processor
ES_NAME	EX41	Select element within ELT_PROC
ES_PARS	0.0	Set element-research parameters
AUTO_DOF_SUP	<true>	Automatic d.o.f. suppression
DRILLING_DOF	<false>	
NODES_T	7	Number of nodes in tangential direction
NODES_R	2	Number of nodes in radial direction
RIN	4.12	Inner radius
ROUT	4.32	Outer radius
E	1.E7	Young's elastic modulus
NU	.25	Poisson's ratio
THICKNESS	.1	Thickness
BC_PROCEDURE	CURVED_BC	Procedure for boundary conditions

2.3.3 ARGUMENT DESCRIPTIONS

2.3.3.1 AUTO_DOF_SUP

Automatic degree of freedom suppression flag (default: <true>). This option provides a convenient way of suppressing any freedoms that do not have any (or adequate) stiffness associated with them — for example, at nodes used to prescribe geometry only; or drilling freedoms in fine meshes composed of elements without normal rotational stiffnesses (see argument DRILLING_DOF).

2.3.3.2 BC_PROCEDURE

Boundary condition procedure name (default: `CURVED_BC`). The term "boundary conditions" refers both to displacement constraints and applied loading. The argument `BC_PROCEDURE` permits you to to supply your own boundary condition procedure, but keep in mind that this may drastically change the problem definition.

2.3.3.3 DRILLING_DOF

Drilling degree of freedom flag (default: `<false>`). Drilling freedoms are defined as rotations normal to the surface of the shell. Leaving this flag off forces all drilling freedoms in the model to be suppressed. Turning it on forces all drilling freedoms to be active — unless they are automatically suppressed using use of the `AUTO_DOF_SUP` argument. Note that while many shell elements do not have any rotational stiffness associated with their own surface-normal directions (at nodes), when shell elements are assembled as facets approximating an arbitrary shell surface, there is usually some misalignment between the element normal and the actual shell normal. This is especially true of "flat" (*e.g.*, 4-node) elements. Hence, *some* rotational stiffness about the *shell* normal is usually present in the model. (A clear exception to this is a flat plate, where element and shell normals are identical.) For a cylindrical shell, the misalignment diminishes only as the number of elements is increased. Most shell elements in the Testbed have their own misalignment tolerance parameter, which determines when the `AUTO_DOF_SUP` argument will automatically suppress the drilling freedom. Note that for elements which *have* drilling stiffness, the `DRILLING_DOF` argument should be set to `<true>` regardless of how `AUTO_DOF_SUP` is set.

2.3.3.4 E

Young's elastic modulus (default: 1.0×10^7).

2.3.3.5 ES_NAME

Element name (default: `EX41`). This is the name of the specific shell-element type you wish to select, within the element processor defined by argument `ES_PROC`. The default shell-element type, `EX41`, is a 4-noded quadrilateral element implemented in Processor `ES1`, and described in The Computational Structural Mechanics Testbed User's Manual (see ref. 2.3-1).

2.3.3.6 ES_PARS

Element research parameters (default: `0.`, ...). This argument allows an optional list of element-dependent parameters that some elements provide, primarily when the element is still undergoing research and refinement.

2.3.3.7 ES_PROC

Element processor (default: `ES2`) This is the name of the structural element (ES) processor that contains the shell element type you wish to employ in the model. The default shell-element, processor `ES2`, is described in The Computational Structural Mechanics Testbed User's Manual.

2.3.3.8 NODES_R

Number of radial nodes (default: 2). This is the number of nodes you wish to have along the radial direction of the curved beam shell model. Note that this number should be consistent with the number of nodes per element. For example, NODES_R can be any number greater than 1 for 4-node quadrilateral elements, whereas it must be an odd number greater than 1 for 9-node quadrilateral elements.

2.3.3.9 NODES_T

Number of tangential nodes (default: 7). This is the number of nodes you wish to have along the tangential direction of the curved beam shell model. Note that this number should be consistent with the number of nodes per element. For example, NODES_T can be any number greater than 1 for 4-node quadrilateral elements, whereas it must be an odd number greater than 1 for 9-node quadrilateral elements.

2.3.3.10 NU

Poisson's ratio (default: 0.25).

2.3.3.11 RIN

Inner radius of curved beam (default: 4.12).

2.3.3.12 ROOT

Outer radius of curved beam (default: 4.32).

2.3.3.13 THICKNESS

Beam thickness (default: 0.1).

2.3.4 USAGE GUIDELINES AND EXAMPLES

Procedure GEN_CURVED_BM may be used by preceding the procedure name by the *call directive. Procedure arguments may be changed from their default values by including any or all of the arguments and their new values when the procedure is called. A space or blank is required between the end of the procedure name and the left parenthesis. If the default values of the procedure arguments are to be used, then only the procedure name is required.

```
*procedure GEN_CURVED_BM (elt_proc      = ES2 ; elt_name  = EX41 ; --
                          elt_pars      = 0.0 ; --
                          auto_dof_sup  = <true> ; --
                          drilling_dof  = <false> ; --
                          nodes_t       = 7   ; nodes_r   = 2    ; --
                          rin           = 4.12  ; rout    = 4.32   ; --
                          E=1.E7 ;    PR=.25    ; thick     = .1  ; --
                          BC_PROCEDURE = CURVED_BC )
```

(E1) To perform an entire analysis using the default options, simply invoke the procedure without any arguments, *i.e.*,

```
*call GEN_CURVED_BM
```

2.3.5 LIMITATIONS

2.3.6 ERROR MESSAGES AND WARNINGS

None.

2.3.7 PROCEDURE FLOWCHART

2.3.8 PROCEDURE LISTING

```
*procedure GEN_CURVED (es_proc      = ES2 ; es_name  = EX41 ; --
                       es_pars      = 0.0 ; --
                       auto_dof_sup  = <true> ; --
                       drilling_dof  = <false> ; --
                       nodes_t       = 7   ; nodes_r   = 2    ; --
                       rin           = 4.12  ; rout    = 4.32   ; --
                       E=1.E7 ;    NU=.25   ; thickness     = .1  ; --
                       BC_PROCEDURE = CURVED_BC )
.   ------------------------------------------------------------------
.   Register Element and Define Macros: ELT_NEN, ELT_NIP, ELT_NSTR, etc.
.   ------------------------------------------------------------------
    *call ES ( function = 'DEFINE ELEMENTS' ; es_proc = [es_proc]; --
               es_name =  [es_name]       ; es_pars = [es_pars] )
[XQT TAB
*def nodes_tot = < [nodes_t] * [nodes_r] >
 START <nodes_tot>
 JLOC
       FORMAT = 2                . use cylindrical coordinate system
.
. DEFINE NODAL COORDINATES
.
*def/e dx = < 90. / ([nodes_t]-1) >
*def/e dy = < < [rout] - [rin] > / ([nodes_r]-1) >
*def/i node = 0
*def/e r  = [rin]
.
*do   $j  = 1,[nodes_r]
      *def/e theta  = 90.
   *do   $i  = 1,[nodes_t]
      *def node = < <node> + 1 >
```

```
      <node>  <r>, <theta>, 0.                . NODE DEFINITION
     *def theta = < <theta> - <dx> >
   *enddo
     *def r = < <r> + <dy> >
*enddo
.
. DEFINE FICTITIOUS ELASTIC MATERIAL PROPERTIES
.
 MATC
 1 1.0 .3
.
.      ====================
.      Define DOF Directions
.      ====================
.
         JREF                      .   Use local cylindrical basis vectors
                                   .   for nodal DOFS:
                                   .   u,v,w = radial, circumfer., axial
         NREF = -1
         1  <nodes_tot>            .   same convention for all nodes
.
. DEFINE LOADS AND BOUNDARY CONDITIONS
.
 *call [BC_PROCEDURE] ( nnx = [nodes_t] ; nny = [nodes_r] ; --
                        nen = <es_nen> ; drilling_dof = [drilling_dof] )
.
. DEFINE REAL MATERIAL/SECTION PROPERTIES
.
[XQT AUS
.
. Build Table of Material Data
TABLE(NI=16,NJ=1): OMB DATA 1 1
.
*def/e12.4 G = < [E] / (2.*(1.+[NU])) >
.
I=1,2,3,4,5,6
J=1
[E] [NU] [E]  <G> <G> <G>
.
. Build Laminate Data Tables
TABLE(NI=3,NJ=1,ITYPE=0): LAM OMB 1 1
I=1,2,3 . (material_type, layer_thickness, angle(deg.)
J=1:          1             [THICKNESS]          0.0
.
[XQT LAU
.
. GENERATE ELEMENTS
.
       [XQT ELD
.
.      Define number of integration (stress) points based on element type
.
      *def/i nst  = < <es_nip>*<_nstr> >
```

```
.
.      Define element attributes
.
    <ES_EXPE_CMD>
.
   NSECT = 1
.
.      Define element nodal connectivity
.
     *call CURV_ELT_CONN (nnx=[nodes_t]; nny=[nodes_r]; nen=<es_nen>)
.      ---------------------------------------
.      Suppress DOFs not supported by elements
.      ---------------------------------------
       *if < [AUTO_DOF_SUP] > /then
          *call ES ( function = 'DEFINE FREEDOMS' )
       *endif
.
*end
*procedure CURV_ELT_CONN ( nnx; nny; nen )
.
.         ============================
. Define Element Connectivity Record for ELD Processor
.         ============================
.
   *if < [nen] /eq 4 > /then
      *do $iy = 1, <[nny]-1>
         *do $ix = 1, <[nnx]-1>
            *def/i n1 =  < (<$iy>-1)*[nnx] + <$ix> >
            *def/i n2 =  < <n1> +   1   >
            *def/i n3 =  < <n2> + [nnx] >
            *def/i n4 =  < <n3> -   1   >
.
.             ======================
              <n1>  <n2>  <n3>  <n4>
.             ======================
.
         *enddo
      *enddo
   *elseif < [nen] /eq 9 > /then
      *do $iy = 1, <[nny]-2>, 2
         *do $ix = 1, <[nnx]-2>, 2
            *def/i n1 =  < (<$iy>-1)*[nnx] + <$ix> >
            *def/i n2 =  < <n1> + 2 >
            *def/i n3 =  < <n2> + (2*[nnx]) >
            *def/i n4 =  < <n3> - 2 >
            *def/i n5 =  < <n1> + 1 >
            *def/i n6 =  < <n2> + [nnx] >
            *def/i n7 =  < <n4> + 1 >
            *def/i n8 =  < <n6> - 2 >
            *def/i n9 =  < <n8> + 1 >
.
.           ==========================================
            <n1> <n2> <n3> <n4> <n5> <n6> <n7> <n8> <n9>
```

```
.           ==========================================
.
         *enddo
      *enddo
   *endif
*end
```

2.3.9 REFERENCES

2.3-1 MacNeal, R. H.; and Harder, R. L.: "A Proposed Set of Problems to Test Finite Element Accuracy," *Finite Elements in Analysis and Design*, Vol. 1, 1985, pp. 3-20.

2.3-2 Stewart, Caroline B.: *The Computational Structural Mechanics Testbed User's Manual.* NASA TM-100644, October 1989.

2.4 Procedure GEN_PLATE

2.4.1 GENERAL DESCRIPTION

Procedure **GEN_PLATE** is used to generate flat or warped 4-sided plate finite element models. The plate edges are defined to be straight with the surface defined as the bi-linearly interpolation of the edges. This type of interpolated surface is also known as a Coon's surface (see refs. 2.4-2 and 2.4-3).

2.4.2 PROCEDURE USAGE

Procedure **GEN_PLATE** may be used by preceding the procedure name by the ***call** directive, and following it by a list of arguments enclosed in parentheses. Procedure arguments are order-independent, and most have default values thus making them optional. The formal syntax is as follows:

```
*call GEN_PLATE ( arg1 = val1 ; arg2 = val2 ; ...)
```

where **arg1** and **arg2** represent argument names, and **val1** and **val2** represent their corresponding values. Note that semi-colons are required between arguments, and a double dash (--) preceded by a space may be used to continue the argument list on the next line.

The allowable arguments for procedure **GEN_PLATE** are summarized in the following table, along with their default values (if they exist). Exceptions to this rule are noted in the following section under detailed argument descriptions.

Argument	Default Value	Meaning
ES_PROC	ES1	Generic element processor
ES_NAME	EX97	Generic element name
ES_PARS	0.0	Element research parameters
XYZ1	1,0,0	Cartesian coordinates of point 1.
XYZ2	1,0,1	Cartesian coordinates of point 2.
XYZ3	1,90,1	Cartesian coordinates of point 3.
XYZ4	1,90,0	Cartesian coordinates of point 4.
NODES_1	7	Number of nodes along edge 1
NODES_2	7	Number of nodes along edge 2 including duplicate nodes if plate closes
EDGE_WEIGHTS	1,1,1,1	Plate section property procedure
BC_PROCEDURE	' '	Boundary condition procedure
DRILLING_DOF	<true>	Drilling dof suppression flag
AUTO_DOF_SUP	<true>	Automatic dof suppression flag
SECTION_PRC	' '	Plate section property procedure
NSECT	1	Plate section property number
The following values not used if SECTION_PRC is specified.		
E	30.E6	Young's Modulus
NU	0.3	Poisson's ratio
WTDEN	0.1	Weight Density
THICKNESS	.1	Plate thickness

2.4.3 ARGUMENT DESCRIPTIONS

2.4.3.1 AUTO_DOF_SUP

Automatic degree of freedom suppression flag (default: <true>). This option provides a convenient way of suppressing any freedoms that do not have any (or adequate) stiffness associated with them — for example, at nodes used to prescribe geometry only; or drilling freedoms in fine meshes composed of elements without normal rotational stiffness (see argument DRILLING_DOF).

2.4.3.2 BC_PROCEDURE

Name of user provided boundary condition procedure (default: ' '). The term "boundary conditions" refers both to displacement constraints and applied loading. If a boundary conditions procedure is provided, the following call will be performed. The macrosymbol <es_nen> equals the number of element nodes.

```
*call [BC_PROCEDURE] ( nodes_1 = [nodes_1] ; --
                       nodes_2 = [nodes_2] ; --
                       es_nodes = <es_nen> ; --
                       drilling_dof = [drilling_dof] )
```

No action is taken if a boundary condition procedure name is not provided.

2.4.3.3 DRILLING_DOF

Drilling degree of freedom flag (default: <true>). Drilling freedoms are defined as rotations normal to the surface of the plate. Setting this flag set to <false> forces all drilling freedoms in the model to be suppressed. Setting it to true forces all drilling freedoms to be active — unless they are automatically suppressed by use of the AUTO_DOF_SUP argument. Note that while many plate elements do not have any rotational stiffness associated with their own surface-normal directions (at nodes), when plate elements are assembled as facets approximating an arbitrary plate surface, there is usually some misalignment between the element normal and the actual plate normal. This is especially true of "flat" (*e.g.*, 4-node) elements. Hence, *some* rotational stiffness about the *plate* normal is usually present in the model. (A clear exception to this is a flat plate, where element and plate normals are identical.) For a curved plate, the misalignment diminishes only as the number of elements is increased. Most plate elements in the Testbed have their own misalignment tolerance parameter, which determines when the AUTO_DOF_SUP argument will automatically suppress the drilling freedom. Note that for elements which *have* drilling stiffness, the DRILLING_DOF argument should be set to <true> regardless of how AUTO_DOF_SUP is set.

2.4.3.4 E

Young's modulus (default: 30.E6). This argument is ignored if SECTION_PRC parameter is specified. See the description for SECTION_PRC for more detail.

2.4.3.5 EDGE_WEIGHTS

Node placement can be weighted along each surface edge according to the EDGE_WEIGHTS parameter. The input format requires a list of four edge-node placement weightings representing the node weighting for edge1, edge2, edge3, and edge4 (default: 1.,1.,1.,1.).

The weighting value for a given edge represents the length of the last element divided by the length of the first element along that edge. The edge orientation arrows in figure 2.4-1 point from the first element to the last element along each edge. In the case of 9-node quad elements, the midside and center nodes are positioned at the appropriate locations based on the elements natural coordinate system.

The procedure interprets negative weight values to mean the positive reciprocal. For example, a value of -5.0 is identical to a value of 0.2.

2.4.3.6 ES_NAME

Element name (default: EX97). This argument is the name of the specific plate-element type you wish to select, within the element processor defined by argument ES_PROC. The default plate-element type, EX97, is a 9-node quadrilateral element implemented in processor ES1, and described reference 2.4-1.

2.4.3.7 ES_PARS

Element research parameters (default: 0., ...). This argument is an optional list of element-dependent parameters that some elements provide, primarily when the element is still undergoing research and refinement.

2.4.3.8 ES_PROC

Element processor (default: ES1) This argument is the name of the structural element (ES) processor that contains the plate element type you wish to employ in the model. The default plate-element, processor ES1, is described in The Computational Structural Mechanics Testbed User's Manual.

2.4.3.9 NODES_1

Number of nodes on edge 1 including the nodes at the surface corners (default: 7). This argument is also the number of nodes on edge 3. This number should be consistent with the element type selected. For example, NODES_1 can be any number greater than 1 for 4-node quadrilateral elements, whereas it must be an odd number greater than 1 for 9-node quadrilateral elements.

2.4.3.10 NODES_2

Number of nodes on edge 2 including the nodes at the surface corners (default: 7). This argument is also the number of nodes on edge 4. This number should be consistent with the element type selected. For example, NODES_2 can be any number greater than 1 for 4-node quadrilateral elements, whereas it must be an odd number greater than 1 for 9-node quadrilateral elements.

2.4.3.11 NSECT

Plate section property number (default: 1). The NSECT value is required when defining the element using the processor ELD. See the description of SECTION_PRC for more detail.

2.4.3.12 NU

Poisson's ratio (default: 0.3). This argument is ignored if the SECTION_PRC input parameter is specified. See the description of SECTION_PRC for more detail.

2.4.3.13 XYZ1

The cartesian coordinates (x, y, z) which define corner number 1 of the model surface. The form of the input is three real values, each separated by a comma (default: 1.,0.,0.). The surface is defined by four edges which are defined as a linear interpolation in cartesian coordinates of four endpoints, or "corner" points.

2.4.3.14 XYZ2

The cartesian coordinates (x, y, z) defining the corner number 2 of the model surface (default: 1.,0.,1.).

2.4.3.15 XYZ3

The cartesian coordinates (x, y, z) defining the corner number 3 of the model surface (default: 1.,90.,1.).

2.4.3.16 XYZ4

The cartesian coordinates (x, y, z) defining the corner number 4 of the model surface (default: 1.,90.,0.).

2.4.3.17 SECTION_PRC

Name of a user supplied procedure to define the plate section properties (default = ' '). If a section properties procedure is provided, the following call will be performed.

```
*call [section_prc] ( nsect = [nsect] )
```

The effect of the default is to allow the procedure to generate an isotropic material section based on the input parameters **E**, **NU**, **WTDEN**, and **THICKNESS**. The section number is defined by the input parameter **NSECT**. If the call parameter **SECTION_PRC** is defined by the user, then call parameters **E**, **NU**, **WTDEN**, and **THICKNESS** are ignored by procedure **GEN_PLATE**.

2.4.3.18 THICKNESS

Thickness of the plate wall (default = 1.0). This argument is ignored if **SECTION_PRC** parameter is specified. See the description for **SECTION_PRC** for more detail.

2.4.3.19 WTDEN

Weight density expressed in lb/in.^{3} (default: 0.1 lb/in.^{3}). This argument is ignored if the **SECTION_PRC** input parameter is specified. Processor LAU will convert the weight density to mass density using the gravitational acceleration constant 386.4 in/sec^2.

2.4.4 USAGE GUIDELINES AND EXAMPLES

Procedure **GEN_PLATE** may be invoked using the *call directive. Procedure arguments may be changed from their default values by including any or all of the arguments and their new values in the procedure call. A space or blank is required between the end of the procedure name and the left parenthesis. If the default values of the procedure arguments are to be used, then only the procedure name is required.

```
*procedure GEN_PLATE ( ES_PROC = ES1 ; ES_NAME = EX97 ; ES_PARS = 0.0 ; --
                XYZ1          = 0.,0.,0. ; --
                XYZ2          = 1.,0.,0. ; --
                XYZ3          = 1.,1.,0. ; --
                XYZ4          = 0.,1.,0. ; --
                NODES_1       = 3 ; --
                NODES_2       = 3 ; --
                EDGE_WEIGHTS   = 1.,1.,1.,1. ; --
                BC_PROCEDURE = ' ' ; -- . Boundary condition procedure
                DRILLING_DOF  = <true> ; --
                AUTO_DOF_SUP  = <true> ; --
                SECTION_PRC = ' ' ; --  . Plate section property procedure
                NSECT = 1 ; --          . Plate section property ID
                The following values not used if SECTION_PRC specified
```

```
        E = 30.E6 ; --        . Young's Modulus
        NU = .3 ; --          . Poissons ratio
        THICKNESS = .1 --     . Plate thickness
)
```

2.4.4.1 Mesh Generation

The method of surface generation used by procedure **GEN_PLATE** is described in the section. Terminology depicted on figure 2.4-1 provides a visual interpretation of the parameters used to generate a general plate surface. Node generation capability is provided by the Testbed processor MESH.

To define the plate surface, the user defines four coordinate positions in the cartesian reference frame. These coordinate positions represent the corners of a straight-sided quadrilateral region. The surface of the region is defined as the bi-linear interpolation of the four sides (see refs. 2.4-2 and 2.4-3).

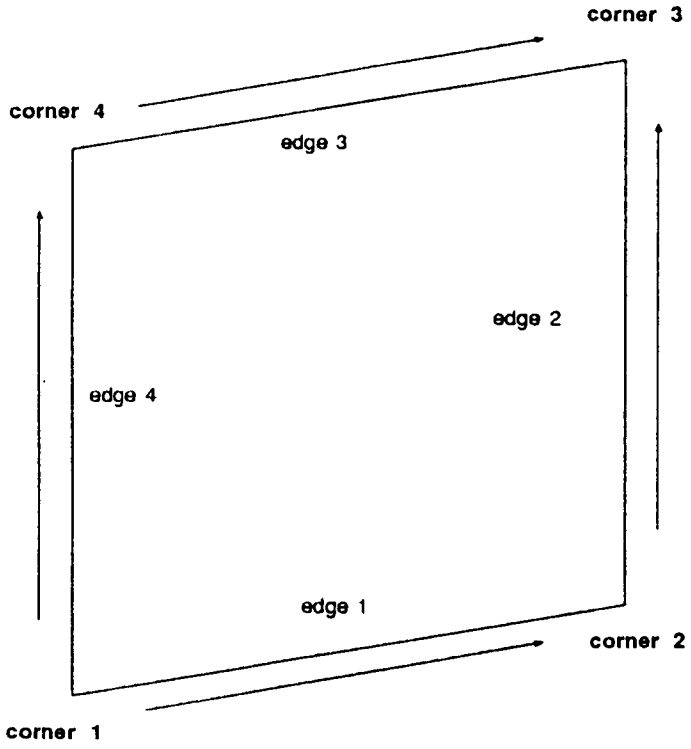


Figure 2.4-1 Generic Plate Surface Topology.

The connectivity of the surface is defined in figure 2.4-1. The user provides the coordinates for corners 1 to 4. Edge 1 of the region starts at corner 1 and ends at corner 2. The remaining edges are defined in a similar manner. The arrows indicate the orientation of the edges and the direction of increasing node numbers.

The topology of the resulting finite element grid is depicted in Figure 2.4-2. Nodes are created first along edge 1, then in successive lines terminating along edge 3. The user defines the number of nodes along edges 1 and 2, which also defines the number of nodes for edges 3 and 4. The relative position of the nodes along each edge may be controlled using the edge weighting parameter `EDGE_WEIGHTS`.

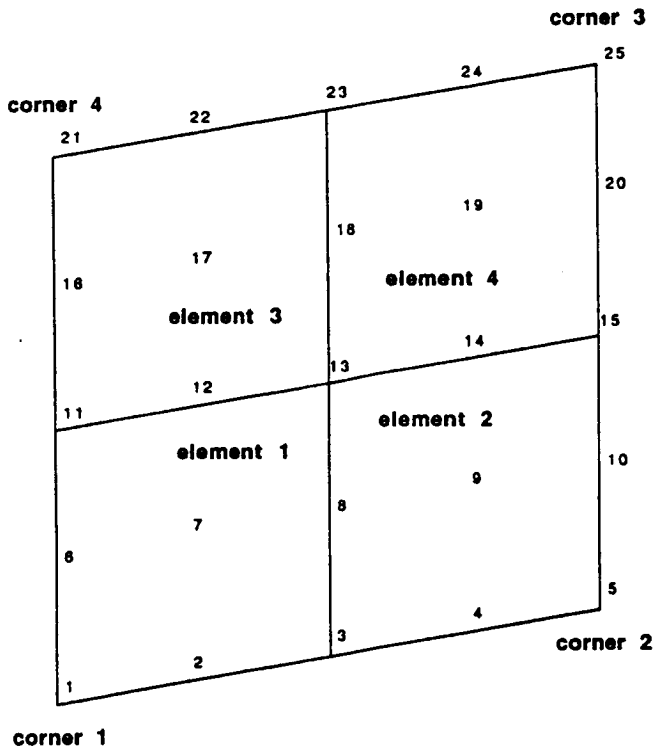


Figure 2.4-2 Node and Element Topology.

2.4.4.2 Flat Plates

In this section are presented examples of how procedure **GEN_PLATE** may be used to create two-dimensional finite element plate models.

In the following example, procedure **GEN_PLATE** is used to generate a flat rectangular plate with length of 10 inches and width of 5 inches using 4-noded quadrilateral elements (see figure 2.4-3).

```
*call GEN_PLATE ( es_proc = 'ES1' ; es_name = 'EX47' ; --
        xyz1          = 0.,0.,0. ; --
        xyz2          = 10.,0.,0. ; --
        xyz3          = 10.,5.,0. ; --
        xyz4          = 0.,5.,0. ; --
        nodes_1       = 9 ; --
        nodes_2       = 5 )
```

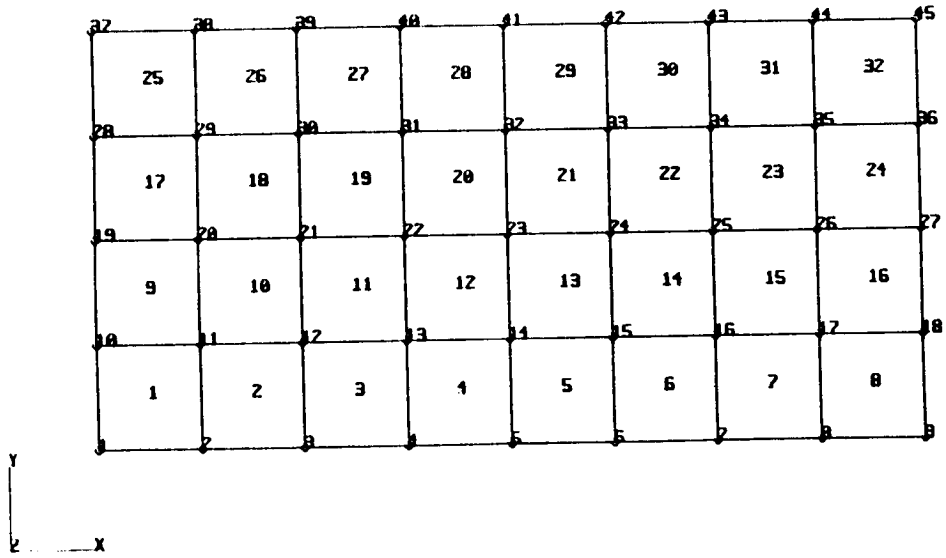
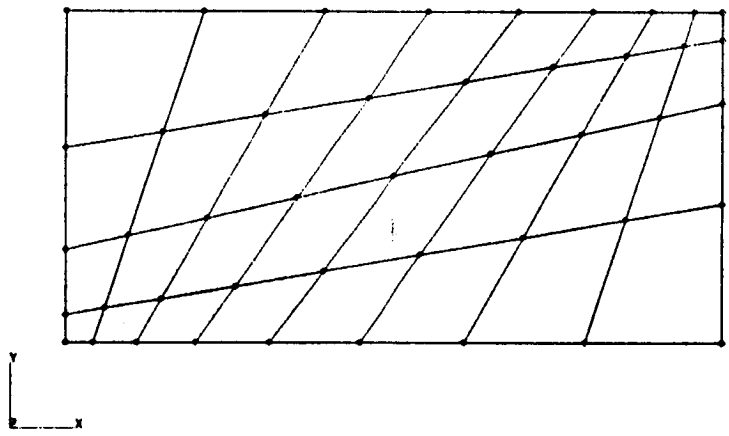


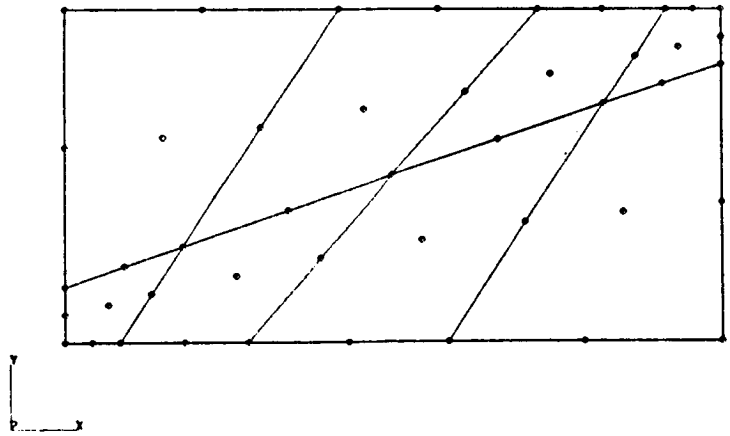
Figure 2.4-3 Rectangular Plate

This example demonstrates the use of the EDGE_WEIGHTS parameter and how it affects both the 4-node and 9-node quadrilateral element mapping (see figure 2.4-4). The EDGE_WEIGHTS specifies elements at the ends of edges 1 through 4 to be 5 times longer than elements at the beginning of the edges.

```
*call GEN_PLATE ( es_proc = 'ES1' ; es_name = 'EX97' ; --
        xyz1          = 0.,0.,0. ; --
        xyz2          = 10.,0.,0. ; --
        xyz3          = 10.,5.,0. ; --
        xyz4          = 0.,5.,0. ; --
        nodes_1       = 9 ; --
        nodes_2       = 5 ; --
        edge_weights  = 5.,-5.,-5.,5. )
```



(a) 4-node elements



(b) 9-node elements

Figure 2.4-4 Rectangular Plate With Weighted Elements

The following example produced the skewed flat plate shown in figure 2.4-5.

```
*call GEN_PLATE ( es_proc = 'ES1' ; es_name = 'EX47' ; --
        xyz1          = 0.,0.,0. ; --
        xyz2          = 10.,1.,0. ; --
        xyz3          = 7.,6.,0. ; --
        xyz4          = 2.,8.,0. ; --
        nodes_1       = 13 ; --
        nodes_2       = 11 )
```

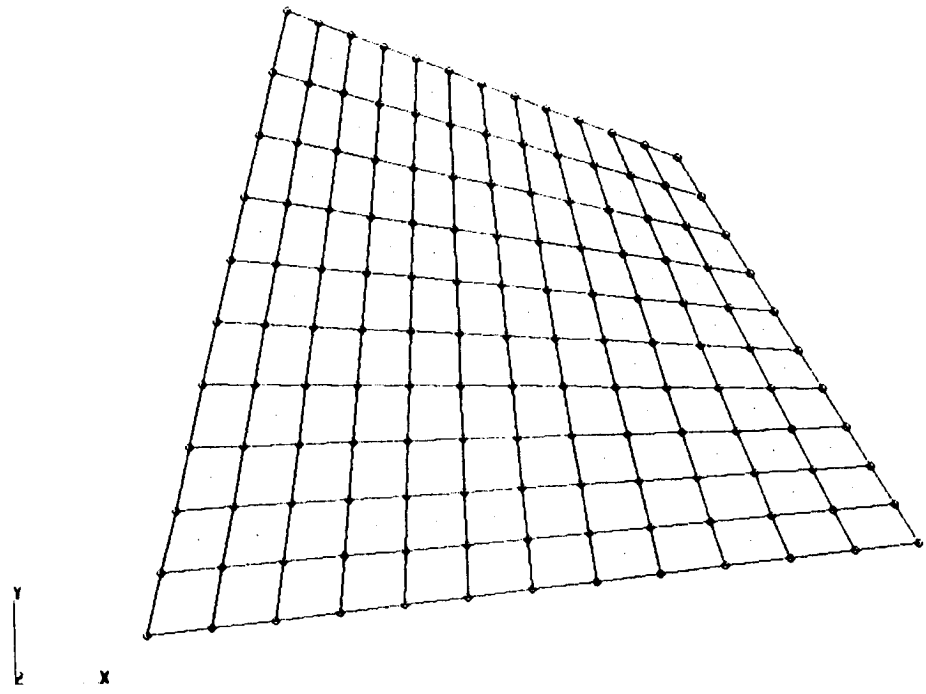


Figure 2.4-5 Skewed Flat Plate

2.4.4.3 Warped Plates

In the following example, procedure **GEN_PLATE** is used to generate the highly warped surface shown in figure 2.4-6.

```
*call GEN_PLATE ( es_proc = 'ES1' ; es_name = 'EX47' ; --
        xyz1          = 0.,0.,0. ; --
        xyz2          = 5.,5.,0. ; --
        xyz3          = 5.,0.,5. ; --
        xyz4          = 0.,5.,5. ; --
        nodes_1       = 21 ; --
        nodes_2       = 23 )
```

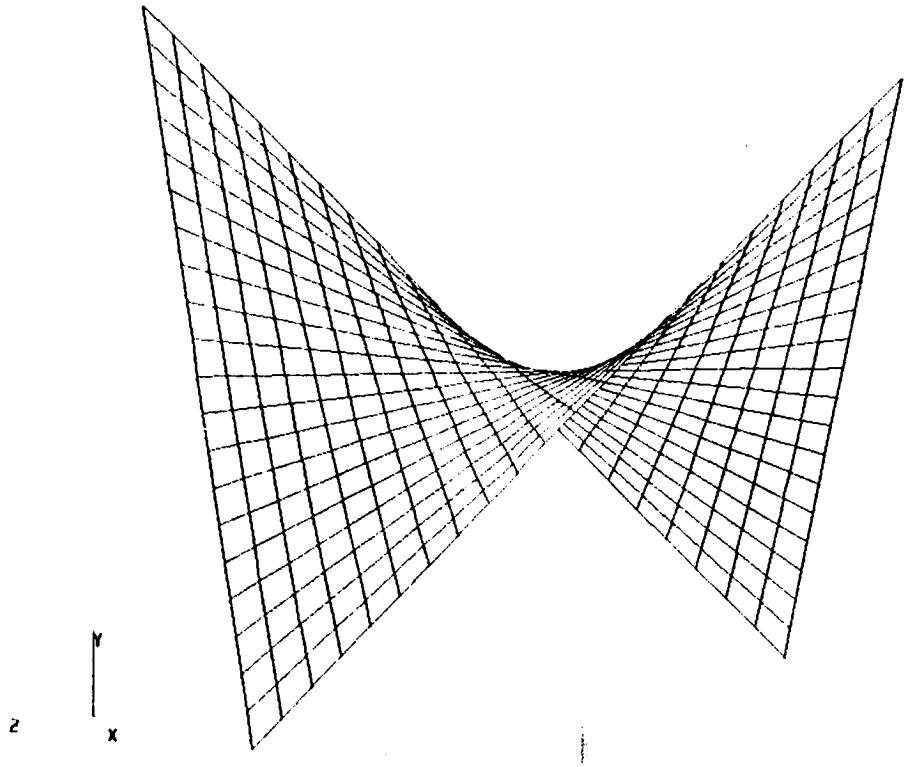


Figure 2.4-6 Warped Plate

2.4.5 LIMITATIONS

None.

2.4.6 ERROR MESSAGES AND WARNINGS

None.

2.4.7 PROCEDURE FLOWCHART

GEN_PLATE (Plate generation procedure)

[BC_PROCEDURE] (user supplied boundary conditions/loads procedure)

[SECTION_PRC] (user supplied section property generation procedure)

2.4.8 PROCEDURE LISTING

```
*procedure GEN_PLATE ( es_proc = es1 ; es_name = ex97 ; es_pars = 0.0 ; --
              xyz1          = 0.,0.,0. ; --
              xyz2          = 1.,0.,0. ; --
              xyz3          = 1.,1.,0. ; --
              xyz4          = 0.,1.,0. ; --
              nodes_1       = 3 ; --
              nodes_2       = 3 ; --
              edge_weights  = 1.,1.,1.,1. ; --
              online        = 0 ; -- . suppress nodes and element output
              bc_procedure = ' ' ; -- . Boundary condition procedure
              drilling_dof  = <true> ; --
              auto_dof_sup  = <true> ; --
              section_prc = ' ' ; --  . Shell section property procedure
              The following values used only if section_prc not specified
                   nsect = 1 ; --       . Shell section property ID
                   E = 30.E6 ; --       . Young's Modulus
                   NU = .3 ; --         . Poisons ratio
                   WTDEN = .1 ; --      . Weight Density
                   thickness = .1 --    . Shell thickness
            )
.
.   ================================================================
.   Model Definition Procedure for GENeric Plate in cartesian coordinates
.   ================================================================
.
.        A general purpose clip procedure to create the finite element
.         mesh for a plate with arbitraty straight sides using 4 or
.         9 noded quadrilateral elements.
.
.        If a procedure to generate shell section properties is not provided,
.         the isotropic section described by E, NU, WTDEN, and THICKNESS will
```

```
.          be automatically generated.  (see [section_prc] parameter)
.
.         A boundary condition procedure should be provided but is
.          optional.  If not provided, no boundary conditions will be
.          defined.  (see [bc_procedure] parameter)
.
*remark ************************************
*remark GEN_PLATE MODEL GENERATION PROCEDURE
*remark ************************************
.
      *def/i n1 = [nodes_1]
      *def/i n2 = [nodes_2]
      *def/e xyz1[1:3] = [xyz1]
      *def/e xyz2[1:3] = [xyz2]
      *def/e xyz3[1:3] = [xyz3]
      *def/e xyz4[1:3] = [xyz4]
      *def/e w[1:4] = [edge_weights]
      *def/e rc[1:4] = <xyz1[1]>,<xyz2[1]>,<xyz3[1]>,<xyz4[1]>
      *def/e tc[1:4] = <xyz1[2]>,<xyz2[2]>,<xyz3[2]>,<xyz4[2]>
      *def/e zc[1:4] = <xyz1[3]>,<xyz2[3]>,<xyz3[3]>,<xyz4[3]>
.
.         ================================================================
.         Register Element and Define Macros: ES_NEN, ES_NIP, ES_NSTR, etc.
.         ================================================================
      *call ES ( function = 'DEFINE ELEMENTS' ; es_proc = [es_proc]; --
                 es_name =  [es_name]       ; es_pars = [es_pars] )
.
.         ================================================================
.         Define nodal coordinates and element connectivities into
.          separate formatted files.  These files are gauranteed to have
.          unique names that are not currently in use in the current
.          directory.
.         ================================================================
.
      *def/i chk_closure = 0
      [xqt mesh
      *if <mesh_err> /then
         *remark  Error occurred during MESH processor execution.
         *remark  GEN_PLATE procedure terminated.
         *eof
      *endif
.
.         ================================================
.         Construct Model Data-base with TESTBED Processors
.         ================================================
.
      [XQT TAB
      START <tot_nodes>
      ONLINE = [online]
      JLOC
      *show macros node_file
      *add <node_file>
.
```

```
.       ====================================================
.       Define Load/Boundary Conditions If Procedure Supplied
.       ====================================================
.
     *if <not(<ifelse([bc_procedure]; ;1;0)>)> /then
       *call [BC_PROCEDURE] ( nodes_1 = <n1> ; --
                              nodes_2 = <n2> ; --
                              es_nodes = <es_nen> ; --
                              drilling_dof = [drilling_dof] )
     *else
        *remark **********************************
        *remark BOUNDARY CONDITIONS NOT SPECIFIED
        *remark **********************************
     *endif
.
.       ================================
.       Define Shell Section Properties
.       ================================
.
     *if <not(<ifelse([section_prc]; ;1;0)>)> /then
        *call [section_prc] ( nsect = [nsect] )
     *else
.
.       ================================================
.       Define the default Material and Section Properties
.       ================================================
.
        *def/e  G = < [E]/(2.*(1+[NU])) >
.
        [XQT AUS
.
.       Build Table of Material Data
.
        TABLE(ni=16,nj=1): OMB DATA 1 1
        i = 1,2,3,4,5,6,7,8,9
        j = 1
.       E11 NU12 E22 G12 G13 G23 ALPHA1 ALPHA2 WTDEN
        [E] [NU] [E] <G> <G> <G>     0.     0. [WTDEN]
.
.       Build Laminate Data Tables
.
        TABLE(ni=3,nj=1,itype=0): LAM OMB [nsect] 1
        i =  1,2,3
        j =  1  .   matl #    layer thickness   matl angle
                       1        [thickness]          0.
.
        [XQT LAU
        ONLINE = 2
     *endif
.
.       =================
.       Generate Elements
.       =================
```

```
.
      [XQT ELD
      <ES_EXPE_CMD>
      NSECT = [nsect]
.
.        Define element nodal connectivity
.
*show macros elem_file
      *add <elem_file>
.
.        ======================================
.        Suppress DOFs not supported by elements
.        ======================================
.
      *if < [AUTO_DOF_SUP] > /then
         *call ES ( function = 'DEFINE FREEDOMS' )
      *endif
*end
```

2.4.9 REFERENCES

2.4-1 Stewart, Caroline B.: *The Computational Structural Mechanics Testbed User's Manual.* NASA TM-100644, October 1989.

2.4-2 Cook, William A.: "Body Oriented (Natural) Coordinates For Generating Three-Dimensional Meshes." *International Journal For Numerical Methods in Engineering*, 1974, Volume 8, pp. 27-43.

2.4-3 Forrest, A. R.: *On Coons and Other Methods for the Representation of Curved Surfaces.* Computer Graphics and Image Processing, 1972, Volume 1, pp. 341-359.

THIS PAGE LEFT BLANK INTENTIONALLY.

2.5 Procedure GEN_SHELL

2.5.1 GENERAL DESCRIPTION

Procedure GEN_SHELL is used to generate a class of curved shell finite element models. All surfaces are modeled as bi-linearly interpolated surfaces (*i.e.*, Coon's surfaces) in cylindrical coordinate space (see refs. 2.5-2 and 2.5-3). Interpolation in cylindrical coordinates is especially well suited for generating shells of revolution, such as cylinders, cones, annular plates, and spiraling surfaces.

2.5.2 PROCEDURE USAGE

Procedure GEN_SHELL may be used by preceding the procedure name by the `*call` directive, and following it by a list of arguments enclosed in parentheses. Procedure arguments are order-independent, and most have default values thus making them optional. The formal syntax is as follows:

```
*call GEN_SHELL ( arg1 = val1 ; arg2 = val2 ; ...)
```

where **arg1** and **arg2** represent argument names, and **val1** and **val2** represent their corresponding values. Note that semi-colons are required between arguments, and a double dash (--) preceeded by a space may be used to continue the argument list on the next line.

The allowable arguments for procedure GEN_SHELL are summarized in the following table, along with their default values (if they exist). Exceptions to this rule are noted in the following section under detailed argument descriptions.

Argument	Default Value	Meaning
ES_PROC	ES1	Generic element processor
ES_NAME	EX97	Generic element name
ES_PARS	0.0	Element research parameters
RTZ1	1,0,0	Cylindrical coordinates of point 1.
RTZ2	1,0,1	Cylindrical coordinates of point 2.
RTZ3	1,90,1	Cylindrical coordinates of point 3.
RTZ4	1,90,0	Cylindrical coordinates of point 4.
NODES_1	7	Number of nodes along edge 1
NODES_2	7	Number of nodes along edge 2 including duplicate nodes if shell closes
EDGE_WEIGHTS	1,1,1,1	Shell section property procedure
JREF	' '	Joint dof reference frame
BC_PROCEDURE	' '	Boundary condition procedure
DRILLING_DOF	<false>	Drilling dof suppression flag
AUTO_DOF_SUP	<false>	Automatic dof suppression flag
SECTION_PRC	' '	Shell section property procedure
NSECT	1	Shell section property number
The following values not used if SECTION_PRC is specified.		
E	30.E6	Young's Modulus
NU	0.3	Poisson's ratio
WTDEN	0.1	Weight Density
THICKNESS	.1	Shell thickness

2.5.3 ARGUMENT DESCRIPTIONS

2.5.3.1 AUTO_DOF_SUP

Automatic degree of freedom suppression flag (default: <false>). This option provides a convenient way of suppressing any freedoms that do not have any (or adequate) stiffness associated with them — for example, at nodes used to prescribe geometry only; or drilling freedoms in fine meshes composed of elements without normal rotational stiffness (see argument DRILLING_DOF).

2.5.3.2 BC_PROCEDURE

Name of user provided boundary condition procedure (default: ' '). The term "boundary conditions" refers both to displacement constraints and applied loading. If a boundary condition procedure is provided, the following call will be performed. The macrosymbol <es_nen> equals the number of element nodes, while the macrosymbols <n1> and <n2> equal the number of nodes actually generated along edges one and two.

```
*call [BC_PROCEDURE] ( nodes_1 = <n1> ; --
                       nodes_2 = <n2> ; --
                       es_nodes = <es_nen> ; --
```

```
                                    drilling_dof = [drilling_dof] )
```

No action is taken if a boundary condition procedure name is not provided.

2.5.3.3 DRILLING_DOF

Drilling degree of freedom flag (default: <false>). Drilling freedoms are defined as rotations normal to the surface of the shell. Leaving this flag set to <false> forces all drilling freedoms in the model to be suppressed. Turning it on forces all drilling freedoms to be active — unless they are automatically suppressed by use of the AUTO_DOF_SUP argument. Note that while many shell elements do not have any rotational stiffness associated with their own surface-normal directions (at nodes), when shell elements are assembled as facets approximating an arbitrary shell surface, there is usually some misalignment between the element normal and the actual shell normal. This is especially true of "flat" (*e.g.*, 4-node) elements. Hence, *some* rotational stiffness about the *shell* normal is usually present in the model. (A clear exception to this is a flat plate, where element and shell normals are identical.) For a cylindrical shell, the misalignment diminishes only as the number of elements is increased. Most shell elements in the Testbed have their own misalignment tolerance parameter, which determines when the AUTO_DOF_SUP argument will automatically suppress the drilling freedom. Note that for elements which *have* drilling stiffness, the DRILLING_DOF argument should be set to <true> regardless of how AUTO_DOF_SUP is set.

2.5.3.4 E

Young's modulus (default: 30.E6). This argument is ignored if SECTION_PRC parameter is specified. See the description for SECTION_PRC for more detail.

2.5.3.5 EDGE_WEIGHTS

Node placement can be weighted along each surface edge according to the EDGE_WEIGHTS parameter. The input format requires a list of four edge-node placement weightings representing the node weighting for edge1, edge2, edge3, and edge4 (default: 1.,1.,1.,1.).

The weighting value for a given edge represents the length of the last element divided by the length of the first element along that edge. The edge orientation arrows in figure 2.5-1 point from the first element to the last element along each edge. In the case of 9-node quad elements, the midside and center nodes are positioned at the appropriate locations based on the elements natural coordinate system.

The procedure interprets negative weight values to mean the positive reciprocal. For example, a value of -5.0 is identical to a value of 0.2.

2.5.3.6 ES_NAME

Element name (default: EX97). This argument is the name of the specific shell-element type you wish to select, within the element processor defined by argument ES_PROC. The default shell-element type, EX97, is a 9-node quadrilateral element implemented in processor ES1, and described reference 2.5-1.

2.5.3.7 ES_PARS

Element research parameters (default: 0., ...). This argument is an optional list of element-dependent parameters that some elements provide, primarily when the element is still undergoing research and refinement.

2.5.3.8 ES_PROC

Element processor (default: ES1) This argument is the name of the structural element (ES) processor that contains the shell element type you wish to employ in the model. The default shell-element, processor ES1, is described in The Computational Structural Mechanics Testbed User's Manual.

2.5.3.9 JREF

Joint degree of freedom (dof) reference frame (default: -1 for global cylindrical). The user may provide any alternate frame which has been created prior to calling this procedure. A negative value causes the frame to be interpreted as a cylindrical reference frame.

2.5.3.10 NODES_1

Number of nodes on edge 1 including the nodes at the surface corners (default: 7). This argument is also the number of nodes on edge 3. This number should be consistent with the element type selected. For example, NODES_1 can be any number greater than 1 for 4-node quadrilateral elements, whereas it must be an odd number greater than 1 for 9-node quadrilateral elements.

2.5.3.11 NODES_2

Number of nodes on edge 2 including the nodes at the surface corners (default: 7). This argument is also the number of nodes on edge 4. This number should be consistent with the element type selected. For example, NODES_2 can be any number greater than 1 for 4-node quadrilateral elements, whereas it must be an odd number greater than 1 for 9-node quadrilateral elements.

2.5.3.12 NSECT

Shell section property number (default: 1). The NSECT value is required when defining the element using the processor ELD. See the description of SECTION_PRC for more detail.

2.5.3.13 NU

Poisson's ratio (default: 0.3). This argument is ignored if the SECTION_PRC input parameter is specified. See the description of SECTION_PRC for more detail.

2.5.3.14 RTZ1

The cylindrical coordinates (r, θ, z) which define corner number 1 of the model surface (θ in degrees). The form of the input is three real values, each separated by a comma (default: 1.,0.,0.). The surface is defined by four edges which are defined as a linear interpolation in cylindrical coordinates of four endpoints, or "corner" points.

2.5.3.15 RTZ2

The cylindrical coordinates (r, θ, z) defining the corner number 2 of the model surface (default: 1.,0.,1.).

2.5.3.16 RTZ3

The cylindrical coordinates (r, θ, z) defining the corner number 3 of the model surface (default: 1.,90.,1.).

2.5.3.17 RTZ4

The cylindrical coordinates (r, θ, z) defining the corner number 4 of the model surface (default: 1.,90.,0.).

2.5.3.18 SECTION_PRC

Name of a user supplied procedure to define the plate section properties (default = ' '). If a section properties procedure is provided, the following call will be performed.

```
*call [section_prc] ( nsect = [nsect] )
```

The effect of the default is to allow the procedure to generate an isotropic material section based on the input parameters **E**, **NU**, **WTDEN**, and **THICKNESS**. The section number is defined by the input parameter **NSECT**. If the call parameter **SECTION_PRC** is defined by the user, then call parameters **E**, **NU**, **WTDEN**, and **THICKNESS** are ignored by procedure **GEN_SHELL**.

2.5.3.19 THICKNESS

Thickness of the shell wall (default = 1.0). This argument is ignored if **SECTION_PRC** parameter is specified. See the description for **SECTION_PRC** for more detail.

2.5.3.20 WTDEN

Weight density expressed in lb/in.^{3} (default: `0.1` lb/in.^{3}). This argument is ignored if the **SECTION_PRC** input parameter is specified. Processor LAU will convert the weight density to mass density using the gravitational acceleration constant 386.4 in/sec^2.

2.5.4 USAGE GUIDELINES AND EXAMPLES

Procedure **GEN_SHELL** may be invoked using the *call directive. Procedure arguments may be changed from their default values by including any or all of the arguments and their new values in the procedure call. A space or blank is required between the end of the procedure name and the left parenthesis. If the default values of the procedure arguments are to be used, then only the procedure name is required.

```
*procedure GEN_SHELL ( ES_PROC = ES1 ; ES_NAME = EX97 ; ES_PARS = 0.0 ; --
                RTZ1          = 1.,0.,0. ; --
                RTZ2          = 1.,0.,1. ; --
```

```
                RTZ3            = 1.,90.,1. ; --
                RTZ4            = 1.,90.,0. ; --
                NODES_1         = 7 ; --
                NODES_2         = 7 ; --
                EDGE_WEIGHTS     = 1.,1.,1.,1. ; --
                BC_PROCEDURE = ' ' ; -- . Boundary condition procedure
                DRILLING_DOF   = <true> ; --
                AUTO_DOF_SUP   = <true> ; --
                SECTION_PRC = ' ' ; --   . Shell section property procedure
                NSECT = 1 ; --             . Shell section property ID
--  .             The following values not used if SECTION_PRC specified
                    E = 30.E6 ; --       . Young's Modulus
                    NU = .3 ; --         . Poissons ratio
                    THICKNESS = .1 --    . Shell thickness
              )
```

2.5.4.1 Mesh Generation

The method of surface generation used by procedure GEN_SHELL is described in the section. Termology depicted on figure 2.5-1 provides a visual interpretation of the parameters used to generate a curved surface.

To the define the shell surface, the user definess four coordinate positions in a cylindrical reference frame. These four positions represent four corners of a four sided region. The sides of the region (which will also be referred to as edges) are defined by linearly interpolating between the coordinate values of the corner points. The surface of the region is defined as a bi-linear interpolation of the four sides, also known as a Coon's surface (see refs. 2.5-2 and 2.5-3). It must be remembered that since interpolations are performed in cylindrical coordinates, the surface and its edges will not generally be flat or straight, but rather curved.

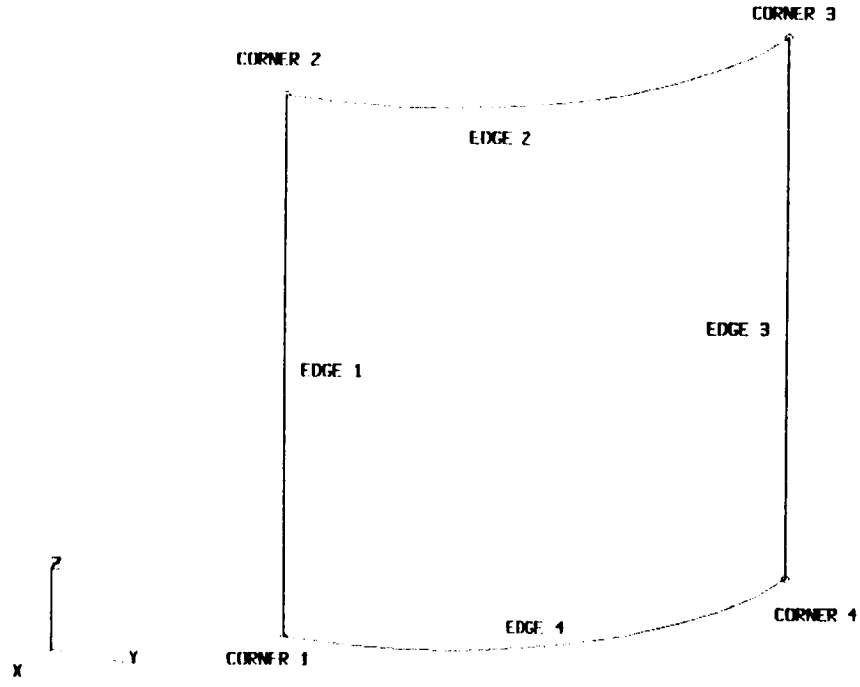


Figure 2.5-1 Generic Shell Surface Topology.

The connectivity of the surface is defined in figure 2.5-1. The user provides the coordinates for corners 1 to 4. Edge 1 of the region starts at corner 1 and ends at corner 2. The remaining edges are defined in a similar manner. The arrows indicate the orientation of the edges and the direction of increasing node numbers.

The topology of the resulting finite element grid is depicted in Figure 2.5-2. Nodes are created first along edge 1, then in successive lines terminating along edge 3. The user defines the number of nodes along edges 1 and 2, which also defines the number of nodes for edges 3 and 4. The relative position of the nodes along each edge may be controlled using the edge weighting parameter `EDGE_WEIGHTS`.

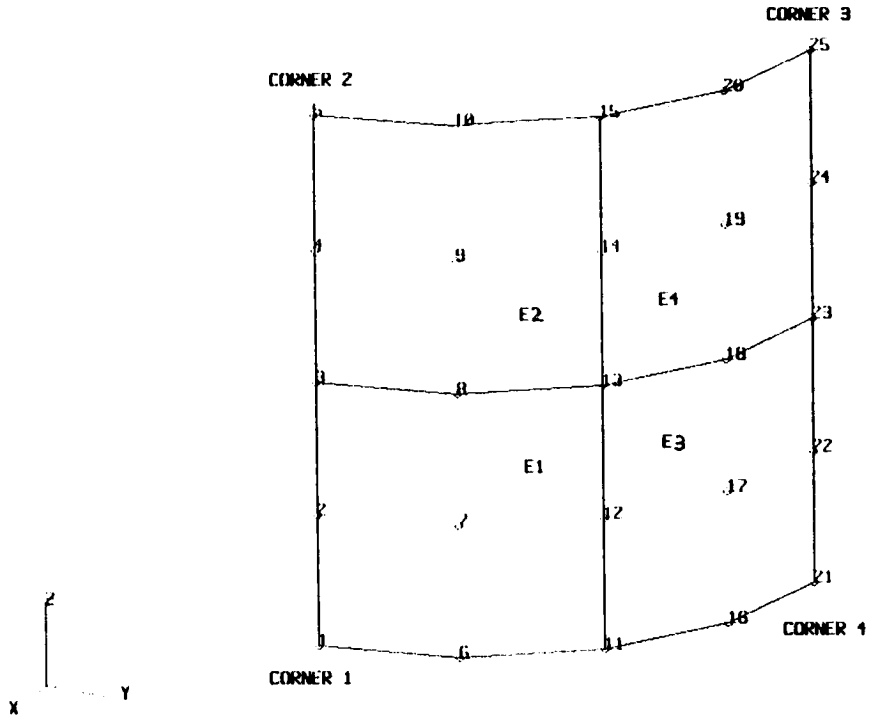


Figure 2.5-2 Node and Element Topology.

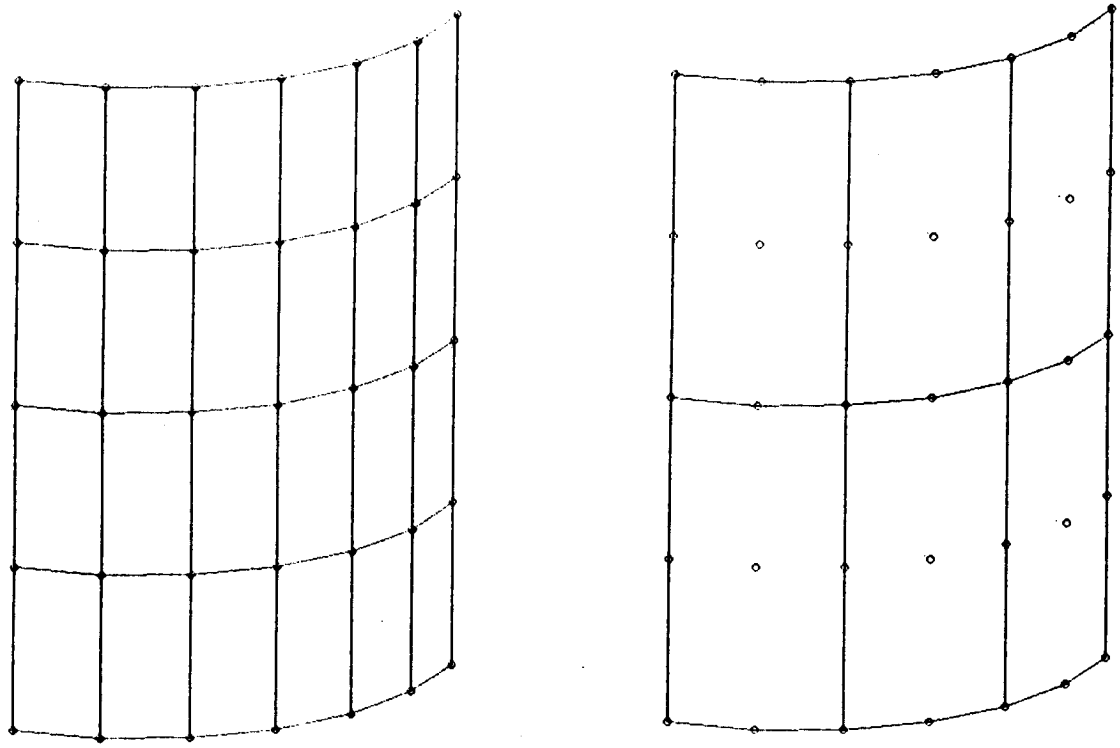
Recognizing that the surfaces generated by procedure **GEN_SHELL** are four sided surfaces in cylindrical coordinates, it will be shown, by example, how to generate segments of cylinders, cones, annular plates, spirals, and more general surfaces. For applications which require a complete axisymmetric surface, such as a 360 degree cylinder, the procedure has the capability of joining the resulting finite element mesh where two sides of the region are coincident. Closure occurs automatically but is checked only along edges 1 and 3. Closure will not occur between edges 2 and 4. Note also that the user must request the number of nodes along edge 2 as though the surface were not closed. This requirement is to say that the user should not presume closure will occur.

2.5.4.2 Cylindrical Shell Sections

In this section there are presented examples of how procedure **GEN_SHELL** may be used to create various shell segments on a right circular cylindrical surface.

In the following example, procedure **GEN_SHELL** is used to generate a cylindrical segment with radius of 5 inches and length of 10 inches. Only 90 degrees of the cylinder is generated (see figure 2.5-3).

```
*call GEN_SHELL ( es_proc = 'ES1' ; es_name = 'EX47' ; --
        rtz1          = 5.,0.,0. ; --
        rtz2          = 5.,0.,10. ; --
        rtz3          = 5.,90.,10. ; --
        rtz4          = 5.,90.,0. ; --
        nodes_1       = 5 ; --
        nodes_2       = 7 )
```



(a) 4-node elements (b) 9-node elements

Figure 2.5-3 90 Degree Cylindrical Segment

This example creates a complete 360 degree cylindrical shell using 9-node quadrilateral elements (see figure 2.5-4). Note that midside nodes are not shown. The input specifies elements at the end of edges 1 and 3 to be 5 times longer that elements at the beginning of the same edges. Closure of the cylinder is obtained by defining corner points 1 and 4, and corner points 2 and 3, to be coincident in the r and z directions, with a difference in θ of 360 degrees.

```
*call GEN_SHELL ( es_proc = 'ES1' ; es_name = 'EX97' ; --
        rtz1          = 5.,0.,0. ; --
        rtz2          = 5.,0.,10. ; --
        rtz3          = 5.,360.,10. ; --
        rtz4          = 5.,360.,0. ; --
        nodes_1       = 11 ; --
        nodes_2       = 25 ; --
        edge_weights  = 5.,1.,5.,1. )
```

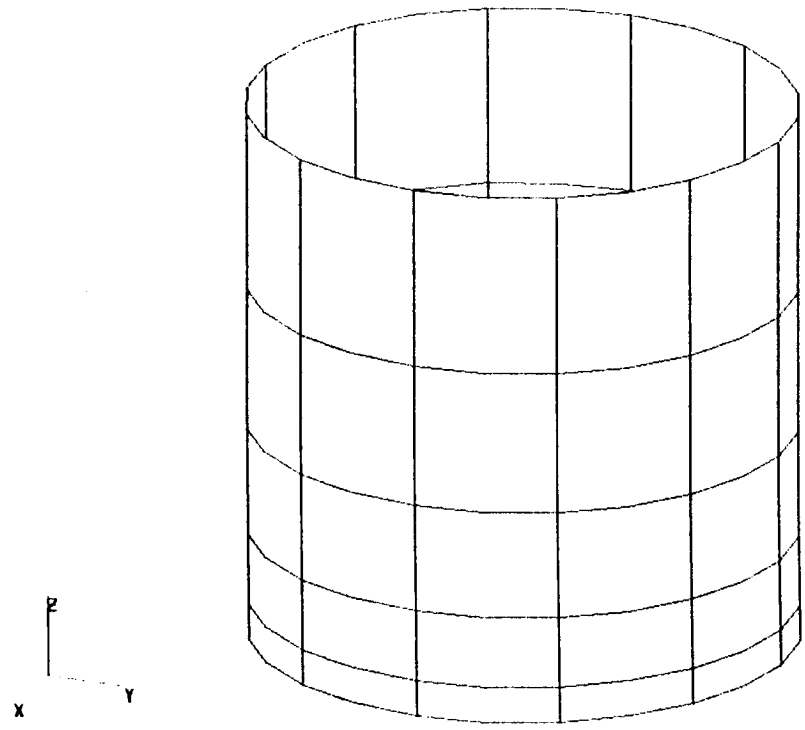


Figure 2.5-4 Right Circular Cylinder - 9-node Elements

By simply increasing the z coordinate values for corner points 3 and 4, the cylinder wall can be made to spiral about the z axis. The following example illustrates this technique. The resulting spiral is shown in figure 2.5-5. This configuration may be used to model a spring.

```
*call GEN_SHELL ( es_proc = 'ES1' ; es_name = 'EX47' ; --
        rtz1          = 5.,0.,0. ; --
        rtz2          = 5.,0.,2. ; --
        rtz3          = 5.,720.,12. ; --
        rtz4          = 5.,720.,10. ; --
        nodes_1       = 5 ; --
        nodes_2       = 37 )
```

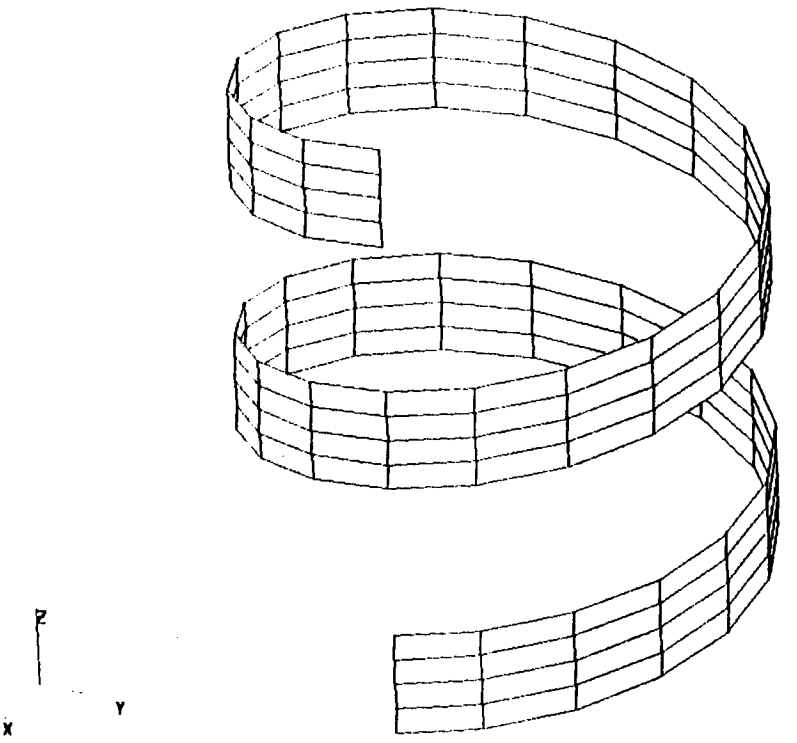


Figure 2.5-5 Spiraling Cylinder Wall

The following example produced the cylinder of skewed elements shown in figure 2.5-6.

```
*call GEN_SHELL ( es_proc = 'ES1' ; es_name = 'EX47' ; --
        rtz1          = 5.,0.,0. ; --
        rtz2          = 5.,90.,10. ; --
        rtz3          = 5.,450.,10. ; --
        rtz4          = 5.,360.,0. ; --
        nodes_1       = 7 ; --
        nodes_2       = 25 )
```

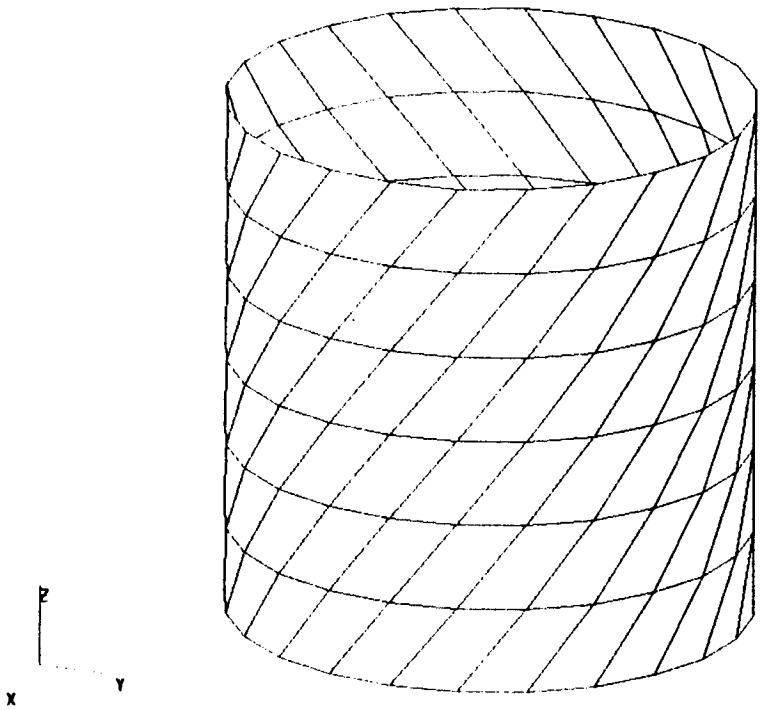


Figure 2.5-6 Cylinder With Skewed Elements

2.5.4.3 Conical Shell Sections

In this section, examples of how procedure GEN_SHELL may be used to create shell segments on a conical surface are presented.

This example creates a complete 360 degree conical shell using 9-node quadrilateral elements (see figure 2.5-7). Elements are defined at the top of the cones to be 1/5 as long axially as those at the base. Closure of the cone is obtained by defining corner points 1 and 4, and corner points 2 and 3, to be coincident in r and z directions, with a difference in θ of 360 degrees.

```
*call GEN_SHELL ( es_proc = 'ES1' ; es_name = 'EX97' ; --
        rtz1          = 5.,0.,0. ; --
        rtz2          = 1.,0.,10. ; --
        rtz3          = 1.,360.,10. ; --
        rtz4          = 5.,360.,0. ; --
        nodes_1       = 9 ; --
        nodes_2       = 25 ; --
        edge_weights  = -4.,1.,-4.,1. )
```

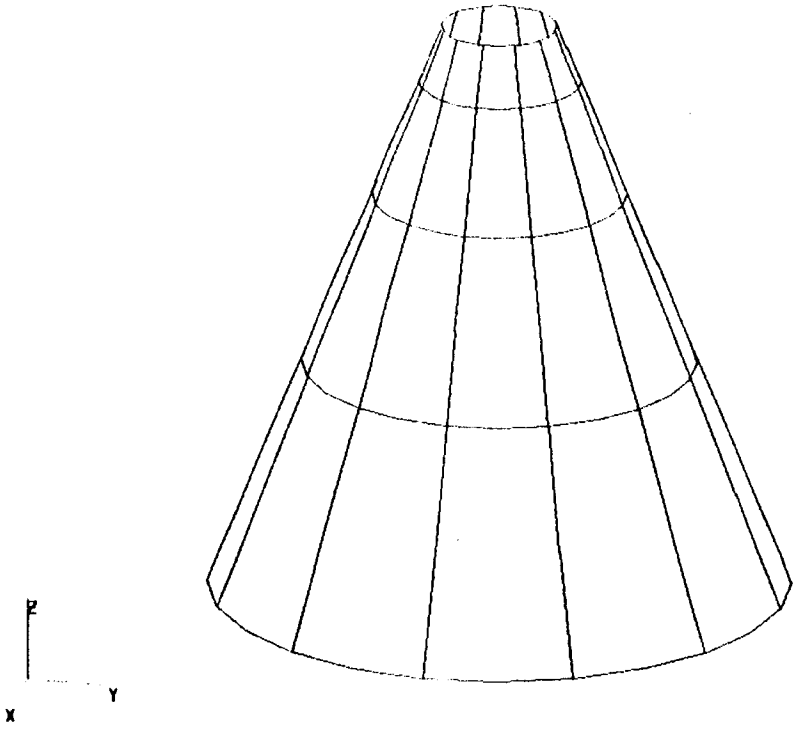


Figure 2.5-7 Conical Shell - 9-Node Elements

In the following example, procedure GEN_SHELL is used to generate an unusual shell which lies on the conical surface of the previous example. The shell spans 360 degrees at the top but only spans 180 degrees at the base (see figure 2.5-8).

```
*call GEN_SHELL ( es_proc = 'ES1' ; es_name = 'EX47' ; --
        rtz1          = 5.,90.,0. ; --
        rtz2          = 1.,0.,10. ; --
        rtz3          = 1.,360.,10. ; --
        rtz4          = 5.,270.,0. ; --
        nodes_1       = 11 ; --
        nodes_2       = 21 ; --
        edge_weights   = -3.,1.,-3.,1. )
```

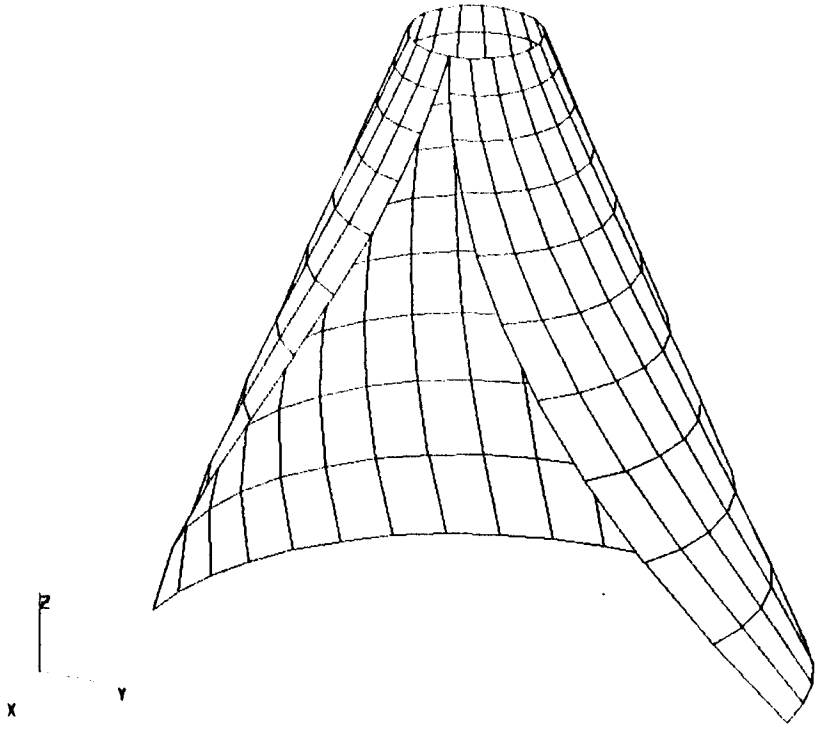


Figure 2.5-8 Unusual Conical Shell

2.5.4.4 Annular Plates

In this section examples of how procedure **GEN_SHELL** may be used to create annular shell segments are presented.

This example creates a 90 degree annular shell segment using 9-node quadrilateral elements (see figure 2.5-9). Note that the midside nodes are not shown. The plate has an inner radius of 1 inch and the outer radius of 5 inches. Element size weighting is also demonstrated.

```
*call GEN_SHELL ( es_proc = 'ES1' ; es_name = 'EX97' ; --
        rtz1          = 1.,0.,0. ; --
        rtz2          = 5.,0.,0. ; --
        rtz3          = 5.,90.,0. ; --
        rtz4          = 1.,90.,0. ; --
        nodes_1       = 11 ; --
        nodes_2       = 11 ; --
        edge_weights  = 6.,1.,6.,1. )
```

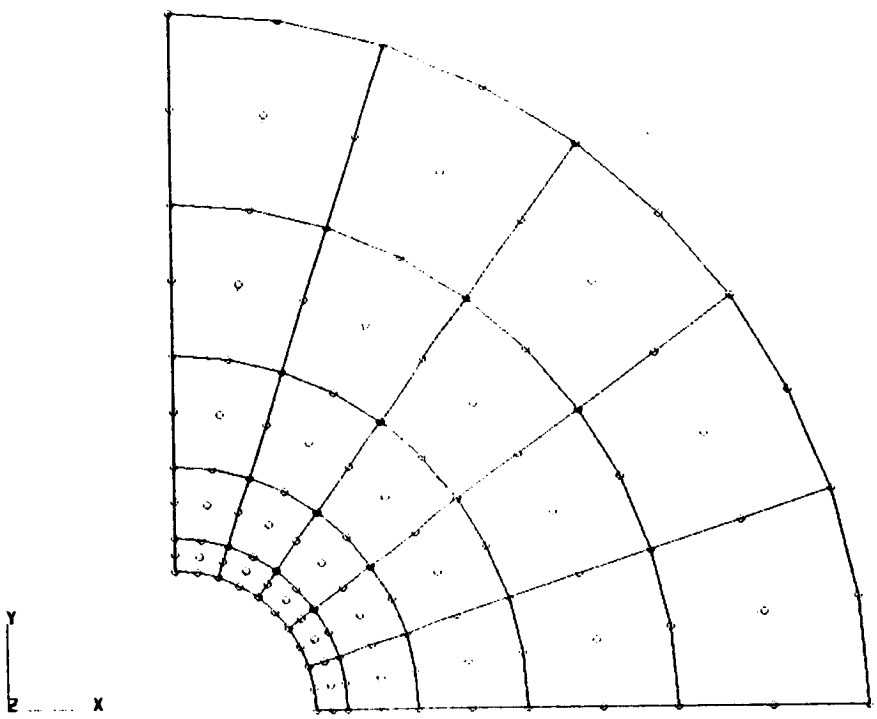


Figure 2.5-9 Flat Annular Shell

By simply increasing the z coordinate values for corner points 3 and 4, the annular surface can be made to spiral about the z axis. The following example illustrates this technique. The resulting spiral is shown in figure 2.5-10.

```
*call GEN_SHELL ( es_proc = 'ES1' ; es_name = 'EX47' ; --
        rtz1          = 3.,0.,0. ; --
        rtz2          = 5.,0.,0. ; --
        rtz3          = 5.,720.,10. ; --
        rtz4          = 3.,720.,10. ; --
        nodes_1       = 5 ; --
        nodes_2       = 33 )
```

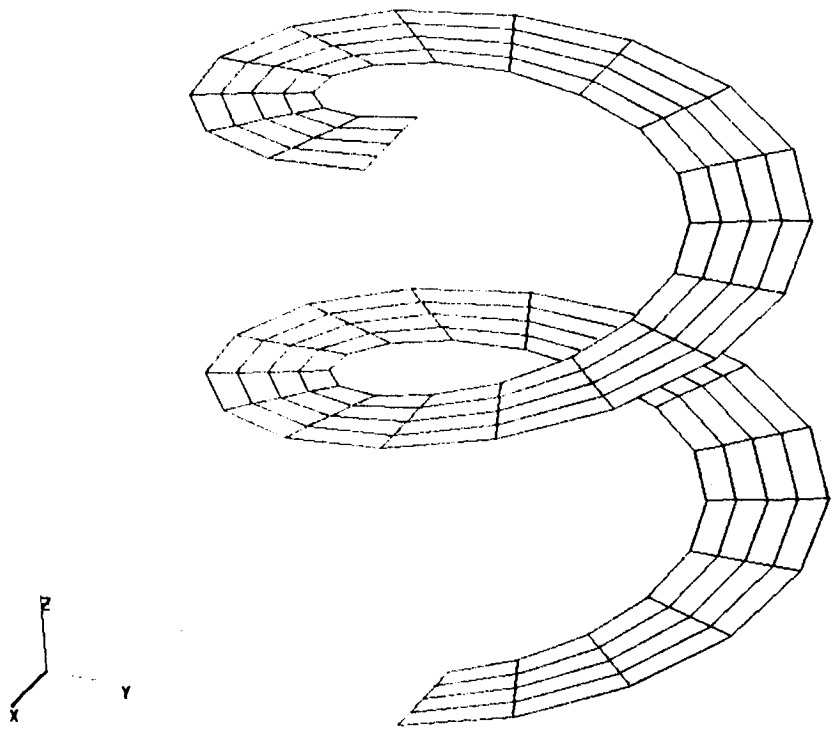


Figure 2.5-10 Spiraling Shell

2.5.4.5 Exotic Shell Sections

In this section examples of how procedure GEN_SHELL may be used to create unusual shell segments are presented.

This example creates a spiraling surface which changes from a flat to a vertical shell while increasing in z direction and decreasing in radius (see figure 2.5-11).

```
*call GEN_SHELL ( es_proc = 'ES1' ; es_name = 'EX97' ; --
        rtz1          = 3.,0.,0. ; --
        rtz2          = 5.,0.,0. ; --
        rtz3          = 1.,720.,12. ; --
        rtz4          = 1.,720.,10. ; --
        nodes_1       = 5 ; --
        nodes_2       = 45 )
```

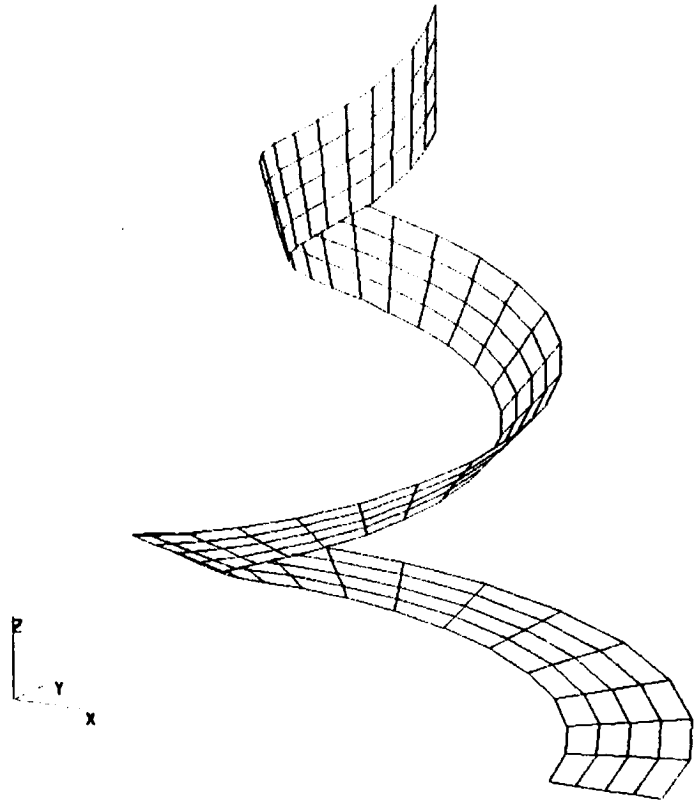


Figure 2.5-11 Exotic Spiral Shell

In the following example, procedure **GEN_SHELL** is used to generate a vertical coiled shell (see figure 2.5-12).

```
*call GEN_SHELL ( es_proc = 'ES1' ; es_name = 'EX47' ; --
        rtz1          = 0.,0.,0. ; --
        rtz2          = 0.,0.,1. ; --
        rtz3          = 3.,1080.,1. ; --
        rtz4          = 3.,1080.,0. ; --
        nodes_1       = 5 ; --
        nodes_2       = 85 )
```

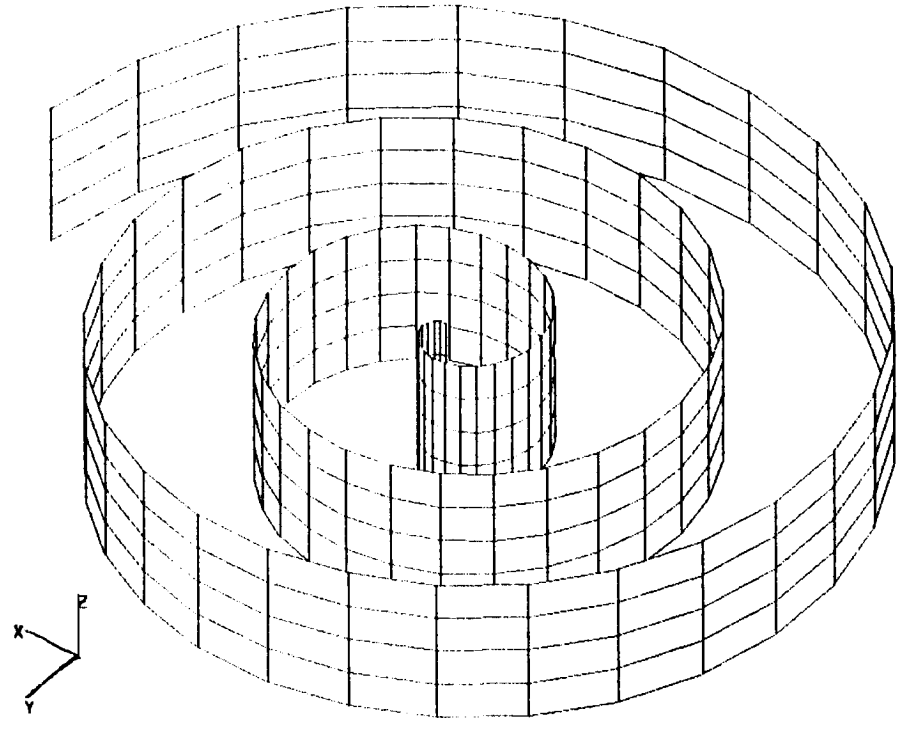


Figure 2.5-12 Vertical Coiled Shell

This example uses procedure **GEN_SHELL** to generate a flat coiled shell (see figure 2.5-13).

```
*call GEN_SHELL ( es_proc = 'ES1' ; es_name = 'EX47' ; --
        rtz1          = 0.,0.,0. ; --
        rtz2          = 1.,0.,0. ; --
        rtz3          = 9.,720.,0. ; --
        rtz4          = 5.5,720.,0. ; --
        nodes_1       = 5 ; --
        nodes_2       = 51 )
```

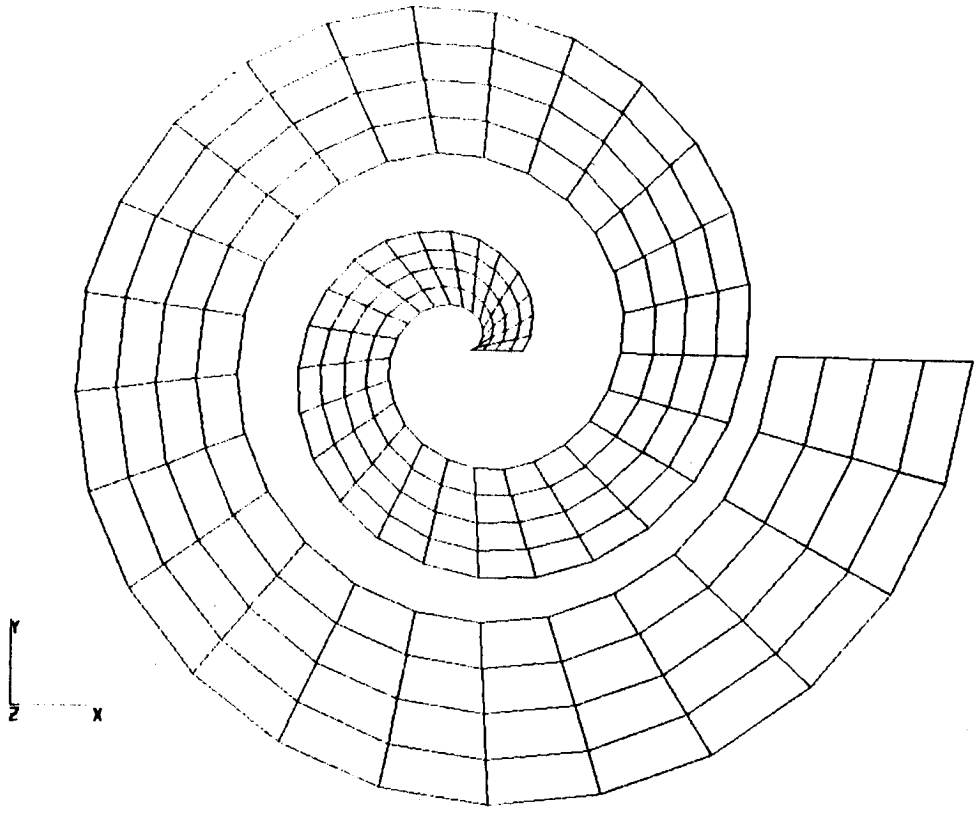


Figure 2.5-13 Flat Coiled Shell

Finally, an arbitrary shell is created to demonstrate the generality of the procedure GEN_SHELL (see figure 2.5-14).

```
*call GEN_SHELL ( es_proc = 'ES1' ; es_name = 'EX47' ; --
        rtz1          = -2.,-25.,0. ; --
        rtz2          = 7.,190.,-3. ; --
        rtz3          = -2.,230.,7. ; --
        rtz4          = 6.,0.,10. ; --
        nodes_1       = 25 ; --
        nodes_2       = 25 )
```

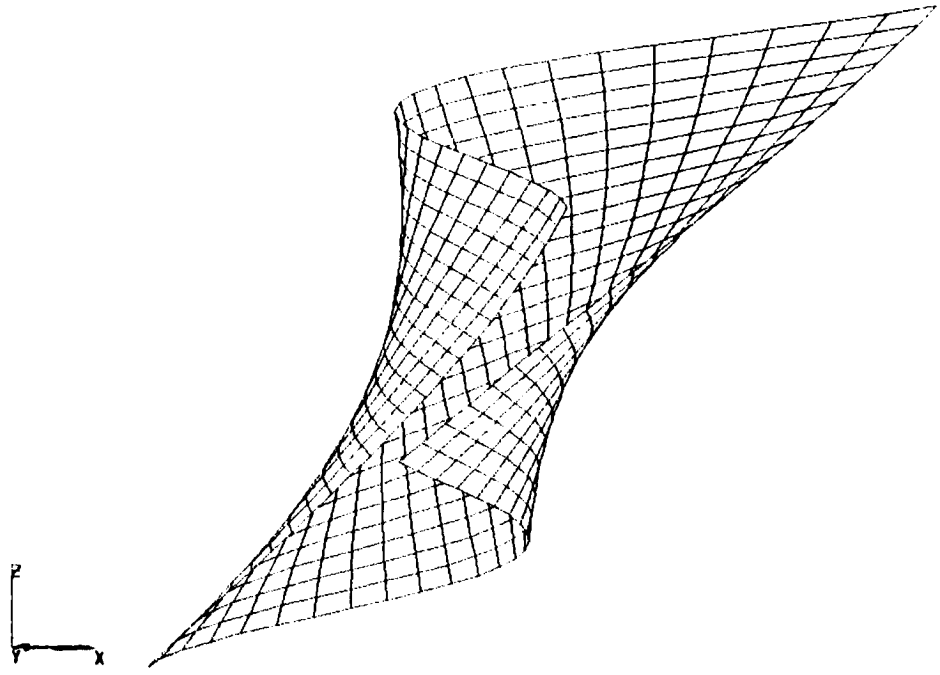


Figure 2.5-14 Arbitrary Shell

2.5.5 LIMITATIONS

As previously stated, to obtain closed axisymmetric surface models, the user must define edges 1 and 3 to be the coincident edges.

2.5.6 ERROR MESSAGES AND WARNINGS

None.

2.5.7 PROCEDURE FLOWCHART

GEN_SHELL	(Shell generation procedure)
[BC_PROCEDURE]	(user supplied boundary conditions/loads procedure)
[SECTION_PRC]	(user supplied section property generation procedure)
CYL_NODES	(surface node generation procedure)
CYL_ELT_CONN	(element connectivity definition procedure)

2.5.8 PROCEDURE LISTING

```
*procedure GEN_SHELL ( es_proc = es1 ; es_name = ex97 ; es_pars = 0. ; --
               rtz1          = 1.,0.,0. ; --
               rtz2          = 1.,0.,1. ; --
               rtz3          = 1.,90.,1. ; --
               rtz4          = 1.,90.,0. ; --
               nodes_1       = 7 ; --
               nodes_2       = 7 ; --
               edge_weights  = 1.,1.,1.,1. ; --
               jref          = -1 ; -- . Joint dof reference frame
               bc_procedure = ' ' ; -- . Boundary condition procedure
               online        = 0 ; -- . supress nodes/elts output
               drilling_dof  = <true> ; --
               auto_dof_sup  = <true> ; --
               section_prc = ' ' ; --  . Shell section property procedure
             The following values used only if section_prc not specified
                   nsect = 1 ; --       . Shell section property ID
                   E = 30.E6 ; --       . Young's Modulus
                   NU = .3 ; --         . Poisons ratio
                   WTDEN = .1 ; --      . Weight Density
                   thickness = .1 --    . Shell thickness
               )
.
.   ========================================================================
.   Model Definition Procedure for GENeric Shell in cylindrical coordinates
.   ========================================================================
.
.        A general purpose clip procedure to create the finite element
```

```
.          mesh for a partial or complete cylindrical shell using 4 or
.          9 noded  quadrilateral elements.
.
.         Note that when generating a 360 degree closed shell,
.          the caller should specify the number of circumferential nodes as
.          if the cylindrical shell were not closed, or in other words,
.          the line of nodes where closure occurs should be counted twice.
.          There will only be one set of nodes actually create where
.          closure occurs.
.
.         There is no verification performed to detect overlapping or
.          otherwise improbable element generation.  This should be done
.          prior to calling this utility.
.
.         If a procedure to generate shell section properties is not provided,
.          the isotropic section described by E, NU, WTDEN, and THICKNESS will
.          be automatically generated.  (see [section_prc] parameter)
.
.         A boundary condition procedure should be provided but is
.          optional.  If not provided, no boundary conditions will be
.          defined.  (see [bc_procedure] parameter)
.
*remark ************************************
*remark GEN_SHELL MODEL GENERATION PROCEDURE
*remark ************************************
.
     *def/i n1 = [nodes_1]
     *def/i n2 = [nodes_2]
     *def/e rtz1[1:3] = [rtz1]
     *def/e rtz2[1:3] = [rtz2]
     *def/e rtz3[1:3] = [rtz3]
     *def/e rtz4[1:3] = [rtz4]
     *def/e w[1:4] = [edge_weights]
     *def/e rc[1:4] = <rtz1[1]>,<rtz2[1]>,<rtz3[1]>,<rtz4[1]>
     *def/e tc[1:4] = <rtz1[2]>,<rtz2[2]>,<rtz3[2]>,<rtz4[2]>
     *def/e zc[1:4] = <rtz1[3]>,<rtz2[3]>,<rtz3[3]>,<rtz4[3]>
.
.         ================================================================
.         Register Element and Define Macros: ES_NEN, ES_NIP, ES_NSTR, etc.
.         ================================================================
     *call ES ( function = 'DEFINE ELEMENTS' ; es_proc = [es_proc]; --
                es_name =  [es_name]       ; es_pars = [es_pars] )
.
.         ================================================================
.         Define nodal coordinates and element connectivities into
.          separate formatted files.  These files are gauranteed to have
.          unique names that are not currently in use in the current
.          directory.
.         ================================================================
.
     *def/i chk_closure = 1
     [xqt mesh
     *if <mesh_err> /then
```

```
      *remark  Error occurred during MESH processor execution.
      *remark  GEN_SHELL procedure terminated.
      *eof
   *endif

.
.      ================================================
.      Construct Model Data-base with TESTBED Processors
.      ================================================
.
   [XQT TAB
   START <tot_nodes>
   ONLINE = [online]
   JLOC
   FORMAT = 2                  . use cylindrical coordinate system
   *show macros node_file
   *add <node_file>
.
.     =====================
.     Define DOF Directions
.     =====================
.
      JREF                     .   Use local cylindrical basis vectors
                               .   for nodal DOFS:
                               .   u,v,w = radial, circumfer., axial
      NREF = [jref]
      1  <tot_nodes>           .   same convention for all nodes
.
.      ======================================================
.      Define Load/Boundary Conditions If Procedure Supplied
.      ======================================================
.
   *if <not(<ifelse([bc_procedure]; ;1;0)>)> /then
     *call [BC_PROCEDURE] ( nodes_1 = <n1> ; --
                            nodes_2 = <n2> ; --
                            es_nodes = <es_nen> ; --
                            drilling_dof = [drilling_dof] )
   *else
      *remark **********************************
      *remark BOUNDARY CONDITIONS NOT SPECIFIED
      *remark **********************************
   *endif
.
.      ==============================
.      Define Shell Section Properties
.      ==============================
.
   *if <not(<ifelse([section_prc]; ;1;0)>)> /then
      *call [section_prc] ( nsect = [nsect] )
   *else
.
.      ================================================
.      Define the default Material and Section Properties
.      ================================================
```

```
.
        *def/e  G = < [E]/(2.*(1+[NU])) >
.
         [XQT AUS
.
.        Build Table of Material Data
.
         TABLE(ni=16,nj=1): OMB DATA 1 1
         i = 1,2,3,4,5,6,7,8,9
         j = 1
.        E11 NU12 E22 G12 G13 G23 ALPHA1 ALPHA2 WTDEN
         [E] [NU] [E] <G> <G> <G>     0.      0. [WTDEN]
.
.        Build Laminate Data Tables
.
         TABLE(ni=3,nj=1,itype=0): LAM OMB [nsect] 1
         i =  1,2,3
         j =  1  .   matl #    layer thickness   matl angle
                        1        [thickness]          0.
.
        [XQT LAU
        ONLINE = 2
      *endif
.
.         ==================
.         Generate Elements
.         ==================
.
      [XQT ELD
      <ES_EXPE_CMD>
      NSECT = [nsect]
.
.        Define element nodal connectivity
.
*show macros elem_file
      *add <elem_file>
.
.        ======================================
.        Suppress DOFs not supported by elements
.        ======================================
.
      *if < [AUTO_DOF_SUP] > /then
         *call ES ( function = 'DEFINE FREEDOMS' )
      *endif
*end
```

2.5.9 REFERENCES

2.5-1 Stewart, Caroline B.: *The Computational Structural Mechanics Testbed User's Manual.* NASA TM-100644, October 1989.

2.5-2 Cook, William A.: "Body Oriented (Natural) Coordinates For Generating Three-Dimensional Meshes." *International Journal For Numerical Methods in Engineering*, 1974, Volume 8, pp. 27-43.

2.5-3 Forrest, A. R.: *On Coons and Other Methods for the Representation of Curved Surfaces.* Computer Graphics and Image Processing, 1972, Volume 1, pp. 341-359.

THIS PAGE LEFT BLANK INTENTIONALLY.

3.0 Solution Procedures

The procedures documented in this chapter are for specific analysis tasks. These procedures provide examples of how to perform common structural analysis tasks (*e.g.*, static solution, eigensolution) using the command language and processors available in the CSM Testbed Software System.

A summary of the procedures found in this chapter is provided in Table 3.0-1.

Table 3.0-1 Solution Procedures

Procedure Name	*Analysis Description*
L_DYNAMIC_0	Linear transient dynamic analysis using modal analysis
L_DYNAMIC_1	Linear transient dynamic analysis using Newmark algorithm
L_STABIL_1	Linear stability (buckling eigenvalue) analysis with prescribed prestress
L_STABIL_2	Linear stability (buckling eigenvalue) analysis with linearly-computed prestress
L_STATIC	Linear static analysis
L_VIBRAT_0	Linear vibration (eigenvalue) analysis about unstressed state
L_VIBRAT_1	Linear vibration (eigenvalue) analysis about a prescribed prestressed state
L_VIBRAT_2	Linear vibration (eigenvalue) analysis about a linearly-computed prestressed state
NL_STATIC_1	Nonlinear static analysis; modified Newton iteration with arc-length control
NL_STATIC_2	Advanced Riks method
NL_DYNAMIC_1	Nonlinear Dynamic Analysis

THIS PAGE LEFT BLANK INTENTIONALLY.

3.1 Processor L_DYNAMIC_0

THIS SECTION UNDER PREPARATION

THIS PAGE LEFT BLANK INTENTIONALLY.

3.2 Procedure L_DYNAMIC_1

3.2.1 GENERAL DESCRIPTION

Procedure **L_DYNAMIC_1** performs a linear transient dynamic analysis using either the Newmark-β implicit direct time integration procedure outlined in reference 3.2-1. When Procedure **L_DYNAMIC_1** is called, a transient response calculation by direct integration of the system equations with a fixed time step is performed. Procedure **L_DYNAMIC** calls Procedure **NEWMARK** which implements the well-known Newmark integration method for second order, coupled systems. Parameters such as the names of the system stiffness and mass matrices, the time step, and the total number of time steps in the analysis are formal arguments to Procedure **L_DYNAMIC_1**. In Procedure **NEWMARK**, extensive use is made of the CLAMP macro expression capability for calculating integration constants and controlling the algorithm. The initial acceleration at time t = 0 is calculated from the given initial displacement and velocity vectors. This is done by using processor AUS to set up the equations of motion at t=0, and processors INV and SSOL to solve for the acceleration. At each subsequent time step, processor AUS is used to set up the recursion relations, and processor SSOL is used to solve for the displacement vector at the next time step. Then velocity and acceleration vectors are calculated and selectively printed.

3.2.2 THEORY

3.2.2.1 Introduction

The equations of motion for an undamped, linear elastic structure at time $t + \Delta t$ are

$$\mathbf{M}\ddot{\mathbf{u}}_{t+\Delta t} + \mathbf{K}\mathbf{u}_{t+\Delta t} = \mathbf{P}_{t+\Delta t} \qquad (3.2-1)$$

where

$\mathbf{M}$ is the mass matrix

$\mathbf{K}$ is the linear elastic stiffness matrix

$\mathbf{P}_{t+\Delta t}$ is the load vector at time $t + \Delta t$

$\mathbf{u}_{t+\Delta t}$ is the displacement vector at time $t + \Delta t$

$\ddot{\mathbf{u}}_{t+\Delta t}$ is the acceleration vector at time $t + \Delta t$

3.2.2.2 Newmark-β Method

The Newmark-β method is an implicit direct time integration procedure that is based on the following assumptions:

$$\dot{\mathbf{u}}_{t+\Delta t} = \dot{\mathbf{u}}_t + [(1-\gamma)\ddot{\mathbf{u}}_t + \gamma\ddot{\mathbf{u}}_{t+\Delta t}]\Delta t \qquad (3.2-2)$$

$$\mathbf{u}_{t+\Delta t} = \mathbf{u}_t + \Delta t\dot{\mathbf{u}}_t + [(\frac{1}{2}-\beta)\ddot{\mathbf{u}}_t + \beta\ddot{\mathbf{u}}_{t+\Delta t}](\Delta t)^2 \qquad (3.2-3)$$

where

$\dot{\mathbf{u}}_{t+\Delta t}$ is the velocity vector at time $t + \Delta t$

Δt is time step size

The parameters γ and β determine integration accuracy and stability. When $\gamma = \frac{1}{2}$ and $\beta = \frac{1}{6}$, the linear acceleration method is obtained (*i.e.*, the acceleration is assumed to vary linearly over a time step). When $\gamma = \frac{1}{2}$ and $\beta = \frac{1}{4}$, Newmark's original, constant-average-acceleration method (also called the trapezoidal rule) is obtained.

3.2.3 ALGORITHM

The Procedure **L_DYNAMIC_1** closely follows the computational procedure presented in reference 3.2-1. Briefly, an outline of the procedure is as follows:

1. Select time step size, Δt, and parameters γ and β. Calculate integration constants:

$$\gamma \geq \frac{1}{2}\ ; \quad \beta \geq \frac{1}{4}\left(\frac{1}{2} + \gamma\right)^2\ ;$$

$$a_o = \frac{1}{\beta(\Delta t)^2}; \qquad a_1 = \frac{\gamma}{\beta \Delta t}; \qquad a_2 = \frac{1}{\beta(\Delta t)};$$
$$a_3 = \frac{1}{2\beta} - 1; \qquad a_4 = \frac{\gamma}{\beta} - 1; \qquad a_5 = \frac{\Delta t}{2}\left(\frac{\gamma}{\beta} - 2\right);$$
$$a_6 = \Delta t(1 - \gamma); \qquad a_7 = \gamma \Delta t$$

2. Initialize displacements $\mathbf{u_o}$, velocities $\dot{\mathbf{u}}_\mathbf{o}$, and accelerations $\ddot{\mathbf{u}}_\mathbf{o}$.

3. Form effective stiffness matrix $\hat{\mathbf{K}}$

$$\hat{\mathbf{K}} = \mathbf{K} + a_o \mathbf{M}$$

4. Decompose $\hat{\mathbf{K}}$

$$\hat{\mathbf{K}} = \mathbf{L}\mathbf{D}\mathbf{L}^T$$

For each time step:

5. Calculate effective loads $\hat{\mathbf{R}}_{t+\Delta t}$

$$\hat{\mathbf{R}}_{t+\Delta t} = \mathbf{P}_{t+\Delta t} + \mathbf{M}(a_o \mathbf{u}_t + a_2 \dot{\mathbf{u}}_t + a_3 \ddot{\mathbf{u}}_t)$$

6. Solve for displacements at time $t + \Delta t$

$$\mathbf{L}\mathbf{D}\mathbf{L}^T \mathbf{u}_{t+\Delta t} = \hat{\mathbf{R}}_{t+\Delta t}$$

7. Calculate accelerations and velocities at time $t + \Delta t$

$$\ddot{\mathbf{u}}_{t+\Delta t} = a_o(\mathbf{u}_{t+\Delta t} - \mathbf{u}_t) - a_2 \dot{\mathbf{u}}_t - a_3 \ddot{\mathbf{u}}_t$$
$$\dot{\mathbf{u}}_{t+\Delta t} = \dot{\mathbf{u}}_t + a_6 \ddot{\mathbf{u}}_t + a_7 \ddot{\mathbf{u}}_{t+\Delta t}$$

This procedure neglects damping and assumes that a single, constant time step size Δt is used throughout the analysis.

3.2.4 PROCEDURE USAGE

Procedure **L_DYNAMIC_1** is used by preceding the procedure name by the ***call** directive. Procedure arguments may be changed from their default values by including any or all of the arguments and their new values when the procedure is called. A space or blank is required between the end of the procedure name and the left parenthesis. If the default values of the procedure arguments are to be used, then only the procedure name is required.

```
*call L_DYNAMIC_1 ( arg1 = val1 ; arg2 = val2 ; ...)
```

where **arg1** and **arg2** represent argument names, and **val1** and **val2** represent their corresponding values. Note that semi-colons are required between arguments, and a double dash (--) may be used to continue the argument list on the next line.

The allowable arguments for Procedure **L_DYNAMIC_1** are summarized in the following table, along with their default values (if they exist). Note that arguments without defaults are generally *mandatory*, while those with defaults are generally optional. Exceptions to this rule are noted in the following section under detailed argument descriptions.

For Procedure **L_DYNAMIC_1**, the following table lists each argument, its default value and meaning.

Argument	Default Value	Meaning
DELT	-	Time increment, Δt
NSTEP	-	Number of time steps
BETA	0.25	Time integrator parameter
GAMMA	0.50	Time integrator parameter

3.2.5 ARGUMENT DESCRIPTION

3.2.5.1 BETA

Newmark-β time integrator parameter, β (default: **1/4**).

3.2.5.2 DELT

Time step size. This argument specifies the size of the time step to be used in the analysis. A constant step size is assumed per procedure call.

3.2.5.3 GAMMA

Newmark-β time integrator parameter γ (default: **1/6**).

3.2.5.4 NSTEP

Number of time steps to march. This argument specifies the number of time steps to march in the transient response prediction using a constant time step size of DELT.

3.2.6 PROCEDURE FLOWCHART

L_DYNAMIC_1	(main procedure)
NEWMARK	(Newmark-β time integration)

3.2.7 LIMITATIONS

None.

3.2.8 ERROR MESSAGES AND WARNINGS

None.

3.2.9 USAGE GUIDELINES AND EXAMPLES

Procedure L_DYNAMIC_1 is used by preceding the procedure name by the *call directive. Procedure arguments may be changed from their default values by including any or all of the arguments and their new values when the procedure is called. A space or blank is required between the end of the procedure name and the left parenthesis.

```
*call L_DYNAMIC_1 ( delt = 0.02 ; --
                    nstep = 100 ; --
                    beta = 0.25 ; --
                    gamma = 0.50 )
```

3.2.10 PROCEDURE LISTING

3.2.11 REFERENCES

3.2-1 Bathe, K. J., *Finite Element Procedures in Engineering Analysis*, Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1982, pp. 511-512.

3.3 Procedure L_STABIL_1

3.3.1 GENERAL DESCRIPTION

Procedure L_STABIL_1 performs linear stability analysis using an eigensolver selected by the global macrosymbol **eigensolver_name** (*e.g.*, EIG2, LAN, LANZ) and structural element (ES*i*) processors based on the generic element processor template. The procedure assumes that the finite element model, loads, and boundary conditions have already been generated, that the prebuckling stress state has been specified, and that the buckling loads and mode shapes need to be calculated. The prebuckling stress state (*i.e.*, prestress state) is specified by prescribing values for procedure L_STABIL_1 arguments. A linear stability analysis is performed using this prescribed stress state.

3.3.2 THEORY

Linear elastic stability analyses may be formulated using the concept of adjacent equilibrium. Membrane forces in beams, plates, and shells result in an equilibrium configuration in which the deformation pattern is tangent to the midsurface of the structure. However, another equilibrium configuration involving out-of-plane deflections and rotations may be adjacent to this membrane state. Buckling occurs when this membrane strain energy is converted to bending strain energy. The linear elastic stability analysis is an eigenvalue problem to calculate the critical load for the bifurcation in the solution (*e.g.*, change from a membrane state to a bending state). This eigenvalue problem can be written as

$$\mathbf{K}\boldsymbol{\phi}_i + \lambda_i \mathbf{K}_g(\sigma)\boldsymbol{\phi}_i = \mathbf{0} \quad i = 1, 2, \ldots \tag{3.3-1}$$

where

$\mathbf{K}$ = assembled linear elastic stiffness matrix
$\mathbf{K}_g(\sigma)$ = assembled geometric stiffness matrix
$\boldsymbol{\phi}_i$ = i-th eigenvector or modeshape
λ_i = i-th eigenvalue or buckling load factor

The matrix denoted by $\mathbf{K}_g$ has been called the initial stress stiffness matrix, the differential stiffness matrix, the geometric stiffness matrix, and the stability coefficient matrix (*e.g.*, see ref. 3.3-1). It is independent of the elastic properties of the structure and dependent on the geometry, displacement field, and state of stress. Herein the matrix $\mathbf{K}_g$ will be referred to as the geometric stiffness matrix.

A general formulation for the geometric stiffness matrix is presented in reference 3.3-1. Strains can be written as

$$\epsilon = \epsilon_{\mathbf{L}} + \epsilon_{\mathbf{NL}} \tag{3.3-2}$$

where $\epsilon_{\mathbf{L}}$ contains the linear strain-displacement terms and $\epsilon_{\mathbf{NL}}$ contains the higher-order or nonlinear strain-displacement terms. For a given stress state σ_0, elastic strain energy

is stored and can be expressed as

$$U = U_L + U_{NL} \qquad (3.3-3)$$

where

$$U = \frac{1}{2}\int_V \epsilon^T \sigma_0 dV \qquad (3.3-4a)$$

$$U_L = \frac{1}{2}\int_V \epsilon_{\mathbf{L}}^T \sigma_0 dV \qquad (3.3-4b)$$

$$U_{NL} = \frac{1}{2}\int_V \epsilon_{\mathbf{NL}}^T \sigma_0 dV \qquad (3.3-4c)$$

The geometric stiffness matrix is derived from the strain energy produced by stresses acting through displacements associated with the nonlinear strain-displacement relations. These relations couple the membrane and bending effects. Typically the strain-displacements relations are written using index notation as

$$\epsilon_{ij} = \frac{1}{2}\left[\partial_i u_j + \partial_j u_i + \partial_i u_k \partial_j u_k\right] \qquad (3.3-5)$$

where $u_i = (u, v, w)$ for $i = 1, 2, 3$, respectively, ∂_i denotes differentiation with respect to ith coordinate direction and summation over repeated indices is implied. Let $\mathbf{d}$ denote the nodal degrees of freedom, then

$$\boldsymbol{\delta} = \mathbf{G}\mathbf{d} \qquad (3.3-6)$$

where

$$\boldsymbol{\delta} = \{\partial_x u,\ \partial_y u,\ \partial_z u,\ \partial_x v,\ \partial_y v,\ \partial_z v,\ \partial_x w,\ \partial_y w,\ \partial_z w\}$$

The coefficients of $\mathbf{G}$ are obtained by differentiating the element shape functions. Finally the matrix $\mathbf{Q}$ is defined as

$$\mathbf{Q} = \begin{bmatrix} \partial_x u & 0 & 0 & \partial_x v & 0 & 0 & \partial_x w & 0 & 0 \\ 0 & \partial_y u & 0 & 0 & \partial_y v & 0 & 0 & \partial_y w & 0 \\ 0 & 0 & \partial_z u & 0 & 0 & \partial_z v & 0 & 0 & \partial_z w \\ \partial_y u & \partial_x u & 0 & \partial_y v & \partial_x v & 0 & \partial_y w & \partial_x w & 0 \\ 0 & \partial_z u & \partial_y u & 0 & \partial_z v & \partial_y v & 0 & \partial_z w & \partial_y w \\ \partial_z u & 0 & \partial_x u & \partial_z v & 0 & \partial_x v & \partial_z w & 0 & \partial_x w \end{bmatrix} \qquad (3.3-7)$$

With these definitions the nonlinear strains given by equation (3.3-2) can be written as

$$\epsilon_{\mathbf{NL}} = \frac{1}{2}\mathbf{Q}\mathbf{G}\mathbf{d} \qquad (3.3-8)$$

The vector of initial stresses is

$$\sigma_0 = \{\sigma_{x0}\ \sigma_{y0}\ \sigma_{z0}\ \tau_{xy0}\ \tau_{yz0}\ \tau_{xz0}\} \qquad (3.3-9)$$

Substituting equations (3.3-6) through (3.3-9) into equation (3.3-4c) gives

$$U_{NL} = \frac{1}{2}\mathbf{d}^T\left(\int_V \mathbf{G}^T\mathbf{Q}^T\boldsymbol{\sigma}_0\, dV\right) \qquad (3.3-10)$$

However, the term $\mathbf{Q}^T\,\boldsymbol{\sigma}_0$ can be written as

$$\mathbf{Q}^T\boldsymbol{\sigma}_0 = \begin{bmatrix} \mathbf{s} & 0 & 0 \\ 0 & \mathbf{s} & 0 \\ 0 & 0 & \mathbf{s} \end{bmatrix}\boldsymbol{\delta} = \overline{\mathbf{S}}(\boldsymbol{\sigma}_0)\boldsymbol{\delta} \qquad (3.3-11)$$

where

$$\mathbf{s} = \begin{bmatrix} \sigma_{x0} & \tau_{xy0} & \tau_{xz0} \\ \tau_{xy0} & \sigma_{y0} & \tau_{yz0} \\ \tau_{xz0} & \tau_{yz0} & \sigma_{z0} \end{bmatrix} \qquad (3.3-12)$$

With these expressions, a general form of the geometric stiffness matrix can be written as

$$\mathbf{K}_g(\boldsymbol{\sigma}) = \int_V \mathbf{G}^T\overline{\mathbf{S}}(\boldsymbol{\sigma}_0)\mathbf{G}dV \qquad (3.3-13)$$

which is symmetric and explicitly dependent on the stress state.

The stress state used to form the geometric stiffness matrix may be obtained in two ways. The first way is first to perform a linear static stress analysis for the given load set and constraint set. This way is used in procedure **L_STABIL_2**. The second way is to specify, in advance, the values of the stress components given in equation (3.3-11) (*i.e.*, specify the prestress state). This way is used in procedure **L_STABIL_1**.

3.3.3 ALGORITHM

The algorithm used to solve equation (3.3-1) depends on the value of the global macrosymbol **eigensolver_name**. Processor EIG2 is used if **eigensolver_name** is defined to be **EIG2**. This processor uses a nodal-block sparse matrix approach as described in reference 3.3-2. Processor LAN is used if **eigensolver_name** is defined to be **LAN**. Processor LANZ is used if **eigensolver_name** is defined to be **LANZ**. These processors are based on the Lanczos algorithm as described in references 3.3-2 to 3.3-4.

3.3.4 PROCEDURE USAGE

Procedure **L_STABIL_1** may be invoked by the ***call** directive, and following it by a list of arguments separated by semicolons(;) and enclosed in parentheses. Procedure arguments are order-independent, and most have default values thus making them optional. The formal syntax is as follows:

```
*call L_STABIL_1 ( arg1 = val1; arg2 = val2; ...)
```

where **argi** are argument names and **vali** are the corresponding values. The following are valid arguments for procedure **L_STABIL_1**; note that those arguments without default values are mandatory, while the others are optional.

Argument	Default Value	Meaning
PS_1	--	Prebuckling membrane stress resultant N_x^o
PS_2	--	Prebuckling membrane stress resultant N_y^o
PS_3	--	Prebuckling membrane stress resultant N_{xy}^o
BCON_SET	1	Constraint set for buckling analysis
ERROR_TOL	.0001	Convergence criterion for eigenvalues
FUNCTION	ALL	Select function to be performed by procedure
INIT_VECTOR	0	Number of initial vectors used to span the subspace
ISEQ	0	Resequencing method to be used
LDI	1	Local device index
LOAD_SET	1	Load set number
KNAME	K	First word of the name of the dataset containing the assembled stiffness matrix
MAX_ITERS	20	Maximum number of iterations allowed
N_GROUPS	1	
N_MODES	1	Number of eigenvalues to converge
PRINT	<false>	Flag to print displacement solution, internal forces, and element stresses, and eigenvectors
RENUMBER	<true>	Flag to resequence node numbers for equation solver
SHIFT	0.0	Eigenvalue shift

Tables 3.3-1, 3.3-2, and 3.3-3 list the datasets used or created by procedure **L_STABIL_1**, the procedures invoked by procedure **L_STABIL_1**, and the processors invoked by procedure **L_STABIL_1**, respectively.

Table 3.3-1 Datasets Input/Output by procedure L_STABIL_1			
Dataset	*Description*	*Input*	*Output*
AMAP..ic2.isize	Factorization Map for INV		√
BUCK.EVAL.*i.j*†	Buckling eigenvalues		√
BUCK.MODE.*i.j*†	Buckling eigenvalues		√
<ES_NAME>.EFIL.0.*nnod*	Element Computational Data	√	√
ES.SUMMARY	ES Processor Status	√	√
DEF.<ES_NAME>.0.*nnod*	Element Definition (Connectivity)	√	
DIR.<ES_NAME>.0.*nnod*	Element EFIL Directory	√	
INV.KSHF.*j*†	Factored Shifted System Matrix		√
INV.<KNAME>.*j*†	Factored System Matrix		√
JDF1.BTAB.1.8	Model Summary	√	
KG.SPAR.jdf2	Assembled geometric stiffness matrix		√
KMAP..ic2.isize	Model Connectivity Map		√
<KNAME>.SPAR.jdf2	Assembled system matrix		√

† i = <load_set> and j = <cons_set>

Table 3.3-2 Sub-Procedures Invoked by procedure L_STABIL_1		
Procedure	*Type*	*Function*
ES	External	Element utility procedure
FACTOR	External	Factors assembled stiffness matrix
L_STABIL_1	Internal	Main procedure

Table 3.3-3 Processors Invoked by procedure L_STABIL_1		
Procedure	*Type*	*Function*
AUS	Internal	Arithmetic Utilities
E	Internal	Initializes EFIL datasets
EIG2	Internal	Solve eigenvalue problem using subspace iteration
ES*i*	External	Element processors based on GEP
K	Internal	Assemble system matrix
LAN	External	Solve eigenvalue problem using Lanczos method
LANZ	External	Solve eigenvalue problem using Lanczos method
RSEQ	Internal	Resequences nodes for equation solving
TOPO	Internal	Generates nodal topology maps
VPRT	Internal	Print SYSVEC system vectors

3.3.5 ARGUMENT DESCRIPTION

3.3.5.1 BCON_SET

Constraint set number for buckling analysis (default: `1`). This argument selects which constraint set to use in solving the linear stability problem.

3.3.5.2 ERROR_TOL

Convergence criterion for eigenvalues (default: `0.0001`). For the k-th iteration, the error measure for the i-th eigenvalue is

$$\epsilon_i^k = \frac{\|(\lambda_i^k - \lambda_i^{k-1})\|}{\|\lambda_i^k\|}$$

The i-th eigenvalue is converged if ϵ_i^k is smaller than `ERROR_TOL`.

3.3.5.3 FUNCTION

Select function to be performed by procedure `L_STABIL_1` (default: `ALL`). This procedure may be used to perform two functions. For `FUNCTION=ALL`, the element data are initialized and elemental stiffness matrices formed; nodal resequencing may be performed, the mesh

topology is analyzed, the system stiffness matrix is assembled and factored, and the eigenproblem is solved. For **FUNCTION=EIGEN**, procedure **L_STABIL_1** uses a prescribed prestress state in solving the eigenvalue problem. Using the **FUNCTION** argument, the user may solve for a variety of constraint (boundary conditions) sets on a given model subjected to a variety of loading conditions.

3.3.5.4 INIT_VECTOR

Number of initial vectors used to span the subspace (default: 0). This argument defines the number of trial vectors used to initiate the subspace iteration. If **INIT_VECTOR=0**, the number of initial vectors will be calculated by the procedure as

```
INIT_VECTOR=MINIMUM (2*N_MODES, N_MODES + 8)
```

3.3.5.5 ISEQ

Resequencing method to be used (default: 0). If the argument **RENUMBER** is **<true>**, then nodal resequencing will be performed using processor RSEQ. The method used by processor RSEQ to resequence the nodes depends on the value of **ISEQ**. If the argument **ISEQ** is greater than or equal to zero, then that method will be used (*i.e.*, **method=0,1,2,3**; see Section 6.1 of the CSM Testbed User's Manual, ref. 3.3-2).

3.3.5.6 KNAME

First word of the dataset name containing the assembled stiffness matrix (default: **K**).

3.3.5.7 LDI

Logical device index (default: 1).

3.3.5.8 MAX_ITERS

Maximum number of iterations (default: 20). This argument specifies the maximum number of iterations that can be used per call to eigensolver.

3.3.5.9 N_GROUPS

3.3.5.10 N_MODES

Number of converged eigenvalues desired (default: 1). This argument specifies the number of eigenvalues to calculate to a convergence criterion of **ERROR_TOL**.

3.3.5.11 PRINT

Flag to print modeshapes (default: **<false>**). If printing of these computed results is requested, processor VPRT will be used to print the buckling modeshapes.

3.3.5.12 PS_1

Prescribed membrane stress resultant N_x^o for the prestressed state.

3.3.5.13 PS_2

Prescribed membrane stress resultant N_y^o for the prestressed state.

3.3.5.14 PS_3

Prescribed membrane stress resultant N_{xy}^o for the prestressed state.

3.3.5.15 RENUMBER

Flag to resequence node numbers prior to equation solving (default: `<true>`). If the argument `RENUMBER=<true>`, then processor RSEQ will be used to perform nodal resequencing, otherwise no resequencing will be performed. Note that the nodal resequencing may greatly reduce the time required to factor and solve the linear system of equations.

3.3.5.16 SHIFT

Eigenvalue shift (default: `0.0`). Converged eigenvalue will only be obtained for eigenvalues greater than `SHIFT`. The shift parameter refers to the shift in the buckling load factor.

3.3.6 PROCEDURE FLOWCHART

`L_STABIL_1`	(main procedure)
`INITIALIZE`	(initialize)
`STIFFNESS`	(form $\mathbf{K}$)
`STIFFNESS`	(form $\mathbf{K}_g$)
`FACTOR`	(factor using buckling boundary conditions)
`EIGEN`	(perform eigenvalue analysis)

3.3.7 LIMITATIONS

None.

3.3.8 ERROR MESSAGES AND WARNINGS

None.

3.3.9 USAGE GUIDELINES AND EXAMPLES

Procedure L_STABIL_1 may be used by preceding the procedure name by the *call directive. Procedure arguments may be changed from their default values by including any or all the arguments and their new values when the procedure is called. A space or blank is required between the end of the procedure name and the left parenthesis.

```
*call L_STABIL_1 ( FUNCTION = ALL ; -- . Select function
                   BCON_SET  = 1  ; -- . Select buckling constraint set
                   ERROR_TOL = .0001 ; -- . Eigenvalue convergence criterion
                   INIT_VECTOR = 0 ; -- . Number of initial vectors
                   ISEQ      = -1  ; -- . Select resequencing method
                   KNAME     =  K  ; -- . First word of stiffness matrix
                                     -- . dataset name
                   MAX_ITERS = 20;  -- Maximum number of iterations
                   N_MODES   =  1 ;  -- Number of eigenvalues
                   N_GROUPS  =  1 ;  --
                   PS_1          ;  --
                   PS_2          ;  --
                   PS_3          ;  --
                   PRINT     =  <true> ;  -- . PRINT flag
                   RENUMBER  =  <true> ;  -- . RESEQUENCING flag
                   SHIFT     =  0.0 ; Eigenvalue shift
                                      )
```

Before procedure L_STABIL_1 is called, the global macrosymbol **eigensolver_name** should be defined as described in Section 3.3.3; otherwise, the default value of EIG2 will be used.

3.3.10 PROCEDURE LISTING

3.3.11 REFERENCES

3.3-1 Cook, Robert D.: *Concepts and Applications of Finite Element Analysis.* (Second Edition). John Wiley and Sons, Inc., New York 1981.

3.3-2 Stewart, Caroline B.: *The Computational Structural Mechanics Testbed User's Manual.* NASA TM-100644, October 1989.

3.3-3 Bostic, S. W. and Fulton R. E.: *A Lanczos Eigenvalue Method on a Parallel Computer.* AIAA Paper No. 87-0725-CP.

3.3-4 Jones, Mark T. and Patrick, Merrell L.: *The Use of Lanczos's Method to Solve the Large Generalized Symmetric Definite Eigenvalue Problem.* NASA CR-181914, September 1989. (Also available as ICASE Report No. 89-69).

THIS PAGE LEFT BLANK INTENTIONALLY.

3.4 Procedure L_STABIL_2

3.4.1 GENERAL DESCRIPTION

Procedure L_STABIL_2 performs linear stability analysis using an eigensolver selected by the global macrosymbol **eigensolver_name** (*e.g.*, EIG2, LAN, LANZ) and structural element (ES*i*) processors based on the generic element processor template. The procedure assumes that the finite element model, loads, and boundary conditions have already been generated, and that the nodal displacements, reaction forces, element stresses, and buckling loads and mode shapes need to be calculated. The applied loads may be due to a combination of specified forces and displacements, and one constraint (*i.e.*, boundary condition) set is permitted per procedure call. A linear elastic stress analysis is performed first using procedure L_STATIC (see Section 3.5) to calculate the prebuckling stress state (*i.e.*, prestress state). After the linear static solution, a linear stability analysis is performed.

3.4.2 THEORY

Linear elastic stability analyses may be formulated using the concept of adjacent equilibrium. Membrane forces in beams, plates, and shells result in an equilibrium configuration in which the deformation pattern is tangent to the midsurface of the structure. However, another equilibrium configuration involving out-of-plane deflections and rotations may be adjacent to this membrane state. Buckling occurs when this membrane strain energy is converted to bending strain energy. The linear elastic stability analysis is an eigenvalue problem to calculate the critical load for the bifurcation in the solution (*e.g.*, change from a membrane state to a bending state). This eigenvalue problem can be written as

$$\mathbf{K}\phi_i + \lambda_i \mathbf{K}_g(\sigma)\phi_i = 0 \quad i = 1, 2, \ldots \qquad (3.4-1)$$

where

$\mathbf{K}$ = assembled linear elastic stiffness matrix
$\mathbf{K}_g(\sigma)$ = assembled geometric stiffness matrix
ϕ_i = i-th eigenvector or modeshape
λ_i = i-th eigenvalue or buckling load factor

The matrix denoted by $\mathbf{K}_g$ has been called the initial stress stiffness matrix, the differential stiffness matrix, the geometric stiffness matrix, and the stability coefficient matrix (*e.g.*, see ref 3.4-1). It is independent of the elastic properties of the structure and dependent on the geometry, displacement field, and state of stress. Herein the matrix $\mathbf{K}_g$ will be referred to as the geometric stiffness matrix.

A general formulation for the geometric stiffness matrix is presented in reference 3.4-1. Strains can be written as

$$\epsilon = \epsilon_L + \epsilon_{NL} \qquad (3.4-2)$$

where ϵ_L contains the linear strain-displacement terms and ϵ_{NL} contains the higher-order or nonlinear strain-displacement terms. For a given stress state σ_0, elastic strain energy is stored and can be expressed as

$$U = U_L + U_{NL} \qquad (3.4-3)$$

where

$$U = \frac{1}{2}\int_V \epsilon^T \sigma_0 dV \qquad (3.4-4a)$$

$$U_L = \frac{1}{2}\int_V \epsilon_L^T \sigma_0 dV \qquad (3.4-4b)$$

$$U_{NL} = \frac{1}{2}\int_V \epsilon_{NL}^T \sigma_0 dV \qquad (3.4-4c)$$

The geometric stiffness matrix is derived from the strain energy produced by stresses acting through displacements associated with the nonlinear strain-displacement relations. These relations couple the membrane and bending effects. Typically the strain-displacements relations are written using index notation as

$$\epsilon_{ij} = \frac{1}{2}\left[\partial_i u_j + \partial_j u_i + \partial_i u_k \partial_j u_k\right] \qquad (3.4-5)$$

where $u_i = (u, v, w)$ for $i = 1, 2, 3$, respectively, ∂_i denotes differentiation with respect to ith coordinate direction and summation over repeated indices is implied. Let **d** denote the nodal degrees of freedom, then

$$\boldsymbol{\delta} = \mathbf{G}\mathbf{d} \qquad (3.4-6)$$

where

$$\boldsymbol{\delta} = \{\partial_x u,\ \partial_y u,\ \partial_z u,\ \partial_x v,\ \partial_y v,\ \partial_z v,\ \partial_x w,\ \partial_y w,\ \partial_z w\}$$

The coefficients of **G** are obtained by differentiating the element shape functions. Finally the matrix **Q** is defined as

$$\mathbf{Q} = \begin{bmatrix} \partial_x u & 0 & 0 & \partial_x v & 0 & 0 & \partial_x w & 0 & 0 \\ 0 & \partial_y u & 0 & 0 & \partial_y v & 0 & 0 & \partial_y w & 0 \\ 0 & 0 & \partial_z u & 0 & 0 & \partial_z v & 0 & 0 & \partial_z w \\ \partial_y u & \partial_x u & 0 & \partial_y v & \partial_x v & 0 & \partial_y w & \partial_x w & 0 \\ 0 & \partial_z u & \partial_y u & 0 & \partial_z v & \partial_y v & 0 & \partial_z w & \partial_y w \\ \partial_z u & 0 & \partial_x u & \partial_z v & 0 & \partial_x v & \partial_z w & 0 & \partial_x w \end{bmatrix} \qquad (3.4-7)$$

With these definitions the nonlinear strains given by equation (3.4-2) can be written as

$$\epsilon_{NL} = \frac{1}{2}\mathbf{Q}\mathbf{G}\mathbf{d} \qquad (3.4-8)$$

The vector of initial stresses is

$$\sigma_0 = \{\sigma_{x0}\ \sigma_{y0}\ \sigma_{z0}\ \tau_{xy0}\ \tau_{yz0}\ \tau_{zz0}\} \qquad (3.4-9)$$

Substituting equations (3.4-6) through (3.4-9) into equation (3.4-4c) gives

$$U_{NL} = \frac{1}{2}\mathbf{d}^T\left(\int_V \mathbf{G}^T\mathbf{Q}^T\sigma_0\ dV\right) \qquad (3.4-10)$$

However, the term $\mathbf{Q}^T\ \sigma_0$ can be written as

$$\mathbf{Q}^T\sigma_0 = \begin{bmatrix} \mathbf{s} & 0 & 0 \\ 0 & \mathbf{s} & 0 \\ 0 & 0 & \mathbf{s} \end{bmatrix} \delta = \overline{\mathbf{S}}(\sigma_0)\delta \qquad (3.4-11)$$

where

$$\mathbf{s} = \begin{bmatrix} \sigma_{x0} & \tau_{xy0} & \tau_{xz0} \\ \tau_{xy0} & \sigma_{y0} & \tau_{yz0} \\ \tau_{xz0} & \tau_{yz0} & \sigma_{z0} \end{bmatrix} \qquad (3.4-12)$$

With these expressions, a general form of the geometric stiffness matrix can be written as

$$\mathbf{K}_g(\sigma) = \int_V \mathbf{G}^T\overline{\mathbf{S}}(\sigma_0)\mathbf{G}dV \qquad (3.4-13)$$

which is symmetric and explicitly dependent on the stress state.

The stress state used to form the geometric stiffness matrix may be obtained in two ways. The first way is first to perform a linear static stress analysis for the given load set and constraint set. This way is used in procedure L_STABIL_2. The second way is to specify, in advance, the values of the stress components given in equation (3.4-11) (*i.e.*, specify the prestress state). This way is used in procedure L_STABIL_1.

3.4.3 ALGORITHM

The algorithm used to solve equation (3.4-1) depends on the value of the global macrosymbol **eigensolver_name**. Processor EIG2 is used if **eigensolver_name** is defined to be **EIG2**. This processor uses a nodal-block sparse matrix approach as described in reference 3.4-2. Processor LAN is used if **eigensolver_name** is defined to be **LAN**. Processor LANZ is used if **eigensolver_name** is defined to be **LANZ**. These processors are based on the Lanczos algorithm as described in references 3.4-2 to 3.4-4.

3.4.4 PROCEDURE USAGE

Procedure L_STABIL_2 may be invoked by the *call directive, and following it by a list of arguments separated by semicolons(;) and enclosed in parentheses. Procedure arguments are order-independent, and most have default values thus making them optional. The formal syntax is as follows:

```
*call L_STABIL_2 ( arg1 = val1; arg2 = val2; ...)
```

where **arg***i* are argument names and **val***i* are the corresponding values. The following are valid arguments for procedure L_STABIL_2; note that those arguments without default values are mandatory, while the others are optional.

Argument	Default Value	Meaning
BCON_SET	1	Constraint set for buckling analysis
CONS_SET	1	Constaint set number for prestress analysis
DIRECTION	0	Direction for element stress output
ERROR_TOL	.0001	Convergence criterion for eigenvalues
FUNCTION	ALL	Select function to be performed by procedure
INIT_VECTOR	0	Number of initial vectors used to span the subspace
ISEQ	0	Resequencing method to be used
KNAME	K	First word of the name of the dataset containing the assembled stiffness matrix
LDI	1	Logical device index
LOAD_SET	1	Load set number
LOCATION	CENTROIDS	Location of the evaluation points for element stresses
MAX_ITERS	20	Maximum number of iterations allowed
N_MODES	1	Number of eigenvalues to converge
NVAL_METH	3	Method to be used for global smoothing
PRINT	<false>	Flag to print displacement solution, internal forces, element stresses, and eigenvectors
REACTION	<false>	Flag to compute internal forces or reactions
RENUMBER	<true>	Flag to resequence node numbers for equation solver
RHS	APPL.FORC	First two words of the dataset name for the right-hand side system vector
SHIFT	0.0	Eigenvalue shift
SMOOTH	<false>	Flag to compute smoothed global stresses
SOLN	STAT.DISP	First two words of the dataset name for the displacement solution
STRESS	<false>	Flag to compute element stresses (resultants)

Tables 3.4-1, 3.4-2, and 3.4-3 list the datasets used or created by procedure L_STABIL_2, the procedures invoked by procedure L_STABIL_2, and the processors invoked by procedure L_STABIL_2, respectively.

Table 3.4-1 Datasets Input/Output by procedure L_STABIL_2			
Dataset	*Description*	*Input*	*Output*
AMAP..ic2.isize	Factorization map for INV		√
APPL.FORC.*i*.1†	Applied force vector	√	
APPL.MOTI.*i*.1†	Specified displacement vector	√	
BUCK.EVAL.*i.j*†	Buckling eigenvalues		√
BUCK.MODE.*i.j*†	Buckling eigenvalues		√
<ES_NAME>.EFIL.0.*nnod*	Element Computational Data	√	√
ES.SUMMARY	ES Processor Status	√	√
DEF.<ES_NAME>.0.*nnod*	Element Defn. (Connectivity)	√	
DIR.<ES_NAME>.0.*nnod*	Element EFIL Directory	√	
INT.FORC.*i.j*†	System Internal Force Vector		√
INV.KSHF.*j*†	Factored Shifted System Matrix		√
INV.<KNAME>.*j*†	Factored System Matrix		√
JDF1.BTAB.1.8	Model Summary	√	
KG.SPAR.jdf2	Assembled geometric stiffness matrix		√
KMAP..ic2.isize	Model connectivity map		√
<KNAME>.SPAR.jdf2	Assembled system matrix		√
STAT.DISP.*i.j*†	System Displacement Vector		√
STAT.REAC.*i.j*†	System Reaction Force Vector		√
STRS.<ES_NAME>.*i.j*†	Element Stresses		√

† i = <load_set> and j = <cons_set>

Table 3.4-2 Sub-Procedures Invoked by procedure L_STABIL_2		
Procedure	*Type*	*Function*
ES	External	Element utility procedure
FACTOR	External	Factors assembled stiffness matrix
L_STABIL_2	Internal	Main procedure
L_STATIC	Internal	Linear static analysis

Table 3.4-3 Processors Invoked by procedure L_STABIL_2		
Procedure	*Type*	*Function*
AUS	Internal	Arithmetic Utilities
E	Internal	Initializes EFIL datasets
EIG2	Internal	Solve eigenvalue problem using subspace iteration
ES*i*	External	Element processors based on GEP
K	Internal	Assemble system matrix
LAN	External	Solve eigenvalue problem using Lanczos method
LANZ	External	Solve eigenvalue problem using Lanczos method
RSEQ	Internal	Resequences nodes for equation solving
TOPO	Internal	Generates nodal topology maps
VPRT	Internal	Print system vectors

3.4.5 ARGUMENT DESCRIPTION

3.4.5.1 BCON_SET

Constraint set number for buckling analysis (default: 1). This argument selects which constraint set to use in solving the linear stability problem.

3.4.5.2 CONS_SET

Constraint set number for prestress solution (default: 1). This argument selects which constraint set to use in solving the linear system of equations.

3.4.5.3 DIRECTION

Direction for the element stress (stress resultant) output (default: 0). The element stress coordinate system will be used if **direction=0**. The material axes (x_m, y_m, z_m) will be used if **direction=1**; the material axes (y_m, z_m, x_m) will be used for **direction=2**; and the material axes (z_m, x_m, y_m) will be used for **direction=3**. For isotropic materials, the first material axis is replaced by the corresponding global axis (see Section 4.3.3.9 of the CSM Testbed User's Manual, ref. 3.4-2).

3.4.5.4 ERROR_TOL

Convergence criterion for eigenvalues (default: 0.0001). For the k-th iteration, the error measure for the k-th eigenvalue is

$$\epsilon_i^k = \frac{\|(\lambda_i^k - \lambda_i^{k-1})\|}{\|\lambda_i^k\|}$$

The i-th eigenvalue is converged if ϵ_i^k is smaller than ERROR_TOL.

3.4.5.5 FUNCTION

Select function to be performed by procedure L_STABIL_2 (default: ALL). This procedure may be used to perform four functions. For **FUNCTION=ALL**, the element data are initialized and elemental stiffness matrices formed; nodal resequencing may be performed, the mesh topology is analyzed, the system stiffness matrix is assembled and factored, the displacement solution is obtained, optionally element stresses (stress resultants) and internal nodal forces (reactions) computed and the eigenproblem is solved. For **FUNCTION=FACT_SOLV**, procedure L_STABIL_2 assumes that the system stiffness matrix has previously been assembled and that nodal resequencing has been performed. The procedure then proceeds to factor the system stiffness matrix, solves for the displacement solution, optionally computes the element stresses (stress resultants) and internal nodal forces (reactions) and solves the eigenproblem. For **FUNCTION=SOLV**, procedure L_STABIL_2 assumes that the system stiffness has previously been formed and factored. The procedure then proceeds to solve for the displacement solution, optionally computes the element stresses (stress resultants) and internal nodal forces (reactions) and solves the eigenproblem. For **FUNCTION=EIGEN**, procedure L_STABIL_2 uses a previously computed prestress state in solving the eigenvalue problem. Using the **FUNCTION** argument, the user may solve for a variety of constraint (boundary conditions) sets on a given model subjected to a variety of loading conditions.

3.4.5.6 INIT_VECTOR

Number of initial vectors used to span the subspace (default: 0). This argument defines the number of trial vectors used to initiate the subspace iteration. If **INIT_VECTOR=0**, the number of initial vectors will be calculated by the procedure as

```
INIT_VECTOR=MINIMUM (2*N_MODES, N_MODES + 8)
```

3.4.5.7 ISEQ

Resequencing method to be used (default: 0). If the argument RENUMBER is <true>, then nodal resequencing will be performed using processor RSEQ. The method used by processor RSEQ to resequence the nodes depends on the value of ISEQ. If the argument ISEQ is greater than or equal to zero, then that method will be used (*i.e.*, method=0,1,2,3; see Section 6.1 of the CSM Testbed User's Manual, ref. 3.4-2).

3.4.5.8 KNAME

First word of the dataset name containing the assembled stiffness matrix (default: K).

3.4.5.9 LDI

Logical device index (default: 1).

3.4.5.10 LOAD_SET

Load set number (default: 1). This argument selects which load set to use as a right-hand side vector.

3.4.5.11 LOCATION

Location of the evaluation points for the element stresses or stress resultants (default: CENTROIDS). The element stresses or stress resultants are optionally computed by calling procedure STRESS (see Section 6.4). This argument may have four values. For LOCATION=INTEG_PTS, the element stresses are computed at the element integration points. For LOCATION=CENTROIDS, the element stresses are computed at the element centroid. For LOCATION=NODES, the element stresses are extrapolated from the integration points to be element nodes. These element nodal stresses will be discontinuous across interelement boundaries. For LOCATION=ALL, the element stresses are computed at the element integration points, element centroid, and element nodes.

3.4.5.12 MAX_ITERS

Maximum number of iterations (default: 20). This argument specifies the maximum number of iterations that can be used per call to eigensolver.

3.4.5.13 N_MODES

Number of converged eigenvalues desired (default: 1). This argument specifies the number of eigenvalues to calculate to a convergence criterion of ERROR_TOL.

3.4.5.14 NVAL_METH

Select method to be used for computing the smoothed global stresses (default: 3). Processor NVAL is used to compute the smoothed global stresses using the method defined by the argument NVAL_METH (see Section 12.5 of reference 3.4-2). If NVAL_METH=1, a topological interpolation of the element centroidal stresses is performed, and the element stresses must have been computed using LOCATION=CENTROIDS. If NVAL_METH=2, a projected least-squares interpolation of the element centroidal stresses is performed, and the stresses must have been computed using LOCATION=CENTROIDS. If NVAL_METH=3, the element nodal stresses (discontinuous across interelement boundaries) are averaged, and the element stresses must have been computed using LOCATION=NODES. Using LOCATION=ALL will generate element stresses at the element centroids, element nodes, and element gauss points. Acceptable values of LOCATION for specific values of NVAL_METH are as follows:

NVAL_METH	LOCATION
1	CENTROIDS, ALL
2	CENTROIDS, ALL
3	NODES, ALL

3.4.5.15 PRINT

Flag to print displacement solution, internal forces and element stresses (default: <false>). If printing of these computed results is requested, processor VPRT will be used to print the displacement solution and internal forces and processor PESR will be used to print the element stresses.

3.4.5.16 REACTION

Flag to compute the internal nodal forces (default: <true>) If the argument REACTION=<true>, then the internal forces will be computed by calling procedure INT_FORC (see Section 6.2).

3.4.5.17 RENUMBER

Flag to resequence node numbers prior to equation solving (default: <true>). If the argument RENUMBER=<true>, then processor RSEQ will be used to perform nodal resequencing, otherwise no resequencing will be performed. Note that the nodal resequencing may greatly reduce the time required to factor and solve the linear system of equations.

3.4.5.18 RHS

First two words of the dataset name for the right-hand side system vector (default: APPL.FORC).

3.4.5.19 SHIFT

Eigenvalue shift (default: 0.0). Converged eigenvalues will only be obtained for values greater than SHIFT. The shift parameter refers to the shift in the buckling load factor.

3.4.5.20 SMOOTH

Flag to compute smoothed global stresses (default: **<false>**). If the argument SMOOTH=**<true>**, then smoothed global stresses will be computed by processor NVAL using the method defined by the argument **NVAL_METH**.

3.4.5.21 SOLN

First two words of the dataset name for the displacement solution (default: **STAT.DISP**).

3.4.5.22 STRESS

Flag to compute element stresses or stress resultants (default: **<false>**). If the argument STRESS=**<true>**, then the element stresses will be computed at the location and in the direction specified by the arguments **LOCATION** and **DIRECTION**, respectively, by calling procedure **STRESS** (see Section 6.4).

3.4.6 PROCEDURE FLOWCHART

L_STABIL_2	(main procedure)
L_STATIC	(linear static analysis procedure)
ES	(initialize, form $\mathbf{K}$)
FACTOR	(factor assembled stiffness matrix)
SOLVE	(solve linear system of equations)
STRESS	(stress/strain recovery procedure)
ES	(calculate element and nodal stress/strain)
INT_FORCE	(internal force procedure)
ES	(internal force calculation)
ES	(form $\mathbf{K}_g$)
FACTOR	(factor using buckling boundary conditions)

3.4.7 LIMITATIONS

None.

3.4.8 ERROR MESSAGES AND WARNINGS

None.

3.4.9 USAGE GUIDELINES AND EXAMPLES

Procedure L_STABIL_2 may be used by preceding the procedure name by the *call directive. Procedure arguments may be changed from their default values by including any or all the arguments and their new values when the procedure is called. A space or blank is required between the end of the procedure name and the left parenthesis.

```
*call L_STABIL_2 ( FUNCTION = ALL ; -- . Select function
           CONS_SET  = 1  ; -- . Select constraint set
           BCON_SET  = 1  ; -- . Select buckling constraint set
           DIRECTION =  0  ; -- . Select direction for element stresses
           ERROR_TOL = .0001 ; --
           INIT_VECTOR = 0 ; --
           ISEQ      =  0  ; -- . Select resequencing method
           KNAME     =  K  ; -- . First word of stiffness matrix
                       9    -- . dataset name
           LOAD_SET  = 1  ; -- . Select load set
           LOCATION  =  CENTROIDS ; -- Select location of element
                                    -- . stress evaluation points
           MAX_ITERS = 20;  -- Maximum number of iterations
           N_MODES   =  1 ;    -- Number of eigenvalues
           NVAL_METH = 3 ; --
           PRINT      =  <false> ;  -- . PRINT flag
           REACTION  =  <false> ; -- . REACTIONS flag
           RENUMBER  =  <true> ;  -- . RESEQUENCING flag
           RHS       =  APPL. FORC ; -- . First two words of RHS
                                    -- . vector dataset
           SHIFT     =  0.0 ; Eigenvalue shift
           SMOOTH    =  <false> ; -- compute smoothed global stresses?
           SOLN      =  STAT. DISP ; -- . First two words of SOLN
                                    -- . dataset
           STRESS    =  <false> -- . STRESS flag
                                   )
```

Before procedure L_STABIL_2 is called, the global macrosymbol **eigensolver_name** should be defined as described in Section 3.4.3; otherwise, the default value of **EIG2** will be used.

3.4.10 PROCEDURE LISTING

3.4.11 REFERENCES

3.4-1 Cook, Robert D.: *Concepts and Applications of Finite Element Analysis.* (Second Edition). John Wiley and Sons, Inc., New York 1981.

3.4-2 Stewart, Caroline B.: *The Computational Structural Mechanics Testbed User's Manual.* NASA TM-100644, October 1989.

3.4-3 Bostic, S. W. and Fulton R. E.: *A Lanczos Eigenvalue Method on a Parallel Computer.* AIAA Paper No. 87-0725-CP.

3.4-4 Jones, Mark T. and Patrick, Merrell L.: *The Use of Lanczos's Method to Solve the Large Generalized Symmetric Definite Eigenvalue Problem.* NASA CR-181914, September 1989. (Also available as ICASE Report No. 89-69).

3.5 Procedure L_STATIC

3.5.1 GENERAL DESCRIPTION

Procedure L_STATIC performs linear static analysis, using an equation solver selected by defining the global macrosymbol **solver_name** (*e.g.*, INV, BAND, ITER, SPK) and user specified structural element (ES*i*) processors based on the generic element processor template. The procedure assumes that the finite element model, loads, and boundary conditions, have already been generated, and that the nodal displacements, reaction forces, and element stresses need to be calculated. The applied loads may be due to a combination of specified forces and displacements, and one constraint (*i.e.*, boundary condition) set is permitted per procedure call.

3.5.2 THEORY

Mathematically, procedure L_STATIC solves the linear static problem:

$$\mathbf{K}\,\mathbf{d} = \mathbf{f} \qquad (3.5-1)$$

where

$\mathbf{K}$ = assembled linear stiffness matrix
$\mathbf{d}$ = generalized displacement vector
$\mathbf{f}$ = external force vector

Note that both the translational and rotational displacements in **d** are assumed to be infinitesimally small (linear strain-displacement relations are employed), and that the material (stress-strain relation) is assumed to be both linear and elastic. Once the displacement vector, **d**, is computed by solution of equation (3.5-1), the reaction forces are computed by multiplying those rows of the stiffness matrix **K** at which the displacements are prescribed (zero or non-zero) by the generalized displacement vector **d**. Finally, element stresses may be optionally computed.

3.5.3 ALGORITHM

The algorithm used to solve equation 3.5-1 depends on the value of the global macrosymbol **solver_name**. Processors INV and SSOL are used if **solver_name** is defined to be **INV**. These processors use nodal-block sparse matrix approach as described in references 3.5-1, 3.5-2 and 3.5-3. Processor BAND (**meth=28**) is used if **solver_name** is defined to be **BAND**. This algorithm is a $\mathbf{LDL}^T$ direct method based on an Choleski variable bandwidth method with loop unrolling as described in references 3.5-1 and 3.5-4. Processor ITER (**meth=0**) is used if **solver_name** is defined to be **ITER**. This algorithm is an iterative method based on a conjugate gradient method with diagonal scaling and sparse storage of the system matrix as described in references 3.5-1 and 3.5-4. Processor SPK is used if **solver_name** is defined to be **SPK**. This algorithm uses the vector-sum column Cholesky algorithms to

factor a general sparse matrix as described in references 3.5-1 and 3.5-5. Processor SPK contains a subset of the SPARSPAK-A package of FORTRAN programs designed to solve effectively large sparse systems of linear equations by direct methods (see reference 3.5-6).

3.5.4 PROCEDURE USAGE

Procedure L_STATIC may be invoked by the *call directive, and following it by a list of arguments separated by semicolons(;) and enclosed in parentheses. Procedure arguments are order-independent, and most have default values thus making them optional. The formal syntax is as follows:

```
*call L_STATIC ( arg1 = val1; arg2 = val2; ...)
```

where **arg***i* are argument names and **val***i* are the corresponding values. The following are valid arguments for procedure L_STATIC; note that those arguments without default values are mandatory, while the others are optional.

Argument	Default Value	Meaning
CONS_SET	1	Constraint set number
DIRECTION	1	Direction for element stress output
FUNCTION	ALL	Select function to be performed by procedure
ISEQ	-1	Resequencing method to be used
KNAME	K	First word of the name of the dataset containing the assembled stiffness matrix
LDI	1	Logical device index
LOAD_SET	1	Load set number
LOCATION	NODES	Location of the evaluation points for element stresses
NVAL_METH	3	Method to be used for global smoothing
PRINT	<false>	Flag to print computed results
REACTION	<false>	Flag to compute internal forces or reactions
RENUMBER	<true>	Flag to resequence node numbers for equation solver
RHS	APPL.FORC	First two words of the dataset name for the right-hand side system vector
SMOOTH	<false>	Flag to compute smoothed global stresses
SOLN	STAT.DISP	First two words of the dataset name for the displacement solution
STRESS	<false>	Flag to compute element stresses (resultants)

Tables 3.5-1, 3.5-2, and 3.5-3 list the datasets used or created by procedure L_STATIC, the procedures invoked by procedure L_STATIC, and the processors invoked by procedure L_STATIC, respectively.

Table 3.5-1 Datasets Input/Output by procedure L_STATIC			
Dataset	*Description*	*Input*	*Output*
AMAP..ic2.isize	Factorization Map for INV		√
APPL.FORC.*i*.1†	Applied force vector	√	
APPL.MOTI.*i*.1†	Specified displacement vector	√	
<ES_NAME>.EFIL.0.*nnod*	Element Computational Data	√	√
ES.SUMMARY	ES Processor Status	√	√
DEF.<ES_NAME>.0.*nnod*	Element Defn. (Connectivity)	√	
DIR.<ES_NAME>.0.*nnod*	Element EFIL Directory	√	
INT.FORC.*i.j*†	System Internal Force Vector		√
INV.<KNAME>.*i.j*†	Factored System Matrix		√
JDF1.BTAB.1.8	Model Summary	√	
JLOC.BTAB.2.5	Nodal Coordinates	√	
KMAP..ic2.isize	Model Connectivity Map		√
<KNAME>.SPAR.jdf2	Assembled System Matrix		√
PROP.BTAB.*	Material/Section Properties	√	
QJJT.BTAB.2.9	Nodal Transformations	√	
STAT.DISP.*i.j*†	System Displacement Vector		√
STAT.REAC.*i.j*†	System Reaction Force Vector		√
STRS.<ES_NAME>.*i.j*†	Element Stresses		√

† i = <load_set> and j = <cons_set>

Table 3.5-2 Sub-Procedures Invoked by procedure L_STATIC		
Procedure	*Type*	*Function*
ES	External	Element utility procedure
FACTOR	External	Factors assembled stiffness matrix
INT_FORCE	External	Computes internal forces
L_STATIC	Internal	Main procedure
SOLVE	External	Solves for the displacement solution
STRESS	External	Computes element stresses

Table 3.5-3 Processors Invoked by procedure L_STATIC		
Procedure	*Type*	*Function*
E	Internal	Initializes EFIL datasets
ES*i*	External	Element processors based on GEP
K	Internal	Assemble material stiffness matrix
RSEQ	Internal	Resequences nodes for equation solving
TOPO	Internal	Generates nodal topology maps
VPRT	Internal	Print SYSVEC-format vectors

3.5.5 ARGUMENT DESCRIPTION

3.5.5.1 CONS_SET

Constraint set number (default: 1). This argument selects which constraint set to use in solving the linear system of equations.

3.5.5.2 DIRECTION

Direction for the element stress (stress resultant) output (default: 1). The element stress coordinate system will be used if DIRECTION=0. The material axes (x_m, y_m, z_m) will be used if DIRECTION=1; the material axes (y_m, z_m, x_m) will be used for DIRECTION=2; and the material axes (z_m, x_m, y_m) will be used for DIRECTION=3. For isotropic materials, the first material axis is replaced by the corresponding global axis (see Section 4.3.3.9 of the CSM Testbed User's Manual, ref. 3.5-1).

3.5.5.3 FUNCTION

Select function to be performed by procedure L_STATIC (default: ALL). This procedure may be used to perform three functions. For FUNCTION=ALL, the element data are initialized and elemental stiffness matrices formed; nodal resequencing may be performed, the mesh topology is analyzed, the system stiffness matrix is assembled and factored, the displacement solution is obtained and optionally element stresses (stress resultants) and internal nodal forces (reactions) computed. For FUNCTION=FACT_SOLV, procedure L_STATIC assumes that the system stiffness matrix has previously been assembled and that nodal resequencing has been performed. The procedure then proceeds to factor the system stiffness matrix, solves for the displacement solution, and optionally computes the element stresses (stress resultants) and internal nodal forces (reactions). For FUNCTION=SOLV, procedure L_STATIC assumes that the system stiffness has previously been formed and factored. The procedure then proceeds to solve for the displacement solution and optionally computes the element stresses (stress resultants) and internal nodal forces (reactions). Using the FUNCTION argument, the user may solve for a variety of constraint (boundary conditions) sets on a given model subjected to a variety of loading conditions.

3.5.5.4 ISEQ

Resequencing method to be used (default: -1). If the argument RENUMBER is <true>, then nodal resequencing will be performed using processor RSEQ. The method used by processor RSEQ to resequence the nodes depends on the value of ISEQ. If the argument ISEQ is greater than or equal to zero, then that method will be used (*i.e.*, method=0,1,2,3; see Section 6.1 of the CSM Testbed User's Manual, ref. 3.5-1). If the argument ISEQ has a value of -1, then a default method will be used depending on the value of the global macrosymbol <solver_name>.

These default value are as follows:

<solver_name>	ISEQ
INV	0
BAND	2
ITER	1
SPK	1

3.5.5.5 KNAME

First word of the dataset name containing the assembled stiffness matrix (default: K).

3.5.5.6 LDI

Logical device index (default: 1).

3.5.5.7 LOAD_SET

Load set number (default: 1). This argument selects which load set to use as a right-hand side vector.

3.5.5.8 LOCATION

Location of the evaluation points for the element stresses or stress resultants (default: NODES). The element stresses or stress resultants are optionally computed by calling procedure STRESS (see Section 6.4). This argument may have four values. For LOCATION=INTEG_PTS, the element stresses are computed at the element integration points. For LOCATION=CENTROIDS, the element stresses are computed at the element centroid. For LOCATION=NODES, the element stresses are extrapolated from the integration points to be element nodes. These element nodal stresses will be discontinuous across interelement boundaries. For LOCATION=ALL, the element stresses are computed at the element integration points, element centroid, and element nodes.

3.5.5.9 NVAL_METH

Select method to be used for computing the smoothed global stresses (default: 3). Processor NVAL is used to compute the smoothed global stresses using the method defined by the argument NVAL_METH (see Section 12.5 of reference 6.4-1). If NVAL_METH=1, a topological interpolation of the element centroidal stresses is performed, and the element stresses must have been computed using LOCATION=CENTROIDS. If NVAL_METH=2, a projected least-squares interpolation of the element centroidal stresses is performed, and the stresses must have been computed using LOCATION=CENTROIDS. If NVAL_METH=3, the element nodal stresses (discontinuous across interelement boundaries) are averaged, and the element stresses must have been computed using LOCATION=NODES. Using LOCATION=ALL will generate element stresses at the element centroids, element nodes, and element gauss points. Acceptable values of LOCATION for specific values of NVAL_METH are as follows:

NVAL_METH	LOCATION
1	CENTROIDS, ALL
2	CENTROIDS, ALL
3	NODES, ALL

3.5.5.10 PRINT

Flag to print computed results such as the displacement solution, internal forces, and element and nodal stresses (default: <false>). If printing of these computed results is requested, processor VPRT will be used to print the displacement solution and internal forces, processor PESR will be used to print the element stresses and processor PNSR will be used to print nodal stresses.

3.5.5.11 REACTION

Flag to compute the internal nodal forces (default: <true>) If the argument REACTION=<true>, then the internal forces will be computed by calling procedure INT_FORCE (see Section 6.2).

3.5.5.12 RENUMBER

Flag to resequence node numbers prior to equation solving (default: <true>). If the argument RENUMBER=<true>, then processor RSEQ will be used to perform nodal resequencing, otherwise no resequencing will be performed. Note that the nodal resequencing may greatly reduce the time required to factor and solve the linear system of equations given by equation 3.5-1.

3.5.5.13 RHS

First two words of the dataset name for the right-hand side system vector (default: APPL.FORC).

3.5.5.14 SMOOTH

Flag to compute smoothed global stresses (default: <false>). If the argument SMOOTH=<true>, then smoothed global stresses will be computed by processor NVAL using the method defined by the argument NVAL_METH.

3.5.5.15 SOLN

First two words of the dataset name for the displacement solution (default: STAT.DISP).

3.5.5.16 STRESS

Flag to compute element stresses or stress resultants (default: <false>). If the argument STRESS is defined to be <true>, then the element stresses will be computed at the location and in the direction specified by the arguments LOCATION and DIRECTION, respectively, by calling procedure STRESS (see Section 6.4).

3.5.6 PROCEDURE FLOWCHART

L_STATIC	(main procedure)
ES	(initialize, form K)
FACTOR	(factor assembled stiffness matrix)
SOLVE	(solve linear system of equations)
STRESS	(stress/strain recovery procedure)
ES	(calculate element stresses and/or strains)
INT_FORCE	(internal force procedure)
ES	(internal force calculation)

3.5.7 LIMITATIONS

Procedure L_STATIC assumes that all datasets either required or generated reside on library one (LDI=1).

3.5.8 ERROR MESSAGES AND WARNINGS

None.

3.5.9 USAGE GUIDELINES AND EXAMPLES

Procedure L_STATIC may be used by preceding the procedure name by the *call directive. Procedure arguments may be changed from their default values by including any or all the arguments and their new values when the procedure is called. A space or blank is required between the end of the procedure name and the left parenthesis.

```
*call L_STATIC ( FUNCTION = ALL ; -- . Select function
              CONS_SET  =  1  ; -- . Select constraint set
              DIRECTION =  0  ; -- . Select direction for element stresses
              ISEQ      = -1  ; -- . Select resequencing method
              KNAME     =  K  ; -- . First word of stiffness matrix
                                -- . dataset name
              LOAD_SET  =  1  ; -- . Select load set
              LOCATION  =  CENTROIDS ; -- Select location of element
                                       -- . stress evaluation points
              PRINT     =  <true> ;  -- . PRINT flag
              REACTION  =  <false> ; -- . REACTIONS flag
              RENUMBER  =  <true> ;  -- . RESEQUENCING flag
              RHS       =  APPL. FORC ; -- . First two words of RHS
                                        -- . vector dataset
              SOLN      =  STAT. DISP ; -- . First two words of SOLN
                                        -- . dataset
              STRESS    =  <false> -- . STRESS flag
                                     )
```

Before procedure L_STATIC is called, the global macrosymbol solver_name should be defined as described in Section 3.5.3; otherwise, the default value of INV will be used.

3.5.10 PROCEDURE LISTING

3.5.11 REFERENCES

3.5-1 Stewart, Caroline B.: *The Computational Structural Mechanics Testbed User's Manual.* NASA TM-100644, October 1989.

3.5-2 Whetstone, W. D.: "Computer Analysis of Large Linear Frames": *Journal of the Structural Division,* ASCE, Vol. 95, No. ST11, November 1969, pp. 2401-2417.

3.5-3 Regelbrugge, M. E. and Wright, M. A.: *The Computational Structural Mechanics Testbed Matrix Processors Internal Logic and Dataflow Descriptions.* NASA CR-181742, March 1989.

3.5-4 Poole, Eugene L. and Overman, Andrea L.: *The Solution of Linear Systems of Equations with a Structural Analysis Code of the NAS Cray-2.* NASA CR-4159, September 1988.

3.5-5 Chu, Eleanor and George, J. Alan: *Sparse Matrix Methods Research Using the CSM Testbed Software System.* NASA CR-4219, March 1989.

3.5-6 Chu, E.C.H.; George, J. A.; Liu, W-H.; and Ng, E. G-Y.: *User's Guide for SPARSPAK-A: Waterloo Sparse Linear Equations Package.* Technical Report CS-84-36, University of Waterloo, Waterloo Ontario, Canada, 1984.

3.5-7 Stewart, Caroline B.: *The Computational Structural Mechanics Testbed Data Library Description.* NASA TM-100645, October 1988.

THIS PAGE LEFT BLANK INTENTIONALLY.

3.6 Procedure L_VIBRAT_0

3.6.1 GENERAL DESCRIPTION

Procedure L_VIBRAT_0 performs a linear vibration analysis about an unstressed state. The eigensolver is selected by defining the global macrosymbol **eigensolver_name** to be the name of the desired processor (*e.g.*, EIG2, LAN, LANZ), and the structural element processors (ES*i*) form the elemental stiffness and mass matrices. The procedure assumes that the finite element model and boundary conditions have already been generated and that the vibration modeshapes and frequencies need to be calculated.

3.6.2 THEORY

Linear vibration analyses are formulated using the equations of motion for an undamped structure. For the case of no external forces, the equations of motion are

$$\mathbf{M}\ddot{\mathbf{u}} + \mathbf{K}\mathbf{u} = 0 \qquad (3.6-1)$$

where

$\mathbf{M}$ = assembled mass matix (consistent or diagonal)
$\mathbf{K}$ = assembled linear stiffness matrix
$\ddot{\mathbf{u}}$ = acceleration vector
$\mathbf{u}$ = displacement vector

If harmonic motion is assumed, then

$$\mathbf{u} = \phi_i \, sin \, \omega_i t \qquad i = 1, 2, \ldots \qquad (3.6-2)$$

where

ϕ_i = ith eigen modeshape
ω_i = ith circular frequency (radians per second)

The ith cyclic frequency f_i (in hertz) is

$$f_i = \frac{\omega_i}{2\pi}$$

and the period T_i (in seconds) is

$$T_i = \frac{1}{f_i}$$

Substituting equation (3.6-2) in equation (3.6-1) gives

$$(\mathbf{K} - \lambda_i \mathbf{M})\phi_i = 0 \qquad i = 1, 2, \ldots \qquad (3.6-3)$$

where

$$\lambda_i = \omega_i^2.$$

If the user has specified the reset parameter **GRAV** in processor LAU to a value of unity then the matrix **M** has the meaning of a "weight" matrix instead of a "mass" matrix. This reset parameter is important for interpreting the eigenvalues calculated by the various eigensolvers.

3.6.3 ALGORITHM

The algorithm used to solve equation (3.6-1) depends on the value of the global macrosymbol **eigensolver_name**. Processor EIG2 is used if **eigensolver_name** is defined to be **EIG2**. This processor uses a nodal-block sparse matrix approach as described in reference 3.6-2. Processor LAN is used if **eigensolver_name** is defined to be **LAN**. Processor LANZ is used if **eigensolver_name** is defined to be **LANZ**. These processors are based on the Lanczos algorithm as described in references 3.6-2 and 3.6-3. If this global macrosymbol is not defined, procedure **L_VIBRAT_0** will set it to **EIG2**.

3.6.4 PROCEDURE USAGE

Procedure **L_VIBRAT_0** may be invoked by the ***call** directive, and following it by a list of arguments separated by semicolons(;) and enclosed in parentheses. Procedure arguments are order-independent, and most have default values thus making them optional. The formal syntax is as follows:

```
*call L_VIBRAT_0 ( arg1 = val1; arg2 = val2; ...)
```

where **arg*i*** are argument names and **val*i*** are the corresponding values. The following are valid arguments for procedure **L_VIBRAT_0**; note that those arguments without default values are mandatory, while the others are optional.

Argument	Default Value	Meaning
ERROR_TOL	.0001	Convergence criterion for eigenvalues
FUNCTION	ALL	Select function to be performed by procedure
INIT_VECTORS	0	Number of initial vectors used to span the subspace
ISEQ	0	Resequencing method to be used
LDI	1	Logical device index
KNAME	K	First word of the name of the dataset containing the assembled stiffness matrix
MASS_TYPE	CONSISTENT	Type of mass matrix
MAX_ITERS	20	Maximum number of iterations
N_MODES	1	Number of eigenvalues to converge
PRINT	<false>	Flag to print eigenvectors
RENUMBER	<true>	Flag to resequence node numbers for equation solver
SHIFT	0.0	Eigenvalue shift
VCON_SET	1	Constraint set for vibration analysis

Tables 3.6-1, 3.6-2, and 3.6-3 list the datasets used or created by procedure L_VIBRAT_0, the procedures invoked by procedure L_VIBRAT_0, and the processors invoked by procedure L_VIBRAT_0, respectively.

Table 3.6-1 Datasets Input/Output by procedure L_VIBRAT_0

Dataset	*Description*	*Input*	*Output*
AMAP..ic2.isize	Factorization Map for INV		√
CEM.SPAR	Consistent Mass Matrix		√
<ES_NAME>.EFIL.0.*nnod*	Element Computational Data	√	√
ES.SUMMARY	ES Processor Status	√	√
DEF.<ES_NAME>.0.*nnod*	Element Defn. (Connectivity)	√	
DEM.DIAG	Diagonal (Lumped) Mass Matrix		√
DIR.<ES_NAME>.0.*nnod*	Element EFIL Directory	√	
INV.KSHF.*j*†	Factored Shifted System Matrix		√
INV.<KNAME>.*j*†	Factored System Matrix		√
JDF1.BTAB.1.8	Model Summary	√	
KG.SPAR.jdf2	Assembled geometric stiffness matrix		√
KMAP..ic2.isize	Model Connectivity Map		√
<KNAME>.SPAR.jdf2	Assembled system matrix		√
VIBR.EVAL.1.*j*†	Vibration eigenvalues		√
VIBR.MODE.1.*j*†	Vibration eigenvalues		√

† j = <cons_set>

Table 3.6-3 Processors Invoked by Procedure L_VIBRAT_0		
Procedure	*Type*	*Function*
AUS	Internal	Arithmetic utilities
E	Internal	Initializes EFIL datasets
EIG2	Internal	Solve eigenvalue problem using subspace iteration
ES*i*	External	Element processors based on GEP
K	Internal	Assemble system matrix
LAN	External	Solve eigenvalue problem using Lanczos method
LANZ	External	Solve eigenvalue problem using Lanczos method
RSEQ	Internal	Resequences nodes for equation solving
TOPO	Internal	Generates nodal topology maps
VPRT	Internal	Print SYSVEC system vectors

3.6.5 ARGUMENT DESCRIPTION

3.6.5.1 ERROR_TOL

Convergence criterion for eigenvalues (default: `0.0001`). For the k-th iteration, the error measure for the i-th eigenvalue is

$$\epsilon_i^k = \frac{\|(\lambda_i^k - \lambda_i^{k-1})\|}{\|\lambda_i^k\|}$$

The i-th eigenvalue is converged if ϵ_i^k is smaller than `ERROR_TOL`.

3.6.5.2 FUNCTION

Select function to be performed by procedure `L_VIBRAT_0` (default: `ALL`). This procedure may be used to perform two functions. For `FUNCTION=ALL`, the element data are initialized and elemental stiffness matrices formed; nodal resequencing may be performed, the mesh topology is analyzed, the system stiffness matrix is assembled and factored, and the eigenproblem is solved. For `FUNCTION=EIGEN`, procedure `L_VIBRAT_0` uses a previously computed prestress state in solving the eigenvalue problem. Using the `FUNCTION` argument, the user may solve for a variety of constraint (boundary conditions) sets on a given model.

3.6.5.3 INIT_VECTOR

Number of initial vectors used to span the subspace (default: 0). This argument defines the number of trial vectors used to initiate the subspace iteration. If INIT_VECTOR=0, the number of initial vectors will be calculated by the procedure as

```
INIT_VECTOR=MINIMUM (2*N_MODES, N_modes + 8)
```

3.6.5.4 ISEQ

Resequencing method to be used (default: 0). If the argument RENUMBER is <true>, then nodal resequencing will be performed using processor RSEQ. The method used by processor RSEQ to resequence the nodes depends on the value of ISEQ. If the argument ISEQ is greater than or equal to zero, then that method will be used (*i.e.*, method=0,1,2,3; see Section 6.1 of the CSM Testbed User's Manual, ref. 3.6-1).

3.6.5.5 LDI

Logical device index (default: 1).

3.6.5.6 KNAME

First word of the dataset name containing the assembled stiffness matrix (default: K).

3.6.5.7 MASS_TYPE

Type of mass matrix (default: CONSISTENT). If MASS_TYPE=CONSISTENT, the element processor will generate consistent element mass matrices that will be assembled by processor K to form the system mass matrix. If MASS_TYPE=DIAGONAL, the element processor will generate a diagonal or lumped mass matrix.

3.6.5.8 MAX_ITERS

Maximum number of iterations (default: 20). This argument specifies the maximum number of iterations that can be used per call to eigensolver.

3.6.5.9 N_MODES

Number of converged eigenvalues desired (default: 1). This argument specifies the number of eigenvalues to calculate to a convergence criterion of ERROR_TOL.

3.6.5.10 PRINT

Flag to print displacement solution, internal forces and element stresses (default: <false>). If printing of these computed results is requested, processor VPRT will be used to print the vibration modeshapes.

3.6.5.11 RENUMBER

Flag to resequence node numbers prior to equation solving (default: <true>). If the argument RENUMBER=<true>, then processor RSEQ will be used to perform nodal resequencing, otherwise no resequencing will be performed. Note that the nodal resequencing may greatly reduce the time required to factor and solve the linear system of equations.

3.6.5.12 SHIFT

Eigenvalue shift (default: 0.0). Converged eigenvalue will only be obtained for eigenvalues greater than **SHIFT**. The shift parameter refers to the frequency squared (*i.e.*, ω^2) for vibration problems.

3.6.5.13 VCON_SET

Constraint set number for vibration analysis (default: 1). This argument selects which constraint set to use in solving the linear vibration problem.

3.6.6 PROCEDURE FLOWCHART

L_VIBRAT_0	(main procedure)
ES	(initialize, form **K** and **M**)
FACTOR	(factor using vibration boundary conditions)

3.6.7 LIMITATIONS

None.

3.6.8 ERROR MESSAGES AND WARNINGS

None.

3.6.9 USAGE GUIDELINES AND EXAMPLES

Procedure L_VIBRAT_0 may be used by preceding the procedure name by the *call directive. Procedure arguments may be changed from their default values by including any or all the arguments and their new values when the procedure is called. A space or blank is required between the end of the procedure name and the left parenthesis.

```
*call L_VIBRAT_0 ( FUNCTION = ALL ; -- . Select function
                   BCON_SET  = 1   ; -- . Select vibration constraint set
                   ERROR_TOL = .0001 ; -- . Eigenvalue convergence criterion
                   INIT_VECTOR = 0 ; -- . Number of initial vectors
                   ISEQ      = -1  ; -- . Select resequencing method
                   KNAME     =  K  ; -- . First word of stiffness matrix
                                     -- . dataset name
                   MAX_ITERS = 20;  -- Maximum number of iterations
                   N_MODES   =  1 ;  -- Number of eigenvalues
                   PRINT     =  <true> ;  -- . PRINT flag
                   RENUMBER  =  <true> ;  -- . RESEQUENCING flag
                   SHIFT     =  0.0       -- . Eigenvalue shift
                                        )
```

Before procedure L_VIBRAT_0 is called, the global macrosymbol **eigensolver_name** should be defined as described in Section 3.6-3; otherwise, the default value of EIG2 will be used.

3.6.10 PROCEDURE LISTING

3.6.11 REFERENCES

3.6-1 Stewart, Caroline B.: *The Computational Structural Mechanics Testbed User's Manual.* NASA TM-100644, October 1989.

3.6-2 Bostic, S. W. and Fulton R. E.: *A Lanczos Eigenvalue Method on a Parallel Computer.* AIAA Paper No. 87-0725-CP.

3.6-3 Jones, Mark T. and Patrick, Merrell L.: *The Use of Lanczos's Method to Solve the Large Generalized Symmetric Definite Eigenvalue Problem.* NASA CR-181914, September 1989. (Also available as ICASE Report No. 89-69).

THIS PAGE LEFT BLANK INTENTIONALLY.

3.7 Procedure L_VIBRAT_1

3.7.1 GENERAL DESCRIPTION

Procedure L_VIBRAT_1 performs a linear vibration analysis about a prescribed prestressed state. The eigensolver is selected by defining the global macrosymbol **eigensolver_name** to be the name of the desired processor (*e.g.*, EIG2, LAN), and the structural element processors (ES*i*) form the elemental stiffness and mass matrices. The procedure assumes that the finite element model and boundary conditions have already been generated, that the prestressed state has been prescribed, and that the vibration modeshapes and frequencies need to be calculated.

3.7.2 THEORY

Linear vibration analyses are formulated using the equations of motion for an undamped structure. For the case of prestressed state, the equations of motion are

$$\mathbf{M}\ddot{\mathbf{u}} + \mathbf{K}\mathbf{u} + \mathbf{K}_g(\sigma)\mathbf{u} = 0 \qquad (3.7-1)$$

where

$\mathbf{M}$ = assembled mass matix (consistent or diagonal)
$\mathbf{K}$ = assembled linear stiffness matrix
$\mathbf{K}_g(\sigma)$ = assembled geometric stiffness matrix for given stress state
$\ddot{\mathbf{u}}$ = acceleration vector
$\mathbf{u}$ = displacement vector
σ = prestress state

The prestressed state may be defined in several ways. Procedure L_VIBRAT_1 assumes that a membrane prestressed state (N_x^o, N_y^o, N_{xy}^o) is explicitly prescribed.

If harmonic motion is assumed, then

$$\mathbf{u} = \phi_i \, sin \, \omega_i t \qquad i = 1, 2, \ldots \qquad (3.7-2)$$

where

ϕ_i = ith eigen modeshape
ω_i = ith circular frequency (radians per second)

The ith cyclic frequency f_i (in hertz) is

$$f_i = \frac{\omega_i}{2\pi}$$

and the period T_i (in seconds) is

$$T_i = \frac{1}{f_i}$$

Substituting equation (3.7-2) in equation (3.7-1) gives

$$(\mathbf{K} + \mathbf{K}_g(\sigma) - \lambda_i \mathbf{M})\phi_i = 0 \qquad i = 1, 2, \ldots \qquad (3.7-3)$$

where

$$\lambda_i = \omega_i^2.$$

If the user has specified the reset parameter **GRAV** in processor LAU to a value of unity then the matrix **M** has the meaning of a "weight" matrix instead of a "mass" matrix. This reset parameter is important for interpreting the eigenvalues calculated by the various eigensolvers.

3.7.3 ALGORITHM

The algorithm used to solve equation (3.7-1) depends on the value of the global macrosymbol **eigensolver_name**. Processor EIG2 is used if **eigensolver_name** is defined to be **EIG2**. This processor uses a nodal-block sparse matrix approach as described in reference 3.7-2. Processor LAN is used if **eigensolver_name** is defined to be **LAN**. Processor LANZ is used if **eigensolver_name** is defined to be **LANZ**. These processors are based on the Lanczos algorithm as described in references 3.7-2 to 3.7-3. If this global macrosymbol is not defined, procedure **L_VIBRAT_1** will set it to **EIG2**.

3.7.4 PROCEDURE USAGE

Procedure **L_VIBRAT_1** may be invoked by the ***call** directive, and following it by a list of arguments separated by semicolons(;) and enclosed in parentheses. Procedure arguments are order-independent, and most have default values thus making them optional. The formal syntax is as follows:

```
*call L_VIBRAT_1 ( arg1 = val1; arg2 = val2; ...)
```

where **arg***i* are argument names and **val***i* are the corresponding values. The following are valid arguments for procedure **L_VIBRAT_1**; note that those arguments without default values are mandatory, while the others are optional.

Argument	Default Value	Meaning
PS_1	--	Prescribed membrane stres resultant N_x^o
PS_2	--	Prescribed membrane stres resultant N_y^o
PS_3	--	Prescribed membrane stres resultant N_{xy}^o
ERROR_TOL	.0001	Convergence criterion for eigenvalues
FUNCTION	ALL	Select function to be performed by procedure
INIT_VECTORS	0	Number of initial vectors used to span the subspace
ISEQ	0	Resequencing method to be used
LDI	1	Logical device index
KNAME	K	First word of the name of the dataset containing the assembled stiffness matrix
MASS_TYPE	CONSISTENT	Type of mass matrix
MAX_ITERS	20	Maximum number of iterations
N_GROUPS	1	
N_MODES	1	Number of eigenvalues to converge
PRINT	<false>	Flag to print eigenvectors
RENUMBER	<true>	Flag to resequence node numbers for equation solver
SHIFT	0.0	Eigenvalue shift
VCON_SET	1	Constraint set for vibration analysis

Tables 3.7-1, 3.7-2, and 3.7-3 list the datasets used or created by procedure L_VIBRAT_1, the procedures invoked by procedure L_VIBRAT_1, and the processors invoked by procedure L_VIBRAT_1, respectively.

Table 3.7-1 Datasets Input/Output by procedure L_VIBRAT_1			
Dataset	*Description*	*Input*	*Output*
AMAP..ic2.isize	Factorization Map for INV		√
CEM.SPAR	Consistent Mass Matrix		√
<ES_NAME>.EFIL.0.*nnod*	Element Computational Data	√	√
ES.SUMMARY	ES Processor Status	√	√
DEF.<ES_NAME>.0.*nnod*	Element Defn. (Connectivity)	√	
DEM.DIAG	Diagonal (Lumped) Mass Matrix		√
DIR.<ES_NAME>.0.*nnod*	Element EFIL Directory	√	
INV.KSHF.*j*†	Factored Shifted System Matrix		√
INV.<KNAME>.*j*†	Factored System Matrix		√
JDF1.BTAB.1.8	Model Summary	√	
KG.SPAR.jdf2	Assembled geometric stiffness matrix		√
KMAP..ic2.isize	Model Connectivity Map		√
<KNAME>.SPAR.jdf2	Assembled system matrix		√
VIBR.EVAL.1.*j*†	Vibration eigenvalues		√
VIBR.MODE.1.*j*†	Vibration eigenvalues		√

† j = <vcon_set>

Table 3.7-2 Sub-Procedures Invoked by procedure L_VIBRAT_1		
Procedure	*Type*	*Function*
ES	External	Element utility procedure
FACTOR	External	Factors assembled stiffness matrix
L_VIBRAT_1	Internal	Main procedure

Table 3.7-3 Processors Invoked by Procedure L_VIBRAT_1		
Procedure	*Type*	*Function*
AUS	Internal	Arithmetic utilities
E	Internal	Initializes EFIL datasets
EIG2	Internal	Solve eigenvalue problem using subspace iteration
ES*i*	External	Element processors based on GEP
K	Internal	Assemble system matrix
LAN	External	Solve eigenvalue problem using Lanczos method
LANZ	External	Solve eigenvalue problem using Lanczos method
RSEQ	Internal	Resequences nodes for equation solving
TOPO	Internal	Generates nodal topology maps
VPRT	Internal	Print SYSVEC system vectors

3.7.5 ARGUMENT DESCRIPTION

3.7.5.1 ERROR_TOL

Convergence criterion for eigenvalues (default: `0.0001`). For the k-th iteration, the error measure for the i-th eigenvalue is

$$\epsilon_i^k = \frac{\|(\lambda_i^k - \lambda_i^{k-1})\|}{\|\lambda_i^k\|}$$

The i-th eigenvalue is converged if ϵ_i^k is smaller than **ERROR_TOL**.

3.7.5.2 FUNCTION

Select function to be performed by procedure **L_VIBRAT_1** (default: **ALL**). This procedure may be used to perform two functions. For **FUNCTION=ALL**, the element data are initialized and elemental stiffness matrices formed; nodal resequencing may be performed, the mesh topology is analyzed, the system stiffness matrix is assembled and factored, and the eigenproblem is solved. For **FUNCTION=EIGEN**, procedure **L_VIBRAT_1** uses a prescribed prestress state in solving the eigenvalue problem. Using the **FUNCTION** argument, the user may solve for a variety of constraint (boundary conditions) sets on a given model.

3.7.5.3 INIT_VECTOR

Number of initial vectors used to span the subspace (default: 0). This argument defines the number of trial vectors used to initiate the subspace iteration. If INIT_VECTOR=0, the number of initial vectors will be calculated by the procedure as

```
INIT_VECTOR=MINIMUM (2*N_MODES, N_modes + 8)
```

3.7.5.4 ISEQ

Resequencing method to be used (default: 0). If the argument RENUMBER is <true>, then nodal resequencing will be performed using processor RSEQ. The method used by processor RSEQ to resequence the nodes depends on the value of ISEQ. If the argument ISEQ is greater than or equal to zero, then that method will be used (*i.e.*, method=0,1,2,3; see Section 6.1 of the CSM Testbed User's Manual, ref. 3.7-1).

3.7.5.5 LDI

Logical device index (default: 1).

3.7.5.6 KNAME

First word of the dataset name containing the assembled stiffness matrix (default: K).

3.7.5.7 MASS_TYPE

Type of mass matrix (default: CONSISTENT). If MASS_TYPE=CONSISTENT, the element processor will generate consistent element mass matrices that will be assembled by processor K to form the system mass matrix. If MASS_TYPE=DIAGONAL, the element processor will generate a diagonal or lumped mass matrix.

3.7.5.8 N_GROUPS

Element group number (default: 1).

3.7.5.9 MAX_ITERS

Maximum number of iterations (default: 20). This argument specifies the maximum number of iterations that can be used per call to eigensolver.

3.7.5.10 N_MODES

Number of converged eigenvalues desired (default: 1). This argument specifies the number of eigenvalues to calculate to a convergence criterion of ERROR_TOL.

3.7.5.11 PRINT

Flag to print modeshapes (default: <false>). If printing of these computed results is requested, processor VPRT will be used to print the vibration modeshapes.

3.7.5.12 PS_1

Prescribed membrane stress resultant N_x^o for the prestressed state.

3.7.5.13 PS_2

Prescribed membrane stress resultant N_y^o for the prestressed state.

3.7.5.14 PS_3

Prescribed membrane stress resultant N_{xy}^o for the prestressed state.

3.7.5.15 RENUMBER

Flag to resequence node numbers prior to equation solving (default: <true>). If the argument RENUMBER=<true>, then processor RSEQ will be used to perform nodal resequencing, otherwise no resequencing will be performed. Note that the nodal resequencing may greatly reduce the time required to factor and solve the linear system of equations.

3.7.5.16 SHIFT

Eigenvalue shift (default: 0.0). Converged eigenvalue will only be obtained for eigenvalues greater than SHIFT. The shift parameter refers to the frequency squared (*i.e.*, ω^2) for vibration problems.

3.7.5.17 VCON_SET

Constraint set number for vibration analysis (default: 1). This argument selects which constraint set to use in solving the linear vibration problem.

3.7.6 PROCEDURE FLOWCHART

L_VIBRAT_1	(main procedure)
ES	(initialize, form K and M)
ES	(form **Kg**)
FACTOR	(factor using vibration boundary conditions)

3.7.7 LIMITATIONS

None.

3.7.8 ERROR MESSAGES AND WARNINGS

None.

3.7.9 USAGE GUIDELINES AND EXAMPLES

Procedure **L_VIBRAT_1** may be used by preceding the procedure name by the ***call** directive. Procedure arguments may be changed from their default values by including any or all the arguments and their new values when the procedure is called. A space or blank is required between the end of the procedure name and the left parenthesis.

```
*call L_VIBRAT_1 ( FUNCTION = ALL ; -- . Select function
                   VCON_SET  = 1  ; -- . Select vibration constraint set
                   ERROR_TOL = .0001 ; -- . Eigenvalue convergence criterion
                   INIT_VECTOR = 0 ; -- . Number of initial vectors
                   ISEQ      =  0  ; -- . Select resequencing method
                   KNAME     =  K  ; -- . First word of stiffness matrix
                                     -- . dataset name
                   MAX_ITERS = 20;  -- Maximum number of iterations
                   N_MODES   =  1 ;  -- Number of eigenvalues
                   N_GROUPS  =  1 ;  --
                   PS_1          ;  --
                   PS_2          ;  --
                   PS_3          ;  --
                   PRINT     =  <true> ;  -- . PRINT flag
                   RENUMBER  =  <true> ;  -- . RESEQUENCING flag
                   SHIFT     =  0.0 ; Eigenvalue shift
                                      )
```

Before procedure **L_VIBRAT_1** is called, the global macrosymbol **eigensolver_name** should be defined as described in Section 3.7-3; otherwise, the default value of **EIG2** will be used.

3.7.10 PROCEDURE LISTING

3.7.11 REFERENCES

3.7-1 Stewart, Caroline B.: *The Computational Structural Mechanics Testbed User's Manual.* NASA TM-100644, October 1989.

3.7-2 Bostic, S. W. and Fulton R. E.: *A Lanczos Eigenvalue Method on a Parallel Computer.* AIAA Paper No. 87-0725-CP.

3.7-3 Jones, Mark T. and Patrick, Merrell L.: *The Use of Lanczos's Method to Solve the Large Generalized Symmetric Definite Eigenvalue Problem.* NASA CR-181914, September 1989. (Also available as ICASE Report No. 89-69).

3.8 Procedure L_VIBRAT_2

3.8.1 GENERAL DESCRIPTION

Procedure **L_VIBRAT_2** performs a linear vibration analysis about a prescribed prestressed state. The eigensolver is selected by defining the global macrosymbol **eigensolver_name** to be the name of the desired processor (*e.g.*, EIG2, LAN), and the structural element processors (ES*i*) form the elemental stiffness and mass matrices. The procedure assumes that the finite element model and boundary conditions have already been generated, that the prestressed state has been prescribed, and that the vibration modeshapes and frequencies need to be calculated.

3.8.2 THEORY

Linear vibration analyses are formulated using the equations of motion for an undamped structure. For the case of prestressed state, the equations of motion are

$$\mathbf{M}\ddot{\mathbf{u}} + \mathbf{K}\mathbf{u} + \mathbf{K}_g(\sigma)\mathbf{u} = 0 \qquad (3.8-1)$$

where

$\mathbf{M}$ = assembled mass matix (consistent or diagonal)
$\mathbf{K}$ = assembled linear stiffness matrix
$\mathbf{K}_g(\sigma)$ = assembled geometric stiffness matrix for given stress state
$\ddot{\mathbf{u}}$ = acceleration vector
$\mathbf{u}$ = displacement vector
σ = prestress state

The prestressed state may be defined in several ways. Procedure **L_VIBRAT_2** assumes that the prestressed state will be first calculated using procedure **L_STATIC**.

If harmonic motion is assumed, then

$$\mathbf{u} = \phi_i \; sin \; \omega_i t \qquad i = 1,2,\ldots \qquad (3.8-2)$$

where

ϕ_i = ith eigen modeshape
ω_i = ith circular frequency (radians per second)

The ith cyclic frequency f_i (in hertz) is

$$f_i = \frac{\omega_i}{2\pi}$$

and the period T_i (in seconds) is

$$T_i = \frac{1}{f_i}$$

Substituting equation (3.8-2) in equation (3.8-1) gives

$$(\mathbf{K} + \mathbf{K}_g(\sigma) - \lambda_i \mathbf{M})\phi_i = 0 \qquad i = 1, 2, \ldots \qquad (3.8-3)$$

where

$$\lambda_i = \omega_i^2.$$

If the user has specified the reset parameter **GRAV** in processor LAU to a value of unity then the matrix M has the meaning of a "weight" matrix instead of a "mass" matrix. This reset parameter is important for interpreting the eigenvalues calculated by the various eigensolvers.

3.8.3 ALGORITHM

The algorithm used to solve equation (3.3-1) depends on the value of the global macrosymbol **eigensolver_name**. Processor EIG2 is used if **eigensolver_name** is defined to be **EIG2**. This processor uses a nodal-block sparse matrix approach as described in reference 3.3-2. Processor LAN is used if **eigensolver_name** is defined to be **LAN**. Processor LANZ is used if **eigensolver_name** is defined to be **LANZ**. These processors are based on the Lanczos algorithm as described in references 3.3-2 to 3.3-3. If this global macrosymbol is not defined, procedure **L_VIBRAT_2** will set it to **EIG2**.

3.8.4 PROCEDURE USAGE

Procedure **L_VIBRAT_2** may be invoked by the ***call** directive, and following it by a list of arguments separated by semicolons(;) and enclosed in parentheses. Procedure arguments are order-independent, and most have default values thus making them optional. The formal syntax is as follows:

```
*call L_VIBRAT_2 ( arg1 = val1; arg2 = val2; ...)
```

where **arg***i* are argument names and **val***i* are the corresponding values. The following are valid arguments for procedure **L_VIBRAT_2**; note that those arguments without default values are mandatory, while the others are optional.

Argument	Default Value	Meaning
CONS_SET	1	Constaint set number for prestress analysis
DIRECTION	0	Direction for element stress output
ERROR_TOL	.0001	Convergence criterion for eigenvalues
FUNCTION	ALL	Select function to be performed by procedure
INIT_VECTORS	0	Number of initial vectors used to span the subspace
ISEQ	0	Resequencing method to be used
LDI	1	Logical device index
KNAME	K	First word of the name of the dataset containing the assembled stiffness matrix
LOAD_SET	1	Load set number
MASS_TYPE	CONSISTENT	Type of mass matrix
MAX_ITERS	20	Maximum number of iterations
N_GROUPS	1	
N_MODES	1	Number of eigenvalues to converge
PRINT	<false>	Flag to print computed solutions
REACTION	<false>	Flag to compute internal forces or reactions
RENUMBER	<true>	Flag to resequence node numbers for equation solver
SHIFT	0.0	Eigenvalue shift
STRESS	<false>	Flag to compute element stresses (resultants)
VCON_SET	1	Constraint set for vibration analysis

Tables 3.8-1, 3.8-2, and 3.8-3 list the datasets used or created by procedure L_VIBRAT_2, the procedures invoked by procedure L_VIBRAT_2, and the processors invoked by procedure L_VIBRAT_2, respectively.

Table 3.8-1 Datasets Input/Output by procedure L_VIBRAT_2			
Dataset	*Description*	*Input*	*Output*
AMAP..ic2.isize	Factorization Map for INV		√
CEM.SPAR	Consistent Mass Matrix		√
<ES_NAME>.EFIL.0.*nnod*	Element Computational Data	√	√
ES.SUMMARY	ES Processor Status	√	√
DEF.<ES_NAME>.0.*nnod*	Element Defn. (Connectivity)	√	
DEM.DIAG	Diagonal (Lumped) Mass Matrix		√
DIR.<ES_NAME>.0.*nnod*	Element EFIL Directory	√	
INV.KSHF.*j*†	Factored Shifted System Matrix		√
INV.<KNAME>.*j*†	Factored System Matrix		√
JDF1.BTAB.1.8	Model Summary	√	
KG.SPAR.jdf2	Assembled geometric stiffness matrix		√
KMAP..ic2.isize	Model Connectivity Map		√
<KNAME>.SPAR.jdf2	Assembled system matrix		√
VIBR.EVAL.1.*j*†	Vibration eigenvalues		√
VIBR.MODE.1.*j*†	Vibration eigenvalues		√

† j = <vcon_set>

Table 3.8-2 Sub-Procedures Invoked by procedure L_VIBRAT_2		
Procedure	*Type*	*Function*
ES	External	Element utility procedure
FACTOR	External	Factors assembled stiffness matrix
L_STATIC	Internal	Static solution procedure
L_VIBRAT_2	Internal	Main procedure

Table 3.8-3 Processors Invoked by Procedure L_VIBRAT_2		
Procedure	*Type*	*Function*
AUS	Internal	Arithmetic utilities
E	Internal	Initializes EFIL datasets
EIG2	Internal	Solve eigenvalue problem using subspace iteration
ES*i*	External	Element processors based on GEP
K	Internal	Assemble system matrix
LAN	External	Solve eigenvalue problem using Lanczos method
LANZ	External	Solve eigenvalue problem using Lanczos method
RSEQ	Internal	Resequences nodes for equation solving
TOPO	Internal	Generates nodal topology maps
VPRT	Internal	Print SYSVEC system vectors

3.8.5 ARGUMENT DESCRIPTION

3.8.5.1 CONS_SET

Constraint set number for prestress solution (default: `1`). This argument selects which constraint set to use in solving the linear system of equations.

3.8.5.2 DIRECTION

Direction for the element stress (stress resultant) output (default: `0`). The element stress coordinate system will be used if `direction=0`. The material axes (x_m, y_m, z_m) will be used if `direction=1`; the material axes (y_m, z_m, x_m) will be used for `direction=2`; and the material axes (z_m, x_m, y_m) will be used for `direction=3`. For isotropic materials, the first material axis is replaced by the corresponding global axis (see Section 4.3.3.9 of the CSM Testbed User's Manual, ref. 3.4-2).

3.8.5.3 ERROR_TOL

Convergence criterion for eigenvalues (default: `0.0001`). For the k-th iteration, the error measure for the i-th eigenvalue is

$$\epsilon_i^k = \frac{\|(\lambda_i^k - \lambda_i^{k-1})\|}{\|\lambda_i^k\|}$$

The i-th eigenvalue is converged if ϵ_i^k is smaller than `ERROR_TOL`.

3.8.5.4 FUNCTION

Select function to be performed by procedure L_VIBRAT_2 (default: ALL). This procedure may be used to perform four functions. For FUNCTION=ALL, the element data are initialized and elemental stiffness matrices formed; nodal resequencing may be performed, the mesh topology is analyzed, the system stiffness matrix is assembled and factored, the displacement solution is obtained, optionally element stresses (stress resultants) and internal nodal forces (reactions) computed and the eigenproblem is solved. For FUNCTION=FACT_SOLV, procedure L_VIBRAT_2 assumes that the system stiffness matrix has previously been assembled and that nodal resequencing has been performed. The procedure then proceeds to factor the system stiffness matrix, solves for the displacement solution, optionally computes the element stresses (stress resultants) and internal nodal forces (reactions) and solves the eigenproblem. For FUNCTION=SOLV, procedure L_VIBRAT_2 assumes that the system stiffness has previously been formed and factored. The procedure then proceeds to solve for the displacement solution, optionally computes the element stresses (stress resultants) and internal nodal forces (reactions) and solves the eigenproblem. For FUNCTION=EIGEN, procedure L_VIBRAT_2 uses a previously computed prestress state in solving the eigenvalue problem. Using the FUNCTION argument, the user may solve for a variety of constraint (boundary conditions) sets on a given model subjected to a variety of loading conditions.

3.8.5.5 INIT_VECTOR

Number of initial vectors used to span the subspace (default: 0). This argument defines the number of trial vectors used to initiate the subspace iteration. If INIT_VECTOR=0, the number of initial vectors will be calculated by the procedure as

```
INIT_VECTOR=MINIMUM (2*N_MODES, N_modes + 8)
```

3.8.5.6 ISEQ

Resequencing method to be used (default: -1). If the argument RENUMBER is <true>, then nodal resequencing will be performed using processor RSEQ. The method used by processor RSEQ to resequence the nodes depends on the value of ISEQ. If the argument ISEQ is greater than or equal to zero, then that method will be used (*i.e.*, method=0,1,2,3; see Section 6.1 of the CSM Testbed User's Manual, ref. 3.8-1).

3.8.5.7 KNAME

First word of the dataset name containing the assembled stiffness matrix (default: K).

3.8.5.8 LDI

Logical device index (default: 1).

3.8.5.9 LOAD_SET

Load set number (default: 1). This argument selects which load set to use as a right-hand side vector.

3.8.5.10 MASS_TYPE

Type of mass matrix (default: CONSISTENT). If MASS_TYPE=CONSISTENT, the element processor will generate consistent element mass matrices that will be assembled by processor K to form the system mass matrix. If MASS_TYPE=DIAGONAL, the element processor will generate a diagonal or lumped mass matrix.

3.8.5.11 MAX_ITERS

Maximum number of iterations (default: 20). This argument specifies the maximum number of iterations that can be used per call to eigensolver.

3.8.5.12 N_MODES

Number of converged eigenvalues desired (default: 1). This argument specifies the number of eigenvalues to calculate to a convergence criterion of ERROR_TOL.

3.8.5.13 PRINT

Flag to print modeshapes (default: <false>). If printing of these computed results is requested, processor VPRT will be used to print the vibration modeshapes.

3.8.5.14 REACTION

Flag to compute the internal nodal forces (default: <true>) If the argument REACTION=<true>, then the internal forces will be computed by calling procedure INT_FORC (see Section 6.2).

3.8.5.15 RENUMBER

Flag to resequence node numbers prior to equation solving (default: <true>). If the argument RENUMBER=<true>, then processor RSEQ will be used to perform nodal resequencing, otherwise no resequencing will be performed. Note that the nodal resequencing may greatly reduce the time required to factor and solve the linear system of equations.

3.8.5.16 SHIFT

Eigenvalue shift (default: 0.0). Converged eigenvalue will only be obtained for eigenvalues greater than SHIFT. The shift parameter refers to the frequency squared (*i.e.*, ω^2) for vibration problems.

3.8.5.17 STRESS

Flag to compute element stresses or stress resultants (default: <false>). If the argument STRESS=<true>, then the element stresses will be computed at the location and in the direction specified by the arguments LOCATION and DIRECTION, respectively, by calling procedure STRESS (see Section 6.4).

3.8.5.18 VCON_SET

Constraint set number for vibration analysis (default: 1). This argument selects which constraint set to use in solving the linear vibration problem.

3.8.6 PROCEDURE FLOWCHART

L_VIBRAT_2	(main procedure)
L_STATIC	(linear static analysis procedure)
ES	(initialize, form **K** and **M**)
FACTOR	(factor assembled stiffness matrix)
SOLVE	(solve linear system of equations)
STRESS	(stress/strain recovery procedure)
ES	(calculate element and nodal stress/strain)
INT_FORCE	(internal force procedure)
ES	(internal force calculation)
ES	(form $\mathbf{K}_g$)
FACTOR	(factor using vibration boundary conditions)

3.8.7 LIMITATIONS

None.

3.8.8 ERROR MESSAGES AND WARNINGS

None.

3.8.9 USAGE GUIDELINES AND EXAMPLES

Procedure L_VIBRAT_2 may be used by preceding the procedure name by the *call directive. Procedure arguments may be changed from their default values by including any or all the arguments and their new values when the procedure is called. A space or blank is required between the end of the procedure name and the left parenthesis.

```
*call L_VIBRAT_2 ( FUNCTION = ALL ; -- . Select function
                   VCON_SET  = 1   ; -- . Select vibration constraint set
                   ERROR_TOL = .0001 ; -- . Eigenvalue convergence criterion
                   INIT_VECTOR = 0 ; -- . Number of initial vectors
                   ISEQ      = -1  ; -- . Select resequencing method
```

```
KNAME     = K  ; -- . First word of stiffness matrix
                  -- . dataset name
MAX_ITERS = 20;  -- Maximum number of iterations
N_MODES   = 1 ;  -- Number of eigenvalues
CONS_SET  = 1  ; -- . Select constraint set
DIRECTION = 0  ; -- . Select direction for element stresses
LOAD_SET  = 1  ; -- . Select load set
REACTION  = <false> ; -- . REACTIONS flag
STRESS    = <false> -- . STRESS flag
PRINT     = <true> ;  -- . PRINT flag
RENUMBER  = <true> ;  -- . RESEQUENCING flag
SHIFT     = 0.0 ; Eigenvalue shift
                          )
```

Before procedure L_VIBRAT_2 is called, the global macrosymbol **eigensolver_name** should be defined as described in Section 3.8-3; otherwise, the default value of **EIG2** will be used.

3.8.10 PROCEDURE LISTING

3.8.11 REFERENCES

3.8-1 Stewart, Caroline B.: *The Computational Structural Mechanics Testbed User's Manual.* NASA TM-100644, October 1989.

3.8-2 Bostic, S. W. and Fulton R. E.: *A Lanczos Eigenvalue Method on a Parallel Computer.* AIAA Paper No. 87-0725-CP.

3.8-3 Jones, Mark T. and Patrick, Merrell L.: *The Use of Lanczos's Method to Solve the Large Generalized Symmetric Definite Eigenvalue Problem.* NASA CR-181914, September 1989. (Also available as ICASE Report No. 89-69).

THIS PAGE LEFT BLANK INTENTIONALLY.

3.9 Procedure NL_STATIC_1

3.9.1 GENERAL DESCRIPTION

Procedure **NL_STATIC_1**, written by G. M. Stanley of Lockheed Palo Alto Research Laboratory, performs nonlinear static analysis based on a modified Newton/Raphson incremental strategy for automatic load-step control (*e.g.*, ref. 3.1). Procedure **NL_STATIC_1** relies on the Generic Element Processor (*i.e.*, structural element processors, ES*i*) and hence has a corotational option for geometric nonlinearity that enables arbitrarily large rotations.

Procedure **NL_STATIC_1** solves a nonlinear algebraic equation system of the form:

$$\mathbf{f}^{int}(\mathbf{d}) = \mathbf{f}^{ext}(\lambda) \qquad (3.9-1)$$

where $\mathbf{f}^{int}$ is the nonlinear internal force vector for the discrete (*i.e.*, finite element) system, $\mathbf{f}^{ext}$ is the external force vector, $\mathbf{d}$ is the displacement vector, and λ is a load factor. The basic modified Newton/Raphson procedure solves this system of equations by linearizing it at each load level (*i.e.*, fixed λ), leading to the solution of the following linear equations:

$$\begin{aligned} \mathbf{K}\,\delta\mathbf{d}_{n+1}^{(i+1)} &= \mathbf{r}(\mathbf{d}_{n+1}^{(i)}, \lambda_{n+1}) \\ \mathbf{d}_{n+1}^{(i+1)} &= \mathbf{d}_{n+1}^{(i)} + \delta\mathbf{d}_{n+1}^{(i+1)} \end{aligned} \qquad (3.9-2)$$

within an iteration loop where $n+1$ is the current (fixed) load-step number, $i+1$ is the iteration number at that load-step, $\delta\mathbf{d}$ is the iterative displacement change, and $\mathbf{r} = \mathbf{f}^{ext} - \mathbf{f}^{int}$ is the nonlinear residual force vector. $\mathbf{K}$ is the effective tangent stiffness matrix, which is updated only at the first iteration of selected load steps (though typically at every step). Iteration is continued until the inner product of $\delta\mathbf{d}$ and $\mathbf{r}$ become smaller than a user-specified error tolerance.

An arc-length constraint is added to the above equations so that i) the user doesn't have to select the load increment ($\Delta\lambda = \lambda_{n+1} - \lambda_n$), and ii) the solution algorithm can automatically traverse limit points – maxima and minima in the load-displacement "curve". The user need only specify the initial load factor, λ_1; the constraint equation converts this to an "arc-length" increment in load-displacement space, and adaptively adjusts this increment based on the iterative performance of the algorithm. Details of this algorithm are given in the theory and algorithm sections.

3.9.2 THEORY

3.9.2.1 Introduction

Procedure **NL_STATIC_1** performs a quasi-static analysis of a system of nonlinear equilibrium equations using an adaptive arc-length-controlled Newton/Raphson incremental/iterative solution algorithm. The arc-length method adopted here is based on a variant

of Crisfield's algorithm (ref. 3.1), where instead of using a nonlinear (quadratic) arc-length constraint equation in conjunction with the linearized equilibrium equations, the constraint equation is consistently linearized as well. This eliminates the pitfalls associated with quadratic root selection in Crisfield's algorithm. By combining this modification with i) polynomial extrapolation of converged solutions to obtain step-predictor solutions, ii) an energy error norm that properly weights translational and rotational freedoms, and iii) generalization to large rotations, specified displacements and live loads; the present algorithm is both more general and more robust than the basic algorithm. Additional enhancements to overcome hard singularities at limit and bifurcation points are currently under development; and will be incorporated as procedures at a later date.

NOTATION

$\mathbf{d}$	Displacement vector.
$\delta\mathbf{d}$	Iterative change in $\mathbf{d}$.
$\Delta\mathbf{d}$	Incremental (load-step) change in $\mathbf{d}$.
$\mathbf{d}_n^i$	Displacement vector at iteration i of step n.
$\Delta\mathbf{d}_n^i$	Incremental (step) change in $\mathbf{d}$. $\Delta\mathbf{d}_{n+1}^{(i+1)} = \mathbf{d}_{n+1}^{(i)} - \mathbf{d}_n$
$\delta\widehat{\mathbf{d}}$	Tangential displacement $= \mathbf{K}^{-1}\widehat{\mathbf{f}}^{ext}$
$\Delta\ell$	Incremental arc-length (step) parameter.
$\widehat{\mathbf{f}}^{ext}$	External force vector – base load.
$\mathbf{f}^{int}$	Internal force vector.
$\mathbf{r}$	Residual force vector.
$\mathbf{K}$	Stiffness matrix.
$\delta\overline{\mathbf{d}}$	Basic (fixed-load) iterative change in $\mathbf{d}$.
$\delta\mathbf{d}^*$	Linear combination of $\delta\overline{\mathbf{d}}$ and $\delta\widehat{\mathbf{d}}$.
num_cuts	Number of times load step has been cut in half at current step.
num_div	Number of consecutive iterations at which divergence occurs.
c_n	Extrapolation coefficient corresponding to step n.
$\widehat{\epsilon}$	Relative error in energy norm.
ϵ	Absolute error in energy norm.
ϵ_{ref}	Reference value of ϵ; initialized as zero.
ϵ_{tol}	Relative error tolerance (default: 10^{-4}).
λ	Load factor.
$\Delta\lambda$	Incremental (load-step) change in λ.
$\mathbf{K}^{us}$	Stiffness submatrix coupling specified (s) displacement components with force components corresponding to unknown (u) displacements.
$\widehat{\mathbf{d}}^s$	Base value of specified displacement vector.
$\mathbf{d}^s$	Current value of specified displacement vector. $\mathbf{d}^s = \lambda\,\widehat{\mathbf{d}}^s$.

3.9.2.2 Nonlinear Equations

The set of nonlinear equations solved by the present algorithm consist of: i) the system of equilibrium equations for the discrete (finite element) model, and ii) a single constraint equation governing the maximum "arc-length" to be traversed in a single solution increment along a curve in load-displacement space. These equations may be expressed as follows:

$$\begin{aligned} \mathbf{r}(\mathbf{d},\lambda) &= 0 \quad \text{(Equilibrium)} \\ c(\mathbf{d},\lambda) &= 0 \quad \text{(Arc-length Constraint)} \end{aligned} \qquad (3.9-3)$$

where $\mathbf{d}$ is the displacement vector representing all of the degrees-of-freedom (DOFs) for the discrete model, and λ is an external load parameter. In conventional load- or displacement-controlled solution algorithms, λ is usually specified by the user. In an arc-length controlled algorithm like the present one, λ is treated as an additional unknown. Thus, there are just as many independent equations in equation 3.9-3 as there are unknowns. The vector $\mathbf{r}$ represents the *residual* (or out-of-balance) force vector, which is identically zero at equilibrium.

For the special case of:

- Displacement-independent loading
- Proportional (one-parameter) force loading
- Load-independent arc-length constraint

$\mathbf{r}$ and c take on the following form:

$$\boxed{\begin{aligned} \mathbf{r}(\mathbf{d},\lambda) &= \lambda \widehat{\mathbf{f}}^{ext} - \mathbf{f}^{int}(\mathbf{d}) \\ c(\mathbf{d},\lambda) &= \|\Delta\mathbf{d}\|^2 - \Delta\ell^2 \end{aligned}} \qquad (3.9-4)$$

where $\widehat{\mathbf{f}}^{ext}$ is a normalized (base-value) external load vector, $\mathbf{f}^{int}$ is the nonlinear internal force vector, $\Delta\mathbf{d}$ is an incremental displacement vector, defined as

$$\Delta\mathbf{d} = \mathbf{d} - \mathbf{d}_n \qquad (3.9-5)$$

in which $\mathbf{d}_n$ is the known displacement vector at a nearby (previous) configuration, and $\Delta\ell$ is a prescribed arc-length parameter defining the size of the increment.

The problem statement then is to solve equation 3.9-3 for a statically connected set of configurations, $(\mathbf{d},\lambda)$, representing the load-displacement "history" of the structural model. Note that due to the nonlinear nature of the equations, this history is not always unique. For example, at pure bifurcation points, the present algorithm cannot determine the correct path, unless some sort of imperfection or "trigger" is introduced by the user – thus converting the bifurcation point into a limit point. Limit points (*i.e.*, local maximums and minimums in the load-displacement curve) cause no difficulty for the present algorithm – except when a load-step happens to lie "too close" to the actual limit point (see procedure **NL_STATIC_2** for a solution to this rare but frustrating problem).

3.9.2.3 Linearized Equations

To solve the above nonlinear equations, we use the modified Newton/Raphson algorithm, which requires their linearization – *i.e.*, first-order Taylor series expansion of the simultaneous equations. The solution to the linearized equations is then used to update the nonlinear solution in an iterative process that continues until convergence has been obtained at a given configuration. Furthermore, the final configuration is typically obtained as a series of *steps*, or increments, with a new iteration cycle occuring within each step; and with information from preceding steps used to predict a starting solution at the new step. (This is

commonly called an "incremental/iterative" algorithm.) Note that the *iterative* changes in displacement and load-factor, which we shall denote $\delta\mathbf{d}$ and $\delta\lambda$, respectively, are different than the *incremental* changes, denoted by $\Delta\mathbf{d}$ and $\Delta\lambda$. The incremental load/displacement changes are used to advance from one converged solution $(\mathbf{d}_n, \lambda_n)$ to another $(\mathbf{d}_{n+1}, \lambda_{n+1}$, while the iterative changes all refer to successive approximations $(\mathbf{d}_{n+1}^{(i)}, \lambda_{n+1}^{(i)})$ of the same target configuration $(\mathbf{d}_{n+1}, \lambda_{n+1})$. Thus, $\delta\mathbf{d}$ converges to zero at a given configuration, while $\Delta\mathbf{d}$ converges to a value dictated by the arc-length constraint equation 3.9-4b*

Linearization of equations 3.9-4 about a reference configuration $(\overline{\mathbf{d}}, \overline{\lambda})$ results in the following expressions:

$$\begin{aligned} \mathbf{r}(\mathbf{d},\lambda) &\approx \mathbf{r}(\overline{\mathbf{d}},\overline{\lambda}) + \frac{\partial \mathbf{r}}{\partial \mathbf{d}}(\overline{\mathbf{d}},\overline{\lambda})\delta\mathbf{d} + \frac{\partial \mathbf{r}}{\partial \lambda}(\overline{\mathbf{d}},\overline{\lambda})\delta\lambda = 0 \\ c(\mathbf{d},\lambda) &\approx c(\overline{\mathbf{d}},\overline{\lambda}) + \frac{\partial c}{\partial \mathbf{d}}(\overline{\mathbf{d}},\overline{\lambda})\delta\mathbf{d} + \frac{\partial c}{\partial \lambda}(\overline{\mathbf{d}},\overline{\lambda})\delta\lambda = 0 \end{aligned} \qquad (3.9-6)$$

where for the special assumptions listed in Section 3.9.9.1, the partial derivatives in equation 3.9-6 become:

$$\begin{aligned} \frac{\partial \mathbf{r}}{\partial \mathbf{d}}(\overline{\mathbf{d}},\overline{\lambda}) &= -\mathbf{K}(\overline{\mathbf{d}}), \qquad \frac{\partial \mathbf{r}}{\partial \lambda}(\overline{\mathbf{d}},\overline{\lambda}) = \widehat{\mathbf{f}}^{ext} \\ \frac{\partial c}{\partial \mathbf{d}}(\overline{\mathbf{d}},\overline{\lambda}) &= 2\Delta\overline{\mathbf{d}}, \qquad \frac{\partial c}{\partial \lambda}(\overline{\mathbf{d}},\overline{\lambda}) = 0 \end{aligned} \qquad (3.9-7)$$

where $\mathbf{K} = \partial\mathbf{f}^{int}/\partial\mathbf{d}$ is the tangent stiffness matrix. Substituting equation 3.9-7 into equation 3.9-6 leads to the simultaneous linear equations:

$$\boxed{\begin{aligned} \overline{\mathbf{K}}\,\delta\mathbf{d} &= \overline{\mathbf{r}} + \delta\lambda\widehat{\mathbf{f}}^{ext} \\ 2\Delta\overline{\mathbf{d}}\cdot\delta\mathbf{d} &= \overline{c} \end{aligned}} \qquad (3.9-8)$$

where

$$\overline{\mathbf{r}} = \mathbf{r}(\overline{\mathbf{d}},\overline{\lambda}), \qquad \overline{c} = c(\overline{\mathbf{d}},\overline{\lambda}), \qquad \overline{\mathbf{K}} = \begin{cases} \mathbf{K}(\overline{\mathbf{d}}) \text{ for True Newton} \\ \mathbf{K}(\mathbf{d}_n) \text{ for Modified Newton} \end{cases} \qquad (3.9-9)$$

Solving the first of equations 3.9-8 for $\delta\mathbf{d}$ leads to

$$\delta\mathbf{d} = \overline{\mathbf{K}}^{-1}\left[\overline{\mathbf{r}} + \widehat{\mathbf{f}}^{ext}\delta\lambda\right] = \delta\overline{\mathbf{d}} + \delta\widehat{\mathbf{d}}\,\delta\lambda \qquad (3.9-10)$$

where

$$\boxed{\begin{aligned} \delta\overline{\mathbf{d}} &= \overline{\mathbf{K}}^{-1}\overline{\mathbf{r}} \\ \delta\widehat{\mathbf{d}} &= \overline{\mathbf{K}}^{-1}\widehat{\mathbf{f}}^{ext} \end{aligned}} \qquad (3.9-11)$$

* The reason for using a series of increments, instead of solving for the final solution in one single increment, is that the Newton/Raphson method converges only if the starting solution is sufficiently "close" to the final solution.

Similarly, we can solve for $\delta\lambda$, by substituting equation 3.9-10 into equation 3.9-8b, *i.e.*,

$$\bar{c} + 2\Delta\bar{\mathbf{d}} \cdot \delta\bar{\mathbf{d}} + 2(\Delta\bar{\mathbf{d}} \cdot \delta\hat{\mathbf{d}})\delta\lambda = 0 \qquad (3.9-12)$$

which yields

$$\delta\lambda = \frac{-\bar{c} - 2\Delta\bar{\mathbf{d}} \cdot \delta\bar{\mathbf{d}}}{2\Delta\bar{\mathbf{d}} \cdot \delta\hat{\mathbf{d}}} = \frac{\Delta\ell^2 - \|\Delta\bar{\mathbf{d}}\|^2 - 2\Delta\bar{\mathbf{d}} \cdot \delta\bar{\mathbf{d}}}{2\Delta\bar{\mathbf{d}} \cdot \delta\hat{\mathbf{d}}} \qquad (3.9-13)$$

3.9.2.4 Update Procedure

The solution update procedure – from one iteration to the next at a fixed arc-length increment from a converged solution – consists of the following two (sequential) equations:

$$\boxed{\begin{aligned} 1)\ \delta\lambda &= \frac{\Delta\ell^2 - 2\Delta\bar{\mathbf{d}} - \|\Delta\bar{\mathbf{d}}\|^2}{2\Delta\bar{\mathbf{d}} \cdot \delta\hat{\mathbf{d}}} \\ 2)\ \delta\mathbf{d} &= \delta\bar{\mathbf{d}} + \delta\hat{\mathbf{d}}\,\delta\lambda \end{aligned}} \qquad (3.9-14)$$

where $\delta\bar{\mathbf{d}}$ and $\delta\hat{\mathbf{d}}$ are first computed using equation 3.9-11, and $\Delta\bar{\mathbf{d}}$ is the displacement increment from the previous converged solution to the previous iterate of the current solution, *i.e.*, $\bar{\mathbf{d}} - \mathbf{d}_n$.

Then we simply apply the update formulas:

$$\begin{aligned} \mathbf{d} &= \bar{\mathbf{d}} + \delta\mathbf{d} \\ \lambda &= \bar{\lambda} + \delta\lambda \end{aligned} \qquad (3.9-15)$$

where special consideration must be given to equation 3.9-15a in the case of problems with large-rotational degrees-of-freedom (see Section 3.9.2.10).

3.9.2.5 Predictor Solutions; arc-length-Constraint and Extrapolation Techniques

To obtain a prediction for $\mathbf{d}_{n+1}$ and λ_{n+1} at the beginning of step $n+1$ (so that the Newton iterations may occur within a sufficiently small neighborhood of the converged solution), we use one of the following two procedures:

1) The arc-length constraint equation 3.9-4b

or

2) Quadratic extrapolation of a series of converged solutions

The first approach (*arc-length constraint*) leads to the following predictor equations:

$$\boxed{\begin{aligned} \mathbf{d}_{n+1}^{(1)} &= \mathbf{d}_n + \underbrace{\Delta\lambda_{n+1}^{(1)}\,\delta\widehat{\mathbf{d}}_{n+1}}_{\Delta\mathbf{d}_{n+1}^{(1)}} \\ \lambda_{n+1}^{(1)} &= \lambda_n \pm \underbrace{\frac{\Delta\ell_{n+1}}{\|\delta\widehat{\mathbf{d}}_{n+1}\|}}_{\Delta\lambda_{n+1}^{(1)}} \end{aligned}} \qquad (3.9-16)$$

where the sign in front of $\Delta\lambda$ in equation 3.9-16b is taken as the sign of the determinant of $\overline{\mathbf{K}}$. Equation 3.9-16a was obtained by noting that

$$\Delta\mathbf{d}_{n+1}^{(1)} = \delta\mathbf{d}_{n+1}^{(1)} \qquad (3.9-17)$$

which from equation 3.9-10 can be written as

$$\delta\mathbf{d}_{n+1}^{(1)} = \delta\overline{\mathbf{d}}_{n+1}^{(1)} + \delta\lambda_{n+1}^{(1)}\,\delta\widehat{\mathbf{d}}_{n+1} = \Delta\lambda_{n+1}^{(1)}\,\delta\widehat{\mathbf{d}}_{n+1} \qquad (3.9-18)$$

since

$$\delta\overline{\mathbf{d}}_{n+1}^{(1)} = \overline{\mathbf{K}}^{-1}\,\mathbf{r}_n = \overline{\mathbf{K}}^{-1}\,\mathbf{0} = \mathbf{0} \qquad (3.9-19)$$

and

$$\delta\lambda_{n+1}^{(1)} = \Delta\lambda_{n+1}^{(1)} \qquad (3.9-20)$$

Then, equation 3.9-16b follows by substituting equation 3.9-16a into the arc-length constraint equation

$$\|\Delta\mathbf{d}_{n+1}^{(1)}\| = \Delta\ell \qquad (3.9-21)$$

The second approach (*quadratic extrapolation*) leads to the following predictor equations:

$$\boxed{\begin{aligned} \mathbf{d}_{n+1}^{(1)} &= c_n\,\mathbf{d}_n + c_{n-1}\,\mathbf{d}_{n-1} + c_{n-2}\,\mathbf{d}_{n-2} \\ \lambda_{n+1}^{(1)} &= c_n\,\lambda_n + c_{n-1}\,\lambda_{n-1} + c_{n-2}\,\lambda_{n-2} \end{aligned}} \qquad (3.9-22)$$

where $(\mathbf{d}_n, \lambda_n)$, $(\mathbf{d}_{n-1}, \lambda_{n-1})$, and $(\mathbf{d}_{n-2}, \lambda_{n-2})$ are the converged solutions at the three previous (consecutive) load steps, and the c_i are the quadratic Lagrange interpolation functions:

$$\begin{aligned} c_n &= \frac{(\ell_{n+1}-\ell_{n-2})(\ell_{n+1}-\ell_{n-1})}{(\ell_n-\ell_{n-2})(\ell_n-\ell_{n-1})} \\ c_{n-1} &= \frac{(\ell_{n+1}-\ell_{n-2})(\ell_{n+1}-\ell_n)}{(\ell_{n-1}-\ell_{n-2})(\ell_{n-1}-\ell_n)} \\ c_{n-2} &= \frac{(\ell_{n+1}-\ell_{n-1})(\ell_{n+1}-\ell_n)}{(\ell_{n-2}-\ell_{n-1})(\ell_{n-2}-\ell_n)} \end{aligned} \qquad (3.9-23)$$

The arc-length parameter, ℓ, is computed by arbitrarily setting $\ell_{n-2} = 0$, and accumulating the incremental parameters, $\Delta\ell$. Thus,

$$\begin{aligned} \ell_{n-2} &= 0 \\ \ell_{n-1} &= \Delta\ell_{n-1} \\ \ell_n &= \ell_{n-1} + \Delta\ell_n \\ \ell_{n+1} &= \ell_n + \Delta\ell_{n+1} \end{aligned} \qquad (3.9-24)$$

where the arc-length increments, $\Delta\ell_{n-2}$, $\Delta\ell_{n-1}$, $\Delta\ell_n$, and $\Delta\ell_{n+1}$ are defined by the adaptive algorithm described in Section 3.9.2.6.

Note that unlike the arc-length-based predictor equations 3.9-16, the above extrapolation formulas do not require any additional information to determine the direction (*i.e.*, sign) of the load increment. This is because the curvature of the quadratic polynomial used for extrapolation automatically senses, and enables the traversal of, turning points in the solution path.

Quadratic extrapolation is usually much more efficient than the arc-length constraint approach for computing the predictor solutions. In some cases, extrapolation can reduce the number of load steps by up to an order of magnitude. However, in other cases – *e.g.*, around very sharp turns in load-displacement space – quadratic extrapolation may be too *smooth* to capture the sudden changes. In such cases, small, arc-length-controlled predictor steps may be the most practical strategy for getting through the critical phases. (Adaptive algorithms for selectively switching between extrapolation and arc-length step predictors are recommended as a topic for future research.)

3.9.2.6 Adaptive Load-Step (arc-length) Selection

To advance the solution from one load-step (n) to the next (n+1), it is also necessary to select a new arc-length increment, $\Delta\ell_{n+1}$. Note that this parameter, which represents a distance traversed along the load-displacement "curve", is (by definition) always positive,* but may grow or shrink depending on solution difficulty. To automate the process, we use the following simple, but fairly robust, heuristic algorithm suggested by Crisfield (ref. 3.1):

$$\Delta\ell_{n+1} = \frac{\text{Desired number of iterations}}{\text{Actual number of iterations}} \times \Delta\ell_n \qquad (3.9-25)$$

where *actual* refers to the number of iterations required for convergence at the previous step (n), and the *desired* number of iterations is user-specified and typically around 4. This causes the step-size to grow or shrink in direct proportion to the convergence rate of the nonlinear solution, and generally leads to a nearly constant number of iterations (*i.e.*, computations) per step. In fact the number of load-steps will automatically adjust according to the curvature of the load-displacement curve, with more steps being required around turning points (high curvature) and less along smooth stretches (low curvature).

* This is in contrast to conventional load-controlled algorithms, which require explicit selection of the load increment, – a parameter that may change sign at limit or bifurcation points.

Note that the initial value of the arc-length increment, $\Delta\ell_1$, must be specified – at least indirectly – by the user. However, since the user typically has no physical insight about the magnitude of $\Delta\ell_1$, we employ the arc-length constraint equation 3.9-10 to compute it in terms of the initial load-factor, $\Delta\lambda_1$ – which the user can typically estimate based on some prior linear analysis. The expression used to compute the initial arc-length increment is thus

$$\Delta\ell_1 = \lambda_1 \|\delta\hat{\mathbf{d}}_1\| \qquad (3.9-26)$$

which was derived by rewriting equation 3.9-16b as

$$\Delta\ell_{n+1} = \Delta\lambda_{n+1} \|\delta\hat{\mathbf{d}}_{n+1}\| \qquad (3.9-27)$$

setting $n = 0$ and noting that

$$\Delta\lambda_1 = \lambda_1 - \lambda_0 = \lambda_1 \qquad (3.9-28)$$

The "tangential" displacement, $\delta\hat{\mathbf{d}}_1$, in equation 3.9-26 is simply the linear solution obtained with the normalized external load vector as right-hand-side, *i.e.*,

$$\delta\hat{\mathbf{d}}_1 = \mathbf{K}^{-1}(\mathbf{0})\hat{\mathbf{f}}^{ext} \qquad (3.9-29)$$

3.9.2.7 Specified Displacements

The presence of specified-displacement loads (in addition to specified-force loads) affects the above algorithm in three subtle – but important – ways:

i) It modifies the definition of the residual force vector, $\mathbf{r}$, since the *internal* force vector, $\mathbf{f}^{int}$, becomes a function of the load-factor, λ.

ii) Due to the above dependence, the derivative of $\mathbf{r}$ with respect to λ gains an additional term, which in turn modifies the definition of the tangential displacement vector, $\delta\hat{\mathbf{d}}$.

iii) The specified displacement components must be included in all displacement norms and inner products appearing in the arc-length constraint equation, and in the computation of nonlinear error estimates.

The above modifications can be expressed mathematically as follows. First, define a *specified displacement vector*, $\mathbf{d}^s$, which has zeros everywhere except in the components that are user-specified. Further assume that this specified displacement vector is scaled by the same load-factor, λ, as the external force vector, *i.e.*,

$$\mathbf{d}^s = \lambda\hat{\mathbf{d}}^s \qquad (3.9-30)$$

where $\hat{\mathbf{d}}^s$ is a normalized vector containing the reference ($\lambda = 1$) values of specified displacement. The expression for the residual force vector, equation 3.9-4, can then be re-written as

$$\mathbf{r}(\mathbf{d},\lambda) = \lambda\hat{\mathbf{f}}^{ext} - \mathbf{f}^{int}(\mathbf{d},\mathbf{d}^s) \qquad (3.9-31)$$

and its derivative with respect to λ in equation 3.9-7 becomes

$$\frac{\partial \mathbf{r}}{\partial \lambda} = \hat{\mathbf{f}}^{ext} - \frac{\partial \mathbf{f}^{int}}{\partial \mathbf{d}^s}\frac{\partial \mathbf{d}^s}{\partial \lambda} = \hat{\mathbf{f}}^{ext} - \frac{\partial \mathbf{f}^{int}}{\partial \mathbf{d}^s}\hat{\mathbf{d}}^s \qquad (3.9-32)$$

But:

$$\frac{\partial \mathbf{f}^{int}}{\partial \mathbf{d}^s} = \mathbf{K}^{us} \qquad (3.9-33)$$

where $\mathbf{K}^{us}$ is the partition of the tangent stiffness matrix that couples specified displacement increments to unknown force increments, *i.e.*, the *extended* stiffness matrix may be partitioned as

$$\mathbf{K}^* = \begin{bmatrix} \mathbf{K}^{uu} & \mathbf{K}^{us} \\ \mathbf{K}^{su} & \mathbf{K}^{ss} \end{bmatrix} \qquad (3.9-34)$$

where $\mathbf{K}^{uu}$ is the *active* block of the stiffness matrix, *i.e.*, in the present terminology:

$$\mathbf{K} = \mathbf{K}^{uu} \qquad (3.9-35)$$

Thus, the new load-derivative of the residual force vector may be expressed as

$$\boxed{\frac{\partial \mathbf{r}}{\partial \lambda} = \hat{\mathbf{f}}^{ext} - \mathbf{K}^{us}\hat{\mathbf{d}}^s} \qquad (3.9-36)$$

and the corresponding *tangential displacement* vector from equation 3.9-11b becomes:

$$\boxed{\delta\hat{\mathbf{d}} = \overline{\mathbf{K}}^{-1}\left(\hat{\mathbf{f}}^{ext} - \overline{\mathbf{K}}^{us}\hat{\mathbf{d}}^s\right)} \qquad (3.9-37)$$

where the superposed bar is defined in equation 3.9-9.

Note that the solution indicated in equation 3.9-37 is much like what is required for linear analysis with specified displacements: The right-hand-side is modified by multiplying a part of the extended stiffness matrix times the (normalized) specified displacements. This is in contrast to the solution for $\delta\overline{\mathbf{d}}$ in equation 3.9-11a, where specified displacements are accounted for exclusively through their nonlinear dependence in the residual force vector, $\overline{\mathbf{r}}$.

3.9.2.8 General Loading (*e.g.*, "Live" Loads)

The last restriction we shall lift is the simple form of the external force vector given in equation 3.9-4a, *i.e.*, we shall replace:

$$\mathbf{f}^{ext} = \lambda\hat{\mathbf{f}}^{ext} \qquad (3.9-38)$$

by

$$\mathbf{f}^{ext} = \lambda\hat{\mathbf{f}}^{ext}(\mathbf{d}) \qquad (3.9-39)$$

This allows for the kind of displacement-dependent (or "live") loading that arises from hydrostatic, or follower, forces. The residual force vector then becomes

$$\mathbf{r}(\mathbf{d},\lambda) = \lambda \hat{\mathbf{f}}^{ext}(\mathbf{d}) - \mathbf{f}^{int}(\mathbf{d},\lambda \hat{\mathbf{d}}^{s}) \qquad (3.9-40)$$

and the only modifications to equations 3.9-7 are that i) the tangent stiffness matrix acquires a *load-stiffness* contribution, *i.e.*,

$$\frac{\partial \mathbf{r}}{\partial \mathbf{d}} = -\mathbf{K} = -(\mathbf{K}^{matl} + \mathbf{K}^{geom} + \mathbf{K}^{load}) \qquad (3.9-41)$$

where

$$\mathbf{K}^{load} = -\frac{\partial \mathbf{f}^{ext}}{\partial \mathbf{d}} \qquad (3.9-42)$$

and ii) the external-force contribution to the tangent load vector, $\mathbf{r}_{,\lambda}$, is no longer a constant, *i.e.*,

$$\frac{\partial \mathbf{r}}{\partial \lambda} = \hat{\mathbf{f}}^{ext}(\mathbf{d}) - \mathbf{K}^{us}(\mathbf{d})\hat{\mathbf{d}}^{s} \qquad (3.9-43)$$

Note that we have not allowed for a general external-load history, in which both the magnitude and direction of the external force vector may change from step to step, *e.g.*,

$$\mathbf{f}^{ext} = \mathbf{f}^{ext}(\mathbf{d},\lambda) \qquad (3.9-44)$$

Nor have we allowed for *multiple* load-factors, λ_a, λ_b, ..., which can arise when several independent load systems are acting on a structure. Such complications require straightfoward generalization of the present arc-length algorithm, and will be considered as necessary for future applications.

3.9.2.9 Convergence Criteria

As a measure of the error in the nonlinear equilibrium equations (not including discretization errors), we use an *energy norm*, which is the inner product of the residual force vector and the iterative displacement-change vector. This is effective for two reasons: i) it involves only a single error norm, in contrast to algorithms that check displacement and residual errors independently; and ii) the inner product of force and displacement introduces a natural scaling of different types of generalized freedoms (for example rotational freedoms versus translational freedoms) – by weighting each generalized displacement with its corresponding (conjugate) generalized force. Thus, sensitivity to physical units and choice of independent variables is minimized.

Convergence of the nonlinear solution process at a given load-step, $n+1$, is checked by evaluating the following energy error norm at each iteration, i:

$$\boxed{\|\text{error}\|^{(i)}_{n+1} = \sqrt{\mathbf{r}^{(i)}_{n+1} \cdot \delta\mathbf{d}^{(i)}_{n+1}}} \qquad (3.9-45)$$

and comparing it with some user-specified fraction of a *reference* error norm. Thus, convergence is defined as satisfaction of the condition:

$$\|\text{error}\|_{n+1}^{(i)} \leq \text{tolerance} \times \|\text{error}\|_{n+1}^{ref} \qquad (3.9-46)$$

where *tolerance* is the user-specified fraction, and the *reference* error is defined as:

$$\|\text{error}\|_{n+1}^{ref} = \text{MAX}(\|\text{error}\|_{n+1}^{(1)}, \|\text{error}\|_{n}^{ref}) \qquad (3.9-47)$$

Note that at the first load-step, $\|\text{error}\|_{0}^{ref}$) is assumed to be zero.

The above convergence criterion can be very sensitive to the user-specified error *tolerance*. Typically, a value of 10^{-3} is adequate, but for some problems this may be an order of magnitude too large – or even too small (*i.e.*, causing more iterations than are actually needed). More adaptive and robust error tolerances are a recommended topic for future research.

3.9.2.10 Large Rotations

In problems involving rotational freedoms, *e.g.*, with beam or shell elements, the following modification to equation 3.9-15a is used when the rotation angles become large (say greater than 10 degrees). First, note that the system displacement vector, and its iterative change, are typically partitioned by nodes, *i.e.*,

$$\mathbf{d} = \begin{Bmatrix} \mathbf{d}_1 \\ \mathbf{d}_2 \\ \vdots \\ \mathbf{d}_{\text{Nnodes}} \end{Bmatrix}, \qquad \delta\mathbf{d} = \begin{Bmatrix} \delta\mathbf{d}_1 \\ \delta\mathbf{d}_2 \\ \vdots \\ \delta\mathbf{d}_{\text{Nnodes}} \end{Bmatrix} \qquad (3.9-48)$$

where Nnodes is the total number of nodes in the problem. At nodes with both translational and rotational freedoms we can further partition:

$$\mathbf{d}_A = \begin{Bmatrix} \mathbf{u}_A \\ \mathbf{T}_A \end{Bmatrix}, \qquad \delta\mathbf{d}_A = \begin{Bmatrix} \delta\mathbf{u}_A \\ \delta\boldsymbol{\theta}_A \end{Bmatrix} \qquad (3.9-49)$$

in which

$\mathbf{u}_A$ = translation at node A

$\delta\mathbf{u}_A$ = iterative change in translation at node A

$\mathbf{T}_A$ = rotation triad at node A

$\delta\boldsymbol{\theta}_A$ = iterative change in rotation pseudo-vector at node A

Thus the *total* displacement, $\mathbf{d}_A$, is represented by the combination of a nodal translation, and a nodal triad that describes the orientation of a "rigid body" attached to the node. The iterative change in displacement, $\delta\mathbf{d}_A$, is represented by an iterative change in translation, and an iterative change in a rotation pseudo-vector. The latter quantity uniquely describes iterative changes in the rotation triad.

With the above definitions, the modified version of equation 3.9-15a at an individual node may be written as follows. For the translational freedoms we simply have:

$$\mathbf{u}_A = \overline{\mathbf{u}}_A + \delta\mathbf{u}_A \qquad (3.9-50)$$

whereas for the rotational freedoms we use:

$$\boxed{\mathbf{T}_A = exp(\delta\boldsymbol{\theta}_A)\overline{\mathbf{T}}_A} \qquad (3.9-51)$$

where $exp(\delta\boldsymbol{\theta}_A)$ is the *exponential* of $\delta\boldsymbol{\theta}_A$, which is an orthogonal (rotation) matrix whose rotation angle and direction correspond to the magnitude and direction of the pseudo-vector $\delta\boldsymbol{\theta}_A$. The following explicit expression for this matrix is known as as "Rodriguez' formula" (see, *e.g.*, ref. 3.2):

$$exp(\boldsymbol{\theta}) = \mathbf{I} + \frac{sin\,\theta}{\theta}\boldsymbol{\Theta} + \frac{(1-cos\,\theta)}{\theta^2}\boldsymbol{\Theta}^2 \qquad (3.9-52)$$

where $\boldsymbol{\Theta}$ is the *skew-symmetric* matrix corresponding to the pseudo-vector, $\boldsymbol{\theta}$, *i.e.*,

$$\boldsymbol{\Theta}\mathbf{h} = \boldsymbol{\theta} \times \mathbf{h} \qquad (3.9-53)$$

for any vector $\mathbf{h}$, so that

$$\boldsymbol{\Theta} = \begin{bmatrix} 0 & -\theta_3 & \theta_2 \\ \theta_3 & 0 & -\theta_1 \\ -\theta_2 & \theta_1 & 0 \end{bmatrix}, \qquad \boldsymbol{\theta} = \begin{Bmatrix} \theta_1 \\ \theta_2 \\ \theta_3 \end{Bmatrix} \qquad (3.9-54)$$

and θ is the magnitude of $\boldsymbol{\theta}$, *i.e.*,

$$\theta = \|\boldsymbol{\theta}\| = \sqrt{\theta_1^2 + \theta_2^2 + \theta_3^2} \qquad (3.9-55)$$

where $\theta_i (i = 1,2,3)$ are Cartesian components of $\boldsymbol{\theta}$. Note that the pseudovector, $\boldsymbol{\theta}$, behaves as a true vector in all ways except for vector addition – since the sum of two arbitrary large rotations is not a vectorial sum (see refs. 3.3or 3.4for the rules of rotation pseudovector addition).

The rotation update formula given by equation 3.9-19 need only be performed at nodes with rotational freedoms. It may also be used to perform *incremental* updates as well as iterative updates, *i.e.*,

$$\mathbf{T}_A^{n+1} = exp(\Delta\boldsymbol{\theta}_A)\,\mathbf{T}_A^n \qquad (3.9-56)$$

The above relation is employed in the computation of the *predictor* solutions described by equations 3.9-16 or 3.9-22. For the *extrapolated* predictor (equation 3.9-22), the extrapolated rotation triads are obtained by: i) extrapolating the rotational components of the displacement vector precisely as indicated by equation 3.9-22a; ii) computing the rotation increments between steps n and $n+1$ by subtracting the rotational components of $\mathbf{d}_n$ from those of the extrapolated vector, $\mathbf{d}_{n+1}$; and iii) plugging the corresponding rotation-increments $(\Delta\boldsymbol{\theta}_A)$ into equation 3.9-56. Note that while the rotation components of the "total" displacement vectors, $\mathbf{d}_A^n$ and $\mathbf{d}_A^{n+1}$ are meaningless for large rotations, the difference between these to vectors defines a valid incremental rotation pseudo-vector, $\Delta\boldsymbol{\theta}_A$.

3.9.3 ALGORITHM

General Algorithm

(1) STEP LOOP: for n = 2, 3, 4 ... (step = n+1)

(1.1) Extrapolate Solution for Predictor

call EXTRAP ($\Delta\ell_{n+1}, \Delta\ell_n, \Delta\ell_{n-1} \rightarrow c_n, c_{n-1}, c_{n-2}$)

$$\lambda_{n+1}^{(1)} = c_n\lambda_n + c_{n-1}\lambda_{n-1} + c_{n-2}\lambda_{n-2}$$
$$\mathbf{d}_{n+1}^{(1)} = c_n\mathbf{d}_n + c_{n-1}\mathbf{d}_{n-1} + c_{n-2}\mathbf{d}_{n-2}$$

$$\Delta\mathbf{d}_{n+1}^{(1)} = \mathbf{d}_{n+1}^{(1)} - \mathbf{d}_n \qquad (\mathbf{T}_{n+1}^{(1)} = \mathbf{R}(\Delta\mathbf{d}_{n+1}^{(1)})\mathbf{T}_n)$$
$$\Delta\lambda_{n+1}^{(1)} = \lambda_{n+1}^{(1)} - \lambda_n$$

(1.2) Solve for Tangential Displacement based on Predictor

$$\boxed{\delta\widehat{\mathbf{d}} = \mathbf{K}^{-1}(\mathbf{d}_{n+1}^{(1)})\left[\widehat{\mathbf{f}}^{ext} + \mathbf{K}^{us}\widehat{\mathbf{d}}^{s}\right]}$$

(1.3) Form Residual based on Predictor

$$\mathbf{r}_{n+1}^{(1)} = \mathbf{r}(\mathbf{d}_{n+1}^{(1)}, \lambda_{n+1}^{(1)}) = \lambda_{n+1}^{(1)}\widehat{\mathbf{f}}^{ext} - \mathbf{f}^{int}(\mathbf{d}_{n+1}^{(1)})$$

call CHK_CONV ($\mathbf{r}_{n+1}^{(1)}, \Delta\mathbf{d}_{n+1}^{(1)}, \epsilon_{ref} \rightarrow \widehat{\epsilon}^{(1)}, \epsilon_{ref}$)

(2) ITERATION LOOP: i = 1, 2, ... (iter = i+1 = 2, 3, ...)

(2.1) Solve for Basic Iterative Displacement Change

$$\boxed{\delta\overline{\mathbf{d}} = \mathbf{K}^{-1}(\mathbf{d}_{n+1}^{(1)})\,\mathbf{r}_{n+1}^{(i)}}$$

(2.2) Solve Linearized arc-length Constraint Eqn for New Load Factor

$$\delta\lambda = \frac{\Delta\ell^2 - \Delta\mathbf{d}_{n+1}^{(i)} \cdot \Delta\mathbf{d}_{n+1}^{(i)} - 2(\Delta\mathbf{d}_{n+1}^{(i)} \cdot \delta\overline{\mathbf{d}})}{2(\Delta\mathbf{d}_{n+1}^{(i)} \cdot \delta\widehat{\mathbf{d}})}$$

$$\lambda_{n+1}^{(i+1)} = \lambda_{n+1}^{(i)} + \delta\lambda$$

(2.3) Update Displacements

$$\delta \mathbf{d} = \delta \overline{\mathbf{d}} + \delta\lambda\, \delta \widehat{\mathbf{d}}$$

$$\mathbf{d}_{n+1}^{(i+1)} = \mathbf{d}_{n+1}^{(i)} + \delta \mathbf{d} \qquad (\mathbf{T}_{n+1}^{(i+1)} = \mathbf{R}(\delta \mathbf{d})\, \mathbf{T}_{n+1}^{(i)})$$

$$\Delta \mathbf{d}_{n+1}^{(i+1)} = \Delta \mathbf{d}_{n+1}^{(i)} + \delta \mathbf{d}$$

(2.4) Compute New Residual

$$\mathbf{r}_{n+1}^{(i+1)} = \mathbf{r}(\mathbf{d}_{n+1}^{(i+1)}, \lambda_{n+1}^{(i+1)}) = \lambda_{n+1}^{(i+1)} \widehat{\mathbf{f}}^{ext} - \mathbf{f}^{int}(\mathbf{d}_{n+1}^{(i+1)})$$

(2.5) Check Convergence

call CHK_CONV ($\mathbf{r}_{n+1}^{(i+1)}, \delta \mathbf{d}, \widehat{\epsilon}^{(i)}, \epsilon_{ref}, \widehat{\epsilon}_{tol}$, num_div $\rightarrow$
$\widehat{\epsilon}^{(i+1)}$, <CONVERGED>, <DIVERGED>)

if (<CONVERGED>) then

num_iters_required = iter

$\Delta \ell_{n+2} = \Delta \ell_{n+1} \left(\frac{\text{num\_iters\_desired}}{\text{num\_iters\_required}} \right)$

$n \leftarrow n + 1$

go to (1) STEP LOOP

elseif (<DIVERGED> .or. (iter > max_iters)) then

if (num_cuts < max_cuts) then

$\Delta \ell_{n+1} = \Delta \ell_{n+1}/2$

num_cuts = num_cuts + 1

go to (1.1)

else

STOP

endif

else

$i \leftarrow i + 1$

go to (2) ITER LOOP

endif

Starting Procedure: Step 1 (n=0)

Replace Algorithm Steps (1.1)–(1.2) by:

$$\mathbf{d}_1^{(0)} = \mathbf{0}$$

$$\delta\hat{\mathbf{d}} = \mathbf{K}^{-1}(\mathbf{d}_1^{(0)}) \left[\hat{\mathbf{f}}^{ext} + \mathbf{K}^{us}\hat{\mathbf{d}}^{s}\right]$$

$$\begin{aligned} \lambda_1^{(1)} &= \lambda_{start} \qquad \text{(user specified)} \\ \Delta\lambda_1^{(1)} &= \lambda_1^{(1)} \\ \Delta\ell_1 &= \Delta\lambda_1^{(1)} \|\delta\hat{\mathbf{d}}\| \\ \Delta\mathbf{d}_1^{(1)} &= \Delta\lambda_1^{(1)} \delta\hat{\mathbf{d}} \\ \mathbf{d}_1^{(1)} &= \Delta\mathbf{d}_1^{(1)} \end{aligned}$$

(Also, form $\mathbf{K}(\mathbf{d}_1^{(1)})$ before next solve.)

Starting Procedure: Step 2 (n=1)

Replace Algorithm Steps (1.1)–(1.2) by:

$$\mathbf{d}_2^{(0)} = \mathbf{d}_1$$

$$\delta\hat{\mathbf{d}} = \mathbf{K}^{-1}(\mathbf{d}_2^{(0)}) \left[\hat{\mathbf{f}}^{ext} + \mathbf{K}^{us}\hat{\mathbf{d}}^{s}\right]$$

$$\begin{aligned} \Delta\lambda_2^{(1)} &= \Delta\ell_2 / \|\delta\hat{\mathbf{d}}\| \\ \lambda_2^{(1)} &= \lambda_1 + \Delta\lambda_2^{(1)} \\ \Delta\mathbf{d}_2^{(1)} &= \Delta\lambda_2^{(1)} \delta\hat{\mathbf{d}} \\ \mathbf{d}_2^{(1)} &= \mathbf{d}_1 + \Delta\mathbf{d}_2^{(1)} \qquad (\mathbf{T}_2^{(1)} = \mathbf{R}(\Delta\mathbf{d}_2^{(1)}\, \mathbf{T}_1) \end{aligned}$$

(But don't reform $\mathbf{K}(\mathbf{d}_1^{(1)})$ until step 3.)

CHK_CONV: Procedure to Check Convergence

Input: $\mathbf{r}$, $\delta\mathbf{d}$, $\hat{\epsilon}^{(i)}$, ϵ_{ref}, $\hat{\epsilon}_{tol}$, num_div, iter

Output: $\hat{\epsilon}^{(i+1)}$, num_div, <CONVERGED>, <DIVERGED>

(1) Compute Energy Error Norm: $\epsilon^{(i+1)} = |\mathbf{r} \cdot \delta\mathbf{d}|$

(2) Normalize: $\hat{\epsilon}^{(i+1)} = \sqrt{\frac{\epsilon^{(i+1)}}{\epsilon_{ref}}}$

(3) Check Convergence / Divergence:

```
if ( ε̂^(i+1) ≤ ε̂_tol ) then
        <CONVERGED> = TRUE
else
        if ( ε̂^(i+1) > ε̂^(i) ) then
                num_div = num_div + 1
                if ( num_div > 1 ) then
                        <DIVERGED> = TRUE
                endif
        endif
endif
```

Modification for Iteration = 1 (i=0):

```
(1.5) if ( iter = 1 ) then
        if ( ε^(1) > ε_ref ) then
                ε_ref = ε^(1)
        endif
endif
```

EXTRAP: Procedure to Compute Quadratic Extrapolation Coefficients

Input: $\Delta\ell_{n+1}$, $\Delta\ell_n$, $\Delta\ell_{n-1}$

Output: c_n, c_{n-1}, c_{n-2}

$$\ell_{n-1} = \Delta\ell_{n-1}$$

$$\ell_n = \ell_{n-1} + \Delta\ell_n$$

$$\ell_{n+1} = \ell_n + \Delta\ell_{n+1}$$

$$\Delta\ell_{tot} = \ell_{n+1} - \ell_{n-1}$$

$$c_n = \frac{(\ell_{n+1})(\Delta\ell_{tot})}{(\ell_n)(\Delta\ell_n)}$$

$$c_{n-1} = -\frac{(\ell_{n+1})(\Delta\ell_{n+1})}{(\ell_{n-1})(\Delta\ell_n)}$$

$$c_{n-2} = \frac{(\Delta\ell_{n+1})(\Delta\ell_{tot})}{(\ell_{n-1})(\ell_n)}$$

3.9.4 PROCEDURE USAGE

Procedure **NL_STATIC_1** may be invoked by the ***call** directive:

```
*call NL_STATIC_1 ( arg1 = val1; arg2 = val2; ...)
```

where **arg***i* are argument names and **val***i* are the corresponding values you wish to give them. The following are valid arguments for procedure **NL_STATIC_1**; note that those without default values are mandatory, while the others are optional.

Argument	Default Value	Meaning
BEG_LOAD	--	Starting load factor (>0.)
BEG_STEP	--	Starting step number (>0)
MAX_LOAD	--	Upper_bound on load factor
MIN_LOAD	--	Lower_bound on load factor
MAX_STEPS	--	Maximum steps to compute
COROTATION	1	Corotational Update Option
DES_ITERS	4	Number of iterations per step desired
EXTRAPOLATE	<true>	Perform quadratic extrapolation of solution
FAC_STEPS	1	Steps_per_refactoring
MAX_ITERS	9	Maximum iterations per step
MAX_CUTS	3	Maximum number of successive step cuts
NL_GEOM	2	Geometric Nonlinearity Level (1 or 2)
NOMINAL_DB	NOMINAL.GAL	Results database file
NOMINAL_DS	RESPONSE.HISTORY	Results dataset
N_SELECT	0	Number of nodes for selected disp. output
PATH_SCALE	1.	arc-length scale factor for restarts
SEL_NODES	0	List of nodes for selected output
SEL_DOFS	0	Corresponding list of nodal freedoms (1–6)
TOL_E	1.E-3	Relative error tolerance in energy norm

In the above definitions, the term *step* refers to a load step. The total response is automatically subdivided into load steps, with the starting load factor prescribed by the user – using BEG_LOAD. Subsequent load step sizes are automatically selected by the algorithm, using an arc-length constraint, as described in the theory section.

Table 3.9-1 Datasets Input/Output by procedure NL_STATIC_1				
Dataset	*Description*	*Lib*	*Input*	*Output*
<ES_NAME>.EFIL.*0.nnod*	Element Computational Data	1	√	√
ES.SUMMARY.*0.nnod*	ES Processor Status	1	√	√
DEF.<ES_NAME>.*0.nnod*	Element Defn. (Connectivity)	1	√	
DIR.<ES_NAME>..*0.nnod*	Element EFIL Directory	1	√	
INC.DISP	System Displacement Vector	1	√	√
INT.FORC.*step*	Nodal Rotation Pseudovectors	1	√	√
JDF1.BTAB.1.8	Model Summary	1	√	
JLOC.BTAB.2.5	Nodal Coordinates	1	√	
PROP.BTAB.*	Material/Section Properties	1	√	
QJJT.BTAB.2.9	Nodal Transformations	1	√	
TOT.DISP.*step*	System Displacement Vector	1	√	√
TOT.ROTN.*step*	System Force Vector	1	√	√

where *step* is the load-step number, and ranges consecutively from 1 to the total number of steps computed.

Table 3.9-2 Sub-Procedures Invoked by procedure NL_STATIC_1		
Procedure	*Type*	*Function*
NL_STATIC_1	Internal	Main procedure
CHKCONV	Internal	Check convergence
DEFNS	Internal	Defines recursive macrosymbols
ES	External	Element utility procedure
EXTRAP	Internal	Quadratic extrapolation for next load step
INITIAL	Internal	Initialize displacements and rotations
POSTSTEP	Internal	Print load step summary
POSTRESS	Internal	Archive load step data
RESIDUAL	Internal	Forms residual-force vector
SOLVE	Internal	Solves linear equation systems
STIFFNESS	Internal	Forms and factors stiffness matrix
TANDIS	Internal	Solves for tangential displacements

3.9.5 ARGUMENT DESCRIPTION

3.9.5.1 BEG_LOAD

Starting load factor (λ_1) for the nonlinear analysis. For applied *force* loading, this factor is multiplied by the reference applied force vector to obtain the starting load vector, *i.e.*,

$$\mathbf{f}_1^{ext} = \lambda_1 \bar{\mathbf{f}}^{ext}$$

where $\bar{\mathbf{f}}^{ext}$ is the reference applied force vector stored in dataset APPL.FORC.1. For applied *displacement* loading, the starting load factor is applied to the reference applied displacement vector, which is then used to compute the initial internal force vector, *i.e.*,

$$\mathbf{f}_1^{int(1)} = \mathbf{f}^{int}(\lambda_1 \bar{\mathbf{d}}^{ext})$$

where $\bar{\mathbf{d}}^{ext}$ is the reference applied displacement vector stored in dataset APPL.MOTI.1.

Note that this argument is *irrelevant* for re-start runs (*i.e.*, BEG_STEP > 1).

3.9.5.2 BEG_STEP

This argument defines the number of the first step to be computed in a given nonlinear analysis interval. It is important primarily for analysis re-starts. Initially, BEG_STEP should be set to 1. To continue an analysis in a subsequent run, after having computed and saved "n" steps in the previous run, one would typically set BEG_STEP equal to "n+1". For example, if the 10th step was successfully completed in the first run, then it could be continued in a second run by setting BEG_STEP = 11. However, it is not necessary for BEG_STEP to be larger than any previously computed step. That is, you may *re-compute* a sequence of steps by setting BEG_STEP to the number of the first step to be re-computed. The procedure will automatically use those steps which immediately precede BEG_STEP (*e.g.*, BEG_STEP-1, BEG_STEP-2 and BEG_STEP-3) to smoothly effect the restart.

3.9.5.3 COROTATION

Corotational update switch for large-rotation problems (default: <true>). This switch should be set to <true> when the model involves finite elements that require corotation for geometric nonlinearity. This is true of most beam and shell elements, and may be true for some solid (3D) elements used to model shell structures. Consult the appropriate element processor (ES*i*) section in the CSM Testbed User's Manual (see ref. 3.9-5) for specific guidelines.

3.9.5.4 DEBUG

Procedure debug switch (default: <false>). This switch should only be turned on to obtain additional diagnostic printout for procedure debugging.

3.9.5.5 DES_ITERS

Desired number of iterations allowed for convergence at a given load step (default=4). This parameter is used to adaptively adjust the arc-length increment from one load step to the next, by comparing DES_ITERS with the actual number of iterations required for convergence at the last step.

3.9.5.6 EXTRAPOLATE

Solution extrapolation switch (default: <true>). Extrapolation here refers to a technique for predicting the displacement vector and load factor at the beginning of a new load step, by fitting a quadratic curve through the converged solutions at the three previous load steps. It has been found to be a very effective strategy for accelerating traversal of the load-displacement "curve", *i.e.*, far fewer load steps are usually required with extrapolation turned on, than with it turned off. One exception is near very sharp turns in the load-displacement curve, where extrapolation may be too smooth to follow the curve, and may have to be temporarily suppressed. (Note: EXTRAPOLATE = <false> option has not been fully tested.)

3.9.5.7 FAC_STEPS

Number of load steps between updating (formation and re-factoring) of the stiffness matrix default = 1). **FAC_STEPS** = n implies that re-factoring will be performed every n steps, starting with the first step of the analysis interval (**BEG_STEP**). Best results are often obtained by allowing the procedure to re-factor at the beginning of each load step (**FAC_STEPS** = 1).

3.9.5.8 MAX_CUTS

Maximum number of step cuts permitted during the current nonlinear analysis interval (default=3). A step cut refers to a halving of the arc-length increment used to advance the solution from one step to the next. Step cuts are performed only if the maximum number of iterations are exceeded without converging at a given load step. Note that the relationship between the increment in "arc-length" and the increment in the load-factor, λ, is computed internally by the procedure.

3.9.5.9 MAX_ITERS

Maximum number of iterations allowed for convergence at a given load step. This parameter is used to terminate the iteration process at a given load level. If convergence hasn't been obtained after **MAX_ITERS** iterations, the load (*i.e.*, arc-length) increment is cut in half and the step is repeated – until either convergence has been obtained or **MAX_CUTS** has been exceeded.

3.9.5.10 MAX_LOAD

Maximum load factor. This sets an upper limit on the load level, and thus provides a convenient way of terminating the arc-length controlled solution algorithm. Since the load factor is actually an unknown in procedure **NL_STATIC_1**, there is no way of knowing a-priori how many load steps will be required to attain a particular load level. The nonlinear analysis is terminated when either **MAX_LOAD**, **MIN_LOAD**, **MAX_STEPS** or **MAX_CUTS** is exceeded – whichever comes first.

3.9.5.11 MAX_STEPS

Maximum number of load steps to compute in the current nonlinear analysis run. This provides an implicit limit on analysis run-time. Since the load factor is actually an unknown in procedure **NL_STATIC_1** (controlled by the arc-length constraint), there is no way of knowing a-priori how many load steps will be required to attain a particular load level. The nonlinear analysis is thus terminated whenever **MAX_STEPS**, **MIN_LOAD**, **MAX_LOAD** or **MAX_CUTS** is exceeded – whichever comes first.

3.9.5.12 MIN_LOAD

Minimum load factor. This sets a lower limit on the load level, and thus provides a convenient way of terminating the arc-length controlled solution algorithm. Since the load

factor is actually an unknown in procedure **NL_STATIC_1**, there is no way of knowing a-priori how many load steps will be required to attain a particular load level. The nonlinear analysis is terminated when either **MIN_LOAD**, **MAX_LOAD**, **MAX_STEPS** or **MAX_CUTS** is exceeded – whichever comes first.

3.9.5.13 NL_GEOM

Geometric nonlinearity level: 0, 1, or 2 (default =2). A value of zero means that the problem is geometrically linear; a value of one means that the geometric nonlinearity will be handled globally (*i.e.*, using corotational updates only); and a value of two means that the nonlinear element strain-displacement relations will be used in addition to any global treatment of geometric nonlinearity. If **COROTATION** = **<true>**, options 1 and 2 refer to first-order and second-order corotation, respectively. The latter option can be significantly more accurate than the former for a given finite element model – depending on which element types are involved.

3.9.5.14 NOMINAL_DB

Name of database (GAL file) where a step-wise history of important solution parameters and selected response variables is to be stored (default = **NOMINAL.GAL**).

3.9.5.15 NOMINAL_DS

Name of dataset, within database defined by argument **NOMINAL_DB**, where a step-wise history of important solution parameters and selected response variables is to be stored (default = **RESPONSE.HISTORY**). See the CSM Testbed Dataset Manual (ref. 3.9-6), under dataset **RESPONSE.HISTORY**, for a description of the individual data records stored in this dataset.

3.9.5.16 N_SELECT

Number of user-selected displacement components to be saved in the dataset specified by argument **NOMINAL_DS** (default =0). Values for these displacement components, the locations and directions of which are specified by arguments **SEL_NODES** and **SEL_DOFS**, respectively, are stored at every load-step.

3.9.5.17 PATH_SCALE

This floating point number represents a scale factor to be applied to the incremental arclength (or pathlength) used for the first step of an analysis *re-start* run (default =1.0). If the default (1.0) is used, the arc-length increment from the previous step **BEG_STEP-1** will be used for the first step **BEG_STEP**. Note that this may lead to a different load-step size than if the analysis had continued to step **BEG_STEP** *without* a re-start. To eliminate this difference, the user can set:

[PATH_SCALE] = [DES_ITERS] / act_iters

where "act_iters" is the actual number of iterations required for convergence at step **[BEG_STEP]-1**.

3.9.5.18 SEL_DOFS

List of nodal degrees-of-freedom at which displacement histories are to be saved in dataset **NOMINAL_DS** (default =1.0). There should be **N_SELECT** numbers in the list, in correspondence with the node numbers specified by argument **SEL_NODES**. Values of each number in the list must range between 1 and 6, in correspondence to the nodal degree-of-freedom sequence (*e.g.*, $u, v, w, \theta_x, \theta_y, \theta_z$) specified by the **START** command of processor TAB.

3.9.5.19 SEL_NODES

List of node numbers at which displacement histories are to be saved in dataset **NOMINAL_DS** (default =0). There should be **N_SELECT** numbers in the list, and node numbers can be repeated if more than one nodal degree-of-freedom is to be saved at a node. The corresponding nodal degree-of-freedom for each entry is specified by argument **SEL_DOFS**.

3.9.5.20 TOL_E

Error tolerance used to establish convergence of the nonlinear equilibrium iteration procedure at each load step (default =1.e-3). The iteration loop at a given step is terminated whenever the following condition is met:

$$\epsilon \leq [\text{TOL\_E}]$$

where

$$\epsilon = \sqrt{\frac{\mathbf{r}^{(i)} \cdot \delta\mathbf{d}^{(i)}}{\mathbf{r}^{(1)} \cdot \delta\mathbf{d}^{(1)}}}$$

is the *relative energy error norm*, $\mathbf{r}$ is the residual force vector, $\delta\mathbf{d}$ is the iterative displacement change, and i is the iteration counter.

3.9.6 PROCEDURE FLOWCHART

NL_STABIL_1	(main procedure)
DEFNS	(define recursive macrosymbols)
INITIAL	(initial displacements and rotations)
ES	(initialize element data)
EXTRAP	(quadratic extrapolation for next load step)
STIFFNESS	(form and form stiffness matrix)
TANDIS	(solve for tangential displacements)
SOLVE	(solve linear system of equations)
RESIDUAL	(form residual-force vector)
ES	(calculate residual)
CHKCONV	(check convergence)
POSTSTEP	(print load step summary)
POSTRESS	(archive load step data)

3.9.7 LIMITATIONS

3.9.7.1 Number of Database Libraries

All analysis (computational) data is assumed to be resident on a single database library (*i.e.*, file), and that file is expected to be attached to logical device index 1 before calling procedure **NL_STATIC_1**. Additionally, important analysis statistics and re-start parameters, as well as selected post-processing data, will be deposited in a separate data library, which is automatically opened by procedure **NL_STATIC_1** on logical device index 3. The name of this secondary library is user-specified using procedure argument **NOMINAL_DB**, which has the default setting: **NOMINAL_DB = NOMINAL.GAL**.

3.9.7.2 Element Types

Only shell elements have been tested thus far with procedure **NL_STATIC_1**. While the procedure is potentially compatible with beam and solid elements too, minor modifications to the generic element processor may be required to handle geometric nonlinearity for these elements.

3.9.7.3 Number of Load/Constraint Systems

Only one set of loads and constraints is accommodated by procedure **NL_STATIC_1**. Furthermore, they must be referred to as load set 1 and constraint set 1 in the database.

3.9.7.4 Specified Rotations

Specified (non-zero) rotational freedoms are currently not implemented in procedure **NL_STATIC_1** – unless rotation angles remain moderately small (*i.e.*, less than 10 degrees). For larger rotations, the specification of rotation components constitutes a nonlinear constraint, which must be translated into the motion of nodal rotation triads, and requires a modification to the linearized equilibrium equations. This capability is planned for a future version of procedure **NL_STATIC_1**.

3.9.7.5 Material Nonlinearity

Only *geometric* nonlinearity is accounted for by procedure **NL_STATIC_1**. However, this limitation is really due to current limitations within the generic element processor; the global nonlinear solution algorithm doesn't particularly care about the *source* of the nonlinearity, as long as it is properly represented by the tangent stiffness matrix and the residual force vector.

3.9.7.6 Re-Starting from Step 1

It is currently not permitted to re-start the analysis from step 1 (*i.e.*, re-run the problem from the beginning) unless you either: i) delete the database and recreate the model, or ii) ***enable** the original APPL.MOTI and APPL.FORC datasets. The reason for this is that the original APPL.MOTI and APPL.FORC datasets are copied into new datasets – and marked for deletion – by procedure **NL_STATIC_1** whenever **BEG_STEP** = 1. This limitation will be removed in the next version of **NL_STATIC_1**.

3.9.7.7 EXTRAPOLATE Must be Turned On

Extrapolation is the only form of load/displacement step prediction that has been thoroughly tested. Without extrapolation, procedure **NL_STATIC_1** is supposed to use the arc-length constraint equation to generate a predictor solution (see theory section), and use information regarding the sign of the stiffness determinant to determine the load *direction* (increasing or decreasing) between one step and the next. However, the no-extrapolation option has not yet been quality assured. Note that the extrapolation option is the recommended approach anyway.

3.9.7.8 True-Newton Iteration

Only *modified* Newton/Raphson iteration is provided by the current version of procedure **NL_STATIC_1**. This means that the stiffness matrix is re-formed and re-factored only at the beginning (first iteration) of every **FAC_STEPS** load steps (where **FAC_STEPS** = 1 by default). In the next version, we plan to implement the option for *true* Newton/Raphson iteration, in which the stiffness matrix can be updated at each iteration of selected load steps. This option can be useful for problems with limit points that are nearly as sharp as bifurcation points.

3.9.7.9 Constant Load-Increments

Currently, all load increments (except the first one) are computed automatically by procedure **NL_STATIC_1**, using the arc-length constraint equation. Therefore, it is not possible for the user to fix the load increment, as is done in conventional load-controlled algorithms. Since this option may be useful for benchmarking (and research) purposes, we plan to include it in a future version of the procedure, wherein the arc-length constraint equation will be selectively bypassed.

3.9.7.10 Singularities due to Limit Points and Bifurcations

Procedure **NL_STATIC_1** cannot handle singularities in the stiffness matrix that arise when the load-stepping algorithm lands too close to a limit point, or attempts to traverse a bifurcation point. (Note: Singularities due to limit points are overcome by procedure **NL_STATIC_2**.)

3.9.8 ERROR MESSAGES AND WARNINGS

3.9.8.1 "Non-Convergence at Step *n*. Revise Strategy."

This message means that the maximum number of nonlinear iterations (**MAX_ITERS**) has been exhausted, as well as the maximum number of step cuts (**MAX_CUTS**), and convergence still hasn't been obtained at step n. A possible cure is to re-start the analysis from several steps back, and decrease the arc-length increment at that point (using the **PATH_SCALE** argument). However, just increasing **MAX_ITERS** or **MAX_CUTS**, or even **TOL_E**, may also solve the problem. In other words, re-think the definition of all solution parameters based on the observed behavior of the solution algorithm just prior to the break-down.

3.9.8.2 "Divergence at Step *n*. Revise Strategy."

This message has similar implications to the previous message, but it occurs when the error grows instead of decreases during two successive nonlinear iterations. The difference between divergence and non-convergence is that divergence cannot be cured by increasing MAX_ITERS; and probably should not be "cured" by increasing TOL_E. It generally means that the step-size is too big – or that the error tolerance (TOL_E) has been too big all along, so that changes are occurring suddenly that should have been detected by the solution algorithm at earlier load steps. Thus, you might try re-starting from an earlier step, reducing PATH_INC, and possibly reducing TOL_E as well.

3.9.8.3 "Specified Displacements are Identically Zero"

This is not necessarily an abortive error. As long as either nonzero specified displacements *or* specified forces are defined, the solution can proceed – in which case the message should be taken merely as a warning.

3.9.8.4 "Specified Forces are Identically Zero"

This is not necessarily an abortive error. As long as either nonzero specified displacements *or* specified forces are defined, the solution can proceed – in which case the message should be taken merely as a warning.

3.9.9 USAGE GUIDELINES AND EXAMPLES

Procedure NL_STATIC_1 may be used by preceding the procedure name by the *call directive. Procedure arguments may be changed from their default values by including any or all the arguments and their new values when the procedure is called. A space or blank is required between the end of the procedure name and the left parenthesis.

```
*call NL_STATIC_1 ( --
                    beg_step =1   ; -- . Starting step number (>0)
                    max_steps=1   ; -- . Maximum steps to compute
                    max_iters=9   ; -- . Maximum iterations per step
                    des_iters=4   ; -- . Number of iterations desired
                    fac_steps=1   ; -- . Steps_per_refactoring
                    max_cuts=3    ; -- . Maximum number of step cuts
                    tol_e=1.E-3   ; -- . Energy error tolerance
                    beg_load      ; -- . Starting load factor (>0.)
                    max_load      ; -- . Upper_bound on load factor
                    min_load      ; -- . Lower_bound on load factor
                    path_scale=1.; -- . Path_inc scl_factor (restart)
                    extrapolate=<true> ; --
                    line_search=1.;-- . Initial line-search parameter
                    debug=<FALSE>; -- . Debug_print switch
                    NL_GEOM = 2   ; -- . Geom. Nonlin. Level (1|2)
                    COROTATION=1  ; -- . Corotational Flag (leave on!!)
                    Nominal_DB = NOMINAL.GAL ; -- . Selected Output
                    Nominal_DS = RESPONSE.HISTORY ; --
                    N_SELECT ; SEL_NODES; SEL_DOFS )
```

3.9.9.1 Starting an Analysis

To begin a nonlinear static analysis with procedure **NL_STATIC_1**, it is only necessary that the finite element model be defined. This does not require pre-formation of element stiffness matrices, node renumbering for optimal factorization time, or any form of linear analysis (unless initial geometric imperfections are based on linear displacement modes). Only nodal coordinates/transformations, material properties and element connectivity are pre-requisite to nonlinear analysis. To invoke procedure **NL_STATIC_1**, only those arguments that don't have default values (see procedure usage section) need be specified.

For example, suppose you wanted to start an analysis with an initial load-factor of .1, a maximum load-factor of 1., a minimum load-factor of 0., and compute no more than 20 load-steps. You could then invoke the procedure as follows:

```
*call NL_STATIC_1 ( BEG_STEP = 1 ; MAX_STEPS = 20 ; --
                    BEG_LOAD = .1 ; MAX_LOAD = 1.  ; MIN_LOAD = 0.  )
```

Keep in mind that the number of load-steps actually performed during the above run will depend on the number of adaptively-sized arc-length increments needed to attain the maximum load level. Since it may be difficult to estimate this in advance, you may want to start with only a few load-steps (*e.g.*, set **MAX_STEPS** = 3) to get some experience, and later re-start the analysis with more steps allowed.

3.9.9.2 Re-Starting (or Continuing) an Analysis

To re-start an analysis from a previously computed, converged solution, you need only specify the first step to compute as argument **BEG_STEP**. If the **EXTRAPOLATE** argument is activated (which is the default), the three solution steps immediately preceding the one you wish to compute must be resident on the database. If the **EXTRAPOLATE** argument is turned off, only one preceeding solution step is required. Thus, to continue an analysis that has already produced ten load steps, you could issue the call:

```
*call NL_STATIC_1 ( BEG_STEP = 11; MAX_STEPS = 100 ; --
                    MAX_LOAD = 1.  ; MIN_LOAD = 0.  )
```

Note that the argument **BEG_LOAD** is not necessary for a continuation (or re-start) run; the new load level is automatically computed using the preceeding value of the arc-length increment. However, you can modify the new increment, by using the **PATH_SCALE** argument, which is a scale factor applied to the previous arc-length increment to generate the new increment. The default value of **PATH_SCALE** is 1.0.

Furthermore, on analysis continuations/restarts, you are free to modify any of the other solution parameters (*i.e.*, procedure arguments). For example, you may want to increase MAX_ITERS, reduce TOL_E, or even change DES_ITERS. The default values for these solution parameters are not suitable for all problems; they are useful primarily for gaining experience by trial and error.

3.9.10 PROCEDURE LISTING

```
. =DECK NL_STATIC_1
*procedure NL_STATIC_1 ( --
                    beg_step =1  ; -- . Starting step number (>0)
                    max_steps=1  ; -- . Maximum steps to compute
                    max_iters=9  ; -- . Maximum iterations per step
                    des_iters=4  ; -- . Number of iterations desired
                    fac_steps=1  ; -- . Steps_per_refactoring
                    max_cuts=3   ; -- . Maximum number of step cuts
                    tol_e=1.E-3  ; -- . Energy error tolerance
                    beg_load     ; -- . Starting load factor (>0.)
                    max_load     ; -- . Upper_bound on load factor
                    min_load     ; -- . Lower_bound on load factor
                    path_scale=1.; -- . Path_inc scl_factor (restart)
                    extrapolate=<true> ; --
                    line_search=1.;-- . Initial line-search parameter
                    debug=<FALSE>; -- . Debug_print switch
                    NL_GEOM = 2  ; -- . Geom. Nonlin. Level (1|2)
                    COROTATION=1 ; -- . Corotational Flag (leave on!!)
                    Nominal_DB = NOMINAL.GAL ; -- . Selected Output
                    Nominal_DS = RESPONSE.HISTORY ; --
                    N_SELECT ; SEL_NODES; SEL_DOFS  --
                  )
 *Remark  ------------------------------------------------------------
 *Remark |                                                            |
 *Remark                      N L _ S T A T I C _ 1
 *Remark
 *Remark           CSM Testbed Procedure for Nonlinear Statics:
 *Remark
 *Remark              o  Corotational Newton/Raphson algorithm
 *Remark              o  Linearized Crisfield/Riks arc-length control
 *Remark              o  Applied forces and/or displacements
 *Remark
 *Remark           Authors: G.M. Stanley and C.C. Rankin
 *Remark           Version: MAR-03-1988
 *Remark |                                                            |
 *Remark  ------------------------------------------------------------
```

3.9.11 REFERENCES

3.9-1. Crisfield, M. A.: "A Fast Incremental/Iterative Solution Procedure that Handles Snap-Through." *Computers and Structures*, Vol. 13, 1983, pp. 55-62.

3.9-2. Kane, T. R.; Likins, P. W.; and Levinson, D. A.: *Spacecraft Dynamics*, McGraw-Hill Book Co., New York, 1983.

3.9-3. Argyris, J. H.: "An Excursion into Large Rotations." *International Journal for Numerical Methods in Engineering*, Vol. 32, 1982, pp. 85-155.

3.9-4. Rankin, C. C. and Brogan, F. A.: "An Element-Independent Corotational Procedure for the Treatment of Large Rotations." In *Collapse Analysis of Structures*, edited by Sobel, L.H. and Thomas, K., ASME, New York, 1984, pp. 85-100.

3.9-5. Stewart, Caroline B.: *The Computational Structural Mechanics Testbed User's Manual.* NASA TM-100644, October 1989.

3.9-6. Stewart, Caroline B.: *The Computational Structural Mechanics Testbed Data Library Description.* NASA TM-100645, October 1988.

3.10 Procedure NL_STATIC_2

3.10.1 GENERAL DESCRIPTION

NL_STATIC_2 is a modification to NL_STATIC_1 Riks arclength strategy that allows solution of a limit point problem in the immediate neighborhood of a critical point. Developed and written by C. C. Rankin in consultation with E. Riks, this modified algorithm avoids the singularity in the stiffness matrix at the critical point, thereby permitting a smooth continuation of the solution even when a solution point *coincides exactly* with the critical point. NL_STATIC_2 is preferred over NL_STATIC_1 whenever solution points are desired very near the critical area, or when the singularity of the stiffness is likely to persist over some distance along the solution path. In the absence of numerical difficulties, however, NL_STATIC_1 and NL_STATIC_2 produce *identical* results.

3.10.2 PROCEDURE USAGE

Procedure NL_STATIC_2 may be invoked by the ***call** directive:

```
*call NL_STATIC_2 ( arg1 = val1; arg2 = val2; ...)
```

where **arg***i* are argument names and **val***i* are the corresponding values you wish to give them. The following are valid arguments for NL_STATIC_1; note that those without default values are mandatory, while the others are optional.

Argument	Default Value	Meaning
BEG_LOAD	–	Starting load factor (>0.)
BEG_STEP	–	Starting step number (>0)
MAX_LOAD	–	Upper_bound on load factor
MIN_LOAD	–	Lower_bound on load factor
MAX_STEPS	–	Maximum steps to compute
COROTATION	1	Corotational Update Option
DES_ITERS	4	Number of iterations per step desired
EXTRAPOLATE	<true>	Perform quadratic extrapolation of solution
FAC_STEPS	1	Steps_per_refactoring
MAX_ITERS	9	Maximum iterations per step
MAX_CUTS	3	Maximum number of successive step cuts
NL_GEOM	2	Geometric Nonlinearity Level (1 or 2)
NOMINAL_DB	NOMINAL.GAL	Results database file
NOMINAL_DS	RESPONSE.HISTORY	Results dataset
N_SELECT	0	Number of nodes for selected disp. output
PATH_SCALE	1.	Arclength scale factor for restarts
SEL_NODES	0	List of nodes for selected output
SEL_DOFS	0	Corresponding list of nodal freedoms (1–6)
TOL_E	1.E-3	Relative error tolerance in energy norm
ADV_RIKS	<false>	If true, advanced Riks; else, do NL_STATIC_1

3.10.3 ARGUMENT GLOSSARY

The only additional argument that NL_STATIC_2 has is ADV_RIKS. All other arguments are identical in name and function as their counterparts in NL_STATIC_1.

3.10.3.1 ADV_RIKS

This is a logical parameter, that, when set to <true>, invokes the advanced Riks option. Otherwise, NL_STATIC_2 reduces to NL_STATIC_1.

3.10.4 INPUT/OUTPUT DATASETS

The input/output requirements of NL_STATIC_2 are identical to NL_STATIC_1.

3.10.5 SUB-PROCEDURES AND PROCESSORS

Table 3.10-2 Sub-Procedures Invoked by Procedure NL_STATIC_2

Procedure	*Type*	*Function*
NL_STATIC_2	Internal	Main Procedure
CHK_CONV	Internal	Check convergence
DEFNS	Internal	Defines recursive macrosymbols
ES	External	Element utility Procedure
FACT_STIFF	Internal	Factors Stiffness Matrix
FORM_STIFF	Internal	Forms Stiffness Matrix
RESIDUAL	Internal	Forms residual-force vector
SOLVE	Internal	Solves linear equation systems
STIF_COLMN	Internal	Fixes Freedom *I*, Extracts Stiffness Col. *I*
TWO_BY_TWO	Internal	Computes Advanced Riks Reduced Coefficients

Table 3.10-3 Processors Invoked by Procedure NL_STATIC_2

Procedure	*Type*	*Function*
AUS	Internal	Extracts column *I* of Stiffness
E	Internal	Initializes EFIL datasets
ES*i*	External	Element Processors based on GEP
INV	Internal	Factors stiffness matrix
SSOL	Internal	Solves linear equation systems
VEC	Internal	Performs all vector/pseudo-vector algebra

3.10.6 LIMITATIONS

NL_STATIC_2 has the all the limitations that are listed in Section 4.1.6. However, whereas numerical difficulties can be expected with NL_STATIC_1 near a critical point, no difficulties whatsoever should be expected with NL_STATIC_2 near *limit* (as opposed to bifurcation points). There are two additional software limitations described in the next two subsections.

3.10.6.1 AUS Limitation

Due to hard wired data set naming conventions in processor AUS for system matrices (*e.g.* K.SPAR.36), the number of DOF's per node specified on the START command of processor TAB is presently restricted to be 6. AUS is currently essential to NL_STATIC_2 for computing the product of the stiffness with an elementary vector to extract row I of the stiffness matrix. This restriction will be removed when we make the connection to the Generic Matrix Processor.

3.10.6.2 First Two Steps

Because of the way the extrapolation is set up, NL_STATIC_2 operates as NL_STATIC_1 for the first two steps for a new (not restart) analysis. This limitation is not important, since it is highly unlikely that a critical point will be reached at the very beginning of a solution sequence.

3.10.7 ERROR MESSAGES

All error messages have the same meaning as in NL_STATIC_1 (section 4.1.7).

3.10.8 USAGE GUIDELINES AND EXAMPLES

As they are identical to NL_STATIC_1, please consult section 4.1.8 for usage guidelines.

3.10.9 THEORY

3.10.9.1 Introduction

Procedure NL_STATIC_2, like NL_STATIC_1, performs a quasi-static analysis of a system of nonlinear equilibrium equations using an adaptive arclength-controlled Newton/Raphson incremental/iterative solution algorithm. The function of the arclength constraint, the stepsize selection, and convergence criteria are identical to NL_STATIC_1. Because almost all of the theory underlying NL_STATIC_2 is covered in Section 4.1.9, only the matrix partitioning and pivoting process that avoids the factoring of a singular or near singular stiffness matrix is covered here.

3.10.9.2 The Block Pivot Strategy

The difficulties that occur in NL_STATIC_1 can be readily seen by examining Eq. (8).

$$\begin{aligned} \overline{\mathbf{K}}\,\delta\mathbf{d} &= ???rbbold + \delta\lambda\,\widehat{\mathbf{f}}^{ext} \\ 2\Delta\overline{\mathbf{d}}\cdot\delta\mathbf{d} &= \overline{c} \end{aligned} \qquad (3.10-1)$$

where $\overline{\mathbf{r}}$ is the residual, $\widehat{\mathbf{f}}^{ext}$ is the external load, $\overline{c}$ is the arclength parameter, and $\delta\overline{\mathbf{d}}$ is the desired displacement increment. Clearly, if $\overline{\mathbf{K}}$ is singular, we will have a problem in solving this pair of equations. For *limit points*, however, the extended equation system containing the constraint (second of (57)) is non-singular. A simple and robust method of overcoming the singularity is to select out the equation for a particular freedom and treat it the same way as the constraint equation, solving *three* rather than *two* sets of equations, as illustrated by the following

$$\begin{bmatrix} \widehat{\mathbf{K}} & \widehat{\mathbf{f}}^{I} & -\widehat{\mathbf{f}}^{ext} \\ (\widehat{\mathbf{f}}^{I})^{T} & K_{II} & -f_{I}^{ext} \\ \widehat{\mathbf{c}} & c_{I} & 0 \end{bmatrix} \begin{bmatrix} \delta\widehat{\mathbf{d}} \\ \delta d_{I} \\ \delta\lambda \end{bmatrix} = \begin{bmatrix} \widehat{\mathbf{r}} \\ r_{I} \\ \overline{c} \end{bmatrix} \qquad (3.10-2)$$

where $\widehat{\mathbf{K}}$ is the stiffness matrix with row and column I removed. The hats over vectors also indicate that row I has been deleted. $\widehat{\mathbf{f}}^{I}$ is column I of the stiffness, with K_{II} being the diagonal element for freedom I. The same partitioning also applies to the external force $\mathbf{f}^{ext}$ and the constraint equation $\mathbf{c}$.

If we formally solve the first of (58) for $\delta\widehat{\mathbf{d}}$ and substitute the result into the remaining two equations, we obtain the following two-by-two nonsymmetric system of equations for δd_I and $\delta\lambda$:

$$\begin{bmatrix} K_{II} - (\widehat{\mathbf{f}}^{I})^{T}\widehat{\mathbf{K}}^{-1}\widehat{\mathbf{f}}^{I} & (\widehat{\mathbf{f}}^{I})^{T}\widehat{\mathbf{K}}^{-1}\widehat{\mathbf{f}}^{ext} - f_{I}^{ext} \\ c_{I} - \widehat{\mathbf{c}}^{T}\widehat{\mathbf{K}}^{-1}\widehat{\mathbf{f}}^{I} & \widehat{\mathbf{c}}^{T}\widehat{\mathbf{K}}^{-1}\widehat{\mathbf{f}}^{ext} \end{bmatrix} \begin{bmatrix} \delta d_{I} \\ \delta\lambda \end{bmatrix} = \begin{bmatrix} r_{I} - (\widehat{\mathbf{f}}^{I})^{T}\widehat{\mathbf{K}}^{-1}\widehat{\mathbf{r}} \\ \overline{c} - \widehat{\mathbf{c}}^{T}\widehat{\mathbf{K}}^{-1}\widehat{\mathbf{r}} \end{bmatrix} \qquad (3.10-3)$$

One additional forward/backsolve is required with this algorithm. After each *factoring* of the stiffness matrix, one must solve the pair

$$\begin{aligned} \widehat{\mathbf{K}}\delta\widehat{\mathbf{v}} &= \widehat{\mathbf{f}}^{I} \\ \widehat{\mathbf{K}}\delta\widehat{\mathbf{w}} &= \widehat{\mathbf{f}}^{ext} \end{aligned} \qquad (3.10-4)$$

for vector $\delta\widehat{\mathbf{v}}$ and $\delta\widehat{\mathbf{w}}$. $\delta\widehat{\mathbf{w}}$ corresponds to $\delta\widehat{\mathbf{d}}$ in the second of (11). $\delta\widehat{\mathbf{d}}$ is also used for $\delta\widehat{\mathbf{w}}$ in the description of the NL_STATIC_2 algorithm. The following vector must be solved *every iteration*:

$$\widehat{\mathbf{K}}\widehat{\mathbf{u}} = \widehat{\mathbf{r}} \qquad (3.10-5)$$

This equation corresponds to the first of (11). The two-by-two system (59) becomes

$$\begin{bmatrix} K_{II} - (\widehat{\mathbf{f}}^I)^T \delta\widehat{\mathbf{v}} & (\widehat{\mathbf{f}}^I)^T \delta\widehat{\mathbf{w}} - f_I^{ext} \\ c_I - \widehat{\mathbf{c}}^T \delta\widehat{\mathbf{v}} & \widehat{\mathbf{c}}^T \delta\widehat{\mathbf{w}} \end{bmatrix} \begin{bmatrix} \delta d_I \\ \delta\lambda \end{bmatrix} = \begin{bmatrix} r_I - (\widehat{\mathbf{f}}^I)^T \widehat{\mathbf{u}} \\ \bar{c} - \widehat{\mathbf{c}}^T \widehat{\mathbf{u}} \end{bmatrix} \qquad (3.10-6)$$

$$\begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} \begin{bmatrix} \delta d_I \\ \delta\lambda \end{bmatrix} = \begin{bmatrix} R_1 \\ R_2 \end{bmatrix}$$

We have rewritten the first of (62) with the notation used in the description of the algorithm, where the coefficients A_{ij} and R_i are the reduced Riks coefficients and residuals, respectively; a comparison of the first of (62) with the second yields their values.

Once δd_I and $\delta\lambda$ are known, the full solution becomes the sum

$$\widehat{\delta\mathbf{d}} = \widehat{\mathbf{u}} - \delta d_I \delta\widehat{\mathbf{v}} + \delta\lambda \delta\widehat{\mathbf{w}} \qquad (3.10-7)$$

$\widehat{\delta\mathbf{d}}$ is then expanded by one freedom and the explicit value $\delta\mathbf{d}_I$ previously solved for is inserted. The advantage of (62) is that this system, in contrast to (8) is never singular at a limit point, provided that the component I does not correspond to a zero entry in the tangent displacement vector at the critical point.

3.10.9.3 Selecting Freedom I and the Vector $\widehat{\mathbf{f}}^I$

A simple method for ensuring a nonsingular reduced stiffness $\widehat{\mathbf{K}}$ is to look at the difference between the last converged solution and the predicted solution for the current step. We choose that particular component with the largest absolute value; its index becomes I, and the stiffness column belonging to that freedom becomes $\widehat{\mathbf{f}}^I$ with diagonal K_{II}. In practice, after we have selected the proper index, we generate an elementary vector that is zero except for unity in position I. $\mathbf{f}^I$ is just the product of the stiffness matrix with the elementary vector. $\widehat{\mathbf{f}}^I$ is this same column with row I deleted.

3.10.9.4 Constructing the Solution

Immediately after the stiffness matrix has been factored, we solve (60) for $\delta\widehat{\mathbf{v}}$ and $\delta\widehat{\mathbf{w}}$. Solution for $\delta\widehat{\mathbf{w}}$ is treated like the corresponding solution for $\delta\mathbf{d}$ in (37), with specified displacements being handled exactly as in NL_STATIC_1. The only difference is that the new stiffness has one additional freedom held fixed. In practice, entry I is added to the specified displacement list (and if I has changed since the last step, *the previous freedom is "released" or "unfixed"*). The solutions $\delta\widehat{\mathbf{v}}, \delta\widehat{\mathbf{w}}$ are full system vectors with a zero in the Ith slot. At each iteration, a new basic displacement increment $\widehat{\mathbf{u}}$ is also solved for (62). After solution of the reduced two-by-two system (62), (63) is used to construct the full displacement increment. This is followed by direct insertion of the increment for component I.

3.10.9.5 Computing the Stiffness Determinant

The determinant of the stiffness $\widehat{\mathbf{K}}$ is not equal to that of the original, unconstrained stiffness. However, since our algorithm is a special case of a block Gauss elimination,it is a simple matter to compute it. The full determinant of the stiffness matrix is the product

$$\text{Det}(\mathbf{K}) = \text{Det}(\widehat{\mathbf{K}})A_{11} \qquad (3.10-8)$$

Note that when A_{11} vanishes, the stiffness determinant also vanishes. However, the system (62) is still nonsingular whenever the off diagonal reduced coefficients are nonzero (the extended system).

3.10.10 ALGORITHM FLOW CHART

General Algorithm

(1) STEP LOOP: n = 2, 3, 4 ... (step = n+1)

(1.1) Extrapolate Solution for Predictor

call EXTRAP ($\Delta\ell_{n+1}, \Delta\ell_n, \Delta\ell_{n-1} \rightarrow c_n, c_{n-1}, c_{n-2}$)

$$\lambda_{n+1}^{(1)} = c_n\lambda_n + c_{n-1}\lambda_{n-1} + c_{n-2}\lambda_{n-2}$$
$$\mathbf{d}_{n+1}^{(1)} = c_n\mathbf{d}_n + c_{n-1}\mathbf{d}_{n-1} + c_{n-2}\mathbf{d}_{n-2}$$

$$\Delta\mathbf{d}_{n+1}^{(1)} = \mathbf{d}_{n+1}^{(1)} - \mathbf{d}_n \qquad (\mathbf{T}_{n+1}^{(1)} = \mathbf{R}(\Delta\mathbf{d}_{n+1}^{(1)})\mathbf{T}_n)$$
$$\Delta\lambda_{n+1}^{(1)} = \lambda_{n+1}^{(1)} - \lambda_n$$

(1.2) Form and Assemble Stiffness

(1.3) Find Index of Largest Tangent Component
Extract Column I from Stiffness (including diagonal)

$$I = \text{Arg}\{\max_{\forall k} |\Delta\mathbf{d}(k)_{n+1}^{(1)}|\}$$
$$\hat{\mathbf{f}}^I = \mathbf{K}\mathbf{e}_I$$

(1.4) Fix freedom I to give constrained $\mathbf{K}$ ($\hat{\mathbf{K}}$)

(1.5) Solve for Tangential Displacement based on Predictor

$$\boxed{\delta\hat{\mathbf{d}} = \hat{\mathbf{K}}^{-1}(\mathbf{d}_{n+1}^{(1)})\left[\hat{\mathbf{f}}^{ext} + \mathbf{K}^{us}\hat{\mathbf{d}}^s\right]}$$

(1.6) Solve for perturbation displacement $\delta\hat{\mathbf{v}}$

$$\boxed{\delta\hat{\mathbf{v}} = \hat{\mathbf{K}}^{-1}(\mathbf{d}_{n+1}^{(1)})\,\hat{\mathbf{f}}^I}$$

(1.7) Solve for Riks reduced system coefficients

$$\delta\hat{v}_I = 0.$$
$$\delta\hat{\mathbf{d}}_I = 0.$$
$$A_{11} = \hat{\mathbf{f}}_I^I - \hat{\mathbf{f}}^I \cdot \delta\hat{\mathbf{v}}$$
$$A_{12} = -\hat{\mathbf{f}}_I^{ext} + \hat{\mathbf{f}}^I \cdot (\delta\hat{\mathbf{d}} - \hat{\mathbf{d}}^s)$$
$$A_{21} = 2(\Delta\mathbf{d}(I)_{n+1}^{(1)} - \Delta\mathbf{d}_{n+1}^{(1)} \cdot \delta\hat{\mathbf{v}})$$
$$A_{22} = 2\Delta\mathbf{d}_{n+1}^{(1)} \cdot \delta\hat{\mathbf{d}}$$

(1.7) Form Residual based on Predictor

$$\mathbf{r}_{n+1}^{(1)} = \mathbf{r}(\mathbf{d}_{n+1}^{(1)}, \lambda_{n+1}^{(1)}) = \lambda_{n+1}^{(1)} \widehat{\mathbf{f}}^{ext} - \mathbf{f}^{int}(\mathbf{d}_{n+1}^{(1)})$$

call CHK_CONV ($\mathbf{r}_{n+1}^{(1)}, \Delta\mathbf{d}_{n+1}^{(1)}, \epsilon_{ref} \rightarrow \widehat{\epsilon}^{(1)}, \epsilon_{ref}$)

(2) ITERATION LOOP: i = 1, 2, ... (iter = i+1 = 2, 3, ...)

(2.1) Solve for Basic Iterative Displacement Change

$$\boxed{\delta\overline{\mathbf{d}} = \widehat{\mathbf{K}}^{-1}(\mathbf{d}_{n+1}^{(1)})\, \mathbf{r}_{n+1}^{(i)}}$$

(2.2) Solve for Reduced r.h.s.

$\delta\overline{d}_I = 0.$

$R_1 = \mathbf{r}(I)_{n+1}^{(i)} - \widehat{\mathbf{f}}^I \cdot \delta\overline{\mathbf{d}}$

$R_2 = \Delta\ell^2 - \Delta\mathbf{d}_{n+1}^{(i)} \cdot \Delta\mathbf{d}_{n+1}^{(i)} - 2\Delta\mathbf{d}_{n+1}^{(i)} \cdot \delta\overline{\mathbf{d}}$

(2.3) Solve 2 x 2 Reduced system:

$$\begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} \begin{bmatrix} \delta d_I \\ \delta\lambda \end{bmatrix} = \begin{bmatrix} R_1 \\ R_2 \end{bmatrix}$$

(2.4) Update Displacements and Load Factor

$$\begin{aligned} \delta\mathbf{d} &= \delta\overline{\mathbf{d}} + \delta\lambda\,\delta\widehat{\mathbf{d}} - \delta d_I \delta\widehat{\mathbf{v}} \\ \mathbf{d}_{n+1}^{(i+1)} &= \mathbf{d}_{n+1}^{(i)} + \delta\mathbf{d} \qquad (\mathbf{T}_{n+1}^{(i+1)} = \mathbf{R}(\delta\mathbf{d})\,\mathbf{T}_{n+1}^{(i)}) \\ \Delta\mathbf{d}_{n+1}^{(i+1)} &= \Delta\mathbf{d}_{n+1}^{(i)} + \delta\mathbf{d} \\ \lambda_{n+1}^{(i+1)} &= \lambda_{n+1}^{(i)} + \delta\lambda \end{aligned}$$

(2.4) Compute New Residual

$$\mathbf{r}_{n+1}^{(i+1)} = \mathbf{r}(\mathbf{d}_{n+1}^{(i+1)}, \lambda_{n+1}^{(i+1)}) = \lambda_{n+1}^{(i+1)} \widehat{\mathbf{f}}^{ext} - \mathbf{f}^{int}(\mathbf{d}_{n+1}^{(i+1)})$$

(2.5) Check Convergence

call CHK_CONV ($\mathbf{r}_{n+1}^{(i+1)}, \delta\mathbf{d}, \widehat{\epsilon}^{(i)}, \epsilon_{ref}, \widehat{\epsilon}_{tol}$, num_div $\rightarrow$
$\widehat{\epsilon}^{(i+1)}$, <CONVERGED>, <DIVERGED>)

if (<CONVERGED>) then

num_iters_required = iter

$\Delta\ell_{n+2} = \Delta\ell_{n+1}\left(\frac{\text{num\_iters\_desired}}{\text{num\_iters\_required}}\right)$

$n \leftarrow n + 1$

go to (1) STEP LOOP

elseif (<DIVERGED> .or. (iter > max_iters)) then

if (num_cuts < max_cuts) then

$\Delta\ell_{n+1} = \Delta\ell_{n+1}/2$

num_cuts = num_cuts + 1

go to (1.3)

else

STOP

endif

else

$i \leftarrow i + 1$

go to (2) ITER LOOP

endif

Starting Procedure: Step 1 (n=0)

Replace Algorithm Steps (1.1)-(1.2) by:

$$\mathbf{d}_1^{(0)} = \mathbf{0}$$

$$\delta\hat{\mathbf{d}} = \mathbf{K}^{-1}(\mathbf{d}_1^{(0)}) \left[\hat{\mathbf{f}}^{ext} + \mathbf{K}^{us}\hat{\mathbf{d}}^{s}\right]$$

$$\begin{aligned} \lambda_1^{(1)} &= \lambda_{start} \qquad \text{(user specified)} \\ \Delta\lambda_1^{(1)} &= \lambda_1^{(1)} \\ \Delta\ell_1 &= \Delta\lambda_1^{(1)} \|\delta\hat{\mathbf{d}}\| \\ \Delta\mathbf{d}_1^{(1)} &= \Delta\lambda_1^{(1)} \delta\hat{\mathbf{d}} \\ \mathbf{d}_1^{(1)} &= \Delta\mathbf{d}_1^{(1)} \end{aligned}$$

(Also, form $\mathbf{K}(\mathbf{d}_1^{(1)})$ before next solve.)

Starting Procedure: Step 2 (n=1)

Replace Algorithm Steps (1.1)-(1.2) by:

$$\mathbf{d}_2^{(0)} = \mathbf{d}_1$$

$$\delta\hat{\mathbf{d}} = \mathbf{K}^{-1}(\mathbf{d}_2^{(0)}) \left[\hat{\mathbf{f}}^{ext} + \mathbf{K}^{us}\hat{\mathbf{d}}^{s}\right]$$

$$\begin{aligned} \Delta\lambda_2^{(1)} &= \Delta\ell_2 / \|\delta\hat{\mathbf{d}}\| \\ \lambda_2^{(1)} &= \lambda_1 + \Delta\lambda_2^{(1)} \\ \Delta\mathbf{d}_2^{(1)} &= \Delta\lambda_2^{(1)} \delta\hat{\mathbf{d}} \\ \mathbf{d}_2^{(1)} &= \mathbf{d}_1 + \Delta\mathbf{d}_2^{(1)} \qquad (\mathbf{T}_2^{(1)} = \mathbf{R}(\Delta\mathbf{d}_2^{(1)}\, \mathbf{T}_1) \end{aligned}$$

(But don't reform $\mathbf{K}(\mathbf{d}_1^{(1)})$ until step 3.)

NOTATION

$\mathbf{d}$	Displacement vector.
$\delta\mathbf{d}$	Iterative change in $\mathbf{d}$.
$\Delta\mathbf{d}$	Incremental (load-step) change in $\mathbf{d}$.
$\mathbf{d}_n^i$	Displacement vector at iteration i of step n.
$\Delta\mathbf{d}_n^i$	Incremental (step) change in $\mathbf{d}$. $\Delta\mathbf{d}_{n+1}^{(i+1)} = \mathbf{d}_{n+1}^{(i)} - \mathbf{d}_n$
$\delta\widehat{\mathbf{d}}$	Tangential displacement $= \mathbf{K}^{-1}\widehat{\mathbf{f}}^{ext}$
$\Delta\ell$	Incremental arclength (step) parameter.
$\widehat{\mathbf{f}}^{ext}$	External force vector — base load.
$\mathbf{f}^{int}$	Internal force vector.
$\widehat{\mathbf{f}}^I$	I'th column of $\mathbf{K}$
$\mathbf{r}$	Residual force vector.
$\mathbf{K}$	Stiffness matrix.
$\widehat{\mathbf{K}}$	Stiffness matrix with I'th row and column deleted
$\widehat{\mathbf{v}}$	Solution vector corresponding to $\widehat{\mathbf{K}}^{-1}\widehat{\mathbf{f}}^I$
$\delta\overline{\mathbf{d}}$	Basic (fixed-load) iterative change in $\mathbf{d}$¿
$\delta\mathbf{d}^*$	Linear combination of $\delta\overline{\mathbf{d}}$ and $\delta\widehat{\mathbf{d}}$.
A_{ij}	Advanced-Riks reduced system coefficients
R_i	Advanced-Riks reduced residuals
num_cuts	Number of times load step has been cut in half at current step.
num_div	Number of consecutive iterations at which divergence occurs.
c_n	Extrapolation coefficient corresponding to step n.
$\widehat{\epsilon}$	Relative error in energy norm.
ϵ	Absolute error in energy norm.
ϵ_{ref}	Reference value of ϵ; initialized as zero.
ϵ_{tol}	Relative error tolerance (default: 10^{-4}).
λ	Load factor.
$\Delta\lambda$	Incremental (load-step) change in λ.
$\mathbf{K}^{us}$	Stiffness submatrix coupling specified (s) displacement components with force components corresponding to unknown (u) displacements.
$\widehat{\mathbf{d}}^s$	Base value of specified displacement vector.
$\mathbf{d}^s$	Current value of specified displacement vector. $\mathbf{d}^s = \lambda\widehat{\mathbf{d}}^s$.

3.10.11 PROCEDURE LISTING

```
*procedure NL_STATIC_2 ( --
                      beg_step =1  ; -- . Starting step number (>0)
                      max_steps=1  ; -- . Maximum steps to compute
                      max_iters=9  ; -- . Maximum iterations per step
                      des_iters=4  ; -- . Number of iterations desired
                      fac_steps=1  ; -- . Steps_per_refactoring
                      max_cuts=3   ; -- . Maximum number of step cuts
                      tol_e=1.E-3  ; -- . Energy error tolerance
                      beg_load     ; -- . Starting load factor (>0.)
                      max_load     ; -- . Upper_bound on load factor
                      min_load     ; -- . Lower_bound on load factor
                      path_scale=1.; -- . Path_inc scl_factor (restart)
                      extrapolate=<true> ; --
                      line_search=1.;-- . Initial line-search parameter
                      debug=<FALSE>; -- . Debug_print switch
                      NL_GEOM = 2  ; -- . Geom. Nonlin. Level (1|2)
                      COROTATION=1 ; -- . Corotational Flag (leave on!!)
                      adv_riks = <false> ; -- . Advanced RIKS flag.
                      Nominal_DB = NOMINAL.GAL ; -- . Selected Output
                      Nominal_DS = RESPONSE.HISTORY ; --
                      N_SELECT ; SEL_NODES; SEL_DOFS  --
                     )
 *Remark  ------------------------------------------------------------
 *Remark |                                                            |
 *Remark                    N L _ S T A T I C _ 1
 *if < [adv_riks] > /then
    *remark                   A D V A N C E D   R I K S
 *endif
 *Remark
 *Remark           CSM Testbed Procedure for Nonlinear Statics:
 *Remark
 *Remark              o  Corotational Newton/Raphson algorithm
 *Remark              o  Linearized Crisfield/Riks arc-length control
 *Remark              o  Applied forces and/or displacements
 *Remark
 *Remark           Authors: G.M. Stanley and C.C. Rankin
 *Remark           Version: MAR-03-1988
 *Remark |                                                            |
 *Remark  ------------------------------------------------------------
.
.
. INITIALIZATION
.
 *def/i ns_overwrite == <true>
 *def/i debug        == [debug]
 *def/i max_step     == <[beg_step]+[max_steps]-1>
 *def/i num_iters    == [des_iters]
 *def/d path_scale   == [path_scale]
 *def/i extrapolate  == [extrapolate]
 *def/a NOM_DB       == [Nominal_DB]
 *def/a NOM_DS       == [Nominal_DS]
```

```
*def/i N_SELECT      == [N_SELECT]
*def/i SEL_NODES[1:<N_SELECT>] == [SEL_NODES]
*def/i SEL_DOFS[1:<N_SELECT>]  == [SEL_DOFS]
*def/i NL_GEOM       == [NL_GEOM]
*def/i COROTATION    == [COROTATION]
*CALL DEFNS ( STEP = [BEG_STEP]; ITER = 0 )
*IF < [BEG_STEP] /EQ 1 > /THEN
   *Remark INITIALIZATION:
   *call INITIAL
*ENDIF
.
. BEGIN STEP_LOOP
.
*DO :STEP_LOOP $n = 0, <[max_steps]-1>
    *def/i step == < [beg_step]+<$n> >
    *def/i pass == 1
    *def/i tot_iters == 1
    *if < <<step> /gt 2> /and <<extrapolate> /eq <true>> > /then
       *def/i extrap_this_step == <true>
    *else
       *def/i extrap_this_step == <false>
    *endif
    *Remark
    *remark ------------------------------------------------------------
    *remark
    *Remark       BEGINNING STEP <STEP>
    *remark ------------------------------------------------------------
    *remark
    *call DEFNS ( step = <step>; iter = 1 )
    :NEW_PASS           . Re-entry point for step-size reduction
    [XQT VEC
   *if < <<step> /gt 1> /and <<pass> /eq 1> > /then
     *def/d path_inc  == < <path_scale>*<path_inc_n> >
     *Remark New PATH INCREMENT = <path_inc>
   *endif
.
. . . Generate Displacement PREDICTOR
.
    *if <extrap_this_step> /then
       *Remark Predicting displacements/load via EXTRAPOLATION
       *call EXTRAP ( dx_np1 = <path_inc>    ; cx_n = cx_n    ; --
                      dx_n   = <path_inc_n>  ; cx_nm1 = cx_nm1; --
                      dx_nm1 = <path_inc_nm1>; cx_nm2 = cx_nm2  )
       *def/d lam_np1_i ==  <  (<cx_n>*<lam_n>)     --
                             + (<cx_nm1>*<lam_nm1>) --
                             + (<cx_nm2>*<lam_nm2>) >
        <d_np1_i>  <-  <cx_n> <d_n> +  <cx_nm1> <d_nm1>
        <d_np1_i>  <-    <d_np1_i>  +  <cx_nm2> <d_nm2>
        <d_inc_i>  <-    <d_np1_i>  -  <d_n>
         ROTATE   <T_n> * <d_inc_i>  ->  <T_np1_i>
    *else
       *Remark Using Previous Step as Displacement Predictor
       *remark
```

```
      *remark      Last displacement dataset : <d_n>
      *remark
      *def lam_np1_i == <lam_n>
       <d_np1_i>  <-  <d_n>
       <T_np1_i>  <-  <T_n>
       <d_inc_i>  <-  <d_n>
    *endif
.
    *if < <spec_disp_flag> > /then
      SPECIFY <lam_np1_i> <d_ext> -> <d_np1_i>
    *endif
.
. . . Form Stiffness based on Predictor
.
    *if < < <pass> /eq 1 >  /or  <extrap_this_step> > /then
      *call FORM_STIFF (displacement = <d_np1_i>; --
                        rotation     = <T_np1_i>; --
                        stiffness    = <K_asm> )       . output
.
. . . Find largest tangent component I = <ndef> <fd>
. . . Extract corresponding column of system matrix
. . . Specify freedom NDEF, FD as constrained to zero
.
     *if <adv_riks> /then
        *if <<step> /le 1> /then
          *def/i ndef==0
        *else
          *call STIF_COLMN
        *endif
     *else
        *def/i ndef==0
     *endif
.
. . . Factor stiffness with the added constraint
.
       *call FACT_STIFF
.
. . . Solve for Tangential Displacement
.
       *call TANDIS ( step = <step> ; iter = 1  ; --
                     displacement = <d_np1_i>  ; --
                      rotation     = <T_np1_i>  ; --
                      load_factor  = <lam_np1_i>; --
                      max_load     = [max_load] ; --
                      external_force = <F_ext>  ; --
                      internal_force = <F_int>  ; --
                      specified_disp = <d_ext>  ; --
                      tangent_force  = <F_T>    ; --
                      tangent_disp   = <delta_T> )
.
. . . Solve for incremental displacement based on stiffness column I
.
     *if <<ndef> /gt 0> /then
```

```
        *call SOLVE ( RHS = <K_I>; SOLN = <delta_I> )
      *endif
     *endif
.
. . . Compute Magnitude of Tangential Displacement
.
      NORM <delta_T> -> mag_delta_T  . magnitude of tang. displacement
.
. .  Predict Load Factor and Path Increment for Current Step
.
    *if <  <step> /eq  1 > /then
       *if < <pass> /eq 1 > /then
          *def/d lam_np1_i == [beg_load]
          *def/d lam_inc   == <lam_np1_i>
          *def/d path_inc  == < <lam_inc>*<mag_delta_T> >
          *def/i load_dir  == <SIGN(1.;<lam_inc>)>
       *else
          *def/d lam_inc   == < <path_inc>/<mag_delta_T> >
          *def/d lam_np1_i == <lam_inc>
       *endif
    *elseif < <step> /gt 1 > /then
.
. . . . Path_Increment based on Iterative Performance, Set Direction
.
       *if < <sign_det> /ne <sign_det_n> > /then
          *def/i load_dir  == < -1.*<load_dir_n> >
       *else
          *def/i load_dir  == <load_dir_n>
       *endif
.
. . . . Compute Load_Increment
.
       *if <extrap_this_step> /then
          *def/d lam_inc == < <lam_np1_i> - <lam_n> >
       *else
           *Remark USING CONSTRAINT EQN TO PREDICT LOAD AT STEP <step>
           *def/d lam_inc   == << <path_inc>/<mag_delta_T> >*<load_dir>>
           *def/d lam_np1_i == <<lam_n>+<lam_inc>>
       *endif
    *endif
    *Remark LAMBDA_<step>^ 1 = <lam_np1_i>    , LAMBDA_INC = <lam_inc>
    *if < <lam_np1_i> /gt [max_load] > /then
       *Remark   MAXIMUM LOAD_LEVEL OBTAINED.
       *jump to :EXIT
    *elseif < <lam_np1_i> /le [min_load] > /then
       *Remark   MINIMUM LOAD_LEVEL OBTAINED.
       *jump to :EXIT
    *endif
    *if <<extrap_this_step> /eq <false>> /then
       *Remark MODIFYING PREDICTOR VIA PATH-CONSTRAINT
        <d_inc_i>  <-  <lam_inc> <delta_T>
        <d_np1_i>  <-  <d_n>  +  <d_inc_i>
        ROTATE  <T_n> * <d_inc_i>  ->  <T_np1_i>
```

```
      *if < <spec_disp_flag> > /then
        SPECIFY <lam_np1_i> <d_ext>  ->  <d_np1_i>
      *endif
   *endif
.
. . . Compute 2 x 2 system coefficients
.
       *call TWO_BY_TWO (free = <ndof>)
.
. . . Form (Stress and) Residual_Force Vector
.
    *call RESIDUAL ( step      = <step>     ; --  . input
                     iter      =   1        ; --
                     displacement = <d_np1_i>     ; --
                     rotation     = <T_np1_i>     ; --
                     load_factor      = <lam_np1_i>   ; --
                     external_force   = <F_ext>       ; --
                     internal_force   = <F_int>       ; --
                     residual_force   = <R_np1_i> )   . output
.
. . . Initialize Convergence Criteria
.
    *call CHKCONV ( STEP = 1 ;  ITER = 1                ; --
                    Residual_force   = <R_np1_i>        ; --
                    displacement_inc = <d_inc_i>        ; --
                    load_inc         = <lam_inc>        ; --
                    relaxation       =  s )
    *if <CONVERGENCE> /then
       *def/d lam_np1_ip1 == <lam_np1_i>
       *jump to :CONVERGED
    *endif
.
. -- BEGIN ITERATION LOOP
.
   *DO :ITER_LOOP $i = 1, [max_iters]
       *def/i iter == <<$i>+1>
       *def/i tot_iters == <<tot_iters>+1>
       *if < <<step> /eq 1> /and <<iter> /eq 2> > /then
         *call FORM_STIFF (displacement = <d_np1_i>; --
                           rotation     = <T_np1_i>; --
                           stiffness    = <K_asm> )       . output
.
. . . Find largest tangent component I = <ndof> <fd>
. . . Extract corresponding column of system matrix
. . . Specify freedom NDEF, FD as constrained to zero
.
           *if <adv_riks> /then
               *if <<step> /le 1> /then
                   *def/i ndof==0
               *else
                   *call STIF_COLMN
               *endif
           *else
```

```
            *def/i ndef==0
          *endif
.
. . . Factor stiffness with the added constraint
.
          *call FACT_STIFF
.
. . . . . Update Tangential Displacement if Specified Displs., etc.
.
           *if < <spec_disp_flag> /or <live_load_flag> > /then
              *Remark o  Recomputing Tangential Displacement
              *call TANDIS ( step = <step> ; iter = <$i>; --
                             displacement = <d_np1_i>  ; --
                             rotation     = <T_np1_i>  ; --
                             load_factor  = <lam_np1_i>; --
                             max_load     = [max_load] ; --
                             external_force = <F_ext>  ; --
                             internal_force = <F_int>  ; --
                             specified_disp = <d_ext>  ; --
                             tangent_force  = <F_T>    ; --
                             tangent_disp   = <delta_T> )
           *endif
.
. . . Solve for incremental displacement based on stiffness column I
.
           *if <<ndef> /gt 0> /then
              *call SOLVE ( RHS = <K_I>; SOLN = <delta_I> )
           *endif
.
. . . Compute 2 x 2 system coefficients
.
            *call TWO_BY_TWO (free = <ndef>)
          *endif
.
. . . . . Compute Basic Displacement Iteration (delta_B)
.
         *call SOLVE ( RHS = <R_np1_i>; SOLN = <DELTA_B> )
.
.
. . . . . Compute New Load-Factor and Displacement Component I
. . . . . Advanced Riks (2 x 2) system.
.
          DOT <delta_B> * <d_inc_i> -> db_DOT_Dd
          DOT <d_inc_i> * <d_inc_i> -> Dd_DOT_Dd
         *def/d R_2 == < <path_inc>*<path_inc> >
         *def/d R_2 == < <R_2> - <Dd_DOT_Dd> - <2.0*<db_DOT_Dd>> >
         *if <<ndef> /gt 0> /then
             COMPONENT <ndef> <fd> <r_np1_i> -> r_I
             DOT <delta_B> * <K_I> -> db_DOT_K_I
.
           *def/d R_1 == <<r_I> - <db_DOT_K_I>>
.
. . . . . Solve 2 x 2 RIKS system
```

```
.
            *def/d Rdet = <<<A_11> * <A_22>> - <<A_12> * <A_21>>>
            *def/d dd_I = <<<R_1> * <A_22>> - <<R_2> * <A_12>>>
            *def/d dlam = <<<A_11> * <R_2>> - <<A_21> * <R_1>>>
            *def/d dd_I = <<dd_I> / <Rdet>>
            *def/d dlam = <<dlam> / <Rdet>>
          *else
.
. . . . .  Solve simple 1-D RIKS system for load factor (Normal Riks)
.
            *def/d  dlam = <<R_2>/<A_22>>
          *endif
.
. . . . . Update Load Factor
.
          *def/d lam_np1_ip1 = < <lam_np1_i> + <dlam> >
*remark
*remark *REMARK RANKIN'S ROOT SELECTION:
*remark Dd * Dd = <Dd_dot_Dd>
*remark dlam    = <Dlam>
*remark lambda  = <lam_np1_ip1>
*remark *remark
          :LOAD_LEVEL
.
. . . . .  Update Increments
.
           <delta>      <-  <delta_B>  + <dlam> <delta_T>
           *if <<ndef> /gt 0>/then
.
. . . . . . . Add Solution Due to Perturbation in I'th Displacement Component
. . . . . . .    (Advanced Riks Only)
.
              <delta>      <-  <delta>    - <dd_i> <delta_I>
.
. . . . . . . Set last component
.
              COMPONENT <ndef> <fd> <delta> <- <dd_I>
           *endif
.
. . . . .  Update Displacements, Rotations
.
           <d_np1_ip1>  <-  <d_np1_i>  +  <delta>
           ROTATE  <T_np1_i>  *  <delta>  ->   <T_np1_ip1>
           <d_inc_ip1>  <-  <d_inc_i>  +  <delta>
          *if < <spec_disp_flag> > /then
           SPECIFY <lam_np1_ip1> <d_ext> -> <d_np1_ip1>
          *endif
          *if < <DEBUG> > /then
            *print 1, <d_np1_ip1>
            *print 1, <T_np1_ip1>
          *endif
.
. . . . .  FORM (STRESSES AND) RESIDUAL_FORCE
```

```
.
        *call RESIDUAL ( step      = <step>     ; --   . input
                         iter      = <iter>     ; --
                         displacement = <d_np1_ip1>    ; --
                         rotation     = <T_np1_ip1>    ; --
                         load_factor      = <lam_np1_ip1>  ; --
                         external_force   = <f_ext>        ; --
                         internal_force   = <F_int>       ; --
                         residual_force   = <R_np1_ip1> )  . output
.
. . . . .  CHECK CONVERGENCE
.
        *call CHKCONV   ( step      = <step>       ; --  . input
                          iter      = <iter>       ; --
                          Residual_force   = <R_np1_ip1> ; --
                          displacement_inc = <delta>     ; --
                          load_inc         = 0.0 ; -- . for error norm
                          tol_e            = [tol_e]     ; --
                          max_iters        = [max_iters] ; --
                          convergence = CONVERGENCE; -- . output
                          divergence  = DIVERGENCE ; --
                          relaxation  = s )
      *if <<CONVERGENCE> /eq <TRUE>> /then
        *jump to :CONVERGED
      *elseif   <<DIVERGENCE> /eq <TRUE>> /then
        *if < <pass> /le [max_cuts] > /then
          *jump to :REPEAT
        *else
          *Remark Maximum number of automatic step cuts exhausted.
          *jump to :DIVERGED
        *endif
      *endif
      *def/d lam_np1_i == <lam_np1_ip1>
    :ITER_LOOP
    :NEUTRAL
    *if < <pass> /le [max_cuts] > /then
      *jump to :REPEAT
    *else
      *Remark Maximum number of automatic step cuts exhausted.
    *endif
    *Remark          NON-CONVERGENCE AT STEP <step>.  REVISE STRATEGY.
    *jump to :EXIT
    :REPEAT
     *def/i pass     == < <pass>+1 >
     *def path_inc == < <path_inc>/2. >
     *Remark
     *Remark CONVERGENCE DIFFICULTIES; REPEATING STEP <step>
     *Remark with reduced path_increment = <path_inc>
     *Remark Pass: <pass>
     *jump to :NEW_PASS
    :DIVERGED
    *Remark          DIVERGENCE AT STEP <STEP>.  REVISE STRATEGY.
    *jump to :EXIT
```

```
    :CONVERGED
    *Remark          CONVERGENCE AT STEP <STEP>.
    :NEXTSTEP
    *def/i num_iters     == <iter>
    *def/d path_scale    == <[des_iters]/<num_iters>>
    *def/d path_inc_nm1 == <path_inc_n>
    *def/d path_inc_n    == <path_inc>
    *def/d lam_nm2       == <lam_nm1>
    *def/d lam_nm1       == <lam_n>
    *def/d lam_n         == <lam_np1_ip1>
    *def/i sign_det_n    == <sign_det>
    *def/i load_dir_n    == <load_dir>
    *call POSTSTEP ( step=<step>; iter=<iter> )
 :STEP_LOOP
 :EXIT
*end
. =DECK CHKCONV
*procedure CHKCONV   ( STEP              ; --  . input
                       ITER              ; --
                       DISPLACEMENT_INC  ; --
                       RESIDUAL_FORCE    ; --
                       LOAD_INC          ; --
                       TOL_E             ; --
                       MAX_ITERS         ; --
                       CONVERGENCE=CONVERGENCE ; -- . output
                       DIVERGENCE =DIVERGENCE  ; --
                       relaxation = s )
[XQT VEC
.
. . Initialize
.
   *def/i [CONVERGENCE] == <FALSE>
   *def/i [DIVERGENCE]  == <FALSE>
   *if < [ITER] /gt 1 > /then
      *def/d12.4 ERR_E_I == <ERR_E_IP1>
   *endif
.
. . Compute Current Incremental Energy Norm
.
    DOT  [RESIDUAL_FORCE] * [DISPLACEMENT_INC] -> INC_E_IP1
   *if < [ITER] /le 1 > /then
      *if < <ABS(<INC_E_IP1>)> /gt <REF_E> > /then
         *def/d12.4 REF_E == <ABS(<INC_E_IP1>)>
      *endif
      *def/f7.2 [RELAXATION]  ==  1.0
      *def/i     num_diverges ==  0
   *endif
.
. . Compute Error Norms
.
   *def/d12.4  ERR_E_RAW ==  < <INC_E_IP1>/<REF_E> >
   *def/d12.4  ERR_E_IP1 ==  < <ABS(<ERR_E_RAW>)>^ .5 >
.
```

```
. . Check for displacement convergence
.
   *if  < <ERR_E_IP1> /le [TOL_E] >  /then
.
. . .  Step Converged
.
      *DEF/I [CONVERGENCE] == <TRUE>
      *jump to :BOTTOM_LINE
   *endif
   *if  < [ITER] /le 1 >  :BOTTOM_LINE
.
. . .  Step Not Converged; Check for Divergence
.
      *IF  < <ERR_E_IP1>/<ERR_E_I> /GT 1.0 >  /THEN
         *def num_diverges == < <num_diverges>+1 >
         *if <<num_diverges> /gt 1 > /then
            *DEF/I [DIVERGENCE] == <TRUE>
         *ENDIF
      *ENDIF
.
. . Select Relaxation (i.e., line-search) Parameter: s
.
   *def/f7.2   ls = <[relaxation]>
   *if < <<err_e_ip1>/<err_e_i>> /gt .5 > /then
       COMP <max_nod> <max_dof> [DISPLACEMENT_INC]  ->  max_d_ip1
      *if < <max_d_ip1>*<max_d_i> /lt 0 > /then
         *def  ls = < <ls>-.3 >
         *def  ls = <MAX(<ls>;.4)>
      *elseif < <ABS(<max_d_ip1>)> /lt <ABS(<max_d_i>)> > /then
         *def  ls = < <ls>+.4 >
         *def  ls = <MIN(<ls>;2.)>
      *else
         *def  ls = 1.0
      *endif
   *endif
   *def/f7.2  [relaxation] == <ls>
:BOTTOM_LINE
 NORM/MAX [DISPLACEMENT_INC]   ->   max_d_i  max_nod max_dof
 NORM/MAX [RESIDUAL_FORCE]     ->   max_f_i  max_nod_f max_dof_f
*if < [ITER] /LE 1 > /then
  *def/e12.4 load_factor  = <lam_np1_i>
*else
   *def/e12.4 load_factor  = <lam_np1_ip1>
*endif
*remark
*remark ____________________________________________________________
*remark  ITER=[ITER]  LD=<load_factor>  ERR=<err_e_ip1>  REF=<ref_e>
*remark            delta_d_max=<max_d_i>  node=<max_nod>   dof=<max_dof>
*remark            resid_f_max=<max_f_i>  node=<max_nod_f> dof=<max_dof_f>
*remark
*remark
    *if < <DEBUG> > /then
       *print 1, [DISPLACEMENT_INC]
```

```
      *print 1, [RESIDUAL_FORCE]
    *endif
*END
. =DECK DEFNS
*procedure DEFNS ( step; iter )
.
. Purpose:  Define basic macro_symbols for NLSTATIC3 procedure
.
*def/i np1 = [step]
*def/i n   = < <np1>-1 >
*def/i nm1 = < <n>-1 >
*def/i nm2 = < <n>-2 >
*if < [iter] /eq 0 > /then              . run initialization
   *def/i NS_ldi    ==  1
   *def/i NOM_ldi   ==  3
   *def/i iset      ==  1
   *def/i icon      ==  1
   *def/a f_spec    == APPL.FORC.<iset>.<icon>
   *def/a d_spec    == APPL.MOTI.<iset>.<icon>
.
.  Check for Prescribed Force/Displacement Loading
.
   *find dataset <NS_ldi> <f_spec> /seq=ids
   *if < <ids> /gt 0 > /then
      *Remark Note: Applied forces dataset <f_spec>, will be used
      *def/i spec_force_flag == <true>
   *else
      *def/i spec_force_flag == <false>
   *endif
   *find dataset <NS_ldi> <d_spec> /seq=ids
   *if < <ids> /gt 0 > /then
      *Remark Note: Specified displacement dataset <d_spec> will be used
      *def/i spec_disp_flag == <true>
   *else
      *def/i spec_disp_flag == <false>
   *endif
   *if < <<spec_force_flag>/eq 0> /and <<spec_disp_flag>/eq 0>> /then
        *remark  ; *remark    Procedure stopped: no loads defined
   *endif
   *def/i live_load_flag == <false>
.
   *def/a d_ext     ==  EXT.DISP
   *def/a f_ext     ==  EXT.FORC
   *def/a f_int     ==  INT.FORC
   *def/a f_T       ==  TAN.FORC
   *def/a R_np1_i   ==  RES.FORC
   *def/a R_np1_ip1 ==  RES.FORC
   *def/a d_inc_i   ==  INC.DISP
   *def/a d_inc_ip1 ==  INC.DISP
   *def/a delta_T   ==  HAT.DISI
   *def/a delta_B   ==  BAR.DISI
   *def/a delta_S   ==  STR.DISI
   *def/a delta     ==  TOT.DISI
```

```
   *def/a K_asm      ==  ASM.STIF
   *def/a K_fac      ==  FAC.STIF
   *def/a K_I        ==  COL.STIF.1.1
   *def/a Delta_I    ==  STF.DISI
   *def/i ndef       ==  0
   *def/a d_active   ==  ACT.DISI
.
.   Retrieve Control Parameters from DATA_BASE
.
   *if < [step] /eq 1 > /then
    *def/i sign_det      == 1
    *def/i load_dir      == 1
    *def/d lam_n         == 0.0
    *def/d lam_nm1       == 0.0
    *def/d lam_nm2       == 0.0
    *def/d path_inc_n    == 0.0
    *def/d path_inc_nm1  == 0.0
    *def/d ref_e         == 0.0
    *Remark    STARTING PARAMETERS INITIALIZED.
   *else
    *open <NOM_ldi> <NOM_DB>
    *find dataset <nom_ldi> <nom_ds> /seq=nom_ids
    *g2m /name==lam_n /type=d <nom_ldi> <nom_ids> LOAD.<n>
    *g2m /name==lam_nm1 /type=d <nom_ldi> <nom_ids> LOAD.<nm1>
    *g2m /name==lam_nm2 /type=d <nom_ldi> <nom_ids> LOAD.<nm2>
    *g2m /name==path_inc_n /type=d <nom_ldi> <nom_ids> PATH_INC.<n>
    *g2m /name==path_inc_nm1 /type=d <nom_ldi> <nom_ids> PATH_INC.<nm1>
    *g2m /name==sign_det_n /type=I <nom_ldi> <nom_ids> SIGN_DET.<n>
    *g2m /name==load_dir_n /type=I <nom_ldi> <nom_ids> LOAD_DIR.<n>
    *g2m /name==ref_e /type=d <nom_ldi> <nom_ids> REF_ERR.<n>
    *g2m /name==ndef /type=i <nom_ldi> <nom_ids> NDEF
    *g2m /name==fd /type=i <nom_ldi> <nom_ids> FD
    *close <NOM_ldi>
    *Remark
    *remark ------------------------------------------------------------
    *remark
    *Remark    RE-START PARAMETERS:
    *Remark      LOAD_FACTORS (n,n-i,n-2) = <lam_n>, <lam_nm1>, <lam_nm2>
    *Remark      PATH_INCREMS (n, nm1)    = <path_inc_n>, <path_inc_nm1>
    *Remark      SIGN_DET (n)             = <sign_det_n>
    *Remark      LOAD_DIR (n)             = <load_dir_n>
    *remark
    *remark ------------------------------------------------------------
   *endif
*endif
.
. Define Global Datasets for Upcoming Step
.
*def/a d_nm2      == TOT.DISP.<nm2>
*def/a d_nm1      == TOT.DISP.<nm1>
*def/a d_n        == TOT.DISP.<n>
*def/a d_np1_i    == TOT.DISP.<np1>
*def/a d_np1_ip1  == TOT.DISP.<np1>
```

```
*def/a T_n       == TOT.ROTN.<n>
*def/a T_np1_i   == TOT.ROTN.<np1>
*def/a T_np1_ip1 == TOT.ROTN.<np1>
*end
. =DECK EXTRAP
*procedure EXTRAP ( dx_np1; dx_n; dx_nm1; cx_n; cx_nm1; cx_nm2 )
   *def dx_np1 = [dx_np1]
   *def dx_n   = [dx_n]
   *def dx_nm1 = [dx_nm1]
   *def x_nm1 = <dx_nm1>
   *def x_n   = < <x_nm1>+<dx_n> >
   *def x_np1 = < <x_n>+<dx_np1> >
   *def s_nm1 = < <x_np1>-<x_nm1> >
   *def s_n   = < <x_np1>-<x_n>   >
   *def [cx_n]   ==     < <x_np1>*<s_nm1>/<<x_n>*<dx_n>> >
   *def [cx_nm1] == < -1.*<x_np1>*<s_n>/<<x_nm1>*<dx_n>> >
   *def [cx_nm2] ==     < <s_nm1>*<s_n>/<<x_nm1>*<x_n>>  >
*end
. =DECK FACT_STIFF
*procedure FACT_STIFF
    [XQT INV                          . Factor stiffness
     RESET CON  = 1
     RESET LRA  = 7168
     RESET DZERO = 1.E-10
     RESET SPDP = <csm_precision>
    *remark
    *Remark             MATRIX BEING FACTORED (in Double Precision).
. Define factored-matrix parameters as global macrosymbols
 [XQT VEC
 *remark                Determinant = <coef_det> * 10 ^ <exp10_det>
 *remark                Neg. roots  = <num_neg>
 *def/i sign_det == <SIGN(1.;<coef_det>)>
 *remark                Sign of det = <sign_det>
 *remark
*END
. =DECK FORM_STIFF
*procedure FORM_STIFF ( DISPLACEMENT ; --
                        ROTATION     ; --
                        STIFFNESS )        . output
.
   *Remark             FORMING NEW STIFFNESS MATRIX
.
     *call ES ( function    = 'FORM STIFFNESS/TANG' ; --
                es_nl_geom = <NL_GEOM>             ; --
                es_coro    = <COROTATION>          ; --
                es_dis_ds  = [DISPLACEMENT]        ; --
                es_rot_ds  = [ROTATION] )
    [XQT K                           . Transform/assemble stiffness
*END
. =DECK INITIAL
*procedure INITIAL
.
.  CLEAR INITIAL DISPLACEMENTS AND ROTATIONS   ( n = 0 )
```

```
.
    *g2m /name=parameters /type=i /maxn=18 <NS_LDI> JDF1.BTAB.1.8 DATA.1
    *def/i NNODES = <PARAMETERS[1]>
    *def/i NDOF   = <PARAMETERS[2]>
    *Remark    Problem Dimensions: N_nodes = <NNODES>, N_dofn = <NDOF>
  [XQT VEC
    INIT_DOF CON..<icon> -> DOF.TABL
    *Remark    DOF TABLE initialized.
    INIT_VEC <d_inc_ip1>      <NDOF> BY <NNODES> . Zeroize translations
    INIT_VEC <K_I>      <NDOF> BY <NNODES> . Zeroize translations
    INIT_VEC <Delta_I>  <NDOF> BY <NNODES> . Zeroize translations
    INIT_VEC <d_active> <NDOF> BY <NNODES> . Zeroize translations
    INIT_VEC <d_n>      <NDOF> BY <NNODES> . Zeroize translations
    INIT_VEC <T_n>   3    BY <NNODES> . Zeroize rotation pseudovectors
    *Remark    Displacements initialized.
    *if < <spec_disp_flag> > /then
     <d_ext>  <-  <d_spec>                                  . for VEC
     NORM <d_ext> -> norm_d
     *if < <norm_d> /eq 0. > /then
        *Remark
        *Remark NOTE: Specified displacements are identically zero.
        *Remark
     *endif
     *copy <NS_ldi> = <NS_ldi>, <d_spec>
     <d_spec> <- 0                                          . for SSOL
     *Remark    Specified displacements saved in <d_ext>.
    *endif
    *if <<spec_force_flag>> /then
       <F_ext>  <-  <F_spec>                                . for VEC
       NORM <f_ext> -> norm_f
       *if < <norm_f> /eq 0. > /then
           *Remark
           *Remark NOTE: External forces are identically zero.
           *Remark
       *endif
       *Remark    External force vector saved in <F_ext>.
       *copy <NS_ldi> = <NS_ldi>, <F_spec>                  . for SSOL
    *else
       INIT_VEC <F_ext>  <NDOF> BY <NNODES>
       INIT_VEC <F_spec> <NDOF> BY <NNODES> /single_precision . for SSOL
    *endif
    *Remark    Right-hand-side vector zeroized (in <F_spec>).
    INIT_VEC <F_int> <NDOF> BY <NNODES>
    *Remark    Internal force vector zeroized (in <F_int>).
.
.  INITIALIZE ELEMENT CONFIGURATION
.
  [XQT E
  [XQT RSEQ
         reset method=0,maxcon=35
  [XQT TOPO
         reset maxsub = 40000,lram=8196
  *call ES ( function = 'INITIALIZE' )
```

```
  *Remark    Element configuration initialized.
.
*end
. =DECK POSTRES
*procedure POSTRES ( step )
 [XQT VEC
  *find dataset <NOM_LDI> <NOM_DS> /seq=post_ids
  *if < <post_ids> /le 0 > /then
     *put dataset <NOM_LDI> <NOM_DS> /mrat=2000 /seq=post_ids
  *endif
.
. Save selected displacements on nominal data-base
.
   *def/a dof_names = U, V, W, RU, RV, RW
.
   *do $isel = 1, <N_SELECT>
    *def/i node = <SEL_NODES[<$isel>]>
    *def/i dof  = <SEL_DOFS[<$isel>]>
    *def/a dof_name = <dof_names[<dof>]>
     COMPONENT <node> <dof> TOT.DISP.[step] -> DISP
    *def/a recd_name = DISP_<dof_name>_<node>.[step]
    *Remark <recd_name> = <DISP>
    *m2g /name=disp /type=d <nom_ldi> <post_ids> <recd_name>
.
.    Save Reaction Forces for Current Step
.
     REAC.FORC.[step]  <-  INT.FORC
    *if < [step] /eq 0 > /then
       REAC.FORC.0  <-  0.0
    *endif
    *if < <spec_disp_flag> > /then
        COMPONENT <node> <dof> REAC.FORC.[step] -> FORCE
       *def/a recd_name = FORCE_<dof_name>_<node>.[step]
       *Remark <recd_name> = <FORCE>
       *m2g /name=force /type=d <nom_ldi> <post_ids> <recd_name>
    *endif
   *enddo
*end
. =DECK POSTSTEP
*procedure POSTSTEP ( step; iter )
*Remark ___________________
*Remark STEP [step] SUMMARY :
*Remark ____________________________________________________________
*Remark Load Factor _____________________ <lam_np1_ip1>
*Remark Load Direction __________________ <load_dir>
*Remark Stiffness determinant ___________ <coef_det> * 10 ^ <exp10_det>
*Remark Number of negative roots ________ <num_neg>
*Remark Path_length_increment ___________ <path_inc>
*Remark Relative energy_error ___________ <err_e_ip1>
*Remark Number of Iterations ____________ <num_iters>
*Remark Number of Step Cuts _____________ <<pass>-1>
*Remark Total Number of Iterations ______ <tot_iters>
*Remark ____________________________________________________________
```

```
*open <NOM_ldi> <NOM_DB>
*if < [step] /eq 1 > /then
   *call POSTRES ( step = 0 )
   *def/d load_0=0.
   *m2g /name=load_0 /type=d <NOM_LDI> <NOM_DS> LOAD.0:0
*endif
*call POSTRES  ( step=[step] )
*find dataset <NOM_LDI> <NOM_DS> /seq=nom_ids
*m2g /name=lam_np1_ip1 /type=d <nom_ldi> <nom_ids> LOAD.[STEP]
*m2g /name=path_inc /type=d <nom_ldi> <nom_ids> PATH_INC.[STEP]
*m2g /name=err_e_ip1 /type=d <nom_ldi> <nom_ids> ERROR.[STEP]
*m2g /name=ref_e /type=d <nom_ldi> <nom_ids> REF_ERR.[STEP]
*m2g /name=load_dir /type=i <nom_ldi> <nom_ids> LOAD_DIR.[STEP]
*m2g /name=sign_det /type=i <nom_ldi> <nom_ids> SIGN_DET.[STEP]
*m2g /name=coef_det /type=d <nom_ldi> <nom_ids> COEF_DET.[STEP]
*m2g /name=exp10_det /type=i <nom_ldi> <nom_ids> EXP10_DET.[STEP]
*m2g /name=num_neg /type=d <nom_ldi> <nom_ids> NEG_ROOTS.[STEP]
*m2g /name=num_iters /type=i <nom_ldi> <nom_ids> NUM_ITERS.[STEP]
*m2g /name=tot_iters /type=i <nom_ldi> <nom_ids> TOT_ITERS.[STEP]
*def/i passm1 = <<pass>-1>
*m2g /name=passm1 /type=i <nom_ldi> <nom_ids> NUM_CUTS.[STEP] . cgl, 7/26/88
*m2g /name=ndef /type=i <nom_ldi> <nom_ids> NDEF
*if <<ndef> /eq 0> /then
   *def/i fd==0
*endif
*m2g /name=fd /type=i <nom_ldi> <nom_ids> FD
*close <NOM_ldi>
*end
. =DECK RESIDUAL
*procedure RESIDUAL ( STEP=1; ITER=1    ; --  . input
                      DISPLACEMENT      ; --
                      ROTATION          ; --
                      LOAD_FACTOR       ; --
                      SPECIFIED_DISP    ; --
                      INTERNAL_FORCE    ; --
                      EXTERNAL_FORCE    ; --
                      RESIDUAL_FORCE  )     . output
.
 [XQT VEC
     [INTERNAL_FORCE] <- 0.
 *call ES ( function    = 'FORM FORCE/INT'; --
            es_nl_geom = <NL_GEOM>       ; --
            es_coro    = <COROTATION>    ; --
            es_dis_ds  = [DISPLACEMENT]  ; --
            es_rot_ds  = [ROTATION]      ; --
            es_frc_ds  = [INTERNAL_FORCE] )
 [XQT VEC
 [RESIDUAL_FORCE]  <-  [load_factor] [EXTERNAL_FORCE] - [INTERNAL_FORCE]
*END
. =DECK SOLVE
*procedure SOLVE ( RHS ; SOLN )
.
. Copy Right-Hand-Side Vector to Expected SPAR (Single Precision) Dataset
```

```
.
   [XQT VEC
    APPL.FORC.<iset>.<icon>  <-  [RHS]            /single_precision
.
. Solve
.
   [XQT SSOL
    RESET CON  = <icon>
    RESET SET  = <iset>
    RESET REAC = 0
.
. Copy SPAR Solution Vector to Double Precision Dataset
.
   [XQT VEC
    [SOLN]  <-  STATIC.DISP.<iset>.<icon>
*end
. =DECK STIF_COLMN
*procedure STIF_COLMN
[XQT VEC
   *if <<NDEF> /gt 0> /then
      FREE <NDEF> <FD>
   *endif
   *if <<spec_disp_flag>  > /then
       <d_active> <- <d_inc_i>
       SPECIFY 0. <d_ext> -> <d_active>
       NORM/MAX <d_active> -> D_II NDEF FD
   *else
       NORM/MAX  <d_inc_i> -> D_II NDEF FD
   *endif
[XQT AUS
    SYSVEC : UNIT I 1 1
        I=<FD> : J=<NDEF> : 1.0
    SYSVEC : COL STF 1 1
    DEFINE E = <NS_ldi> UNIT I 1 1
    DEFINE K = <NS_ldi> K SPAR 36
    COL STF 1 1 = PRODUCT (K,E)
[XQT VEC
   FIX <NDEF> <FD>
   <K_I> <- COL.STF.1.1
*END
. =DECK STIFFNESS
*procedure STIFFNESS ( DATA_BASE    ; --  . input
                       STEP         ; --
                       ITER         ; --
                       LOAD_FACTOR  ; --
                       DISPLACEMENT ; --
                       ROTATION     ; --
                       STIFFNESS )       . output
.
   *Remark             FORMING NEW STIFFNESS MATRIX
.
     *call ES ( function    = 'FORM STIFFNESS/TANG' ; --
                es_nl_geom = <NL_GEOM>             ; --
```

```
                es_coro     = <COROTATION>          ; --
                es_dis_ds   = [DISPLACEMENT]        ; --
                es_rot_ds   = [ROTATION] )
    [XQT K                              . Transform/assemble stiffness
    [XQT INV                            . Factor stiffness
     RESET CON  = 1
     RESET LRA  = 7168
     RESET DZERO = 1.E-10
     RESET SPDP = <csm_precision>
    *remark
    *Remark              MATRIX BEING FACTORED (in Double Precision).
. Define factored-matrix parameters as global macrosymbols
 [XQT VEC
 *remark                 Determinant = <coef_det> * 10 ^ <exp10_det>
 *remark                 Neg. roots  = <num_neg>
 *def/i sign_det == <SIGN(1.;<coef_det>)>
 *remark                 Sign of det = <sign_det>
 *remark
*END
. =DECK TANDIS
*procedure TANDIS ( step;     iter; displacement  ; rotation      ; --
                    load_factor   ; max_load      ; external_force; --
                    specified_disp; internal_force; --
                    tangent_force ; tangent_disp )
*if < <spec_disp_flag> > /then     . Load standard spec_disp dataset
   [XQT VEC
       SPECIFY  1.0 [specified_disp]  ->  <d_spec>
*endif
*call SOLVE ( RHS = [external_force]; SOLN = [tangent_disp] )
*if < <spec_disp_flag> > /then     . Clear standard spec_disp dataset
   [XQT VEC
       <d_spec>  <-  0.0
*endif
*end
. =DECK TWO_BY_TWO
*procedure TWO_BY_TWO ( FREE )
.
. . . Compute 2 x 2 system coefficients
.
     [XQT VEC
.
. . . A_22 is used for both ordinary and advanced RIKS
.
        DOT <d_inc_i> * <delta_T> -> A_22
        *def/d A_22 == < 2.0 * <A_22>>
.
. . . The remainder of Procedure is invoked only for Advanced Riks
.
        *if <[FREE] /gt 0> /then
           COMPONENT <ndef> <fd> <K_I> -> K_II
           COMPONENT <ndef> <fd> <f_ext> -> fe_I
           COMPONENT <ndef> <fd> <d_inc_i> -> D_II
           DOT <K_I> * <delta_I> -> K_DOT_d_I
```

```
            DOT <K_I> * <delta_T> -> K_DOT_T
            DOT <d_inc_i> * <delta_I> -> dd_DOT_d_I
            *def/d A_11 == <<K_II> - <K_DOT_d_I>>
            *def/d A_12 == <<K_DOT_T> - <fe_I>>
            *if <<spec_disp_flag>>/then
                DOT <K_I> * <d_ext> -> K_DOT_d_ext
                *def/d A_12 == <<A_12> - <K_DOT_d_ext>>
            *endif
            *def/d A_21 == < 2.0 *<<D_II> - <dd_DOT_d_I>> >
          *endif
*end
```

3.11 Procedure NL_DYNAMIC_1

3.11.1 GENERAL DESCRIPTION

Procedure NL_DYNAMIC_1, written by C. C. Rankin and B. Nour-Omid of Lockheed Palo Alto Research Laboratory, performs nonlinear transient analysis using a one-step, self-starting, implicit integration algorithm containing adjustable parameters. The nonlinear system is solved using a modified Newton/Raphson incremental/iterative solution sequence. Like NL_STATIC_1, procedure NL_DYNAMIC_1 relies on the Generic Element Processor (*i.e.*, structural element processors, ES*i*) and hence has a corotational option for geometric nonlinearity that enables arbitrarily large rotations.

Procedure NL_DYNAMIC_1 solves the transient system

$$\mathbf{M}\ddot{\mathbf{u}} + \mathbf{f}^{int}(\mathbf{u}) = \mathbf{f}^{ext} \qquad (3.11-1)$$

Here **M** is the mass matrix, **u** is the displacement, and dots over quantities refer to differentiation with respect to time. These equations are discretized over time using the following relations involving two parameters θ and α:

$$\mathbf{M}\mathbf{a}_{n+\theta} + (1+\alpha)\mathbf{f}^{int}(\mathbf{d}_{n+\theta}) - \alpha\mathbf{f}^{int}(\mathbf{d}_n) = \mathbf{f}^{ext}_{n+\theta} \qquad (3.11-2)$$

where $\mathbf{d}_{n+\theta}$ and $\mathbf{a}_{n+\theta}$ are the approximations to the displacement and acceleration at time $(n+\theta)\Delta t$, and where Wilson's θ interpolation parameter is used to obtain the desired acceleration according to

$$\mathbf{a}_{n+\theta} = (1-\theta)\mathbf{a}_n + \theta\mathbf{a}_{n+1} \qquad (3.11-3)$$

Velocity **v** and acceleration **a** are related to the displacements at the point $n+\theta$ by

$$\begin{aligned} \mathbf{v}_{n+\theta} &= \mathbf{v}_n + \theta\Delta t[(1-\gamma)\mathbf{a}_n + \gamma\mathbf{a}_{n+\theta}] \\ \mathbf{d}_{n+\theta} &= \mathbf{d}_n + \theta\Delta t\mathbf{v}_n + (\theta\Delta t)^2[(\frac{1}{2}-\beta)\mathbf{a}_n + \beta\mathbf{a}_{n+\theta}] \end{aligned} \qquad (3.11-4)$$

where we have introduced the two standard Newmark parameters β and γ as the last two parameters in our system. The resulting system to be solved is

$$\mathbf{M}\mathbf{d}_{n+\theta} + (\theta\Delta t)^2\beta(1+\alpha)\mathbf{f}^{int}(\mathbf{d}_{n+\theta}) - \tilde{\mathbf{f}}_{n+\theta} = 0 \qquad (8)$$

where the quantity $\tilde{\mathbf{f}}_{n+\theta}$ is known from the last converged time step (see Section 4.3.9.2. for a complete derivation of equations and definition of symbols).

NL_DYNAMIC_1 is designed to solve any well posed initial value transient analysis problem with non-negative mass. This includes problems with initial velocity and/or initial displacement, as well as a generalized imposed external forcing with either built-in or user-suppled time dependence. Currently, the procedure uses the same external loading as NL_STATIC_1, with a multiplier defined by a force-time procedure. The built-in procedures are described in Sections 4.3.3.7 and 4.3.3.8.

Since (66) is a nonlinear system similar to (1), much of what is in NL_STATIC_1 applies to NL_DYNAMIC_1. In particular, we mention the solution of a banded system with similar structure to the stiffness matrix, identical handling of large rotations, similar procedures for archiving results of solutions and retrieving restart information, and very similar convergence and stepsize adjustment procedures. We shall henceforth concentrate on the differences between the static and dynamic algorithms, with particular emphasis on operations involving the mass, time step, and the four adjustable integration parameters. In some respects, NL_DYNAMIC_1 is simpler than its static analogues, since no arclength constraint is needed for a positive-definite mass/stiffness system.

3.11.2 PROCEDURE USAGE

Procedure NL_DYNAMIC_1 may be invoked by the `*call` directive:

```
*call NL_DYNAMIC_1 ( arg1 = val1; arg2 = val2; ...)
```

where **arg***i* are argument names and **val***i* are the corresponding values you wish to give them. The following are valid arguments for NL_DYNAMIC_1; note that those without default values are mandatory, while the others are optional.

Argument	Default Value	Meaning
BEG_STEP	–	Starting step number (>0)
MAX_STEPS	–	Maximum steps to compute
BEG_TIME	–	Problem starting time
MAX_TIME	–	Upper_bound on problem time
DEL_TIME	–	Beginning time step
INT_PARS	1.,0.,.25,.5	Time integration parameters
LOAD_STIFF	<false>	Include load stiffness
FT_PROC	FT_ZERO	Forcing procedure
FT_ARGS	0.	Parameter array for FT_PROC
COROTATION	1	Corotational Update Option
DES_ITERS	4	Number of iterations per step desired
EXTRAPOLATE	<true>	Perform quadratic extrapolation of solution
FAC_STEPS	1	Steps_per_refactoring
MAX_ITERS	9	Maximum iterations per step
MAX_CUTS	3	Maximum number of successive step cuts
NL_GEOM	2	Geometric Nonlinearity Level (1 or 2)
NOMINAL_DB	NOMINAL.GAL	Results database file
NOMINAL_DS	RESPONSE.HISTORY	Results dataset
N_SELECT	0	Number of nodes for selected disp. output
SEL_NODES	0	List of nodes for selected output
SEL_DOFS	0	Corresponding list of nodal freedoms (1–6)
TOL_E	1.E-3	Relative error tolerance in energy norm

In the above definitions, the term *step* refers to a time step. The total response is automatically subdivided into time steps, with the starting time prescribed by the User — using BEG_TIME. The initial time step is set by DEL_TIME, and held constant unless trouble occurs. If convergence is difficult, the time step will be cut, and a new initial value problem will be started at the last converged solution.

3.11.3 ARGUMENT GLOSSARY

3.11.3.1 BEG_STEP

This argument defines the number of the first step to be computed in a given nonlinear analysis interval. It is important primarily for analysis re-starts. Initially, BEG_STEP should be set to 1. To continue an analysis in a subsequent run, after having computed and saved "n" steps in the previous run, one would typically set BEG_STEP equal to "n+1". For example, if the 10th step was successfully completed in the first run, then it could be continued in a second run by setting BEG_STEP = 11. However, it is not necessary for BEG_STEP to be larger than any previously computed step. That is, you may *re-compute* a sequence of steps by setting BEG_STEP to the number of the first step to be re-computed. The procedure will automatically use the step that immediately precedes BEG_STEP (*e.g.*, BEG_STEP-1) to obtain the necessary initial displacements, velocities, and accelerations.

3.11.3.2 BEG_TIME

The starting problem time. For the initial run, the value of this parameter is usually set to zero.

3.11.3.3 COROTATION (default = <true>)

Corotational update switch for large-rotation problems. This switch should be set to <true> when the model involves finite elements that require corotation for geometric nonlinearity. This is true of most beam and shell elements, and may be true for some solid (3D) elements used to model shell structures. Consult the appropriate element Processor (ES*i*) section in the Testbed User's Manual for specific guidelines.

3.11.3.4 DEBUG (default = <false>)

Procedure debug switch. This switch should only be turned on to obtain additional diagnostic printout for procedure debugging.

3.11.3.5 DEL_TIME

The initial time step. If all goes well with the integration, this will be the time increment throughout the analysis. For restart, it is possible to use the time increment that was in force for the previous run. *For restart only*, input zero to cause the procedure to read in and use the previous DEL_TIME.

3.11.3.6 DES_ITERS (default = 4)

Desired number of iterations allowed for convergence at a given load step. This parameter is used to adaptively adjust the arclength increment from one load step to the next, by comparing DES_ITERS with the actual number of iterations required for convergence at the last step.

3.11.3.7 FT_PROC (default = FT_ZERO)

FT_PROC is the name of a procedure that will obtain the load factor (λ) for the time-dependent forcing function. For applied *force* loading, this factor is multiplied by the reference applied force vector to obtain the current load vector, *i.e.*,

$$\mathbf{f}_1^{ext} = \lambda(t)\,\bar{\mathbf{f}}^{ext}$$

where $\bar{\mathbf{f}}^{ext}$ is the reference applied force vector stored in dataset APPL.FORC.1. For applied *displacement* loading, the starting load factor is applied to the reference applied displacement vector, which is then used to compute the initial internal force vector, *i.e.*,

$$\mathbf{f}_1^{int(1)} = \mathbf{f}^{int}[\lambda(t)\,\bar{\mathbf{d}}^{ext}]$$

where $\bar{\mathbf{d}}^{ext}$ is the reference applied displacement vector stored in dataset APPL.MOTI.1.

The user can supply any procedure he desires for FT_PROC. If this is the case, FT_PROC is the *name* of that procedure. FT_PROC has three arguments of its own. T is the current problem time, FT_ARGS is an array of up to six scalars, and F_MAC is the name of the macrosymbol (set by NL_DYNAMIC_1) to contain $\lambda(t)$.

If the user wishes to use the FT procedures supplied here, he has four choices, as illustrated in Fig. 4.3-1.

FT_ZERO	This is the default option, which means that there will be no external forcing.
FT_LIN	Piecewise linear forcing function.
FT_SIN	Sinusoidal forcing function.
FT_EXP	Exponential decay.

3.11.3.8 FT_ARGS

These are the arguments to either the user-supplied forcing function, or *required* for FT_LIN, FT_SIN, or FT_EXP. For these last three procedures (supplied here), we offer the same options that are supplied with the code Structural Analysis of General Shells (STAGS). We have reproduced the forcing function diagram from the STAGS manual, which here appears as Fig. 4.3-1. In this figure, PA is the function $\lambda(t)$, TIME is in units of problem time, and CA1 through CA6 are FT_ARGS[1:6], respectively. A summary for each each case follows:

For FT_LIN (piecewise linear time dependence), the array is as follows:

FT_ARGS[1] Maximum load factor (CA1 in figure).

FT_ARGS[2] Delay time before applied load ramps up (CA2). Can be zero.

FT_ARGS[3] Time at which load reaches maximum (CA3). Can be equal to FT_ARGS[2].

FT_ARGS[4] Time at which load begins to drop (CA4). Can be equal to FT_ARGS[3] if FT_ARGS[3] is not equal to FT_ARGS[2].

FT_ARGS[5] Time at which load drops (CA5).

FT_ARGS[3:5] can be safely set to larger than MAX_TIME if that is what the user wants.

For FT_SIN (sinusoidal time dependence), the array is as follows:

FT_ARGS[1] Peak height of sine wave (see Fig. 4.3-1, CA1)

FT_ARGS[2] Load offset for the sine wave function (CA2). If zero, the sine wave will oscillate to plus/minus FT_ARGS[1].

FT_ARGS[3] Half wavelength (CA3).

FT_ARGS[4] Problem time to first maximum load (CA4).

FT_ARGS[5] Delay time to imposition of load (CA5).

FT_ARGS[6] Time at which all loading will terminate (can be a large number), or CA6 in Fig. 4.3-1.

For FT_EXP (Exponential decay), the array is as follows:

FT_ARGS[1] Peak height of exponential (see Fig. 4.3-1, CA1)

FT_ARGS[2] Delay time before any load is imposed (CA2). Can be zero.

FT_ARGS[3] Half life time of exponential decay.

FT_ARGS[4] This argument is not used. In the Fig. 4.3-1, CA4 must be half CA1. If this is not the case, then FT_ARGS[3] should be recomputed.

Figure 4.3-1 Load Factor Histories

3.11.3.9 INT_PARS (default=1.,0.,.25,.5)

The values contained in this array set the characteristics of the time integrator. INT_PARS[1] is Wilson's θ interpolator (see Theory, Section 4.3.9), INT_PARS[2] is α, and INT_PARS[3:4] are the Newmark β and γ parameters, respectively. A sophisticated user can reset these parameters for special integrator performance characteristics. Unless there is reason to tinker with their default values, it is better for the user to leave these parameters alone.

3.11.3.10 LOAD_STIFF / (default=<false>)

If true, include load stiffness in the total stiffness matrix assembly.

3.11.3.11 MAX_CUTS (default=3)

Maximum number of step cuts permitted during the current nonlinear analysis interval. A step cut refers to a halving of the arclength increment used to advance the solution from one step to the next. Step cuts are performed only if the maximum number of iterations are exceeded without converging at a given load step. Note that the relationship between the increment in "arclength" and the increment in the load-factor, λ, is computed internally by the procedure.

3.11.3.12 MAX_TIME

Problem time for which the analysis is to be terminated.

3.11.3.13 MAX_ITERS

Maximum number of iterations allowed for convergence at a given load step. This parameter is used to terminate the iteration process at a given load level. If convergence hasn't been obtained after MAX_ITERS iterations, the load (*i.e.*, arclength) increment is cut in half and the step is repeated — until either convergence has been obtained or MAX_CUTS has been exceeded.

3.11.3.14 MAX_STEPS

Maximum number of load steps to compute in the current nonlinear analysis run. This provides an implicit limit on analysis run-time. The transient analysis is thus terminated whenever MAX_STEPS or MAX_TIME is exceeded — whichever comes first.

3.11.3.15 NL_GEOM (default = 2)

Geometric nonlinearity level: 0, 1, or 2. 0 $\Rightarrow$ the problem is geometrically linear; 1 $\Rightarrow$ geometric nonlinearity will be handled globally, *e.g.*, using corotational updates only; and 2 $\Rightarrow$ nonlinear element strain-displacement relations should be used in addition to any global treatment of geometric nonlinearity. If COROTATION = <true>, options 1 and 2 refer to first-order and second-order corotation, respectively. The latter option can be significantly more accurate than the former for a given finite element model — depending on which element types are involved.

3.11.3.16 NOMINAL_DB (default = NOMINAL.GAL)

Name of database (GAL file) where a step-wise history of important solution parameters and selected response variables is to be stored.

3.11.3.17 NOMINAL_DS (default = RESPONSE.HISTORY)

Name of dataset, within database defined by argument NOMINAL_DB, where a step-wise history of important solution parameters and selected response variables is to be stored. See the CSM Testbed Dataset Manual, under dataset RESPONSE.HISTORY, for a description of the individual data records stored in this dataset.

3.11.3.18 N_SELECT (default = 0)

Number of user-selected displacement components to be saved in the dataset specified by argument NOMINAL_DS. Values for these displacement components, the locations and directions of which are specified by arguments SEL_NODES and SEL_DOFS, respectively, are stored at every time step.

3.11.3.19 SEL_DOFS (default = 0)

List of nodal DOF's at which displacement histories are to be saved in dataset [NOMINAL_DS]. There should be [N_SELECT] numbers in the list, in correspondence with the node numbers specified by argument SEL_NODES. Values of each number in the list must range between 1 and 6, in correspondence to the nodal DOF sequence (*e.g.*, $u, v, w, \theta_x, \theta_y, \theta_z$) specified by the START command of Processor TAB.

3.11.3.20 SEL_NODES (default = 0)

List of node numbers at which displacement histories are to be saved in dataset [NOMINAL_DS]. There should be [N_SELECT] numbers in the list, and node numbers can be repeated if more than one nodal DOF is to be saved at a node. The corresponding nodal DOF for each entry is specified by argument SEL_DOFS.

3.11.3.21 TOL_E (default = 1.e-3)

Error tolerance used to establish convergence of the nonlinear equilibrium iteration procedure at each load step. The iteration loop at a given step is terminated whenever the following condition is met:

$$\epsilon \leq [\text{TOL\_E}]$$

where

$$\epsilon = \sqrt{\frac{\mathbf{r}^{(i)} \cdot \delta\mathbf{d}^{(i)}}{\mathbf{r}^{(1)} \cdot \delta\mathbf{d}^{(1)}}}$$

is the *relative energy error norm*, $\mathbf{r}$ is the residual force vector, $\delta\mathbf{d}$ is the iterative displacement change, and i is the iteration counter.

3.11.4 INPUT/OUTPUT DATASETS

Table 3.11-1 Datasets Input/Output by Procedure NL_DYNAMIC_1

Dataset	*Description*	*Lib*	*Input*	*Output*
<ES_NAME>.EFIL.*	Element Computational Data	1	√	√
ES.SUMMARY	ES Processor Status	1	√	√
DEF.<ES_NAME>.*	Element Defn. (Connectivity)	1	√	
DIR.<ES_NAME>.*	Element EFIL Directory	1	√	
JDF1.BTAB.*	Model Summary	1	√	
JLOC.BTAB.*	Nodal Coordinates	1	√	
PROP.BTAB.*	Material/Section Properties	1	√	
QJJT.BTAB.*	Nodal Transformations	1	√	
TOT.DISP.*step*	System Displacement Vector	1	√	√
TOT.ROTN.*step*	Nodal Rotation Pseudovectors	1	√	√
TOT.VEL.*step*	System Velocity Vector	1	√	√
TOT.ACC.*step*	System Acceleration Vector	1	√	√
REAC.FORC.*step*	System Internal Force Vector	1		√

where *step* is the time step number, and ranges consecutively from 1 to the total number of steps computed. We must emphasize here that if one wishes to solve an initial value problem with either nonzero initial velocities or displacements (and not a restart), the user must supply TOT.DISP.0 and/or TOT.VEL.0. If they are not supplied, the procedure will initialize these to zero.

3.11.5 SUB-PROCEDURES AND PROCESSORS

Table 3.11-2 Sub-Procedures Invoked by Procedure NL_DYNAMIC_1

Procedure	*Type*	*Function*
NL_DYNAMIC_1	Internal	Main Procedure
CHK_CONVD	Internal	Check convergence
CONSTRAIN	Internal	Impose constraints on freedoms
DEFND	Internal	Defines recursive macrosymbols
ES	External	Element utility Procedure
FACTOR	Internal	Factor stiffness matrix
FORCE	Internal	Forms INTERNAL/EXTERNAL force vector
FT_ZERO	Internal	Zero force-time history
FT_LIN	Internal	Piecewise linear force-time history
FT_SIN	Internal	Sinusoidal force-time history
FT_EXP	Internal	Exponential decay force-time history
MASS	Internal	Compute mass
MASS_STIFF	Internal	Combine mass and stiffness matrices
NL_INITD	Internal	Initialize datasets and set restart
POSTRES	Internal	Save selected data, internal forces
POSTSTPD	Internal	Save data for restart
SOLVE	Internal	Solves linear equation systems
STIFFNESS	Internal	Forms and assembles stiffness matrix

Table 3.11-3 Processors Invoked by Procedure NL_DYNAMIC_1		
Procedure	*Type*	*Function*
AUS	Internal	Combines mass and stiffness matrices
E	Internal	Initializes EFIL datasets
ESi	External	Element Processors based on GEP
INV	Internal	Factors stiffness matrix
SSOL	Internal	Solves linear equation systems
VEC	Internal	Performs all vector/pseudo-vector algebra

3.11.6 LIMITATIONS

Limitations spelled out in Sections 4.1.6.1 through 4.1.6.8 apply also to NL_DYNAMIC_1. In addition, the following limitations also exist:

3.11.6.1 No Damping Included

No damping is included in this first version of NL_DYNAMIC_1. Damping that preserves the bandwidth of the problem will be included in a later version.

3.11.6.2 Initial Velocities and Displacements

These are currently the responsibility of the user. NL_DYNAMIC_1 will expect to see data sets TOT.DISP.0 and TOT.VEL.0 for these types of problems. Currently, there is no way to initialize rotational triads. This is a topic for future development.

3.11.6.3 Diagonal Mass

Currently, only diagonal mass can be used. This limitation will soon be removed.

3.11.6.4 AUS Limitation

Due to hard wired data set naming conventions in processor AUS for system matrices (*e.g.* K.SPAR.36), the number of DOF's per node specified on the START command of processor TAB is presently restricted to be 6. AUS is currently essential to NL_DYNAMIC_1 for computing the sum of the mass and stiffness matrices to obtain the dynamic operator. This restriction will be removed when we make the connection to the Generic Matrix Processor.

3.11.7 ERROR MESSAGES

3.11.7.1 "Non-Convergence at Step n. Revise Strategy."

This message means that the maximum number of nonlinear iterations (MAX_ITERS) has been exhausted, as well as the maximum number of step cuts (MAX_CUTS), and convergence still hasn't been obtained at step n. A possible cure is to re-start the analysis from several steps back, and decrease the arclength increment at that point (using the PATH_SCALE argument). However, just increasing MAX_ITERS or MAX_CUTS, or even TOL_E, may also solve the problem. In other words, re-think the definition of all solution parameters based on the observed behavior of the solution algorithm just prior to the break-down.

3.11.7.2 "Divergence at Step n. Revise Strategy."

This message has similar implications to the previous message, but it occurs when the error grows instead of decreases during two successive nonlinear iterations. The difference between divergence and non-convergence is that divergence cannot be cured by increasing MAX_ITERS; and probably should not be "cured" by increasing TOL_E. It generally

means that the step-size is too big — or that the error tolerance (TOL_E) has been too big all along, so that changes are occurring suddenly that should have been detected by the solution algorithm at earlier load steps. Thus, you might try re-starting from an earlier step, reducing PATH_INC, and possibly reducing TOL_E as well.

3.11.7.3 "Specified Displacements are Identically Zero"

This is not necessarily an abortive error. As long as either nonzero specified displacements *or* specified forces are defined, the solution can proceed — in which case the message should be taken merely as a warning.

3.11.7.4 "Specified Forces are Identically Zero"

This is not necessarily an abortive error. As long as either nonzero specified displacements *or* specified forces are defined, the solution can proceed — in which case the message should be taken merely as a warning.

3.11.8 USAGE GUIDELINES AND EXAMPLES

3.11.8.1 Starting an Analysis

To begin a nonlinear transient analysis with procedure NL_DYNAMIC_1, it is only necessary that the finite element model be defined. This does not require pre-formation of element stiffness matrices, node renumbering for optimal factorization time, or any form of linear analysis (unless initial geometric imperfections are based on linear displacement modes). Only nodal coordinates/transformations, material properties and element connectivity are pre-requisite to nonlinear analysis. To invoke procedure NL_DYNAMIC_1, only those arguments that don't have default values (see PROCEDURE USAGE section) need be specified.

For example, if you wanted to start an analysis with an initial time of 0., maximum time of .01, a time step of .005, and compute no more than 20 time steps with an initial velocity profile defined in data set TOT.VEL.0, you could invoke the procedure as follows:

```
*call NL_DYNAMIC_1 ( BEG_STEP = 1 ; MAX_STEPS = 20 ; -
                     BEG_TIME = .0 ; MAX_TIME = .1 ; -
                     DEL_TIME = 0.005 )
```

Keep in mind that the number of time-steps actually performed during the above run will depend on whether convergence difficulties were encountered. If that happened, the step will be cut, and the number of steps to maximum time will be greater. Since it may be difficult to estimate this in advance, you may want to start with only a few time steps (*e.g.*, set MAX_STEPS = 3) to get some experience, and later re-start the analysis with more steps allowed.

3.11.8.2 Re-Starting (or Continuing) an Analysis

To re-start an analysis from a previously computed, converged solution, you need only specify the first step to compute, the maximum time, and any changes in the forcing function. The only requirement is that a previous complete set of solution data for a converged time step must be present on the database. If, for example, one wishes to restart from step 10 (compute beginning step 11), then the call could be

```
*call NL_DYNAMIC_1 ( BEG_STEP = 11; MAX_STEPS = 100 ; -
                     MAX_TIME= .1 ; DEL_TIME = 0. ; -
                     FT_PROC=FT_LIN ; FT_ARGS=100.,0.,0.,1. )
```

In this case, the user wants to restart the previous example at time .05 with the sudden imposition of a step forcing function with scaled magnitude of 100. By examining Fig. 4.3-1, one can see that the step load is to be continued well beyond the termination time. FT_LIN is one of the procedures supplied here.

It is also possible to modify integration parameters and the time step, as well as any of the other applicable procedure arguments. Note that the parameter DEL_TIME is *always* required. The value zero is used to tell the procedure to look into the database for the last value. *A nonzero value for this parameter overrides the time step saved in the data base.*

3.11.9 THEORY

3.11.9.1 Introduction

Procedure NL_DYNAMIC_1 performs a nonlinear transient analysis of a system using a time integration algorithm that has adjustable parameters which allow for the automatic selection of a family of related transient analyzers. All of them are self starting, require no "historical" vectors, and for a useful range of parameters, are unconditionally stable. All are based on discrete matrix equations of motion that may or may not include damping. The internal and external force vectors can be nonlinear functions of the displacements (unknowns).

3.11.9.2 Development of the Algorithm

The discrete form of the equations of motion are

$$\mathbf{M}\ddot{\mathbf{u}} + \mathbf{f}^{int}(\mathbf{u}) = \mathbf{f}^{ext} \qquad (3.11-5)$$

where for simplicity we have omitted damping, and where the dependence of the internal forces on displacement and the external forces has been emphasized. Here **M** is the mass matrix, **u** is the displacement, and dots over quantities refer to differentiation with respect to time. Our goal is to find a discrete solution to (69) at time $t + \Delta t$ (step $n+1$), given information at time t (step n). In order to do this, we need additional relations between the displacements, velocities, and accelerations being solved for.

We begin with a two parameter family of the equations of motion (69) discretized in time as follows:

$$\mathbf{M}\mathbf{a}_{n+\theta} + (1+\alpha)\mathbf{f}^{int}(\mathbf{d}_{n+\theta}) - \alpha\mathbf{f}^{int}(\mathbf{d}_n) = \mathbf{f}^{ext}_{n+\theta} \qquad (3.11-6)$$

where $\mathbf{d}_{n+\theta}$ and $\mathbf{a}_{n+\theta}$ are the approximations to the displacement and acceleration at time $(n+\theta)\Delta t$, and where Wilson's θ interpolation parameter is used to obtain the desired acceleration according to

$$\mathbf{a}_{n+\theta} = (1-\theta)\mathbf{a}_n + \theta\mathbf{a}_{n+1} \qquad (3.11-7)$$

The parameter α is a measure of the "degree of implicitness" of the integrator. If α is 0, the equations of motion are to be satisfied at some point θ greater than unity. If $\alpha = -1$, we have an explicit system, with the acceleration extrapolated from values at the previous step.

To relate the velocity and acceleration to the displacements at the point $n+\theta$, we introduce the two standard Newmark parameters β and γ:

$$\mathbf{v}_{n+\theta} = \mathbf{v}_n + \theta\Delta t[(1-\gamma)\mathbf{a}_n + \gamma\mathbf{a}_{n+\theta}] \qquad (3.11-8)$$

$$\mathbf{d}_{n+\theta} = \mathbf{d}_n + \theta\Delta t\mathbf{v}_n + (\theta\Delta t)^2[(\frac{1}{2}-\beta)\mathbf{a}_n + \beta\mathbf{a}_{n+\theta}] \qquad (3.11-9)$$

Eq. (72) is required when damping is present. If we define the known quantity

$$\tilde{\mathbf{d}}_{n+\theta} = \mathbf{d}_n + \theta\Delta t\mathbf{v}_n + (\theta\Delta t)^2(\frac{1}{2} - \beta)\mathbf{a}_n \qquad (3.11-10)$$

then, using (73), we have

$$\mathbf{d}_{n+\theta} = \tilde{\mathbf{d}}_{n+\theta} + (\theta\Delta t)^2\beta\mathbf{a}_{n+\theta} \qquad (3.11-11)$$

If we multiply Eq. (70) by $(\theta\Delta t)^2\beta$ and eliminate $\mathbf{a}_{n+\theta}$ using (74), we obtain

$$\mathbf{M}\mathbf{d}_{n+\theta} + (\theta\Delta t)^2\beta(1+\alpha)\mathbf{f}^{int}(\mathbf{d}_{n+\theta}) - \tilde{\mathbf{f}}_{n+\theta} = 0 \qquad (3.11-12)$$

where the quantity $\tilde{\mathbf{f}}_{n+\theta}$ is known from the last converged time step:

$$\tilde{\mathbf{f}}_{n+\theta} = (\theta\Delta t)^2\beta[\mathbf{f}^{ext}_{n+\theta} + \alpha\mathbf{f}^{int}(\mathbf{d}_n)] + \mathbf{M}\tilde{\mathbf{d}}_{n+\theta} \qquad (3.11-13)$$

Eq. (77) is a nonlinear system of equations which must be solved for $\mathbf{d}_{n+\theta}$; these displacements will be solved for using an extrapolated starting solution followed by modified Newton corrections. The solution of this system is very similar to what is in NL_STATIC_1, with the exception of the arclength constraint and load factor equation. Thus, solution of (77) proceeds like a fixed-step static algorithm with a modified "stiffness" matrix and residual vector. In the algorithm description, these are labeled the dynamic operator and dynamic residual, respectively.

3.11.9.3 Derivation of the Dynamic Operator and Residual

The derivation of the dynamic operator and residual follows directly from the linearization of (77). Using a procedure similar to that for static analysis, we carry out a Taylor expansion of (77) and drop all terms of order higher than first. The result is the system

$$[\mathbf{M} + (\theta\Delta t)^2\beta(1+\alpha)\mathbf{K}(\mathbf{d}^{(0)}_{n+\theta})]\delta\mathbf{d}^{(i)}_{n+\theta} = \bar{\mathbf{f}}^{(i)}_{n+\theta} \qquad (3.11-14)$$

where

$$\bar{\mathbf{f}}^{(i)} = \tilde{\mathbf{f}}_{n+\theta} - \mathbf{M}\mathbf{d}^{(i-1)}_{n+\theta} - (\theta\Delta t)^2\beta(1+\alpha)\mathbf{f}^{int}(\mathbf{d}^{(i-1)}_{n+\theta}) \qquad (3.11-15)$$

is the dynamic residual. Here

$$\tilde{\mathbf{f}}_{n+\theta} = (\theta\Delta t)^2\beta[\mathbf{f}^{ext}(t_{n+\theta}) + \alpha\mathbf{f}^{int}(\mathbf{d}_n)] + \mathbf{M}\tilde{\mathbf{d}}_{n+\theta} \qquad (3.11-16)$$

which is computed from information known at the previous step or computed from the known external forcing function. $\delta\mathbf{d}^{(i)}_{n+\theta}$ is the vector of unknown displacement increments at iteration i, time point $t + \theta\Delta t$. One can tell that this is a modified Newton sequence, because of the argument of the stiffness matrix in (78). For true Newton, one would have to reform and refactor the stiffness matrix using the latest displacement information.

3.11.9.4 Recovery of Quantities at Step $n+1$

Once the displacements at point $t+\theta\Delta t$ are known, we need to update the accelerations and compute the new velocities and displacements at the desired new step. Starting with $\mathbf{d}_{n+\theta}$, we use the second of (73) to solve for $\mathbf{a}_{n+\theta}$. Introducing (74), the equation for the acceleration becomes

$$\mathbf{a}_{n+\theta} = \frac{1}{(\theta\Delta t)^2\beta}(\mathbf{d}_{n+\theta} - \tilde{\mathbf{d}}_{n+\theta}) \qquad (3.11-17)$$

We now require the accelerations, velocities, and displacements at time step $n+1$. Using (72), we have for $\mathbf{a}_{n+1}$

$$\mathbf{a}_{n+1} = \frac{1}{\theta}\mathbf{a}_{n+\theta} + (1 - \frac{1}{\theta})\mathbf{a}_n \qquad (3.11-18)$$

Eqs. (72) and (73) can then be used with $\theta = 1$ to interpolate velocities and displacements to step $n+1$, given the new and old accelerations:

$$\begin{aligned} \mathbf{v}_{n+1} &= \mathbf{v}_n + \Delta t[(1-\gamma)\mathbf{a}_n + \gamma\mathbf{a}_{n+1} \\ \mathbf{d}_{n+1} &= \mathbf{d}_n + \Delta t\mathbf{v}_n + \Delta t^2[(\frac{1}{2} - \beta)\mathbf{a}_n + \beta\mathbf{a}_{n+1}] \end{aligned} \qquad (3.11-19)$$

3.11.9.5 The Composition of New Displacements and Velocities

Whenever a new velocity vector is to be updated from a previous vector, the increment is simply added to the vector. For displacements, however, only the translations are handled this way. The *rotational triads* must be updated by the product rule, covered in Section 4.1.9.10. NL_DYNAMIC_1 is no different from NL_STATIC_1 in this respect. However, the reader will notice that the vector $\hat{\mathbf{d}}_{n+\theta}$ is treated like a velocity, with no accompanying rotational update. The reason is that $\hat{\mathbf{d}}_{n+\theta}$ eventually ends up as part of an incremental quantity used to account for inertial effects in the dynamic residual. The rotational freedoms in this vector are accumulated like their translational counterparts. To avoid confusion in the algorithm description, compositions that require special rotational treatment will be denoted by the symbol $\oplus$. Again, this is similar to the static case.

3.11.9.6 The Integration Parameters

The integration parameters allow the user to taylor the integrator to the special needs of his problem. The following table defines what we mean by each parameter. For detailed information on the various integrator options, the reader should consult references at the end of this section.

Parameter	Name
β, γ	Newmark Parameters
θ	Wilson's Collocation Parameter
α	Dissipation Parameter

Table 3.11-4 Parameter names used in the NL_DYNAMIC Procedure.

Special choices of these parameters select out well known integrators whose behavior has been studied extensively. These integrators, the parameter choices, and their regions of stability for stiff linear systems can be found in the following table.

Algorithm	α	β	γ	θ	Remarks
Newmark	0			1	Implicit and unconditionally stable when $2\beta \geq \gamma \geq \frac{1}{2}$
Trapezoidal	0	$\frac{1}{4}$	$\frac{1}{2}$	1	Implicit and unconditionally stable
Linear Acceleration	0	$\frac{1}{6}$	$\frac{1}{2}$	1	Implicit and conditionally stable
Fox–Goodwin	0	$\frac{1}{12}$	$\frac{1}{2}$	1	Implicit and conditionally stable
Central Difference	0	0	$\frac{1}{2}$	1	Explicit and conditionally stable
Wilson–θ	0	$\frac{1}{6}$	$\frac{1}{2}$		Implicit and unconditionally stable when $\theta \geq 1.366025$
Collocation	0		$\frac{1}{2}$		Implicit and unconditionally stable when $\theta \geq 1$ and $\frac{\theta}{2(\theta+1)} \geq \beta \geq \frac{2\theta^2-1}{4(2\theta^3-1)}$
α–Method				1	Implicit and unconditionally stable when $-\frac{1}{3} \leq \alpha \leq 0$, $\gamma = \frac{1}{2}(1-2\alpha)$ and $\beta = \frac{1}{4}(1-\alpha)^2$

Table 3.11-5. Some properties of the algorithms that can be produced using NL_DYNAMIC_1 through the α, β, γ, and θ parameters.

3.11.10 ALGORITHM

General Algorithm

(1) INITIALIZATION

(1.1) Initialize α, β, γ, θ.

(1.2) Initialize $\mathbf{d}_0$, $\mathbf{v}$.

(1.3) Compute time at θ

$$t_\theta = t_0 + \theta\Delta t$$

(1.4) Compute weighted out-of-balance force

$$\bar{\mathbf{f}} = \beta\mathbf{f}^{ext}(t_\theta) + (\frac{1}{2} - \beta)\mathbf{f}^{ext}(t_0) + \mathbf{f}^{int}(\mathbf{d}_0)[\beta(1+\alpha) - \frac{1}{2}]$$

(1.5) Add inertial terms

$$\tilde{\mathbf{f}}_\theta = \mathbf{M}(\mathbf{d}_0 + \theta\Delta t\mathbf{v}_0) + (\Delta t\theta)^2\bar{\mathbf{f}}$$

(1.6) Compute initial estimate of displacements

$$\mathbf{d}_\theta^{(0)} = \mathbf{d}_0 \oplus \theta\Delta t\mathbf{v}_0$$

(1.7) Compute initial dynamic residual

$$\bar{\mathbf{f}}^{(i)} = \tilde{\mathbf{f}}_{n+\theta} - \mathbf{M}\mathbf{d}_{n+\theta}^{(i-1)} - (\theta\Delta t)^2\beta(1+\alpha)\mathbf{f}^{int}(\mathbf{d}_{n+\theta}^{(i-1)})$$

(1.8) Set $n = 0$, $t = t_0 + \Delta t$, $n_{beg} = 1$

(2) STEP LOOP: for $n = n_{beg}, n_{beg} + 1, \ldots$

(2.1) Form and Assemble Stiffness Matrix based on predicted displacements

(2.2) Compute the Dynamic Operator

$$\mathbf{E} = [\mathbf{M} + (\theta\Delta t)^2\beta(1+\alpha)\mathbf{K}(\mathbf{d}_{n+\theta}^{(0)})]$$

(3) ITERATION LOOP: i = 1, 2, ... (iter = i+1 = 2, 3, ...)

(3.1) Solve for incremental displacement change

$$\mathbf{E}\delta\mathbf{d}_{n+\theta}^{(i)} = \bar{\mathbf{f}}_{n+\theta}^{(i)}$$

(3.2) Update displacements

$$\mathbf{d}_{n+\theta}^{(i)} = \mathbf{d}_{n+\theta}^{(i-1)} \oplus \delta\mathbf{d}_{n+\theta}^{(i)}$$

(3.3) Set iteration counter

$$i \leftarrow i+1$$

(3.4) Compute new Dynamic Residual

$$\bar{\mathbf{f}}^{(i)} = \tilde{\mathbf{f}}_{n+\theta} - \mathbf{M}\mathbf{d}_{n+\theta}^{(i-1)} - (\theta\Delta t)^2\beta(1+\alpha)\mathbf{f}^{int}(\mathbf{d}_{n+\theta}^{(i-1)})$$

(3.5) Check Convergence

call CHK_CONV ($\mathbf{r}_{n+1}^{(i+1)}, \delta\mathbf{d}, \widehat{\epsilon}^{(i)}, \epsilon_{ref}, \widehat{\epsilon}_{tol}$, num_div $\rightarrow$
$\widehat{\epsilon}^{(i+1)}$, <CONVERGED>, <DIVERGED>)

if (<CONVERGED>) then

num_iters_required = iter

$n \leftarrow n+1$

go to (2.3) STEP LOOP

elseif (<DIVERGED> .or. (iter > max_iters)) then

if (num_cuts < max_cuts) then

$\Delta t_{n+1} = \Delta t_{n+1}/2$

num_cuts = num_cuts + 1

$n_{beg} = n$

GO TO (1.3)

else

STOP

endif

else

go to (3) ITER LOOP

endif

(2.3) Compute acceleration at $n+\theta$

$$\mathbf{a}_{n+\theta} = \frac{1}{(\theta\Delta t)^2\beta}(\mathbf{d}_{n+\theta} - \tilde{\mathbf{d}}_{n+\theta})$$

(2.4) Compute acceleration at $n+1$

$$\mathbf{a}_{n+1} = \frac{1}{\theta}\mathbf{a}_{n+\theta} + (1-\frac{1}{\theta})\mathbf{a}_n$$

(2.5) Extract velocities

$$\mathbf{v}_{n+1} = \mathbf{v}_n + \Delta t[(1-\gamma)\mathbf{a}_n + \gamma\mathbf{a}_{n+1}]$$

(2.6) If $\theta \neq 1$, recompute displacements

$$\mathbf{d}_{n+1} = \mathbf{d}_n \oplus \{\Delta t\mathbf{v}_n + \Delta t^2[(\frac{1}{2}-\beta)\mathbf{a}_n + \beta\mathbf{a}_{n+1}]\}$$

(2.7) Update counters, timestep

$$n \leftarrow n+1; \qquad t \leftarrow t+1$$

(2.8) Check to see if $t > t_{max}$ or $n > n_{max}$

(2.9) If either statement in 2.8 is true, STOP; else GO TO 2.10

(2.10) Compute $\tilde{\mathbf{d}}_{n+\theta}$

$$\tilde{\mathbf{d}}_{n+\theta} = \mathbf{d}_n + \theta\Delta t\mathbf{v}_n + (\theta\Delta t)^2(\frac{1}{2}-\beta)\mathbf{a}_n$$

(2.11) Compute external loads $\mathbf{f}^{ext}(t)$ at current time t

(2.12) Compute $\tilde{\mathbf{f}}_{n+\theta}$

$$\tilde{\mathbf{f}}_{n+\theta} = (\theta\Delta t)^2\beta[\mathbf{f}^{ext}(t_{n+\theta}) + \alpha\mathbf{f}^{int}(\mathbf{d}_n)] + \mathbf{M}\tilde{\mathbf{d}}_{n+\theta}$$

(2.13) Compute initial estimate of new solution

$$\Delta\mathbf{d}_{n+\theta} = \tilde{\mathbf{d}}_{n+\theta} + (\theta\Delta t)^2\beta\mathbf{a}_n - \mathbf{d}_n$$
$$\mathbf{d}^{(0)}_{n+\theta} = \mathbf{d}_n \oplus \Delta\mathbf{d}_{n+\theta}$$

(2.14) END STEP_LOOP

NOTATION

$\mathbf{d}$	Displacement vector.
$\mathbf{v}$	Velocity vector.
$\mathbf{a}$	Acceleration vector.
$\delta\mathbf{d}$	Iterative change in d.
$\tilde{\mathbf{d}}$	Extrapolated displacement from step n
$\hat{\mathbf{f}}^{ext}$	External force vector — base load.
$\mathbf{f}^{int}$	Internal force vector.
$\mathbf{f}^{ext}$	External force vector.
$\tilde{\mathbf{f}}$	Extrapolated out-of-balance force.
$\bar{\mathbf{f}}$	The dynamic residual.
$\mathbf{K}$	Stiffness matrix.
num_cuts	Number of times load step has been cut in half at current step.
num_div	Number of consecutive iterations at which divergence occurs.
$\hat{\epsilon}$	Relative error in energy norm.
ϵ	Absolute error in energy norm.
ϵ_{ref}	Reference value of ϵ; initialized as zero.
ϵ_{tol}	Relative error tolerance (default: 10^{-4}).
λ	Load factor.
$\hat{\mathbf{d}}^s$	Base value of specified displacement vector.
$\mathbf{d}^s$	Current value of specified displacement vector. $\mathbf{d}^s = \lambda\,\hat{\mathbf{d}}^s$.
t	Problem time.
Δt	Time increment.
$n+\theta$	Time at step $n+\theta\Delta t$

3.11.11 PROCEDURE LISTING

```
*procedure NL_DYNAMIC_1 ( --
                      beg_step =1  ; -- . Starting step number (>0)
                      max_steps=1  ; -- . Maximum steps to compute
                      max_iters=9  ; -- . Maximum iterations per step
                      des_iters=4  ; -- . Number of iterations desired
                      fac_steps=1  ; -- . Steps_per_refactoring
                      max_cuts=3   ; -- . Maximum number of step cuts
                      tol_e=1.E-3  ; -- . Energy error tolerance
                      beg_time     ; -- . Starting time
                      max_time     ; -- . Upper_bound on time
                      del_time     ; -- . Time increment
                      int_pars = 1.,0.,.25,.5  ; -- . Itegration parameters
                      load_stiff=<false> ; -- . Include load stiffness
                      FT_proc=FT_ZERO     ; -- . Name of force-time procedure
                      FT_args            ; -- . Parameter array for above
                      debug=<FALSE>; -- . Debug_print switch
                      NL_GEOM = 2  ; -- . Geom. Nonlin. Level (1|2)
                      COROTATION=1 ; -- . Corotational Flag (leave on!!)
                      Nominal_DB = NOMINAL.GAL ; -- . Selected Output
                      Nominal_DS = RESPONSE.HISTORY ; --
                      N_SELECT ; SEL_NODES; SEL_DOFS  --
                    )
 *Remark  ------------------------------------------------------------
 *Remark |                                                            |
 *Remark                    N L _ D Y N A M I C _ 1
 *Remark
 *Remark           CSM Testbed Procedure for Nonlinear Statics:
 *Remark
 *Remark              o  Corotational Newton/Raphson algorithm
 *Remark              o  Linearized Crisfield/Riks arc-length control
 *Remark              o  Applied forces and/or displacements
 *Remark
 *Remark           Authors: G.M. Stanley and C.C. Rankin
 *Remark           Version: MAR-03-1988
 *Remark |                                                            |
 *Remark  ------------------------------------------------------------
.
.
. INITIALIZATION
.
 *def/i ns_overwrite == <false>
 *def/i debug        == [debug]
 *def/i max_step     == <[beg_step]+[max_steps]-1>
 *def/i num_iters    == [des_iters]
 *def/a NOM_DB       == [Nominal_DB]
 *def/a NOM_DS       == [Nominal_DS]
 *def/i N_SELECT     == [N_SELECT]
 *def/i SEL_NODES[1:<N_SELECT>] == [SEL_NODES]
 *def/i SEL_DOFS[1:<N_SELECT>]  == [SEL_DOFS]
 *def/i NL_GEOM      == [NL_GEOM]
 *def/i COROTATION   == [COROTATION]
```

```
*def/d beg_time      == [beg_time]
*def/d max_time      == [max_time]
*def/d int_pars[1:4] = [int_pars]
.
. . . . . Set time integration parameters
.
*def/d theta == <int_pars[1]>
*def/d alpha == <int_pars[2]>
*def/d newm_bet == <int_pars[3]>
*def/d newm_gam == <int_pars[4]>
.
. . . . . Initialize dataset names, constants
.
*CALL DEFND ( STEP = [BEG_STEP]; ITER = 0 )
*IF [BEG_STEP] /EQ 1  /THEN
   *Remark INITIALIZATION:
   *call NL_INITD
   *def/d t    == [beg_time]
   *def/d dt   == [del_time]
*else
   *if < [del_time] /eq 0.0 > /then
      *def/d dt == <dt_n>
   *else
      *def/d dt == [del_time]
   *endif
   *def/d      == <t_n>
*endif
.
. . . . Set initial step, initial iteration parameters
.
*def/i pass == 1
*def/i tot_iters == 1
*def/i istep == [beg_step]
.
. . . . Set initial time, basic time integration scalars
.
*def/d one_p_alp == <1. + <alpha>>
*def/d b_op_a    == <<newm_bet>*<one_p_alp>>
*def/d h_m_bet   == <.5 - <newm_bet>>
*def/d o_d_th    == <1./<theta>>
*def/d o_m_odth  == <1. - <o_d_th>>
*def/d o_m_gam   == <1. - <newm_gam>>
.
. . . . Reintery for problem restart on failure of convergence
. . . . System is treated as a new initial value problem
.
:NEW_PASS           . Re-entry point for step-size reduction
.
. . . . Set iteration and step-independent integration scalars
.
*def/d th_dt         == <<theta>*<dt>>
*def/d t_wils        == <<t> + <th_dt>>
*def/d th_dt_sq      == <<th_dt>*<th_dt>>
```

```
*def/d th_b          == <<newm_bet>*<th_dt_sq>>
*def/d a_th_b == <<th_b> * <alpha>>
*def/d o_d_thb == <1./<th_b>>
*def/d f0_coef == <<b_op_a> - .5>
*def/d k_coeff == <<th_dt_sq>*<b_op_a>>
*def/d v_n_coeff == <<dt>*<o_m_gam>>
*def/d v_np1_coeff == <<dt>*<newm_gam>>
*def/d dtsq == <<dt>*<dt>>
*def/d h_mb == <<dtsq>*<h_m_bet>>
*def/d dt_b == <<dtsq>*<newm_bet>>
*def/d h_mb_th == <<th_dt_sq>*<h_m_bet>>
.
. . . . Set initial displacement estimate, first step
. . . . Set initial estimate of weighted out-of-balance forces
.
.
. . . . Compute external force weighting constant lambda_n(time)
.
    *call [FT_proc] ( T = <t>              ; --  . Problem time
                      FT_ args = [ft_args] ; --  . Force-time parameters
                      F_mac =  lambda_n       ) . External forcing scale factor
    *call FORCE ( type         = EXTERNAL         ; --
                  nl_geom      = [nl_geom]        ; --
                  nl_load      = <true>           ; --
                  displacement = <d_n>            ; --
                  rotation     = <T_n>            ; --
                  corotation   = [corotation]     ; --
                  load_factor  = <lambda_n>       ; --
                  input_force  = <F_ref>          ; --
                  output_force = <F_ext> )
.
. . . . Set internal force
.
    *call FORCE ( type         = INTERNAL         ; --
                  nl_geom      = [nl_geom]        ; --
                  displacement = <d_n>            ; --
                  rotation     = <T_n>            ; --
                  corotation   = [corotation]     ; --
                  output_force = <F_int> )
. stop_1     *stop
.
. . . . Weighted difference between internal and external force
.
 [XQT VEC
  <a_th> <- <f_ext> - <f_int>
  <f_bar> <- <h_m_bet> <F_ext> + <f0_coef> <F_int>
.
. . . . External force at time = T + <theta> * DT
.
    *call [FT_proc] ( T = <t_wils>         ; --  . Problem time
                      FT_ args = [ft_args] ; --  . Force-time parameters
                      F_mac =  lambda_np1     ) . External forcing scale factor
    *call FORCE ( type         = EXTERNAL         ; --
```

```
                  nl_geom      = [nl_geom]        ; --
                  nl_load      = <true>           ; --
                  displacement = <d_n>            ; --
                  rotation     = <T_n>            ; --
                  corotation   = [corotation]     ; --
                  load_factor  = <lambda_np1>     ; --
                  input_force  = <F_ref>          ; --
                  output_force = <F_ext> )
.
. . . . Add correction to external force
.
 [XQT VEC
  <f_bar> <- <f_bar> + <newm_bet> <F_ext>
  <f_bar> <- <th_dt_sq> <f_bar>
.
. . . . Initial estimate of displacements
.
  <delta> <- <th_dt> <v_n>
  <d_np1_i> <- <d_n> + <delta>
  ROTATE <T_n>  *  <delta> -> <T_np1_i>
  <d_tilde> <- <d_np1_i>
.
. . . . Scaling displacement increment for convergence check
.
. stop_2  *stop
  PROD <inv_mass> <a_th> -> <d_inc_i>
  <d_inc_i> <- <h_mb_th> <d_inc_i> + <delta>
.
. . . . Mass * Displacements + scaled out-of-balance force
.
 PROD <mass> <d_np1_i> -> <f_tilde>
 <f_tilde> <- <f_tilde> + <f_bar>
.
.
.
. . . . BEGIN STEP_LOOP  **********   **********   **********
.
.
.
 *DO :STEP_LOOP $n = <istep>, <[max_steps]-1>
     *def/i step == <$n>
     *Remark
     *remark ------------------------------------------------------------
     *remark
     *Remark       BEGINNING STEP <STEP>
     *remark ------------------------------------------------------------
     *remark
     *call CONSTRAIN ( load_factor    = <lambda_np1 > ; --
                       applied_motion = <d_spec>       ; --
                       displacement   = <d_np1_i>          )
.
. . . Form Stiffness based on Predictor
.
```

```
       *call STIFFNESS ( type         = TANGENT        ; --
                         nl_geom      = [nl_geom]      ; --
                         nl_load      = [load_stiff]   ; --
                         load_factor  = <lambda_np1>   ; --
                         displacement = <d_np1_i>      ; --
                         rotation     = <T_np1_i>      ; --
                         corotation   = [corotation]   ; --
                         stiffness    = <K_asm>           ) . output
.
. . . Combine MASS and STIFFNESS into one matrix <M_p_K>
.
       *call MASS_STIFF ( mass        = <mass>    ; -- . Mass vector
                          stif        = <K_asm>   ; -- . Assembled Stiffness
                          mult        = <k_coeff> ; -- . Stiffness Multiplier
                          M_p_K       = <M_p_K>   )   . The dynamic operator
.
. . . Factor DYNAMIC OPERATOR
.
       *call FACTOR    ( input_matrix = <M_p_K>   ; --
                         output_matrix= <M_p_K> )
.
. . . Obtain MASS RESIDUAL
.
    *call FORCE ( type         = INTERNAL         ; --
                  nl_geom      = [nl_geom]        ; --
                  displacement = <d_np1_i>        ; --
                  rotation     = <T_np1_i>        ; --
                  corotation   = [corotation]     ; --
                  output_force = <F_int> )
.
.
. stop_3 *stop
 [XQT VEC
    PROD <mass> <d_np1_i> -> <R_np1_i>
    <R_np1_i> <- <f_tilde> - <R_np1_i>
    <R_np1_i> <- <R_np1_i> - <k_coeff> <F_int>
.
. . . Initialize Convergence Criteria
.
    *call CHKCNVD ( STEP             = 1               ; --
                    ITER             = 1               ; --
                    Residual_force   = <R_np1_i>       ; --
                    displacement_inc = <d_inc_i>          )
.
.
.
. -- BEGIN ITERATION LOOP   ****** ******* ******* *********
.
.
.
. stop_4 *stop
 *echo,on,ma,md
   *DO :ITER_LOOP $i = 1, [max_iters]
```

```
         *def/i iter == <<$i>+1>
         *def/i tot_iters == <<tot_iters>+1>
.
. . . . .  Compute Basic Displacement Iteration (delta_th)
.
         *call SOLVE ( rhs = <R_np1_i>; soln = <delta_th>; matrix = <M_p_K> )
.
.
. . . . .  Update Displacements, Rotations, Increments
.
 [XQT VEC
          <d_np1_ip1>  <-  <d_np1_i>  +  <delta_th>
          ROTATE  <T_np1_i>  *  <delta_th>   ->   <T_np1_ip1>
         *call CONSTRAIN ( load_factor    = <lambda_np1>; --
                           applied_motion = <d_spec>     ; --
                           displacement   = <d_np1_ip1> )
         *if < <DEBUG> > /then
            *print 1, <d_np1_ip1>
            *print 1, <T_np1_ip1>
         *endif
.
. . . . .  Compute DYNAMIC RESIDUAL
.
     *call FORCE ( type         = INTERNAL         ; --
                   nl_geom      = [nl_geom]        ; --
                   displacement = <d_np1_ip1>      ; --
                   rotation     = <T_np1_ip1>      ; --
                   corotation   = [corotation]     ; --
                   output_force = <F_int> )
.
.
 [XQT VEC
 PROD <mass> <d_np1_ip1> -> <R_np1_ip1>
 <R_np1_ip1> <- <f_tilde> - <R_np1_ip1>
 <R_np1_ip1> <- <R_np1_ip1> - <k_coeff> <F_int>
.
. . . . .  CHECK CONVERGENCE
.
         *call CHKCNVD   ( step      = <step>       ; --  . input
                           iter      = <iter>       ; --
                           Residual_force   = <R_np1_ip1> ; --
                           displacement_inc = <delta_th>     ; --
                           tol_e            = [tol_e]     ; --
                           max_iters        = [max_iters] ; --
                           convergence = CONVERGENCE      ; -- . output
                           divergence  = DIVERGENCE          )
       *if <CONVERGENCE> /eq <TRUE> /then
         *jump to :CONVERGED
       *elseif   <DIVERGENCE> /eq <TRUE> /then
         *if <pass> /le [max_cuts] /then
            *jump to :REPEAT
         *else
            *Remark Maximum number of automatic step cuts exhausted.
```

```
            *jump to :DIVERGED
          *endif
        *endif
      :ITER_LOOP
      :NEUTRAL
      *if <pass> /le [max_cuts] /then
         *jump to :REPEAT
      *else
         *Remark Maximum number of automatic step cuts exhausted.
      *endif
      *Remark           NON-CONVERGENCE AT STEP <step>.  REVISE STRATEGY.
      *jump to :EXIT
      :REPEAT
       *def/i pass      == < <pass>+1 >
       *def/d dt == <<dt>*.5>
       *Remark
       *Remark CONVERGENCE DIFFICULTIES; REPEATING STEP <step>
       *Remark with half original time step
       *Remark New TIME increment is <dt>
       *Remark Pass: <pass>
       *jump to :NEW_PASS
      :DIVERGED
      *Remark           DIVERGENCE AT STEP <STEP>.  REVISE STRATEGY.
      *jump to :EXIT
      :CONVERGED
      *Remark           CONVERGENCE AT STEP <STEP>.
      :NEXTSTEP
      *def/i num_iters     == <iter>
      *def/d path_scale    == <[des_iters]/<num_iters>>
      *def/d lam_n         == <lambda_np1>
      *def/i sign_det_n    == <sign_det>
      *def/i load_dir_n    == <load_dir>
.
. . . . Compute current acceleration
.
 [XQT VEC
  *if <<theta> /ne 1.0> /then
     <a_th>  <- <o_d_th> <d_np1_ip1> - <o_d_th> <d_tilde>
     <a_np1> <- <o_d_th> <a_th> + <o_odth> <a_n>
  *else
     <a_np1>  <- <o_d_th> <d_np1_ip1> - <o_d_th> <d_tilde>
  *endif
.
. . . . Update velocity vector
.
  <a_th> <- <v_n_coeff> <a_n> + <v_np1_coeff> <a_np1>
  <v_np1> <- <v_n> + <a_th>
.
. . . . For strange (non-unity) values of Wilson's parameter, update
. . . .       displacements to step n+1
.
 *if <<theta> /ne 1.0> /then
     <a_th> <- <h_mb> <a_n> + <dt_b> <a_np1>
```

```
     <delta> <- <dt> <v_n> + <a_th>
     <d_np1_i> <- <d_n> + <delta>
     ROTATE <T_n>  *  <delta> -> <T_np1_i>
 *endif
.
. . . . Postprocess and save results of computations at <<STEP> + 1>
.
    *call POSTSTPD ( step=<step>; iter=<iter> )
.
. . . . Reset iteration counters
.
 *def/i pass == 1
 *def/i tot_iters == 1
.
. . . . Update current time
.
 *def/d t_wils == <<t> + <th_dt>>
 *def/d t      == <<t> + <dt>> . can be altered by path_scale
 *stop
.
. . . . Check for max time and max steps
.
 *if <<t> /gt <max_time> > /then
*remark ***************************************************************************
*remark *                                                                         *
*remark *       Maximum problem time <max_time> reached or exceeded.              *
*remark *                        Computation terminated                           *
*remark *                                                                         *
*remark ***************************************************************************
     *jump to :EXIT
 *endif
 *if <<step> /eq <<max_steps>- 1>>/then
*remark ***************************************************************************
*remark *                                                                         *
*remark *       Maximum steps <max_steps> exceeded.                               *
*remark *                Problem time is <t>                                    *
*remark *                                                                         *
*remark ***************************************************************************
     *jump to :EXIT
 *endif
.
. . . . Reset step counter for datasets to step <<step> + 1>
.
 *call DEFND ( STEP = <<STEP> + 1>; ITER=1)
.
. . . . Compute new "historical" displacement d_tilde
.
 [XQT VEC
 <a_th> <- <th_dt> <v_n> + <h_mb_th> <a_n>
 <d_tilde> <- <d_n> + <a_th>
.
. . . . Compute new external force for step n+1
.
```

```
    *call [FT_proc] ( T = <t_wils>          ; --  . Problem time
                      FT_ args = [ft_args] ; --  . Force-time parameters
                      F_mac =  lambda_np1     ) . External forcing scale factor
    *call FORCE ( type         = EXTERNAL          ; --
                  nl_geom      = [nl_geom]         ; --
                  nl_load      = <true>            ; --
                  displacement = <d_n>             ; --
                  rotation     = <T_n>             ; --
                  corotation   = [corotation]      ; --
                  load_factor  = <lambda_np1>     ; --
                  input_force  = <F_ref>           ; --
                  output_force = <F_ext> )
.
. . . . Set internal force if Wilson's constant is not unity
.
    *if<<theta> /ne 1.> /then
       *call FORCE ( type         = INTERNAL          ; --
                     nl_geom      = [nl_geom]         ; --
                     nl_load      = <true>            ; --
                     displacement = <d_n>             ; --
                     rotation     = <T_n>             ; --
                     corotation   = [corotation]      ; --
                     load_factor  = <lambda_np1>     ; --
                     input_force  = <F_ref>           ; --
                     output_force = <F_int> )
     *endif
.
. . . . Weighted sum of internal and external force
.
 [XQT VEC
  <f_bar> <- <th_b> <F_ext> + <a_th_b> <F_int>
 PROD <mass> <d_n> -> <f_tilde>
 <f_tilde> <- <f_tilde> + <f_bar>
.
. . . . Compute new initial estimate, displacements
.
 [XQT VEC
  <delta> <- <d_tilde> - <d_n>
  <delta> <- <delta> + <th_b> <a_n>
  <d_np1_i> <- <d_n> + <delta>
  ROTATE <T_n>  *  <delta> -> <T_np1_i>
.
. . . . End Step LOOP
.
 :STEP_LOOP
 :EXIT
*end
. =DECK CHKCNVD
*procedure CHKCNVD   ( STEP                  ; --  . input
                       ITER                  ; --
                       DISPLACEMENT_INC      ; --
                       RESIDUAL_FORCE        ; --
                       TOL_E                 ; --
```

```
                    MAX_ITERS               ; --
                    CONVERGENCE=CONVERGENCE ; -- . output
                    DIVERGENCE =DIVERGENCE      )
[XQT VEC
.
. . Initialize
.
   *def/i [CONVERGENCE] == <FALSE>
   *def/i [DIVERGENCE]  == <FALSE>
   *if [ITER] /gt 1 /then
      *def/d12.4 ERR_E_I == <ERR_E_IP1>
   *endif
.
. . Compute Current Incremental Energy Norm
.
    DOT  [RESIDUAL_FORCE] * [DISPLACEMENT_INC] -> INC_E_IP1
   *if [ITER] /le 1 /then
      *if <ABS(<INC_E_IP1>)> /gt <REF_E> /then
         *def/d12.4 REF_E == <ABS(<INC_E_IP1>)>
      *endif
      *def/i     num_diverges ==  0
   *endif
.
. . Compute Error Norms
.
   *def/d12.4  ERR_E_RAW ==  < <INC_E_IP1>/<REF_E> >
   *def/d12.4  ERR_E_IP1 ==  < <ABS(<ERR_E_RAW>)>^ .5 >
.
. . Check for displacement convergence
.
   *if  < <ERR_E_IP1> /le [TOL_E] >  /then
.
. . .  Step Converged
.
      *DEF/I [CONVERGENCE] == <TRUE>
      *jump to :BOTTOM_LINE
   *endif
   *if  < [ITER] /le 1 >  :BOTTOM_LINE
.
. . .  Step Not Converged; Check for Divergence
.
      *IF  < <ERR_E_IP1>/<ERR_E_I> /GT 1.0 >  /THEN
         *def num_diverges == < <num_diverges>+1 >
         *if <num_diverges> /gt 1 /then
            *DEF/I [DIVERGENCE] == <TRUE>
         *ENDIF
      *ENDIF
   :BOTTOM_LINE
   *def/e12.4 load_factor  = <lambda_np1>
*remark
*remark ----------------------------------------------------------------
*remark          ITER=[ITER]  T=<t>  LD=<load_factor>
*remark          ERR=<err_e_ip1>     REF=<ref_e>
```

```
*remark
*remark
*END
. =DECK DEFND
*procedure DEFND ( step; iter )
.
. Purpose:  Define basic macro_symbols for NL_DYNAMIC procedure
.
*def/i np1 = [step]
*def/i n   = < <np1>-1 >
*if [iter] /eq 0 /then              . run initialization
   *def/i NS_ldi    == 1
   *def/i NOM_ldi   == 3
   *def/i iset      == 1
   *def/i icon      == 1
   *def/a f_spec    == APPL.FORC.<iset>.1
   *def/a d_spec    == APPL.MOTI.<iset>.1
.
.  Check for Specified Displacement Loading
.
   *find dataset <NS_ldi> <d_spec> /seq=ids
   *if < <ids> /gt 0 > /then
      *Remark Note: Specified displacement dataset <d_spec> will be used
      *def/i spec_disp_flag == <true>
   *else
      *def/i spec_disp_flag == <false>
   *endif
.
   *def/a F_ref     ==  REF.FORC
   *def/a f_ext     ==  EXT.FORC
   *def/a f_int     ==  INT.FORC
   *def/a mass      ==  M.DIAG
   *def/a inv_mass  ==  MINV.DIAG
   *def/a R_np1_i   ==  RES.FORC
   *def/a R_np1_ip1 ==  RES.FORC
   *def/a d_inc_i   ==  INC.DISP
   *def/a d_inc_ip1 ==  INC.DISP
   *def/a v_inc_i   ==  INC.VEL
   *def/a v_inc_ip1 ==  INC.VEL
   *def/a delta_TH  ==  THET.DISI
   *def/a a_TH      ==  THET.ACC
   *def/a delta     ==  TOT.DISI
   *def/a d_tilde   ==  TILD.DISP
   *def/a f_tilde   ==  TILD.FORC
   *def/a f_bar     ==  BAR.FORC
   *def/a K_asm     ==  K
   *def/a K_fac     ==  K
   *def/a M_p_K     ==  MpK
.
.   Retrieve Control Parameters from DATA_BASE
.
   *if [step] /eq 1 /then
      *def/d t_n      == <beg_time>
```

```
      *def/i sign_det == 1
      *def/d ref_e    == 0.0
      *def/d lambda_n == 0.0
      *Remark    STARTING PARAMETERS INITIALIZED.
   *else
      *open <NOM_ldi> <NOM_DB>
      *find dataset <nom_ldi> <nom_ds> /seq=nom_ids
      *g2m /name==t_n        /type=D <nom_ldi> <nom_ids> TIME.<n>
      *g2m /name==sign_det_n /type=I <nom_ldi> <nom_ids> SIGN_DET.<n>
      *g2m /name==lambda_n   /type=D <nom_ldi> <nom_ids> LOAD.<n>
      *g2m /name==ref_e      /type=D <nom_ldi> <nom_ids> REF_ERR.<n>
      *close <NOM_ldi>
      *Remark
      *remark ------------------------------------------------------------
      *remark
      *Remark    RE-START PARAMETERS:
      *Remark      TIME                    = <t_n>
      *Remark      SIGN_DET (n)            = <sign_det_n>
      *Remark      LOAD                    = <lambda_n>
      *remark
      *remark ------------------------------------------------------------
   *endif
*endif
.
. Define Global Datasets for Upcoming Step
.
*def/a d_n       == TOT.DISP.<n>
*def/a d_np1_i   == TOT.DISP.<np1>
*def/a d_np1_ip1 == TOT.DISP.<np1>
*def/a v_n       == TOT.VEL.<n>
*def/a v_np1     == TOT.VEL.<np1>
*def/a a_n       == TOT.ACC.<n>
*def/a a_np1     == TOT.ACC.<np1>
*def/a T_n       == TOT.ROTN.<n>
*def/a T_np1_i   == TOT.ROTN.<np1>
*def/a T_np1_ip1 == TOT.ROTN.<np1>
*end
. =DECK FT_ZERO
*procedure FT_ZERO ( t ; ft_args ; F_mac )
.
. . . . Provide for zero dependence of force on time (no external forces)
.
 *def/d [F_mac] == 0.
.
 *end
. =DECK FT_LIN
*procedure FT_LIN ( t ; ft_args ; F_mac )
.
. . . . Provide piecewise linear forcing function
.
 *def/d ft_args[1:5] = [ft_args]
.
.
```

```
*if <[t] /le <ft_args[2]>> /then
     *def/d [F_mac] == 0.
*elseif <[t] /lt <ft_args[3]>> /then
     *def/d c_fact = <<ft_args[3]>-<ft_args[2]>>
     *if <<c_fact> /le 0.> /then
         *def/d [F_mac] == <ft_args[1]>
     *else
         *def/d [F_mac] == <<<[t]-<ft_args[2]>> * --
                     <ft_args[1]>>/<c_fact>>
     *endif
*elseif <[t] /le <ft_args[4]>> /then
     *def/d [F_mac] == <ft_args[1]>
*elseif <[t] /lt <ft_args[5]>> /then
     *def/d c_fact = <<ft_args[5]>-<ft_args[4]>>
     *if <<c_fact> /le 0.> /then
         *def/d [F_mac] == 0.
     *else
         *def/d [F_mac] == <<<<ft_args[5]> - [t]> * --
                    <ft_args[1]>>/<c_fact>>
     *endif
*else
     *def/d [F_mac] == 0.
*endif
.
.
*end
. =DECK FT_SIN
*procedure FT_SIN ( t ; ft_args ; F_mac )
.
 *def/d ft_args[1:6] = [ft_args]
.
. . . . . Piecewise sinusoidal forcing function
.
 *if <[t] /ge <ft_args[6]>> /then
     *def/d [F_mac] = 0.
 *elseif <[t] /ge <ft_args[5]>> /then
     *if <<ft_args[3]> /le 0.> /then
         *def/d [F_mac] == 0.
     *else
         *def/d tc = <<<[t] - <ft_args[4]>>*<pi>>/<ft_args[3]>>
         *def/d [F_mac] == <<ft_args[2]> + <cos(<tc>)> * <ft_args[1]>>
     *endif
 *else
     *def/d [F_mac] == 0.
 *endif
.
.
*end
. =DECK FT_EXP
*procedure FT_EXP ( t ; ft_args ; F_mac )
.
. . . . Exponential decay
.
```

```
 *def/d ft_args[1:3] = [ft_args]
.
 *if <[t] /lt <ft_args[2]>> /then
     *def/d [F_mac] == 0.
 *else
     *def/d apl = <.6931471806/<ft_args[3]>>
     *def/d axl = <<apl>*<[t] - <ft_args[2]>>>
     *def/d [F_mac] == <<ft_args[1]>* <EXP(-<axl>)>>
 *endif
.
.
*end
. =DECK NL_INITD
*procedure NL_INITD ( ldi=1 )
.  ------------------------------
.  PERFORM STANDARD INITIALIZATION
.  ------------------------------
   *call INITIALIZE ( constraint_set = <icon> )
.  ---------------------------------------------------
.  CLEAR INITIAL DISPLACEMENTS AND ROTATIONS   ( n = 0 )
.  ---------------------------------------------------
   *g2m /name=num_nodes /type=i [ldi] MODEL.SUMMARY NUM_NODES
   *g2m /name=num_dofs  /type=i [ldi] MODEL.SUMMARY NUM_DOFS
   *Remark    Problem Dimensions: N_nodes = <num_nodes>, N_dofn = <num_dofs>
   *find dataset <NS_ldi> <d_n> /seq=ids_disp
   *if < <ids_disp> /gt 0 > /then
      *Remark Note: Initial displacement dataset <d_n> will be used
   *else
      INIT_VEC <d_n> <num_dofs> BY <num_nodes> . Zeroize translations
      *Remark    Displacements (<d_n>) initialized.
   *endif
   *find dataset <NS_ldi> <T_n> /seq=ids_rotn
   *if < <ids_rotn> /gt 0 > /then
      *Remark Note: Initial rotation dataset <T_n> will be used
   *else
      INIT_VEC <T_n>   3        BY <num_nodes> . Zeroize rotation pseudovectors
      *Remark    Rotations (<T_n>) initialized.
   *endif
   *find dataset <NS_ldi> <v_n> /seq=ids_vel
   *if < <ids_vel> /gt 0 > /then
      *Remark Note: Initial velocity dataset <v_n> will be used
   *else
     INIT_VEC <v_n> <num_dofs> BY <num_nodes> . Zeroize velocity
     *Remark    Velocities (<v_n>) initialized.
   *endif
   *find dataset <NS_ldi> <a_n> /seq=ids_acc
   *if < <ids_acc> /gt 0 > /then
      *Remark Note: Initial acceleration dataset <a_n> will be used
   *else
      INIT_VEC <a_n> <num_dofs> BY <num_nodes> . Zeroize acceleration
      *Remark    Accelerations (<a_n>) initialized.
   *endif
.  ---------------------------------------------
```

```
.  FORM INITIAL (BASE-LOAD) EXTERNAL-FORCE VECTOR
.  -----------------------------------------------
   *call FORCE ( type         = EXTERNAL ; --
                 input_force  = <f_spec> ; --
                 output_force = <F_ref>      )
.
.  -----------------------------------------------
.  Form mass and inverse mass (diagonal form)
.  -----------------------------------------------
.
*remark
*remark MASS bypassed for testing purposes defined in BASE.DBC
*remark
.   *call MASS (  type  =  DIAGONAL ; --
.                 mass  =  <mass>       )
   [XQT VEC
      DIAG_INV  <mass> -> <inv_mass>
*end
. =DECK MASS_STIFF
*procedure MASS_STIFF ( mass      ; -- . diagonal mass vector
                        stif      ; -- . assembled stiffness
                        mult      ; -- . the dynamic weighting factor
                        M_p_K     )    . the dynamic operator
.
 [XQT AUS
    DEFINE M = <NS_ldi> M DIAG
    DEFINE K = <NS_ldi> [STIF] SPAR 36
    [M_p_K] SPAR 36 = SUM(M, [MULT] K)
.
*end
. =DECK POSTRES
*procedure POSTRES ( step )
 [XQT VEC
  *find dataset <NOM_LDI> <NOM_DS> /seq=post_ids
  *if < <post_ids> /le 0 > /then
     *put dataset <NOM_LDI> <NOM_DS> /mrat=2000 /seq=post_ids
  *endif
.
. Save selected displacements on nominal data-base
.
   *def/a dof_names = U, V, W, RU, RV, RW
.
   *do $isel = 1, <N_SELECT>
    *def/i node = <SEL_NODES[<$isel>]>
    *def/i dof  = <SEL_DOFS[<$isel>]>
    *def/a dof_name = <dof_names[<dof>]>
     COMPONENT <node> <dof> TOT.DISP.[step] -> DISP
    *def/a recd_name = DISP_<dof_name>_<node>.[step]
    *Remark <recd_name> = <DISP>
    *m2g /name=disp /type=d <nom_ldi> <post_ids> <recd_name>
.
.    Save Reaction Forces for Current Step
.
```

```
     REAC.FORC.[step]  <-  INT.FORC
    *if < [step] /eq 0 > /then
       REAC.FORC.0  <-  0.0
    *endif
    *if < <spec_disp_flag> > /then
        COMPONENT <node> <dof> REAC.FORC.[step] -> FORCE
       *def/a recd_name = FORCE_<dof_name>_<node>.[step]
       *Remark <recd_name> = <FORCE>
       *m2g /name=force /type=d <nom_ldi> <post_ids> <recd_name>
    *endif
   *enddo
*end
. =DECK POSTSTPD
*procedure POSTSTEP ( step; iter )
*Remark ___________________
*Remark STEP [step] SUMMARY :
*Remark ____________________________________________________________
*Remark Load Factor _____________________ <lambda_np1>
*Remark Stiffness determinant ____________ <coef_det> * 10 ^ <exp10_det>
*Remark Number of negative roots _________ <num_neg>
*Remark Relative energy_error ____________ <err_e_ip1>
*Remark Number of Iterations _____________ <num_iters>
*Remark Number of Step Cuts ______________ <<pass>-1>
*Remark Total Number of Iterations _______ <tot_iters>
*Remark ____________________________________________________________
*open <NOM_ldi> <NOM_DB>
*if < [step] /eq 1 > /then
   *call POSTRES ( step = 0 )
   *def/d load_0=0.
   *m2g /name=load_0 /type=d <NOM_LDI> <NOM_DS> LOAD.0:0
*endif
*call POSTRES  ( step=[step] )
*find dataset <NOM_LDI> <NOM_DS> /seq=nom_ids
*m2g /name=lambda_np1 /type=d <nom_ldi> <nom_ids> LOAD.[STEP]
*m2g /name=err_e_ip1  /type=d <nom_ldi> <nom_ids> ERROR.[STEP]
*m2g /name=ref_e      /type=d <nom_ldi> <nom_ids> REF_ERR.[STEP]
*m2g /name=sign_det   /type=i <nom_ldi> <nom_ids> SIGN_DET.[STEP]
*m2g /name=coef_det   /type=d <nom_ldi> <nom_ids> COEF_DET.[STEP]
*m2g /name=exp10_det  /type=i <nom_ldi> <nom_ids> EXP10_DET.[STEP]
*m2g /name=num_neg    /type=d <nom_ldi> <nom_ids> NEG_ROOTS.[STEP]
*m2g /name=num_iters  /type=i <nom_ldi> <nom_ids> NUM_ITERS.[STEP]
*m2g /name=tot_iters  /type=i <nom_ldi> <nom_ids> TOT_ITERS.[STEP]
*m2g /name=t          /type=i <nom_ldi> <nom_ids> TIME.[STEP]
*def/i passm1 = <<pass>-1>
*m2g /name=passm1 /type=i <nom_ldi> <nom_ids> NUM_CUTS.[STEP] . cgl, 7/26/88
*close <NOM_ldi>
*end
```

3.11.12 REFERENCES

3.11-1 Almroth, B.O., Brogan, F.A., and Stanley, G.M., "Structural Analysis of General Shells, Volume II, User's Instructions for STAGSC-1," Lockheed Palo Alto Research Laboratory, Palo Alto, CA, Rept. #LMSC-D633873, December 1982.

3.11-2 Hilber, H.M. and Hughes, T.J.R, "Colocation, Dissipation and Overshoot for Time Integration Schemes in Structural Dynamics," *Earthquake Engineering and Structural Dynamics*, Vol. 6, pp. 99-118, 1978.

3.11-3 Newmark, N.M., "A Method of Computation for Structural Dynamics," *Journal of the Engineering Mechanics Division*, ASCE, pp. 67-94, 1959.

3.11-4 Hilber, H.M., Hughes, T.J.R., and Taylor, R.L., "Improved Numerical Dissipation for Time Integration Algorithms in Structural Dynamics," *Earthquake Engineering and Structural Dynamics*, Vol. 5, pp. 283-292, 1977.

3.11-5 Bathe, K.J., and Wilson, E.L., "Stability and Accuracy Analysis of Direct Integration Methods," *Earthquake Engineering and Structural Dynamics*, pp. 283-291, 1973.

3.11-6 Wilson, E.L., "A Computer Program for Dynamic Stress Analysis of Underground Structures," SESM Report No. 68-1, Division of Structural Engineering and Structural Mechanics, University of California, Berkeley, 1968.

4.0 Application Procedures

The procedures documented in this chapter are representative of the types of procedures that may be written to solve specific application (structural analysis) problems. Many of these high-level procedures invoke other (lower-level) procedures to perform preprocessing, solution, and postprocessing functions; which are described elsewhere in this manual. The use of procedures to perform structural analysis applications can provide users flexibility for parameterizing geometric data (*e.g.*, stiffener spacing) as well as spatial discretization parameters (*e.g.*, number of elements). The problems represented here are also intended to serve as part of a standard series of test problems to assess new structural elements installed in the CSM Testbed.

A summary of the procedures found in this chapter is provided in Tables 4.0-1.

Table 4.0-1. Summary of Application Procedures

Procedure Name	*Problem Description*
CLAMPED_BEAM	Transient response of a clamped beam modeled with 2-D shell elements.
COMPRESSED_CYL	Classical buckling and postbuckling analysis of an axially compressed cylindrical shell; modeled with shell elements and initial imperfections.
COOKS_MEM	Linear in-plane bending response of a wing-like trapezoidal plate modeled with shell elements. (Referred to as Cook's membrane problem in the literature.)
ELASTICA	Classical large rotation analysis of a cantilevered beam with an applied end moment. Final configuration is a complete circle — with 360 degrees rotation at the free edge. Modeled with shell elements.
EULER_COLUMN	Inplane buckling of a column modeled with 2-D shell elements. An overall Euler buckling modeshape is obtained.
FOCUS_PANEL	Analysis of the composite blade-stiffened panel with a discontinuous center stiffener.
FREE_EDGE	Free-edge stress analysis of a 4-ply composite laminate.
GEN_STF_PANEL	Linear buckling analysis of stiffened panel configurations subjected to combined in-plane compression and shear loading. Configurations considered include flat rectangular panels with hat-stiffeners, z-stiffeners and blade stiffeners, as well as a corrugated panel. Modeled with shell elements.
HINGED_CYL	Postbuckling (nonlinear) analysis of a hinged cylindrical shell subjected to concentrated transverse load. Shell exhibits snap-through buckling behavior and requires a solution algorithm employing arc-length step control. Modeled with shell elements.

Table 4.0-1; concluded.

Procedure Name	*Problem Description*
PEAR_CYL	Buckling analysis of a pear-shaped cylindrical shell subjected to axial compression. Modeled with shell elements.
PINCHED_CYL	Linear inextensional bending of a thin pinched cylindrical shell. Modeled with shell elements; two opposing radial forces applied at center; and both free and rigid-diaphragm edges considered. (MacNeal-Harder case)
PW_HOLE	Linear elastic stress analysis of a rectangular isotropic plate with a central circular hole.
RECT_PLATE	Eigenvalue (vibration and buckling) analyses of a rectangular plate subjected to various inplane loading conditions.
RHOMBIC_PLATE	Linear bending of a simply supported rhombic plate under constant pressure. (MacNeal-Harder case)
TRUNCATED_CONE	Linear transient dynamic analysis of impulsively loaded truncated cone.
VIB_2D	Vibration analysis of a bar, beam, or ring modeled with 2-D shell elements.

THIS PAGE LEFT BLANK INTENTIONALLY.

4.1 Processor CLAMPED_BEAM

THIS SECTION UNDER PREPARATION

THIS PAGE LEFT BLANK INTENTIONALLY.

4.2 Procedure COMPRESSED_CYL

4.2.1 GENERAL DESCRIPTION

4.2.1.1 Problem Description

This application problem involves the static buckling and postbuckling analysis of a thin cylindrical shell ($R/t \approx 300$) simply supported along its edges, and subjected to uniform axial compression (see Figure 4.2-1). It is a particularly important classical problem for testing linear and nonlinear performance of shell elements, as well as solution algorithms capable of traversing non-monotonic load-displacement curves. A noteworthy feature of the problem is that the buckling eigenvalues of the cylindrical shell are very closely spaced, even though the mode shapes are radically different. This can pose a challenge for both eigensolvers and nonlinear solution algorithms — which are expected to find and maintain the physically dominant (lowest-energy) mode. (It is interesting that the buckling characteristics of this simple cylindrical shell problem have much in common with the more practical problem of an optimally-stiffened cylinder, which also has many diverse mode shapes occurring at the same critical load level.) Another noteworthy feature is that such problems are notoriously imperfection sensitive. Thus, a small initial geometric imperfection can cause a substantial reduction in the peak axial load capacity, compared with the classical critical buckling load corresponding to a perfect cylinder. This implies that the model must be refined enough to represent the initially imperfect configuration, as well as the finally deformed configuration.

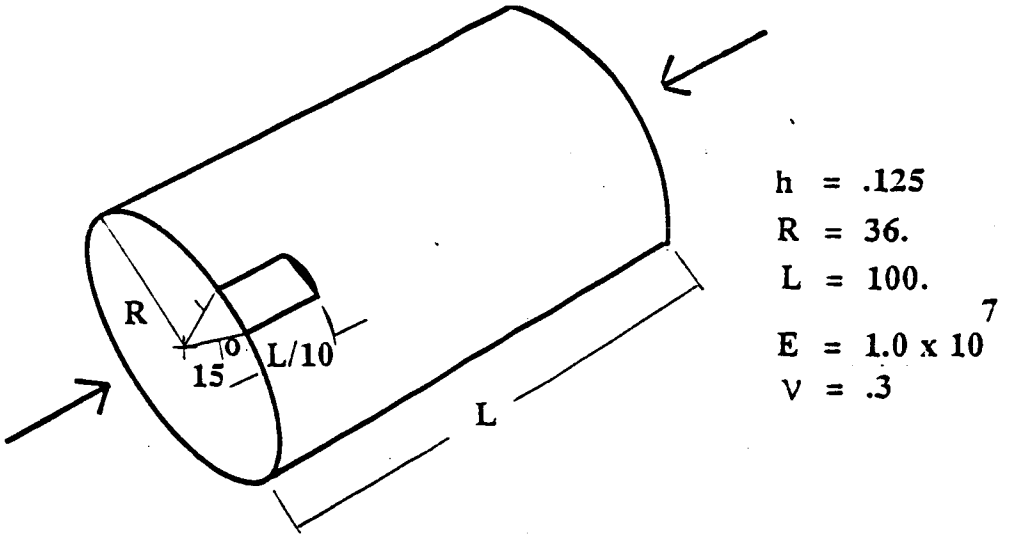


Figure 4.2-1 Compressed Cylinder Problem.

4.2.1.2 Model Description

Procedure COMPRESSED_CYL employs a partial ($L/10 \times 15°$) shell-element based model of the cylindrical shell to capture both linear and nonlinear response. A typical model is shown in Figure 4.2-2. The mesh is rectangular in topology, and restricted to quadrilateral shell elements of either 4- or 9-node variety. Symmetry conditions on all but the one edge that is simply supported enable the model to represent the lowest energy buckling mode with a relatively small number of elements. This particular ("diamond") mode shape consists of 5 axial half-waves and 12 circumferential whole waves over the entire cylinder (in classical terminology: m=5, n=12). Thus, the partial model will contain 1 axial quarter-wave and 1 axial half-wave, and less than 10 shell elements should be required in each direction to obtain engineering accuracy. The axial force is distributed evenly and consistently on the simply supported edge; and is scaled so that the magnitude corresponds to the classical buckling load.

4.2.1.3 Analysis Description

The analysis is conducted in stages, some of which are optional. First, a linear buckling (eigenvalue) analysis is performed (using procedure L_STABIL_1). Due to the selected magnitude of the applied load, the lowest eigenvalue should be close to 1 (for a sufficiently refined mesh). Next, the buckling mode corresponding to the lowest (critical) eigenvalue is scaled and used to perturb the geometry. This is achieved by adding the scaled radial displacements to the initial finite element nodal coordinates. The scale factor is selected so that the maximum radial perturbation is some percentage of the shell thickness (see procedure IMPERFECTION). This initial imperfection is necessary to trigger a realistic postbuckling response in the nonlinear analysis. Due to the extreme imperfection sensitivity of thin axially-compressed cylinders (*e.g.*, see ref. 4.2-3) as little as a 1%-thickness imperfection is adequate to trigger such a response (although a 10%-thickness imperfection is recommended). The nonlinear (postbuckling) analysis is then begun (using procedure NL_STATIC_1). A reasonable description of the load-displacement history can usually be obtained with about 20–30 load steps, from a starting load of about one-tenth of the classical buckling load.

4.2.1.4 Available Solutions

Analytical solutions are available for the buckling (eigenvalue) analysis (*e.g.*, see ref. 4.2-1). Several of the buckling modes corresponding to the classical stability solution are shown in Figure 4.2-2 (Figure 6.9 of ref. 4.2-2) using finite-element meshes. Note that the critical loads (or eigenvalues) for these diverse mode shapes – smooth, axisymmetric, and diamond-pattern – are within 5% of one another. While there are no closed-form solutions for the nonlinear postbuckling response of the axially-compressed cylindrical shell, there are approximate solutions for the "knock-down" factor corresponding to a given initial imperfection, and various numerical solutions are given in the CSM Testbed Applications Manual (see ref. 4.2-4). Figure 4.2-3 gives a sampling of these numerically obtained response curves, for different magnitudes of initial imperfection (50%, 10% and 1% of the shell thickness). Note that with only a 10%-thickness imperfection, the peak load is "knocked-down" to approximately 70% of the value for a perfect (unattainable) shell.

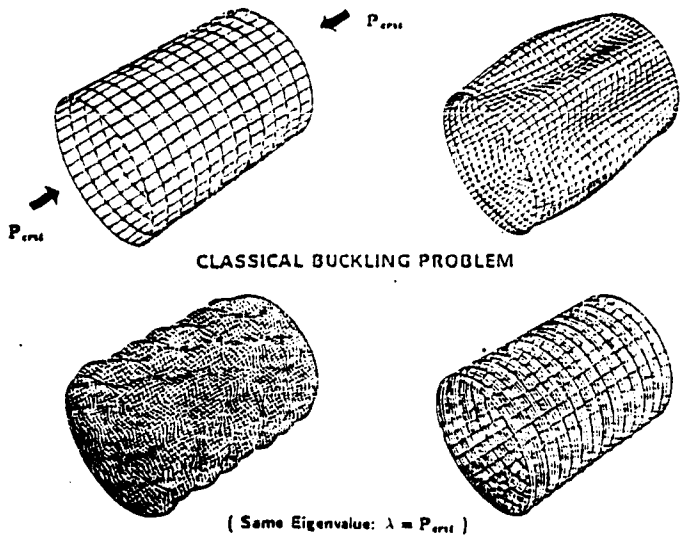


Figure 4.2-2 Buckling of Axially-Compressed Cylinder: Generic Problem Description

Figure 4.2-3 Postbuckling of Axially-Compressed Cylinder:

4.2.2 PROCEDURE USAGE

Procedure **COMPRESSED_CYL** may be used by preceding the procedure name by the ***call** directive, and following it by a list of arguments enclosed in parentheses. Procedure arguments are order-independent, and most have default values thus making them optional. The formal syntax is as follows:

```
*call COMPRESSED_CYL ( arg1 = val1 ; arg2 = val2 ; ...)
```

where **arg1** and **arg2** represent argument names, and **val1** and **val2** represent their corresponding values. Note that semi-colons are required between arguments, and a double dash (--) may be used to continue the argument list on the next line.

The allowable arguments for procedure **COMPRESS_CYL** are summarized in the following table, along with their default values (if they exist). Note that arguments without defaults are generally *mandatory*, while those with defaults are generally optional. Exceptions to this rule are noted in the following section under detailed argument descriptions.

Argument	Default Value	Meaning
ES_PROC	ES1	Element Processor
ES_NAME	EX97	Element name
ES_PARS	0.	Element research parameters
NNODES_A	5	Number of axial nodes
NNODES_C	5	Number of circumferential nodes
SPEC_DIS	<false>	Specified displacements
PRE_STRESS	<false>	Constant pre-stress
AUTO_DOF_SUP	<true>	Automatic d.o.f. suppression
DRILLING_DOF	<false>	Drilling (normal rotational) freedoms
BC_PROCEDURE	CC_BC	Boundary condition procedure
NSTEPS	30	Number of nonlinear load steps
BEG_STEP	1	Starting load step number
MAX_CUTS	3	Maximum number of step cuts (halvings)
BEG_LOAD	.1	Starting load factor
MAX_LOAD	1.0	Maximum load factor
DBC	CC.DBC	Computational database name
DBR	CC.DBR	Results database name
PREP	<true>	Perform pre-processing (model generation)
STABILITY	<true>	Perform linear stability (buckling) analysis
IMPERFECTION	<true>	Superpose imperfections from (buckling) analysis before performing nonlinear(postbuckling) analysis
NUM_IMP_MODES	1	Number of buckling modes used in creating imperfection
IMP_MODES	1	Mode numbers used in creating imperfection
IMP_AMPS	.1	Mode amplitudes
NONLINEAR	<true>	Perform nonlinear (postbuckling) analysis
POST	<false>	Perform post-processing (selected data archival)

4.2.3 ARGUMENT DESCRIPTIONS

4.2.3.1 AUTO_DOF_SUP

Automatic degree of freedom suppression flag (default: <true>). This option provides a convenient way of suppressing any freedoms that do not have any (or adequate) stiffness associated with them — for example, at nodes used to prescribe geometry only; or drilling freedoms in fine meshes composed of elements without normal rotational stiffnesses (see argument DRILLING_DOF).

4.2.3.2 BC_PROCEDURE

Boundary condition procedure name (default: CC_BC for specified forces; CCD_BC for specified displacements). The term "boundary conditions" refers both to displacement constraints and applied loading. Procedures CC_BC and CCD_BC both have the same zero displacement constraints. The only difference is that the former procedure applies axial forces to the simply supported edge, while the latter procedure prescribes non-zero axial displacements on that edge. The argument BC_PROCEDURE permits you to to supply your own boundary condition procedure, but keep in mind that this may drastically change the problem definition, and hence invalidate most of the discussion under Section 4.2.1.

4.2.3.3 BEG_LOAD

Starting load factor in nonlinear analysis (default=.1). This factor is multiplied times the reference load vector to obtain the starting load vector. For example, if specified forces are applied (which is the default option), then BEG_LOAD = .1 means that the first load step to be computed in the nonlinear analysis will be at one-tenth of the classical buckling load level. Note that this argument is *irrelevant* for re-start runs. For more details, refer to the same argument name under solution procedure NL_STATIC_1.

4.2.3.4 BEG_STEP

Number of starting load step in nonlinear analysis (default=1). This is the number of the first step to be computed during an analysis interval. When starting a nonlinear analysis, the first step is obviously 1. When re-starting (*i.e.*, continuing in a subsequent run) a nonlinear analysis, BEG_STEP should be set to the number of the next step to be computed — not to the number of the last step computed. The solution procedure (NL_STATIC_1) will automatically use any previous step(s) required to continue the analysis — providing the necessary steps are available in the database. Currently, the number of consecutive preceding steps required for a restart is three. For more details, refer to the same argument name under solution procedure NL_STATIC_1.

4.2.3.5 DBC

Name of computational database file (default=CC.DBC). This file will contain all model definition data, element computational data, one copy of the assembled and factored stiffness matrices, the buckling eigensolution, and displacement and internal force vectors for every load step computed during the analysis.

4.2.3.6 DBR

Name of results database file (default=CC.DBR). This file will contain one dataset called RESPONSE.HISTORY generated during nonlinear analysis. The dataset will contain record groups — indexed by load step number — for a number of solution parameters, including the load factor and maximum axial displacement components. This database is valuable for obtaining load-displacement plots, and for evaluating the performance of the nonlinear solution strategy employed.

4.2.3.7 DRILLING_DOF

Drilling degree of freedom flag (default: <false>). Drilling freedoms are defined as rotations normal to the surface of the shell. Leaving this flag off forces all drilling freedoms in the model to be suppressed. Turning it on forces all drilling freedoms to be active — unless they are automatically suppressed using use of the AUTO_DOF_SUP argument. Note that while many shell elements do not have any rotational stiffness associated with their own surface-normal directions (at nodes), when shell elements are assembled as facets approximating an arbitrary shell surface, there is usually some misalignment between the element normal and the actual shell normal. This is especially true of "flat" (*e.g.*, 4-node) elements. Hence, *some* rotational stiffness about the *shell* normal is usually present in the model. (A clear exception to this is a flat plate, where element and shell normals are identical.) For a cylindrical shell, the misalignment diminishes only as the number of elements is increased. Most shell elements in the Testbed have their own misalignment tolerance parameter, which determines when the AUTO_DOF_SUP argument will automatically suppress the drilling freedom. Note that for elements which *have* drilling stiffness, the DRILLING_DOF argument should be set to <true> regardless of how AUTO_DOF_SUP is set.

4.2.3.8 ES_NAME

Element name (default: EX97). This argument is the name of the specific shell-element type you wish to select, within the element processor defined by argument ES_PROC. The default shell-element type, EX97, is a 9-noded quadrilateral element implemented in Processor ES1, and described in The Computational Structural Mechanics Testbed User's Manual (see ref. 4.2-1).

4.2.3.9 ES_PARS

Element research parameters (default: 0., ...). This array is an optional list of element-dependent parameters that some elements provide, primarily when the element is still undergoing research and refinement.

4.2.3.10 ES_PROC

Element Processor (default: ES1) This is the name of the structural element (ES) processor that contains the shell element type you wish to employ in the model. The default shell-element, Processor ES1, is described in The Computational Structural Mechanics Testbed User's Manual.

4.2.3.11 IMPERFECTION

Imperfection superposition flag (default <true>). This flag should be turned on if you want geometric imperfections, corresponding to buckling modes, to be superimposed on the initial geometry. Pre-requisites for this option are pre-processing and stability analysis. If you are just starting or re-starting a nonlinear analysis in the current run, and imperfections have already been superposed in a previous run, then you should turn this flag off.

4.2.3.12 IMP_AMPS

List of buckling mode amplitudes to be used in generating the imperfection (default=.1, 10% of shell thickness). This option is relevant only if IMPERFECTION=<true>. The amplitudes in this list are interpreted as the maximum radial displacement to add to the initial nodal coordinates for a given buckling mode. Thus, the maximum coordinate perturbation due to a single buckling mode will occur wherever the mode shape has its maximum radial component.

4.2.3.13 IMP_MODES

List of buckling mode numbers to be used in generating the imperfection (default=1). This option is relevant only if IMPERFECTION=<true>. The numbers in this list should be separated by commas, and there should be a total of NUM_IMP_MODES numbers.

4.2.3.14 MAX_CUTS

Maximum number of load step cuts in nonlinear analysis (default=3). For more details, refer to the same argument name under solution procedure NL_STATIC_1.

4.2.3.15 MAX_LOAD

Maximum load factor in nonlinear analysis (default=1.0). This sets an upper limit for the load level, which can be a convenient way of terminating the arc-length controlled solution algorithm employed within procedure NL_STATIC_1. Since the load factor is actually an unknown in this solution procedure, there is no way of knowing a-priori how many load steps will be required to attain a particular load level. Thus, the analysis will be terminated when either MAX_LOAD is exceeded or NSTEPS is exceeded — whichever comes first. For more details, refer to the same argument name under solution procedure NL_STATIC_1.

4.2.3.16 NNODES_A

Number of axial nodes (default: 7). This is the number of nodes you wish to have along the axial direction of the cylindrical shell model, *i.e.*, along one-tenth of the full cylinder's length. Note that this number should be consistent with the number of nodes per element. For example, NNODES_A can be any number greater than 1 for 4-node quadrilateral elements, whereas it must be an odd number greater than 1 for 9-node quadrilateral elements.

4.2.3.17 NNODES_C

Number of circumferential nodes (default: 7). This argument is the number of nodes you wish to have along the circumferential direction of the cylindrical shell model, *i.e.*, along 15 degrees of circular arclength. Note that this number should be consistent with the number of nodes per element. For example, NNODES_C can be any number greater than 1 for 4-node quadrilateral elements, whereas it must be an odd number greater than 1 for 9-node quadrilateral elements.

4.2.3.18 NONLINEAR

Nonlinear (postbuckling) analysis flag (default=<true>). This flag should be turned on if you want to perform nonlinear analysis in the current run. The pre-requisites are pre-processing, stability analysis and imperfection superposition, all of which may be performed either in a previous run or in the current run — by setting the appropriate arguments (*i.e.*, PREP, STABILITY and IMPERFECTION).

4.2.3.19 NSTEPS

Maximum number of load steps to be computed in the current nonlinear analysis run (default=30). For more details, refer to the same argument name under solution procedure NL_STATIC_1.

4.2.3.20 NUM_IMP_MODES

Number of buckling modes used to represent geometric imperfections for nonlinear analysis (default=1). This option is relevant only if IMPERFECTION=<true>.

4.2.3.21 POST

Postprocessing flag (default=<false>). This flag should be turned on if you want selected response-history parameters to be added to the CC.DBR database. Note that it is not necessary to use this option in order to archive the basic load-displacement curve and solution parameters. It is only needed if you wish to archive special displacement and/or internal force component response histories post-facto.

4.2.3.22 PREP

Preprocessing flag (default=<true>). This flag must be turned on the first time procedure COMPRESSED_CYL is run, as it causes the model to be generated. If subsequent runs are used to perform other stages of the analysis (*e.g.*, nonlinear re-starts), then PREP must be set to <false> for those subsequent runs.

4.2.3.23 PRE_STRESS

Constant pre-stress flag (default: <false>). By setting this flag to <true>, the procedure bypasses computing the linear solution to obtain the prebuckling stresses, and instead prescribes the prebuckling axial stress resultants to be uniform throughout the shell. This

amounts to procedure COMPRESSED_CYL invoking solution procedure L_STABIL_1 instead of procedure L_STABIL_2 to perform the buckling eigenvalue analysis. While the prescribed pre-stress used with this option is equal to the classical buckling value, the results can be somewhat different than those obtained using L_STABIL_2. The reason for this is that due to the simply supported boundary condition on the loaded edge, the prebuckling stress distribution (*i.e.*, both the exact one and that obtained using a linear prebuckling analysis) is not really uniform. Note that even when the uniform PRE_STRESS option is selected, it is used only to obtain the buckling eigenvalues and mode shapes — which are in turn used as initial imperfections for the nonlinear analysis. The uniform pre-stress values are then ignored during the nonlinear analysis, which applies either forces or displacements to the simply-supported edge (depending on argument SPEC_DIS).

4.2.3.24 SPEC_DIS

Specified displacement flag (default: <false>). By setting this flag to <true>, uniform axial end-shortening is imposed instead of the uniform axial loading. This can make a significant difference in both the buckling and postbuckling response, and is not recommended for novice users of this procedure. This is because thin axially-compressed shells are not only imperfection sensitive, but also *boundary condition* sensitive, and uniform axial loading does not correspond (exactly) to uniform axial edge displacements. Note that the reference specified displacement (*i.e.*, end-shortening) magnitude equals .01 inches, and corresponds to an axial load of about .467 times the classical buckling load.

4.2.3.25 STABILITY

Stability (buckling) analysis flag (default: <true>). This flag should be turned on if you want the buckling eigenvalue analysis to be performed in the current run. Preprocessing is a pre-requisite for this option. If you are just performing a nonlinear analysis re-start run, then you should turn this flag off.

4.2.4 USAGE GUIDELINES AND EXAMPLES

Procedure COMPRESS_CYL may be used by preceding the procedure name by the *call directive. Procedure arguments may be changed from their default values by including any or all of the arguments and their new values when the procedure is called. A space or blank is required between the end of the procedure name and the left parenthesis.

```
*call COMPRESS_CYL ( ELT_PROC   = ES1        ; ELT_NAME = EX97 ; --
                     NNODES_A   =  7         ; NNODES_C =  7   ; --
                     SPEC_DIS   =    <false> ; --
                     DRILLING_DOF = <false>  ; --
                     AUTO_DOF_SUP = <true>   ; --
                     PREP       =    <true>  ; --
                     STABILITY  =    <true>  ; --
                     IMPERFECTION = <true>   ; --
                     NONLINEAR  =     <true> ; --
                     POST       =    <true>  ; --
```

```
BEG_STEP  =  1        ; --
NSTEPS    = 10        ; --
BEG_LOAD  =   .1      ; --
MAX_LOAD  =  3.0      ; --
DBc       = CC.DBc    ; DBr    = CC.DBr )
```

(E1) Not suppressing the drilling rotational freedoms can cause strange behavior for some elements during nonlinear analysis. On the other hand, suppressing these freedoms explicitly using the DRILLING_DOF argument may cause some over-stiffening for coarse meshes with some elements. It is probably best to suppress the drilling freedoms explicitly unless the element actually has intrinsic drilling stiffness.

Table 4.2-1 Typical CPU Times for Various Computer Systems

Computer System	Total CPU Time, seconds.
VAX 11/785 VMS 4.7	
MicroVAX ULTRIX 2.2	
SUN SUNVIEW 4.0	
CONVEX C220 VERSION 7.0	
CRAY-2 UNICOS 4.0	

(E2) To perform an entire analysis using the default options, simply invoke the procedure without any arguments, *i.e.*,

```
*call COMPRESSED_CYL
```

This will perform linear buckling eigenvalue analysis and 30 steps of nonlinear analysis with a 7×7 grid of ES1/EX97 shell elements. The time required for this analysis is machine-dependent. Using the default values for the procedure arguments, the amount of CPU time required for this analysis on various computer systems is shown in Table 4.2-1.

4.2.5 LIMITATIONS

(L1) Only 4-node and 9-node shell elements can be employed in the model. (Note: This can be easily modified.)

(L2) Not all of the solution parameters provided by procedure **NL_STATIC_1** are accessible using this procedure. The main reason for this is the CLIP limitation of 25 arguments per procedure. Procedure **COMPRESSED_CYL** is exactly at this limit now.

4.2.6 ERROR MESSAGES AND WARNINGS

If you have to repeat the preprocessing phase, you need to either delete the computational (**.DBC**) database, or delete the **ES.SUMMARY** dataset. Otherwise, the element processors will be run more than once for each element function (*e.g.*, **FORM STIFFNESS**), and the effect will be cumulative. This is due to a lack of explicit initialization in the registration of element processors using procedure **ES**.

4.2.7 PROCEDURE FLOWCHART

COMPRESSED_CYL	(main procedure)
GEN_SHELL	(generate shell model)
CC_BC	(generate boundary conditions/loads)
L_STABIL_2	(perform stability analysis)
IMPERFECTION	(superpose buckling modes as imperfections)
NL_STATIC_1	(perform nonlinear static analysis)
HISTORY	(archive selected displacement/force histories)

4.2.8 PROCEDURE LISTING

4.2.9 REFERENCES

4.2-1 Timoshenko, S. P.; and Gere, J. M.: *Theory of Elastic Stability*, McGraw-Hill, New York, 1961.

4.2-2 Stanley, G. M.: "Continuum-Based Shell Elements," PhD Dissertation, Stanford University, 1985.

4.2-3 Donnel, L. H.: *Beams, Plates, and Shells*, McGraw-Hill, New York, 1976.

4.2-4 CSM Testbed Applications Manual.

THIS PAGE LEFT BLANK INTENTIONALLY.

4.3 Procedure COOK_MEM

4.3.1 GENERAL DESCRIPTION

This section describes a procedure that solves for the deflection of a trapezoidal panel subject to inplane shear and bending, also known as Cook's membrane (see ref. 4.3-1). The geometry of the membrane resembles a wing plan form (see figure 4.3-1) with an applied end shear load. The membrane has constant thickness, and the material is isotropic.

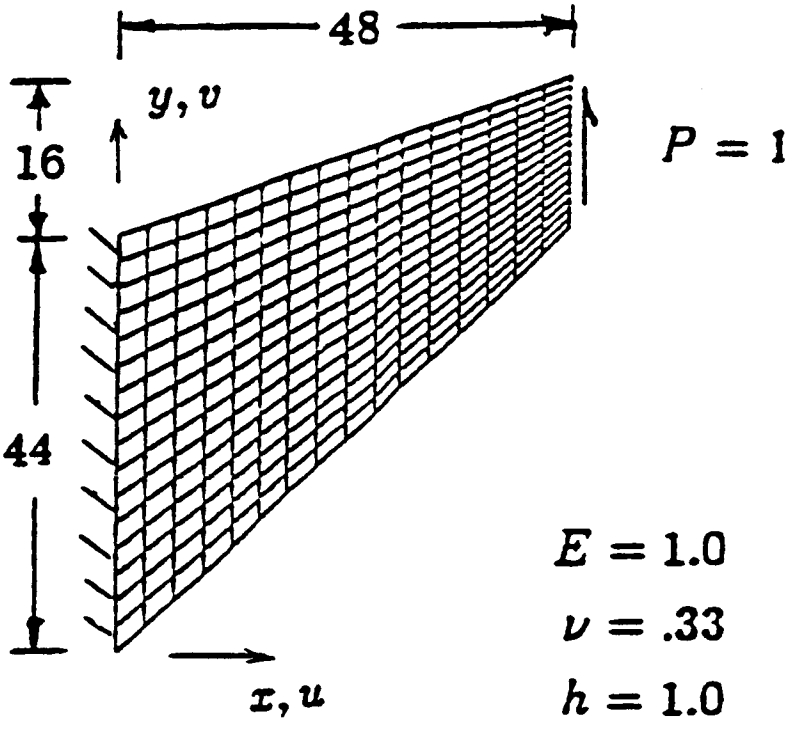


Figure 4.3-1 Cook's Membrane

4.3.1.1 Model Description

4.3.1.2 Analysis Description

4.3.1.3 Available Solutions

4.3.2 PROCEDURE USAGE

Procedure **COOK_MEM** may be used by preceding the procedure name by the ***call** directive, and following it by a list of arguments enclosed in parentheses. Procedure arguments are order-independent, and all have default values thus making them optional. The formal syntax is as follows:

```
*call COOK_MEM ( arg1 = val1 ; arg2 = val2 ; ...)
```

where **arg1** and **arg2** represent argument names, and **val1** and **val2** represent their corresponding values. Note that semi-colons are required between arguments, and a double dash (--) may be used to continue the argument list on the next line.

The allowable arguments for procedure **COOK_MEM** are summarized in the following table, along with their default values (if they exist). Note that arguments without defaults are generally *mandatory*, while those with defaults are generally optional. Exceptions to this rule are noted in the following section under detailed argument descriptions.

Argument	Default Value	Meaning
ES_PROC	**ES1**	Element Processor
ES_NAME	**EX97**	Element name
ES_PARS	**0.**	Element research parameters
NDATA	**4**	Number of grids to analyze
NNODES	**3,5,7,9**	Number of nodes on each edge
AUTO_DOF_SUP	**<true>**	Automatic d.o.f. suppression
DRILLING_DOF	**<false>**	Drilling (normal rotational) freedoms
CM_DB	**CM.DB**	Cook membrane database name

4.3.3 ARGUMENT DESCRIPTIONS

4.3.3.1 AUTO_DOF_SUP

Automatic degree of freedom suppression flag (default: **<true>**). This option provides a convenient way of suppressing any freedoms that do not have any (or adequate) stiffness associated with them — for example, at nodes used to prescribe geometry only; or drilling freedoms in fine meshes composed of elements without normal rotational stiffnesses (see argument **DRILLING_DOF**).

4.3.3.2 CM_DB

Name of Cook membrane database file (default=CM.DB). This file will contain all model data and analysis results.

4.3.3.3 DRILLING_DOF

Drilling degree of freedom flag (default: <false>). Drilling freedoms are defined as rotations normal to the surface of the shell. Leaving this flag off forces all drilling freedoms in the model to be suppressed. Turning it on forces all drilling freedoms to be active — unless they are automatically suppressed using use of the AUTO_DOF_SUP argument. Note that while many shell elements do not have any rotational stiffness associated with their own surface-normal directions (at nodes), when shell elements are assembled as facets approximating an arbitrary shell surface, there is usually some misalignment between the element normal and the actual shell normal. This is especially true of "flat" (*e.g.*, 4-node) elements. Hence, *some* rotational stiffness about the *shell* normal is usually present in the model. (A clear exception to this is a flat plate, where element and shell normals are identical.) For a cylindrical shell, the misalignment diminishes only as the number of elements is increased. Most shell elements in the Testbed have their own misalignment tolerance parameter, which determines when the AUTO_DOF_SUP argument will automatically suppress the drilling freedom. Note that for elements which *have* drilling stiffness, the DRILLING_DOF argument should be set to <true> regardless of how AUTO_DOF_SUP is set.

4.3.3.4 ES_NAME

Element name (default: EX97). This is the name of the specific shell-element type you wish to select, within the element processor defined by argument ES_PROC. The default shell-element type, EX97, is a 9-node quadrilateral element implemented in Processor ES1, and described in The Computational Structural Mechanics Testbed User's Manual (see ref. 4.3-2).

4.3.3.5 ES_PARS

Element research parameters (default: 0., ...). This is an optional list of element-dependent parameters that some elements provide, primarily when the element is still undergoing research and refinement.

4.3.3.6 ES_PROC

Element processor (default: ES1) This is the name of the structural element (ES) Processor that contains the shell element type you wish to employ in the model. The default shell-element, Processor ES1, is described in The Computational Structural Mechanics Testbed User's Manual.

4.3.3.7 NDATA

Specifies the number of model/mesh refinements to analyze (default: 4). The degree of mesh refinement for each analysis is defined by the NNODES parameter.

4.3.3.8 NNODES

A list of integers which represent the number of nodes on each edge of the surface for each analysis to be performed (default: 3,5,9,17). The length of the list is defined by the **NDATA** parameter. The numbers in the list must be consistent with the element type selected. For example, **NNODES** can be any list of numbers greater than 1 for 4-node quadrilateral elements, whereas it must be a list of odd numbers greater than 1 for 9-node quadrilateral elements.

4.3.4 USAGE GUIDELINES AND EXAMPLES

Procedure Procedure **COOK_MEM** may be used by preceding the procedure name by the *call directive. Procedure arguments may be changed from their default values by including any or all of the arguments and their new values when the procedure is called. A space or blank is required between the end of the procedure name and the left parenthesis. If the default values of the procedure arguments are to be used, then only the procedure name is required.

```
*call COOK_MEM  ( ES_PROC = ES1   ; --
                  ES_NAME = EX97  ; --
                  ES_PARS =  0.   ; --
                  NDATA =  4      ; --
                  NNODES = 3,5,9,17     ; --
                  AUTO_DOF_SUP = <true>   ; --
                  DRILLING_DOF = <false> ; --
                  CM_DB = CM.DB   --
                )
```

(E1) To perform an entire analysis using the default options, simply invoke the procedure without any arguments, *i.e.*,

```
*call COOK_MEM
```

This call will perform a linear static analysis for each of the element configurations specified by the default **NNODES** parameter. Also, a fine grid analysis is performed with approximately four times the nodes of the last analysis specified by parameter **NNODES**. The results of the fine grid analysis are used to normalize the results of each requested analysis. All analysis results are presented in table format by the procedure:

```
=======================================
      Cook's Membrane Analysis Data
          Element Name = EX97
=======================================
NODES    DISPLACEMENT       NORMALIZED
 3     0.22677967E+02    0.9440006730146
 5     0.23594924E+02    0.982170233148691
 9     0.23889961E+02    0.994451542428496
 17    0.23989798E+02    0.998607390930779
```

==========

A plot showing the deformed shape of the membrane is provided in figure 4.3-2. The analysis model used 9-node quadrilateral elements with 17 nodes per edge.

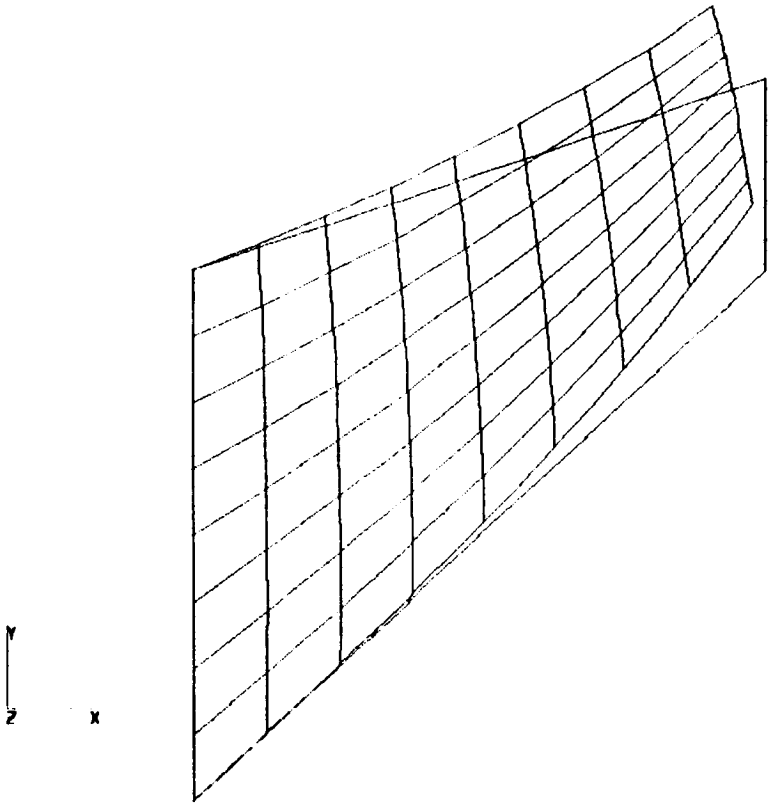


Figure 4.3-2 Cooks Membrane Deflection Plot

The time required for this analysis is machine-dependent. Using the default values for the procedure arguments, the amount of CPU time required for this analysis on various computer systems is shown in Table 4.3-1.

Table 4.3-1 CPU TIMES Table

Computer System	Total CPU Time, seconds.
VAX 11/785 VMS 4.7	
MicroVAX ULTRIX 2.2	
SUN SUNVIEW 4.0	
CONVEX C220 VERSION 7.0	
CRAY-2 UNICOS 4.0	

4.3.5 LIMITATIONS

None.

4.3.6 ERROR MESSAGES AND WARNINGS

None.

4.3.7 PROCEDURE FLOWCHART

```
COOK_MEM          (main procedure)
    GEN_PLATE     (generate model)
        CM_BC     (generate boundary conditions/loads)
    L_STATIC      (perform linear static analysis)
```

4.3.8 PROCEDURE LISTING

4.3.8.1 UNIX Script

cook_mem.com

```
cd $SCR/$USER
cp $CSM_PRC/proclib.gal proclib.gal
chmod u+w proclib.gal
rm CMEX97.*
time testbed <<\endinput
*set echo,off
*set plib=28
*open 28 proclib.gal /old
*add '$CSM_APP/cook_mem/cook_mem.clp'
*call COOK_MEM  ( ES_PROC = ES1  ; -- . Element Processor
                  ES_NAME = EX97 ; -- . Element name
                  ES_PARS =  0.  ; -- . Element parameters
                  NDATA =  4     ; -- . Number of grids to analyze
                  NNODES = 3,5,9,17  ; -- . Number of nodes per side
                  AUTO_DOF_SUP=<true>  ; --
                  DRILLING_DOF=<false> ; --
                  CM_DB = CMEX97.L01  -- . Database
                )
*stop
*close 28 /delete
[xqt exit
\endinput
```

4.3.8.2 CLAMP Procedure

cook_mem.clp

```
*procedure COOK_MEM  ( es_proc = es1  ; -- . element processor
                       es_name = ex97 ; -- . element name
                       es_pars =  0.  ; -- . element parameters
                       ndata =  4  ; -- . Number of grids to analyze
                       nnodes = 3,5,9,17  ; -- . Number of nodes per side
                       auto_dof_sup=<true>  ; --
                       drilling_dof=<false> ; --
                       cm_db = cm.db  -- . database
                     )
.
. ================================================================
. procedure for analysis of in-plane bending of trapezoidal membrane
. ================================================================
.
*def/i ndata = [ndata]
*def/i nnodes[1:<ndata>] = [nnodes]
*def/i nnodes[<<ndata>+1>] = < (<nnodes[<ndata>]>*2) + 1>
*def/i lib = 1
*do $j=1,<<ndata>+1>
   *if < <$j> /lt <<ndata>+1> > /then
      *open <lib>, [cm_db] /new
```

```
   *else
      *open <lib>, zzzscr.lib /new
   *endif
   *def/i nnt = < <nnodes[<$j>]>*<nnodes[<$j>]> >
   *def/e e = 1.0
   *def/e nu = .33
   *def/e h = 1.0
   *call GEN_PLATE ( es_proc      = [es_proc] ; --
                     es_name      = [es_name] ; --
                     es_pars      = [es_pars] ; --
                     nodes_1      = <nnodes[<$j>]> ; --
                     nodes_2      = <nnodes[<$j>]> ; --
                     xyz1         = 48.,44., 0. ; --
                     xyz2         = 48.,60., 0. ; --
                     xyz3         =  0.,44., 0. ; --
                     xyz4         =  0., 0., 0. ; --
                     e = <e> ; nu = <nu> ; thickness  = <h> ; --
                     auto_dof_sup  = [auto_dof_sup]; --
                     drilling_dof  = [drilling_dof]; --
                     bc_procedure  = TP_BC )
.
.         perform static analysis
.
   *def/a solver_name == BAND
   *call L_STATIC
.
.         store the results for this run
.
   [xqt AUS
      MACRO <lib> STAT DISP 1 1 semilocal 2 <6*<nnodes[<$j>]>> 6 'disps
      STOP
.
   *def/e avgdisp = 0.
   *do $jj=1,<nnodes[<$j>]>
   *def/e avgdisp = <<avgdisp> + <disps[<$jj>]> >
   *enddo
   *def/e avgd[<$j>] = < <avgdisp> / <nnodes[<$j>]> >
   *show macros
*enddo
.
.         delete the normalizing analysis
.
*close <lib> /delete
.
.         done
.
*remark
*remark
*remark
*remark
*remark ========================================
*remark       Cook's Membrane Analysis Data
*remark           Element Name = [es_name]
```

```
*remark ========================================
*remark NODES    DISPLACEMENT          NORMALIZED
*remark
*do $j=1,<ndata>
*remark  <nnodes[<$j>]>    <avgd[<$j>]>    <<avgd[<$j>]>/<avgd[<<ndata>+1>]>>
*enddo
*remark ========================================
*remark
*remark
*remark
*remark
.
[xqt EXIT
*end
*procedure TP_BC ( nodes_1 ; nodes_2 ; es_nodes ; drilling_dof=<false> )
.
. ================================================================
. boundary condition procedure for trapezoidal plate analysis
. ================================================================
.
  *def/i nn1 = [nodes_1]
  *def/i nn2 = [nodes_2]
  *def/i nen = [es_nodes]
  *def/i node_A = 1
  *def/i node_B = <nn1>
  *def/i node_D = < <nn1>*<nn2> >
  *def/i node_C = < <node_D> - <nn1> + 1 >
.
. boundary conditions
.   edge c-d gets clamped
.
      constraint = 1
         zero  1 2 3 4 5 6          .   clamp edge c-d
         <node_C>  <node_D>         .       of constant theta
.
        *if < [drilling_dof] /eq <false> > /then
         zero  4                    .   remove drilling dof
         <node_A>  <node_D>         .       everywhere
        *endif
.
. loading conditions
.   edge a-b receives uniform shear in positive global y direction
.
[xqt aus
  alpha
  case title
  1 'in-plane shear (i.e., bending) of trapezoidal membrane'
  sysvec : appl forc 1 1
.
*def/e p = 1.0      . total axial force on model boundary
*if < <nen> /eq 4 > /then
   *def/i nelts = < <nn1> - 1 >
   *def/e12.4 Pelt  = < <p> / <nelts> >
```

```
   *def/e12.4 Pend = < <Pelt>/2. >
   *do $jn = <node_A>, <node_B>
      *if < <<$jn> /eq <node_A>> /or <<$jn> /eq <node_B>> > /then
       i=2 : j=<$jn> : <Pend>
      *else
       i=2 : j=<$jn> : <Pelt>
      *endif
   *enddo
*elseif < <nen> /eq 9 > /then
   *def/i nelts = < (<nn1> - 1)/2 >
   *def/e12.4 Pelt  = < <p> / <nelts> >
   *def/e12.4 Pelt_1 = < 1.*<Pelt>/6. >
   *def/e12.4 Pelt_2 = < 2.*<Pelt>/6. >
   *def/e12.4 Pelt_4 = < 4.*<Pelt>/6. >
.         midside nodes
   *do $j4 = <<node_A> + 1>, <<node_B>-1>, 2
       i=2 : j= <$j4> : <Pelt_4>
   *enddo
.         shared nodes
   *if < <nelts> /gt 1 > /then
      *do $j2 = <<node_A>+2>, <<node_B>-2>, 2
       i=2 : j= <$j2> : <Pelt_2>
      *enddo
   *endif
.         corner nodes
       i=2 : j= <node_A> : <Pelt_1>
       i=2 : j= <node_B> : <Pelt_1>
*endif
*end
```

4.3.9 REFERENCES

4.3-1 Stanley, G. M.: *Continuum-Based Shell Elements.* PhD Dissertation, Stanford University, Stanford, CA, 1985.

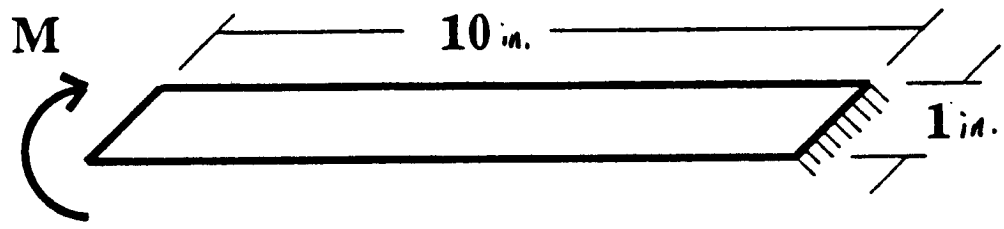
4.3-2 Stewart, Caroline B.: *The Computational Structural Mechanics Testbed User's Manual.* NASA TM-100644, October 1989.

4.4 Procedure ELASTICA

4.4.1 GENERAL DESCRIPTION

4.4.1.1 Problem Description

This section describes a procedure that solves the classical large deflection, large rotation problem of a cantilevered beam with an applied end moment as shown in Figure 4.4-1. The nonlinear response of the cantilevered beam bent into a circle by an applied end moment is determined.



Thickness = 1. in.

M = 2 π in.-lb/in.

E = 120. psi

ν = 0.

Figure 4.4-1 Elastica Problem.

4.4.1.2 Model Description

Procedure **ELASTICA** models the entire strip using 2-D quadrilateral elements. The mesh is generated using procedure **GEN_PLATE**. Two sets of boundary conditions are available. The set in procedure **ELASTICA_BC** imposes a clamped condition at one end of the strip and a moment at the other end. Procedure **EL_ECC_BC** also imposes a clamped condition at one end and an eccentric axial load at the other.

4.4.1.3 Analysis Description

Procedure **ELASTICA** performs either a linear static analysis using procedure **L_STATIC** or a nonlinear static analysis using procedure **NL_STATIC_1**.

4.4.1.4 Available Solutions

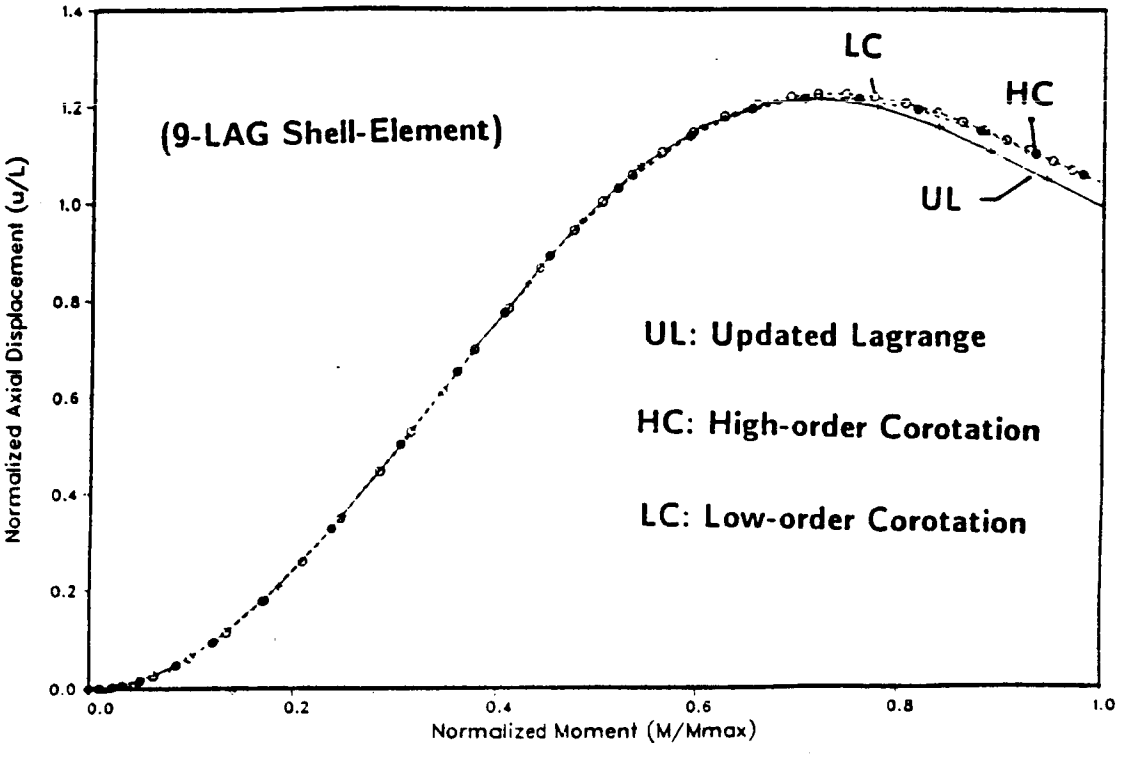


Figure 4.4-2 Nonlinear Response of Elastica Problems.

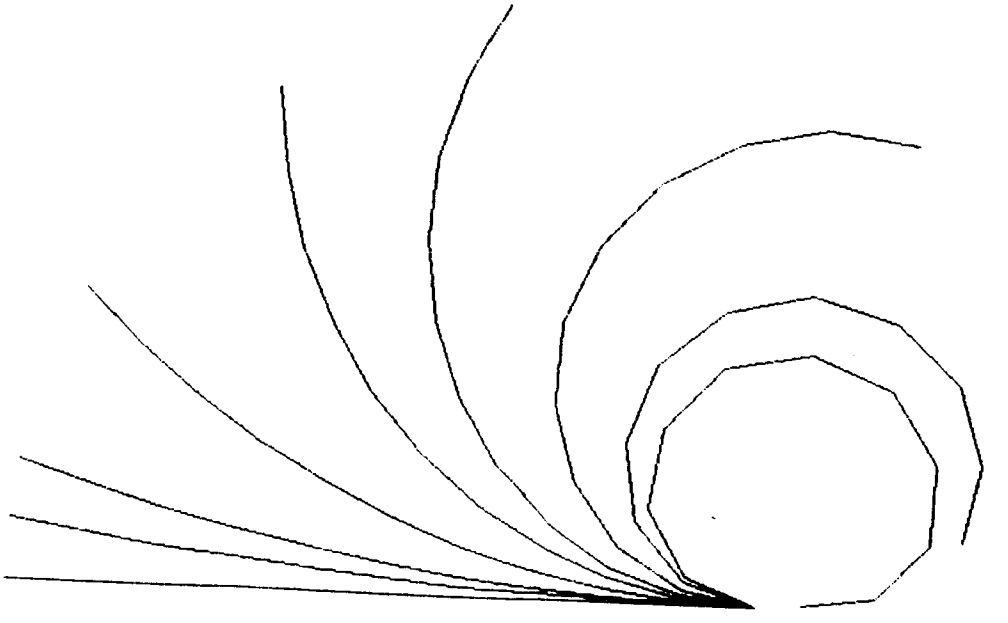


Figure 4.4-3 Sequence of Deformed Shapes.

4.4.2 PROCEDURE USAGE

Procedure **ELASTICA** may be used by preceding the procedure name by the ***call** directive, and following it by a list of arguments enclosed in parentheses. Procedure arguments are order-independent, and most have default values thus making them optional. The formal syntax is as follows:

```
*call ELASTICA ( arg1 = val1 ; arg2 = val2 ; ...)
```

where **arg1** and **arg2** represent argument names, and **val1** and **val2** represent their corresponding values. Note that semi-colons are required between arguments, and a double dash (--) may be used to continue the argument list on the next line.

The allowable arguments for procedure **ELASTICA** are summarized in the following table, along with their default values (if they exist). Note that arguments without defaults are generally *mandatory*, while those with defaults are generally optional. Exceptions to this rule are noted in the following section under detailed argument descriptions.

For the procedure **ELASTICA**, the following table lists each argument, its default value and meaning.

Argument	Default Value	Meaning
ES_PROC	ES1	Select element processor
ES_NAME	EX97	Select element within ES_PROC
ES_PARS	0.	Set element-research parameters
NNODES_X	11	Number of nodes in x-direction
NNODES_Y	3	Number of nodes in y-direction
PREP	<true>	Preprocessing flag
POST	<true>	Postprocessing flag
LINEAR	<true>	Perform Linear Analysis
NONLINEAR	<true>	Perform Nonlinear Analysis
AUTO_DOF_SUP	<false>	Automatic degree of freedom suppression
DRILLING_DOF	<true>	
BEG_STEP	1	Beginning Load Step Number
BEG_LOAD	.01	Beginning Load Factor
NSTEPS	10	Number of Load Steps
DBC	ELASTICA.DBC	Computational Database
DBR	ELASTICA.DBR	Results Database

4.4.3 ARGUMENT DESCRIPTIONS

4.4.3.1 AUTO_DOF_SUP

Automatic degree of freedom suppression flag (default: <true>). This option provides a convenient way of suppressing any freedoms that do not have any (or adequate) stiffness

associated with them — for example, at nodes used to prescribe geometry only; or drilling freedoms in fine meshes composed of elements without normal rotational stiffnesses (see argument DRILLING_DOF).

4.4.3.2 BC_PROCEDURE

Boundary condition procedure name (default: ELASTICA_BC). Two sets of boundary conditions are available. The set in procedure ELASTICA_BC imposes a clamped condition at one end of the strip and a moment at the other end. Procedure EL_ECC_BC also imposes a clamped condition at one end and an eccentric axial load at the other.

4.4.3.3 BEG_LOAD

Starting load factor in nonlinear analysis (default=.1). This factor is multiplied times the reference load vector to obtain the starting load vector. For example, if specified forces are applied (which is the default option), then BEG_LOAD = .1 means that the first load step to be computed in the nonlinear analysis will be at one-tenth of the classical buckling load level. Note that this argument is *irrelevant* for re-start runs. For more details, refer to the same argument name under solution procedure NL_STATIC_1.

4.4.3.4 BEG_STEP

Number of starting load step in nonlinear analysis (default=1). This argument is the number of the first step to be computed during an analysis interval. When starting a nonlinear analysis, the first step is obviously 1. When re-starting (*i.e.*, continuing in a subsequent run) a nonlinear analysis, BEG_STEP should be set to the number of the next step to be computed — not to the number of the last step computed. The solution procedure (NL_STATIC_1) will automatically use any previous step(s) required to continue the analysis — providing the necessary steps are available in the database. Currently, the number of consecutive preceding steps required for a restart is three. For more details, refer to the same argument name under solution procedure NL_STATIC_1.

4.4.3.5 DBC

Name of computational database file (default=ELASTICA.DBC). This file will contain all model definition data, element computational data, one copy of the assembled and factored stiffness matrices, the buckling eigensolution, and displacement and internal force vectors for every load step computed during the analysis.

4.4.3.6 DBR

Name of results database file (default=ELASTICA.DBR). This file will contain one dataset called RESPONSE.HISTORY generated during nonlinear analysis. The dataset will contain record groups — indexed by load step number — for a number of solution parameters, including the load factor and maximum axial displacement components. This database is valuable for obtaining load-displacement plots, and for evaluating the performance of the nonlinear solution strategy employed.

4.4.3.7 DRILLING_DOF

Drilling degree of freedom flag (default: <false>). Drilling freedoms are defined as rotations normal to the surface of the shell. Leaving this flag off forces all drilling freedoms in the model to be suppressed. Turning it on forces all drilling freedoms to be active — unless they are automatically suppressed using use of the AUTO_DOF_SUP argument. Note that while many shell elements do not have any rotational stiffness associated with their own surface-normal directions (at nodes), when shell elements are assembled as facets approximating an arbitrary shell surface, there is usually some misalignment between the element normal and the actual shell normal. This is especially true of "flat" (*e.g.*, 4-node) elements. Hence, *some* rotational stiffness about the *shell* normal is usually present in the model. (A clear exception to this is a flat plate, where element and shell normals are identical.) For a cylindrical shell, the misalignment diminishes only as the number of elements is increased. Most shell elements in the Testbed have their own misalignment tolerance parameter, which determines when the AUTO_DOF_SUP argument will automatically suppress the drilling freedom. Note that for elements which *have* drilling stiffness, the DRILLING_DOF argument should be set to <true> regardless of how AUTO_DOF_SUP is set.

4.4.3.8 ES_NAME

Element name (default: EX97). This argument specifies the name of the specific shell-element type you wish to select, within the element processor defined by argument ES_PROC. The default shell-element type, EX97, is a 9-noded quadrilateral element implemented in processor ES1, and described in The Computational Structural Mechanics Testbed User's Manual (see ref. 4.4-1).

4.4.3.9 ES_PARS

Element research parameters (default: 0., ...). This array allows an optional list of element-dependent parameters that some elements provide, primarily when the element is still undergoing research and refinement.

4.4.3.10 ES_PROC

Element processor (default: ES1) This argument specifies the name of the structural element (ES) processor that contains the shell element type you wish to employ in the model. The default shell-element, processor ES1, is described in The Computational Structural Mechanics Testbed User's Manual (see ref. 4.4-1).

4.4.3.11 NNODES_X

Number of nodes along beam length (default: 11). This argument is the number of nodes along the x-direction of the beam shell model. Note that this number should be consistent with the number of nodes per element. For example, NNODES_X can be any number greater than 1 for 4-node quadrilateral elements, whereas it must be an odd number greater than 1 for 9-node quadrilateral elements.

4.4.3.12 NNODES_Y

Number of nodes along beam depth (default: 3). This argument is the number of nodes along the y-direction of the beam shell model. Note that this number should be consistent with the number of nodes per element. For example, **NNODES_Y** can be any number greater than 1 for 4-node quadrilateral elements, whereas it must be an odd number greater than 1 for 9-node quadrilateral elements.

4.4.3.13 NONLINEAR

Nonlinear (postbuckling) analysis flag (default=<true>). This flag should be turned on to perform nonlinear analysis in the current run.

4.4.3.14 NSTEPS

Maximum number of load steps to be computed in the current nonlinear analysis run (default=30). For more details, refer to the same argument name under solution procedure **NL_STATIC_1**.

4.4.3.15 POST

Postprocessing flag (default=<false>). This flag should be turned on if you want selected response-history parameters to be added to the **ELASTICA.DBR** database. Note that it is not necessary to use this option in order to archive the basic load-displacement curve and solution parameters. It is only needed to archive special displacement and/or internal force component response histories post-facto.

4.4.3.16 PREP

Preprocessing flag (default=<true>). This flag must be turned on the first time procedure **ELASTICA** is run, as it causes the model to be generated. If subsequent runs are used to perform other stages of the analysis (*e.g.*, nonlinear restarts), then **PREP** must be set to <false> for those subsequent runs.

4.4.4 USAGE GUIDELINES AND EXAMPLES

Procedure **ELASTICA** may be used by preceding the procedure name by the ***call** directive. Procedure arguments may be changed from their default values by including any or all of the arguments and their new values when the procedure is called. A space or blank is required between the end of the procedure name and the left parenthesis.

```
*call ELASTICA ( ES_PROC      = ES1  ; --
                 ES_NAME      = EX97 ; --
                 ES_PARS      = 0.0  ; --
                 NNODES_X      = 11   ; --
                 NNODES_Y      =  3   ; --
                 PREP          =<true>; --
                 LINEAR        =<true>; --
```

```
POST          =<false>; --
NONLINEAR     =<false>; --
NSTEPS        = 10    ; --
DBc = ELASTICA.DBc    ; -- . Computational database
DBr = ELASTICA.DBr      -- . Results database)
```

(E1) To perform an entire analysis using the default options, simply invoke the procedure without any arguments, *i.e.*,

```
*call ELASTICA
```

This call will perform linear buckling eigenvalue analysis and 10 steps of nonlinear analysis with a 11×3 grid of ES1/EX97 shell elements. The time required for this analysis is machine-dependent. Using the default values for the procedure arguments, the amount of CPU time required for this analysis on various computer systems is shown in Table 4.4-1.

Table 4.4-1 CPU TIMES Table

Computer System	Total CPU Time, seconds.
VAX 11/785 VMS 4.7	
MicroVAX ULTRIX 2.2	
SUN SUNVIEW 4.0	
CONVEX C220 VERSION 7.0	
CRAY-2 UNICOS 4.0	

(E2) Not suppressing the drilling rotational freedoms can cause strange behavior for some elements during nonlinear analysis. On the other hand, suppressing these freedoms explicitly using the DRILLING_DOF argument may cause some over-stiffening for coarse meshes with some elements. It is probably best to suppress the drilling freedoms explicitly unless the element actually has intrinsic drilling stiffness.

4.4.5 LIMITATIONS

(L1) Only 4-node and 9-node shell elements can be employed in the model. (Note: This can be easily modified.)

4.4.6 ERROR MESSAGES AND WARNINGS

If you have to repeat the preprocessing phase, you need to either delete the computational (.DBC) database, or delete the ES.SUMMARY dataset. Otherwise, the element processors will be run more than once for each element function (*e.g.*, FORM STIFFNESS), and the effect will be cumulative. This is due to a lack of explicit initialization in the registration of element processors using procedure ES.

4.4.7 PROCEDURE FLOWCHART

ELASTICA	(main procedure)
GEN_PLATE	(generate model)
ELASTICA_BC	(generate boundary conditions and end moment)
EL_ECC_BC	(generate boundary conditions and eccentric axial load)
L_STATIC	(perform linear static analysis)
IMPERFECTION	(superpose buckling modes as imperfections)
NL_STATIC_1	(perform nonlinear static analysis)
HISTORY	(archive selected displacement/force histories)

4.4.8 PROCEDURE LISTING

4.4.8.1 UNIX Script

elastica.com

```
cd $SCR/$USER
cp $CSM_PRC/proclib.gal proclib.gal
chmod u+w proclib.gal
rm ELASTICA.*
time testbed << \endinput
 *set echo,off
 *set plib=28
 *open/old 28 proclib.gal
 *add '$CSM_APP/elastica/elastica.clp'
 *def/a solver_name == INV
 *call ELASTICA ( ES_PROC      = ES1  ; --
                  ES_NAME      = EX97 ; --
                  ES_PARS      = 0.0  ; --
                  NNODES_X      = 11   ; --
                  NNODES_Y      =  3   ; --
                  LINEAR=<false>; NONLINEAR     =<true>; --
                  NSTEPS        = 36;drilling_dof=<false>   ; --
                  DBc = ELASTICA.DBc   ; -- . Computational database
                  DBr = ELASTICA.DBr     -- . Results database
                )
 *stop
\endinput
```

4.4.8.2 CLAMP Procedure

elastica.clp

```
*procedure ELASTICA ( ES_PROC      = ES1  ; -- .
                      ES_NAME      = EX97 ; -- .
                      ES_PARS      =  0.  ; --
                      NNODES_X      =  11   ; -- .
                      NNODES_Y      =  3    ; -- .
                      PREP          =<true>; -- .
                      POST          =<true>; -- .
                      LINEAR        =<true>; -- .
                      NONLINEAR     =<true>; -- .
                      AUTO_DOF_SUP  =<false>; -- .
                      DRILLING_DOF  =<true>; -- .
                      BEG_STEP      =  1    ; -- .
                      BEG_LOAD      =   .01; -- .
                      NSTEPS        = 10    ; -- . Number of load steps
                      DBc = ELASTICA.DBc    ; -- . Computational database
                      DBr = ELASTICA.DBr      -- . Results database
                    )
.
. ------------------------------------------------------
. CSM Testbed Procedure for Analysis of ELASTICA Problem
. ------------------------------------------------------
.
*open 1, [DBc]
.
. Generate Plate Model of Cantilever Beam
.
*if < [PREP] > /then
   *call GEN_PLATE ( ES_PROC=[ES_PROC]      ; ES_NAME=[ES_NAME] ;--
                     NODES_1      =[NNODES_X] ; NODES_2 =[NNODES_Y] ;--
                     E = 120.       ; NU = 0. ; THICKNESS   = 1.        ; --
                     xyz1=0.,0.,0.; xyz2=10.,0.,0.; --
                     xyz3=10.,1.,0.; xyz4=0.,1.,0.; --
                     AUTO_DOF_SUP =[auto_dof_sup] ; --
                     DRILLING_DOF =[drilling_dof] ; --
                     BC_PROCEDURE = ELASTICA_BC )
   *toc
*endif
.
. Perform Solution
.
*if < [LINEAR] > /then
   *call L_STATIC
   *if < [POST] > /then
      *open 2, [DBr]
      *def/a ds_r    = [ES_PROC].[ES_NAME].[NNODES_X].[NNODES_Y]
      *call HISTORY ( input_ldi =1 ; input_ds =STAT.DISP.1.1; --
                      output_ldi=2 ; output_ds=<ds_r>       ; --
                      output_rn =LATERAL_DISP; nodes=1; component=3 )
      *call HISTORY ( input_ldi =1 ; input_ds =STAT.DISP.1.1; --
                      output_ldi=2 ; output_ds=<ds_r>       ; --
```

```
                    output_rn =END_ROTATION; nodes=1; component=5 )
      *call HISTORY ( input_ldi =1 ; input_ds =STAT.DISP.1.1; --
                    output_ldi=2 ; output_ds=<ds_r>       ; --
                    output_rn =AXIAL_DISP; nodes=1; component=1 )
      *remark
      *remark
      *remark ------------------------------------------------------
      *remark
      *remark ELASTICA Linear Results
      *remark
      *remark Exact:
      *remark      LATERAL_DISP (compare with W at node 1)  = 31.41
      *remark      END_ROTATION (compare with RV at node 1) =  6.28
      *remark      AXIAL_DISP   (compare with U at node 1)  =  0.0
      *remark
      *remark Computed:
      *print rec 2, <ds_r> LATERAL_DISP /1=drx
      *print rec 2, <ds_r> END_ROTATION /1=drx
      *print rec 2, <ds_r> AXIAL_DISP   /1=drx
      *remark
      *remark ------------------------------------------------------
      *remark
   *endif
 *endif
 *if < [NONLINEAR] > /then
   *def ns_overwrite = <true>
   *call NL_STATIC_1  ( beg_step   = [BEG_STEP]  ; --
                        max_steps  = [NSTEPS]    ; --
                        beg_load   = [BEG_LOAD]  ; --
                        max_load   =1.00         ; --
                        NL_GEOM    =   2         ; --
                        COROTATION =   1         ; --
                        Nominal_DB = [DBr]       ; --
       Nominal_DS = [ES_PROC].[ES_NAME].[NNODES_X].[NNODES_Y] ;--
                        N_SELECT   = 2           ; --
                        SEL_NODES  = 1, 1 ; SEL_DOFS = 1, 3 )
 *toc [DBR]
 *endif
 [xqt exit
 *end
*procedure EL_ECC_BC ( nodes_1; nodes_2; es_nodes; drilling_dof=<false> )
.
. Define Loads and Boundary Conditions for ELASTICA Model
.
*def/i nn_tot = < [nodes_1]*[nodes_2] >
.
. ==========================
. DEFINE BOUNDARY CONDITIONS
. ==========================
.
[xqt TAB
 CON 1
*if < [drilling_dof] /eq <false> > /then
```

```
   ZERO 6                          . Suppress NORMAL ROTATIONS Everywhere
  *do $n=1,<nn_tot>
     <$n>
  *enddo
*endif
 ZERO 2                            . Suppress LATERAL MOTION Everywhere
*do $n=1,<nn_tot>
   <$n>
*enddo
 ZERO 4                            . Suppress TORSIONAL ROTATION Everywhere
*do $n=1,<nn_tot>
   <$n>
*enddo
 ZERO 1,2,3,4,5,6                  . CANTILEVER Right End of Beam
*do $n = [nodes_1], <nn_tot>, [nodes_1]
   <$n>
*enddo
.
. ============
. Define Loads
. ============
.
[xqt AUS
.
. Apply Uniform Eccentric Load Consistently  (over unit width)
.    Px results in  Moment about Y of 2*PI since Eccentricity = .5
.
*def/e12.4 Px   = < -4.*<PI> >
*def/e12.4 Px_1 = < <Px>/2. >
  SYSVEC : APPL FORC 1
*if < [ES_NODES] /eq 4 > /then
   i=1 : j=1             : <Px_1>
   i=1 : j=<[nodes_1]+1>    : <Px_1>
*elseif < [ES_NODES] /eq 9 > /then
  *def/e12.4 Px_1 = <   <Px>/6. >
  *def/e12.4 Px_4 = < 4*<Px>/6. >
   i=1 : j=1              : <Px_1>
   i=1 : j=<[nodes_1]+1>     : <Px_4>
   i=1 : j=<(2*[nodes_1])+1> : <Px_1>
*endif
*end
*procedure ELASTICA_BC ( nodes_1; nodes_2; es_nodes; drilling_dof=<false> )
.
. Define Loads and Boundary Conditions for ELASTICA Model
.
*def/i nn_tot = < [nodes_1]*[nodes_2] >
.
. ==========================
. DEFINE BOUNDARY CONDITIONS
. ==========================
.
[xqt TAB
 CON 1
```

```
*if < [drilling_dof] /eq <false> ) /then
    ZERO 6                          . Suppress NORMAL ROTATIONS Everywhere
   *do $n=1,<nn_tot>
      <$n>
   *enddo
*endif
 ZERO 2                             . Suppress LATERAL MOTION Everywhere
*do $n=1,<nn_tot>
   <$n>
*enddo
 ZERO 4                             . Suppress TORSIONAL ROTATION Everywhere
*do $n=1,<nn_tot>
   <$n>
*enddo
 ZERO 1,2,3,4,5,6                   . CANTILEVER Right End of Beam
*do $n = [nodes_1], <nn_tot>, [nodes_1]
   <$n>
*enddo
.
. ============
. Define Loads
. ============
.
[xqt AUS
.
. Apply Uniform M_y Consistently  (over unit width)
.
*def/e12.4 My   = < 2.*<PI> >
*def/e12.4 My_1 = < <My>/2. >
  SYSVEC : APPL FORC 1
*if < [ES_NODES] /eq 4 > /then
   i=5 : j=1             : <My_1>
   i=5 : j=<[nodes_1]+1>    : <My_1>
*elseif < [ES_NODES] /eq 9 > /then
  *def/e12.4 My_1 = <   <My>/6. >
  *def/e12.4 My_4 = < 4*<My>/6. >
   i=5 : j=1              : <My_1>
   i=5 : j=<[nodes_1]+1>     : <My_4>
   i=5 : j=<(2*[nodes_1])+1> : <My_1>
*endif
*end
```

4.4.9 REFERENCES

4.4-1 Timoshenko, S. P.; and Gere, J. M.: *Theory of Elastic Stability*, McGraw-Hill, New York, 1961.

4.4-2 Stanley, G. M.: "Continuum-Based Shell Elements," PhD Dissertation, Stanford University, 1985.

4.4-3 Donnel, L. H.: *Beams, Plates, and Shells*, McGraw-Hill, New York, 1976.

4.4-4 Stewart, Caroline B.: *The Computational Structural Mechanics Testbed User's Manual.* NASA TM-100644, October 1989.

4.5 Procedure EULER_COLUMN

4.5.1 GENERAL DESCRIPTION

4.5.1.1 Problem Description

This application problem involves a linear bifurcation buckling analysis of a thin isotropic strip subjected to uniform compression (see Figure 4.5-1). This problem is important in that plate and shell elements are frequently used to model stiffeners of stiffened panels. When these panels buckle in an "overall" Euler modeshape the stiffeners must also buckle in their plane (see Figure 4.5-2).

Figure 4.5-1 Euler Column Problem

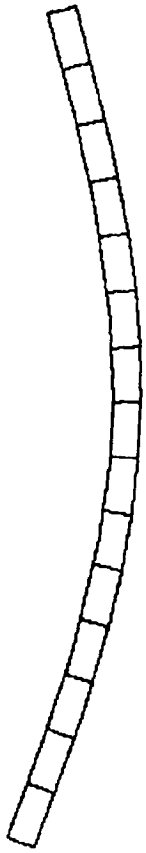


Figure 4.5-2 Euler Buckling Modeshape

4.5.1.2 Model Description

Procedure **EULER_COLUMN** models the entire strip using 2-D quadrilateral elements. The mesh is generated using procedure **GEN_PLATE** and the boundary conditions are specified in procedure **COLUMN_BC**.

4.5.1.3 Analysis Description

Procedure **EULER_COLUMN** performs a single linear buckling analysis with a specified prestress. The solution procedure **L_STABIL_1** is used to perform the buckling analysis.

4.5.1.4 Available Solutions

Analytical solution for the Euler buckling load is readily obtained from reference 4.5-1, page 22-28. The Euler buckling load is given by

$$P_{EULER} = \frac{\pi^2 EI}{L^2}$$

where

E = Young's Modulus

I = Moment of Inertia = $\frac{1}{12}th^3$

L = Column Length

t = Column Thickness

h = Column Width

For a specified uniform membrane stress resultant N_x^o of -1000 lb/in., the corresponding buckling load factor is

$$(N_x)_{EULER} = P_{EULER}/h = \lambda_{EULER} N_x^o$$

For the geometry and material properties used as default values of the procedure arguments, the exact solution for simple-support boundary conditions corresponds to a value of 0.456926 for λ_{EULER} (*i.e.*, the smallest eigenvalue).

4.5.2 PROCEDURE USAGE

Procedure **EULER_COLUMN** may be used by preceding the procedure name by the ***call** directive, and following it by a list of arguments enclosed in parentheses. Procedure arguments are order-independent, and most have default values thus making them optional. The formal syntax is as follows:

```
*call EULER_COLUMN ( arg1 = val1 ; arg2 = val2 ; ...)
```

where **arg1** and **arg2** represent argument names, and **val1** and **val2** represent their corresponding values. Note that semi-colons are required between arguments, and a double dash (--) may be used to continue the argument list on the next line.

The allowable arguments for procedure **EULER_COLUMN** are summarized in the following table, along with their default values (if they exist). Note that arguments without defaults are generally *mandatory*, while those with defaults are generally optional. Exceptions to this rule are noted in the following section under detailed argument descriptions.

Argument	Default Value	Meaning
E	10.E6	Young's Modulus
ES_PROC	ES1	Element Processor
ES_NAME	EX47	Element name
LX	30.0	Length in x-direction
LY	1.0	Length in y-direction
NODES_1	31	Number of nodes along x
NODES_2	3	Number of nodes along y
NU	0.0	Poisson's ratio
PRINT	<false>	Print flag
PS_1	-1000.0	Uniform axial compression prestress
THICKNESS	0.05	Thickness

4.5.3 ARGUMENT DESCRIPTIONS

4.5.3.1 E

Young's elastic modulus (default: 10.0 $\times$ 10^6 psi).

4.5.3.2 ES_NAME

Element name (default: **EX47**). This is the name of the specific shell-element type you wish to select, within the element processor defined by argument **ES_PROC**. The default shell-element type, **EX47**, is a 4-noded quadrilateral element implemented in processor **ES1**, and described in the Computational Structural Mechanics Testbed User's Manual (see ref. 4.5-2).

4.5.3.3 ES_PROC

Element Processor (default: **ES1**) This is the name of the structural element (ES) processor that contains the shell element type you wish to employ in the model. The default shell-element, processor **ES1**, is described in the Computational Structural Mechanics Testbed User's Manual (see refs. 4.5-2).

4.5.3.4 LX

Length of the plate model in the $x-$direction (default: 31.0 inches).

4.5.3.5 LY

Length of the plate model in the $y-$direction (default: 1.0 inches).

4.5.3.6 NODES_1

Number of nodes along x-direction (default: 31).

4.5.3.7 NODES_2

Number of nodes along y-direction (default: 3).

4.5.3.8 NU

Poisson's ratio (default: 0.0).

4.5.3.9 PRINT

Print flag (default: <false>). If the argument PRINT is defined to be <true>, then all computed results (displacements, modeshapes, stresses) will be printed.

4.5.3.10 PS_1

Uniform axial compression prestress (default: -1000 lb/in.).

4.5.3.11 THICKNESS

Thickness of the plate (default: 0.05 inches).

4.5.3.12 WTDEN

Weight density (default: 0.1 lb/in.^{3}). Processor LAU converts the weight density to mass density.

4.5.4 USAGE GUIDELINES AND EXAMPLES

Procedure EULER_COLUMN may be used by preceding the procedure name by the *call directive. Procedure arguments may be changed from their default values by including any or all of the arguments and their new values when the procedure is called. A space or blank is required between the end of the procedure name and the left parenthesis.

```
*call EULER_COLUMN ( ES_PROC  = ES1        ; ES_NAME = EX47 ; --
                     LX = 7.5 ; --
                     LY = 10.0 ; --
                     E = 30.E6 ; --
                     NU = 0.3 ; --
                     PRINT = <false> ; --
                     THICKNESS = 0.1 ; --
                     WTDEN = 0.1 )
```

(E1) To perform an entire analysis using the default options, simply invoke the procedure without any arguments, that is,

```
*call EULER_COLUMN
```

Using the default values for the procedure arguments, the amount of CPU time required for this analysis on various computer systems is shown in Table 4.5-1.

Table 4.5-1 CPU TIMES Table

Computer System	Total CPU Time, seconds.
VAX 11/785 VMS 4.7	
MicroVAX ULTRIX 2.2	
SUN SUNVIEW 4.0	
CONVEX C220 VERSION 7.0	
CRAY-2 UNICOS 4.0	

4.5.5 LIMITATIONS

None

4.5.6 ERROR MESSAGES AND WARNINGS

None.

4.5.7 PROCEDURE FLOWCHART

```
EULER_COLUMN        (main procedure)
    GEN_PLATE       (define model)
        COLUMN_BC   (define simple-support boundary conditions)
```

4.5.8 PROCEDURE LISTING

4.5.8.1 UNIX Script

euler_column.com

```
cd $SCR/$USER
rm euler_column.101
cp $CSM_PRC/proclib.gal proclib.gal
chmod u+w proclib.gal
time testbed  << \endinput
```

```
*open 1 euler_column.101
*set echo off
*def/a eigensolver_name = EIG2
*def/a solver_name = INV
.
*set plib=28
*open 28 proclib.gal /old
*add '$CSM_APP/euler_column/euler_column.clp'
.
.   Inplane Buckling of a Thin Strip
.
*call EULER_COLUMN ( es_name='ex97'; es_proc='es7')
.
[XQT EXIT
\endinput
```

4.5.8.2 CLAMP Procedure

euler_column.clp

```
*procedure EULER_COLUMN (   es_name = 'EX47' ; --
                            es_proc = 'ES1'  ; --
                            e       = 10.0e+6; --
                            nu      = .3     ; --
                            thickness= 0.05  ; --
                            ps_1     = -1000.0; --
                            lx       =30.0 ; --
                            ly       =1.0 ; --
                            nodes_1  = 31  ; --
                            nodes_2  = 3   )
.
*call GEN_PLATE ( es_proc=[es_proc];es_name=[es_name]; --
                  xyz1=0.0,0.0,0.0 ; --
                  xyz2=[lx],0.0,0.0 ; --
                  xyz3=[lx],[ly],0.0 ; --
                  xyz4=0.0,[ly],0.0 ;  --
                  nodes_1=[nodes_1]; nodes_2=[nodes_2]; --
                  nsect=1; e=[e];nu=[nu];thickness=[thickness]; --
                  bc_procedure= 'COLUMN_BC'  )
*toc 1
[XQT TAB
TITLE'EULER COLUMN BUCKLING PROBLEM
.
*call L_STABIL_1 ( ps_1=[ps_1]; print=<true> )
*toc 1
*end
*procedure COLUMN_BC (nodes_1=31; nodes_2=3; --
      es_nodes=<es_nen>; drilling_dof=<true>)
*def/i n1 = 1
*def/i n2 = [nodes_1]
*def/i n3 = < [nodes_1]*[nodes_2] >
*def/i n4 = < [nodes_1]*([nodes_2]-1) + 1 >
CON CASE 1
```

```
ZERO 3 4 5 : <n1>,<n3>
ZERO 2 : <n1>,<n4>,[nodes_1]
ZERO 2 : <n2>,<n3>,[nodes_1]
*if < [drilling_dof] /eq 0 > /then
ZERO 6: <n1>,<n3>
*endif
*end
```

4.5.9 REFERENCES

4.5-1 Brush, Don O. and Almroth, Bo O.: *Buckling of Bars, Plates, and Shells*, McGraw-Hill Book Company, New York, 1975.

4.5-2 Stewart, Caroline B.: *The Computational Structural Mechanics Testbed User's Manual.* NASA TM-100644, October 1989.

4.6 Processor FOCUS_PANEL

THIS SECTION UNDER PREPARATION

THIS PAGE LEFT BLANK INTENTIONALLY.

4.7 Processor FREE_EDGE

THIS SECTION UNDER PREPARATION

THIS PAGE LEFT BLANK INTENTIONALLY.

4.8 Processor GEN_STF_PANEL

THIS SECTION UNDER PREPARATION

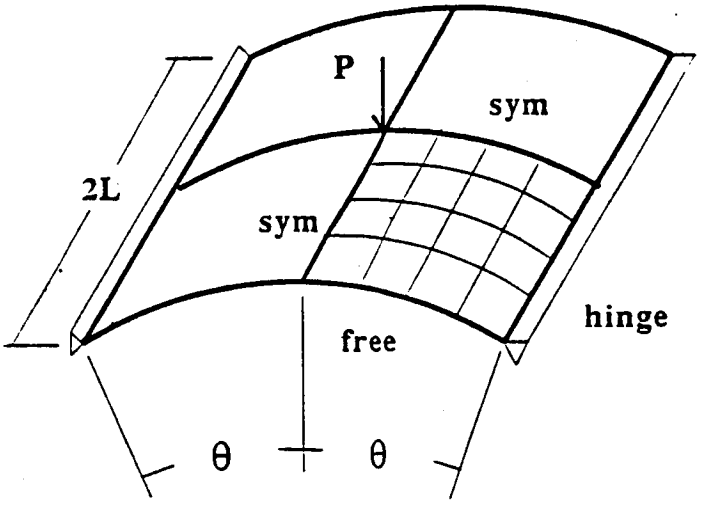
THIS PAGE LEFT BLANK INTENTIONALLY.

4.9 Procedure HINGED_CYL

4.9.1 GENERAL DESCRIPTION

To verify the nonlinear aspects of the shell-element formulation, implementation, and adaptive (Crisfield/Riks) quasi-static solution algorithm, the snap-through of a hinged cylinder under a point force is considered. This problem, which involves only moderate geometric nonlinearities, is nevertheless interesting due to the limit-point characteristics of the load-displacements curve. Furthermore, due to the popularity of the problem, an abundance of numerical results are available for comparison.

This section describes a prcedure that solves the hinged cylinder problem (see figure 4.9-1) which exhibits snap-through behavior.



$R = 2540$ mm
$L = 254$ mm
$h = 12.7$ mm
$\theta = .1$ rad
$E = 3102.75$ N/mm^2
$\nu = .3$

Figure 4.9-1 Hinged Cylinder Problem.

4.9.1.1 Model Description

4.9.1.2 Analysis Description

4.9.1.3 Available Solutions

4.9.2 PROCEDURE USAGE

Procedure **HINGED_CYL** may be used by preceding the procedure name by the ***call** directive, and following it by a list of arguments enclosed in parentheses. Procedure arguments are order-independent, and most have default values thus making them optional. The formal syntax is as follows:

```
*call HINGED_CYL ( arg1 = val1 ; arg2 = val2 ; ...)
```

where **arg1** and **arg2** represent argument names, and **val1** and **val2** represent their corresponding values. Note that semi-colons are required between arguments, and a double dash (--) may be used to continue the argument list on the next line.

The allowable arguments for procedure **HINGED_CYL** are summarized in the following table, along with their default values (if they exist). Note that arguments without defaults are generally *mandatory*, while those with defaults are generally optional. Exceptions to this rule are noted in the following section under detailed argument descriptions.

Argument	Default Value	Meaning
ES_PROC	ES1	Element Processor
ES_NAME	EX97	Element name
ES_PARS	0.	Element research parameters
NNODES_A	5	Number of axial nodes
NNODES_C	5	Number of circumferential nodes
SPEC_DIS	<false>	Specified displacements
AUTO_DOF_SUP	<true>	Automatic d.o.f. suppression
DRILLING_DOF	<false>	Drilling (normal rotational) freedoms
NSTEPS	10	Number of nonlinear load steps
BEG_STEP	1	Starting load step number
MAX_CUTS	3	Maximum number of step cuts (halvings)
BEG_LOAD	.1	Starting load factor
MAX_LOAD	1.0	Maximum load factor
DBC	CC.DBC	Computational database name
DBR	CC.DBR	Results database name
PREP	<true>	Perform pre-processing (model generation)
LINEAR	<false>	Perform nonlinear (post-buckling) analysis
TH_SCALE	1.0	
TOL	.001	

4.9.3 ARGUMENT DESCRIPTIONS

4.9.3.1 AUTO_DOF_SUP

Automatic degree of freedom suppression flag (default: <true>). This option provides a convenient way of suppressing any freedoms that do not have any (or adequate) stiffness associated with them — for example, at nodes used to prescribe geometry only; or drilling freedoms in fine meshes composed of elements without normal rotational stiffnesses (see argument DRILLING_DOF).

4.9.3.2 BEG_LOAD

Starting load factor in nonlinear analysis (default=.1). This factor is multiplied times the reference load vector to obtain the starting load vector. For example, if specified forces are applied (which is the default option), then BEG_LOAD = .1 means that the first load step to be computed in the nonlinear analysis will be at one-tenth of the classical buckling load level. Note that this argument is *irrelevant* for re-start runs. For more details, refer to the same argument name under solution procedure NL_STATIC_1.

4.9.3.3 BEG_STEP

Number of starting load step in nonlinear analysis (default=1). This is the number of the first step to be computed during an analysis interval. When starting a nonlinear analysis, the first step is obviously 1. When re-starting (*i.e.*, continuing in a subsequent run) a nonlinear analysis, BEG_STEP should be set to the number of the next step to be computed — not to the number of the last step computed. The solution procedure (NL_STATIC_1) will automatically use any previous step(s) required to continue the analysis — providing the necessary steps are available in the database. Currently, the number of consecutive preceding steps required for a restart is three. For more details, refer to the same argument name under solution procedure NL_STATIC_1.

4.9.3.4 DBC

Name of computational database file (default=HC.DBC). This file will contain all model definition data, element computational data, one copy of the assembled and factored stiffness matrices, the buckling eigensolution, and displacement and internal force vectors for every load step computed during the analysis.

4.9.3.5 DBR

Name of results database file (default=HC.DBR). This file will contain one dataset called RESPONSE.HISTORY generated during nonlinear analysis. The dataset will contain record groups — indexed by load step number — for a number of solution parameters, including the load factor and maximum axial displacement components. This database is valuable for obtaining load-displacement plots, and for evaluating the performance of the nonlinear solution strategy employed.

4.9.3.6 DRILLING_DOF

Drilling degree of freedom flag (default: <false>). Drilling freedoms are defined as rotations normal to the surface of the shell. Leaving this flag off forces all drilling freedoms in the model to be suppressed. Turning it on forces all drilling freedoms to be active — unless they are automatically suppressed using use of the AUTO_DOF_SUP argument. Note that while many shell elements do not have any rotational stiffness associated with their own surface-normal directions (at nodes), when shell elements are assembled as facets approximating an arbitrary shell surface, there is usually some misalignment between the element normal and the actual shell normal. This is especially true of "flat" (*e.g.*, 4-node) elements. Hence, *some* rotational stiffness about the *shell* normal is usually present in the model. (A clear exception to this is a flat plate, where element and shell normals are identical.) For a cylindrical shell, the misalignment diminishes only as the number of elements is increased. Most shell elements in the Testbed have their own misalignment tolerance parameter, which determines when the AUTO_DOF_SUP argument will automatically suppress the drilling freedom. Note that for elements which *have* drilling stiffness, the DRILLING_DOF argument should be set to <true> regardless of how AUTO_DOF_SUP is set.

4.9.3.7 ES_NAME

Element name (default: EX97). This is the name of the specific shell-element type you wish to select, within the element processor defined by argument ES_PROC. The default shell-element type, EX97, is a 9-noded quadrilateral element implemented in processor ES1, and described in The Computational Structural Mechanics Testbed User's Manual (see ref. 4.9-1).

4.9.3.8 ES_PARS

Element research parameters (default: 0., ...). This is an optional list of element-dependent parameters that some elements provide, primarily when the element is still undergoing research and refinement.

4.9.3.9 ES_PROC

Element processor (default: ES1) This is the name of the structural element (ES) Processor that contains the shell element type you wish to employ in the model. The default shell-element, processor ES1, is described in The Computational Structural Mechanics Testbed User's Manual.

4.9.3.10 MAX_CUTS

Maximum number of load step cuts in nonlinear analysis (default=3). For more details, refer to the same argument name under solution procedure NL_STATIC_1.

4.9.3.11 MAX_LOAD

Maximum load factor in nonlinear analysis (default=1.0). This sets an upper limit for the load level, which can be a convenient way of terminating the arc-length controlled solution

algorithm employed within procedure **NL_STATIC_1**. Since the load factor is actually an unknown in this solution procedure, there is no way of knowing a-priori how many load steps will be required to attain a particular load level. Thus, the analysis will be terminated when either **MAX_LOAD** is exceeded or **NSTEPS** is exceeded — whichever comes first. For more details, refer to the same argument name under solution procedure **NL_STATIC_1**.

4.9.3.12 NNODES_A

Number of axial nodes (default: 7). This is the number of nodes you wish to have along the axial direction of the cylindrical shell model, *i.e.*, along one-tenth of the full cylinder's length. Note that this number should be consistent with the number of nodes per element. For example, **NNODES_A** can be any number greater than 1 for 4-node quadrilateral elements, whereas it must be an odd number greater than 1 for 9-node quadrilateral elements.

4.9.3.13 NNODES_C

Number of circumferential nodes (default: 7). This is the number of nodes you wish to have along the circumferential direction of the cylindrical shell model, *i.e.*, along 15 degrees of circular arclength. Note that this number should be consistent with the number of nodes per element. For example, **NNODES_C** can be any number greater than 1 for 4-node quadrilateral elements, whereas it must be an odd number greater than 1 for 9-node quadrilateral elements.

4.9.3.14 NONLINEAR

Nonlinear (post-buckling) analysis flag (default=<true>). This flag should be turned on if you want to perform nonlinear analysis in the current run. The pre-requisites are pre-processing, stability analysis and imperfection superposition, all of which may be performed either in a previous run or in the current run — by setting the appropriate arguments (*i.e.*, **PREP**, **STABILITY** and **IMPERFECTION**).

4.9.3.15 NSTEPS

Maximum number of load steps to be computed in the current nonlinear analysis run (default=30). For more details, refer to the same argument name under solution procedure **NL_STATIC_1**.

4.9.3.16 POST

Postprocessing flag (default=<false>). This flag should be turned on if you want selected response-history parameters to be added to the **HC.DBR** database. Note that it is not necessary to use this option in order to archive the basic load-displacement curve and solution parameters. It is only needed if you wish to archive special displacement and/or internal force component response histories post-facto.

4.9.3.17 PREP

Preprocessing flag (default=<true>). This flag must be turned on the first time procedure COMPRESSED_CYL is run, as it causes the model to be generated. If subsequent runs are used to perform other stages of the analysis (*e.g.*, nonlinear re-starts), then PREP must be set to <false> for those subsequent runs.

4.9.3.18 SPEC_DIS

Specified displacement flag (default: <false>). By setting this flag to <true>, uniform axial end-shortening is imposed instead of the uniform axial loading. This can make a significant difference in both the buckling and post-buckling response, and is not recommended for novice users of this procedure. This is because thin axially-compressed shells are not only imperfection sensitive, but also *boundary condition* sensitive, and uniform axial loading does not correspond (exactly) to uniform axial edge displacements. Note that the reference specified displacement (*i.e.*, end-shortening) magnitude equals .01 inches, and corresponds to an axial load of about .467 times the classical buckling load.

4.9.4 USAGE GUIDELINES AND EXAMPLES

Procedure HINGED_CYL may be used by preceding the procedure name by the *call directive. Procedure arguments may be changed from their default values by including any or all of the arguments and their new values when the procedure is called. A space or blank is required between the end of the procedure name and the left parenthesis.

```
*call HINGED_CYL ( ES_PROC  = ES1       ; ES_NAME = EX97 ; --
                   NNODES_A  =  5        ; NNODES_C =  5   ; --
                   SPEC_DIS  =    <false> ; --
                   DRILLING_DOF = <false> ; --
                   AUTO_DOF_SUP = <true>  ; --
                   PREP      =    <true>  ; --
                   NONLINEAR =     <true> ; --
                   POST      =    <true>  ; --
                   BEG_STEP  =  1        ; --
                   NSTEPS    = 10        ; --
                   BEG_LOAD  =   .1      ; --
                   MAX_LOAD  =  3.0      ; --
                   DBc       = HC.DBc    ; DBr    = HC.DBr )
```

(E1) To perform an entire analysis using the default options, simply invoke the procedure without any arguments, *i.e.*,

```
*call HINGED_CYL
```

This will perform linear buckling eigenvalue analysis and 30 steps of nonlinear analysis with a 5×5 grid of ES1/EX97 shell elements. The time required for this analysis is machine-dependent. Using the default values for the procedure arguments, the amount of systems is shown in Table 4.9-1.

Table 4.9-1 CPU TIMES Table

Computer System	Total CPU Time, seconds.
VAX 11/785 VMS 4.7	
MicroVAX ULTRIX 2.2	
SUN SUNVIEW 4.0	
CONVEX C220 VERSION 7.0	
CRAY-2 UNICOS 4.0	

(E2) Not suppressing the drilling rotational freedoms can cause strange behavior for some elements during nonlinear analysis. On the other hand, suppressing these freedoms explicitly using the DRILLING_DOF argument may cause some over-stiffening for coarse meshes with some elements. It is probably best to suppress the drilling freedoms explicitly unless the element actually has intrinsic drilling stiffness.

4.9.5 LIMITATIONS

4.9.6 ERROR MESSAGES AND WARNINGS

None.

4.9.7 PROCEDURE FLOWCHART

HINGED_CYL	(main procedure)
GEN_SHELL	(generate model)
HC_BC	(generate boundary conditions and applied loads)
HCD_BC	(generate boundary conditions and specified displacements)
L_STATIC_1	(perform linear static analysis)
NL_STATIC_1	(perform nonlinear static analysis)
HISTORY	(archive selected displacement/force histories)

4.9.8 PROCEDURE LISTING

4.9.9 REFERENCES

4.9-1 Stewart, Caroline B.: *The Computational Structural Mechanics Testbed User's Manual.* NASA TM-100644, October 1989.

4.10 Procedure PEAR_CYL

4.10.1 GENERAL DESCRIPTION

The nonlinear shell response of cylindrical shells with a noncircular cross-section became the subject of intense research in the early 1970's. Early Space Shuttle fuselage configurations were noncircular, and the "pear-shaped" cross-section was a leading candidate. The pear-shaped cylinder shown in Figure 4.10-1 has been adopted by many researchers and the behavior of this shell subject to a uniform end-shortening investigated (*e.g.*, refs. 4.10-1 and 4.10-2). The results in this section are compared with results reported by Hartung and Ball (ref. 4.10-1) and by Almroth and Brogan (ref. 4.10-2). In all cases, only one fourth of the cylinder is modeled. The shell is isotropic with a uniform thickness of 0.01 inches. The boundaries are simply supported.

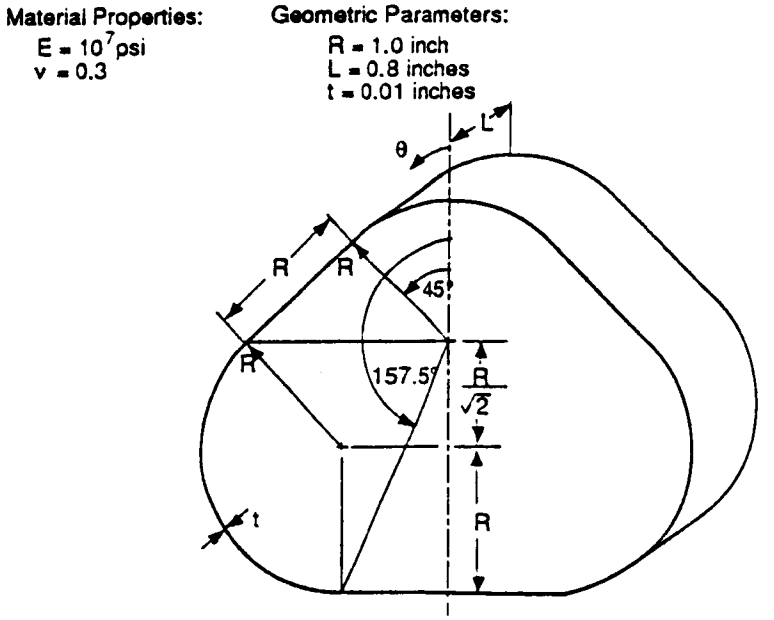


Figure 4.10-1 Pear-shaped cylinder – geometry, properties, and loading.

4.10.1.1 Model Description

The finite element model of the pear-shaped cylinder is generated in segments as shown in figure 4.10-2. Joint locations and element connectivity are defined using the mesh generating capabilities of Testbed processors TAB and ELD. The user must specify only the number of elements along the 45° arc of segment 1 (**NEL_ARC**), the number of elements along the flat edge of segment 2 (**NEL_FLAT**), and the number of elements and through the depth of the cylinder in the z-direction (**NEL_DEPTH**). The finite element mesh, loading, and constraints are then automatically generated. In order to facilitate both the model generation and the results interpretation, several alternate reference frames have been defined.

Reference frames 1, 2, and 3 are defined automatically in processor TAB as global reference frames, rotated $-90°$ about z axis, and rotated 90° about y axis, respectively. Reference frames 4, 5, and 6 are translations of the origin to points A, B, and C, respectively (see figure 4.10-2). Frames 7 and 8 involve rotations about the global z-axis and include no translations.

The results of interest in this analysis are the normal and tangential stresses and displacements. Each joint is therefore assigned a joint reference frame to which displacements, constraints, and loads are referred. The stress component directions are defined by the order of the element connectivity using the element network generators of processor ELD.

Segment 1 of the finite element model is generated using reference frame 4. The FORMAT=2 statement of the TAB/JLOC subprocessor allows for the generation of a regular mesh of nodes on the cylindrical surface of segment 1. The TAB/JLOC input is in cylindrical coordinates with point A as the origin of the cylindrical frame. The joints in this segment are each assigned joint reference frame -4 indicating a cylindrical frame with the joint 1-axis concident with the outward normal at each joint, the joint 3-axis coincident with the global z-axis, and the joint 2-axis coincident with the tangent to each joint.

Segment 2 is also generated using frame 4 although in this case, frame 4 is employed as a rectangular frame. The TAB/JLOC subprocessor generates a regular mesh of nodes connecting the four corners of the rectangular surface of the segment. The TAB/JLOC input is in cartesian coordinates with point A at the origin of the rectangular frame. Each joint in this segment is assigned joint reference frame 7 indicating that the 1- and 2-axes of each joint are coincident with x and y axes rotated 135° about global z. The joint 3-axis and global z-axis are coincident.

Segment 3, a 135° section of arc, is generated using reference frame 5, centered at point B. As with segment 1, the FORMAT=2 statement of the TAB/JLOC subprocessor is used to generate a regular mesh of nodes on the cylindrical segment. The number of elements around the circumference of this segment is assumed to be three times the number of elements around the circumference of Segment 1. The joints in this segment are each assigned joint reference frame -5 indicating a cylindrical frame with the joint 1-axis coincident with the outward normal at each joint, the joint 3-axis coincident with the global z-axis, and the joint 2-axis coincident with the tangent to each joint.

Segment 4 is generated using reference frame 6, centered at point C. Frame 6 is used as a rectangular frame to generate a regular mesh of nodes connecting the four corners of the rectangular surface of the segment. The number of elements along this segment is half that used in segment 2. Each joint in this segment is assigned joint reference frame 8 indicating that the 1- and 2-axes of each joint are coincident with x- and y- axes rotated 270° about global z-axis. The joint 3-axis and global z-axis are coincident.

Three finite element models were considered in this study as shown in Figure 4.10-3. Mesh 1 (3×26) has three nodes through the cylinder depth and 26 nodes around the half-cylinder circumference (see Figure 4.10-3a). The mesh is composed of 50 4-node quadrilateral shell elements (ES1/EX47 or ES5/E410) and a total of 78 nodes. Mesh 2 (5×37) has five nodes through the cylinder depth and 37 nodes around the half-cylinder circumference (see Figure 4.10-3b). This mesh accommodates both 4- and 9-node elements and will contain either 144 4-node (ES1/EX47 or ES5/E410) or 36 9-node (ES1/ES97) quadrilateral shell elements and 185 nodes. Mesh 3 (7×51) has seven nodes through the cylinder depth and 51 nodes around the half-cylinder circumference (see Figure 4.10-3c). Mesh 3 accommodates both 4- and 9-node elements and will contain either 300 4-node (ES1/EX47 or ES5/E410) or 75 9-node (ES1/EX97) quadrilateral shell elements and 357 nodes.

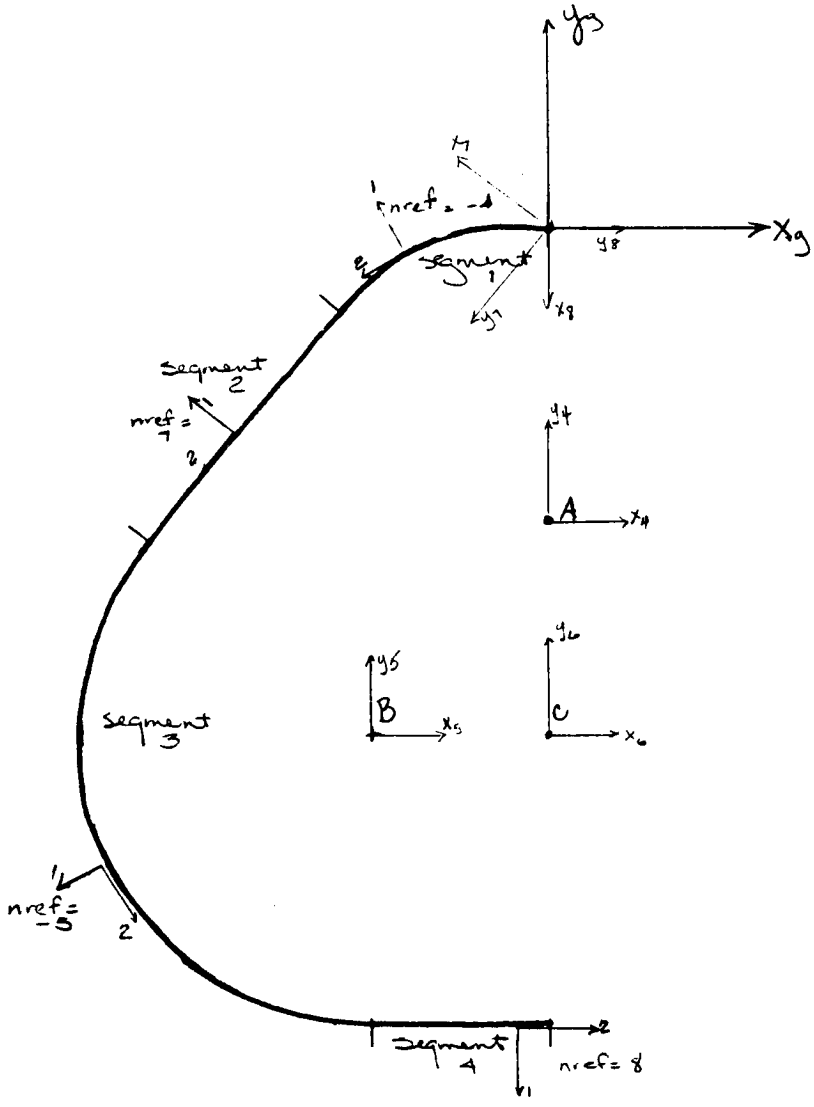
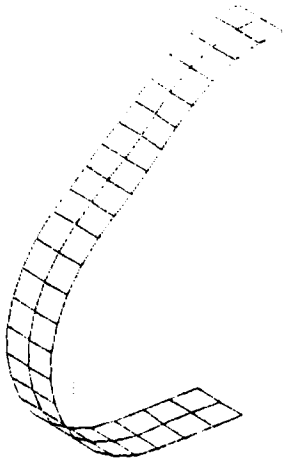
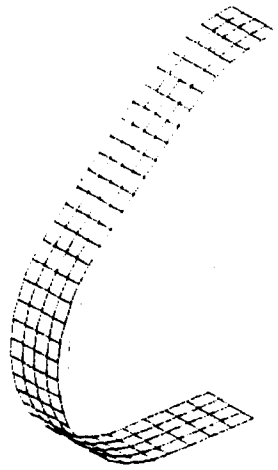
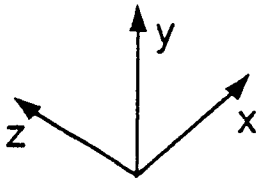


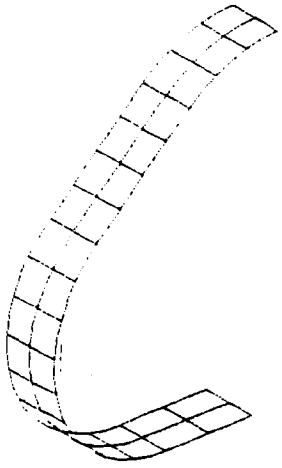
Figure 4.10-2 Model Generation Strategy and Reference Frames



(a) Mesh 1.



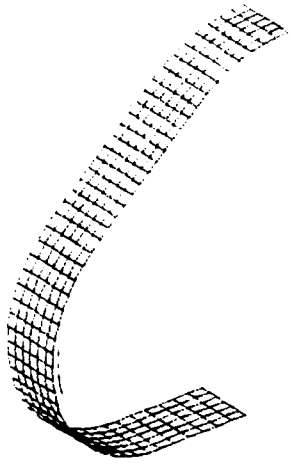


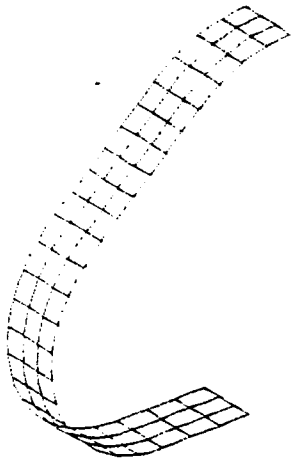


4-node elements

9-node elements

(b) Mesh 2.





4-node elements

9-node elements

(c) Mesh 3.

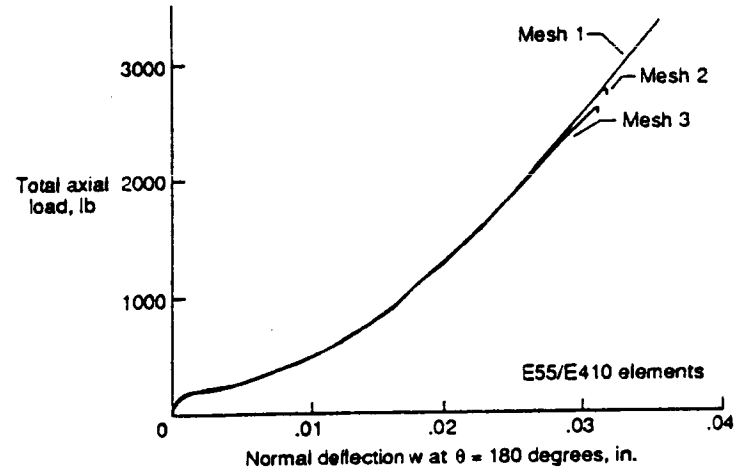
Figure 4.10-3 Finite element models of pear-shaped cylinder.

4.10.1.2 Analysis Description

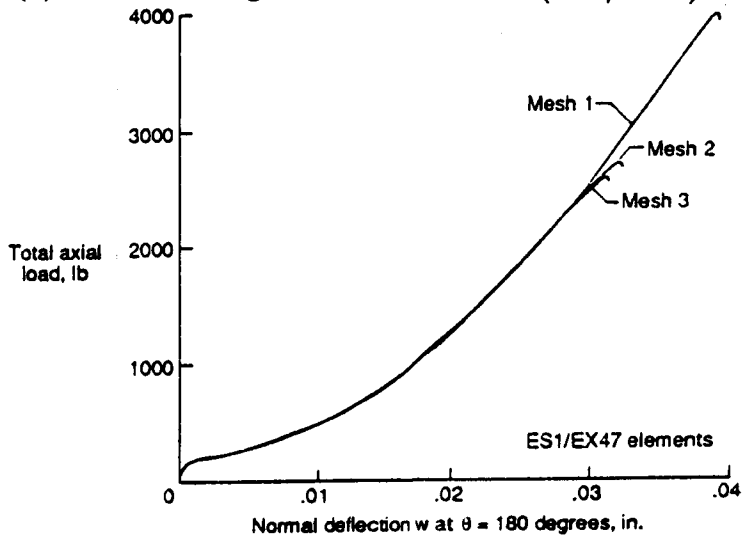
4.10.1.3 Available Solutions

The pear-shaped cylinder is representative of a shell-type structure with a complex nonlinear collapse behavior. The shell response becomes nonlinear at a very low value of applied end-shortening, and the normal deflections of the flat portions of the shell increase rapidly. The nonlinear analysis of the cylinder is performed using the procedure `NL_STATIC_1`. The distribution of the normal displacement (normal to the shell surface) is shown in Figure 4.10-4 as a function of the applied load for the point of maximum normal displacement (at x =0, θ =180°) and for all meshes and element types considered. These results indicate that a converged solution is provided by the ES1/EX97 elements as the curves for Mesh 2 and Mesh 3 in Figure 4.10-4c are nearly identical. These three solutions correspond to a discretization with 185 nodes with either 144 4-node flat classical (ES5/E410) elements, 144 4-node flat shear-flexible (ES1/EX47) elements, or 36 9-node curved shear-flexible (ES1/EX97) elements.

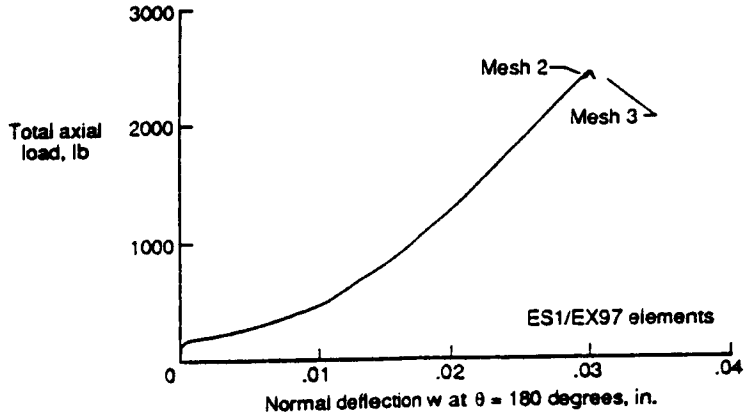
A comparison of the three solutions obtained using Mesh 2 is provided in Figure 4.10-5. The effects of transverse shear flexibility, present in the ES1/EX97 and ES1/EX47 elements but not present in ES5/E410 elements, are apparent near collapse. The response curves begin to separate slightly at approximately half the elastic collapse load with the shear flexible elements exhibiting a consistent, yet slightly lower, stiffness than the response obtained using the ES5/E410 element. The elastic collapse loads obtained using the 4-node flat elements is nearly the same. However, results obtained using higher-order curved elements to model the shell and its response are approximately 10% lower than the collapse loads obtained using the flat elements.



(a) Models using 4-node flat classical (ES5/E410) elements.



(b) Models using 4-node flat shear-flexible (ES1/EX47) elements.



(c) Models using 9-node curved shear-flexible (ES1/EX97) elements.

Figure 4.10-4 Nonlinear response of pear-shaped cylinder.

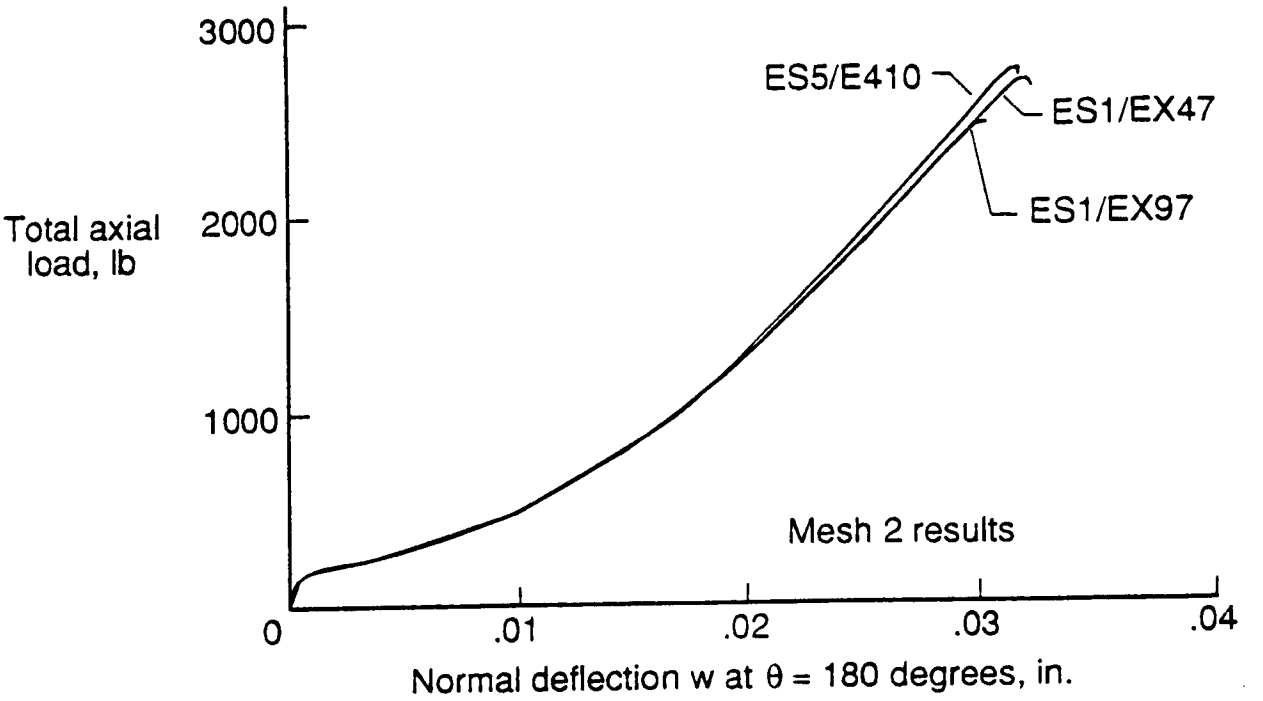


Figure 4.10-5 Nonlinear response of pear-shaped cylinder – Mesh 2.

Hartung and Ball (ref. 4.10-1) reported an elastic collapse load of 2372 pounds using a finite difference version of STAGS. Several years later, Almroth and Brogan (ref. 4.10-2) performed a convergence study using the finite element version of STAGS and estimated that the elastic collapse load was between 2300 and 2400 pounds. These results, along with the Testbed results, and results from independent elastic collapse analyses using the STAGSC-1 computer code, are summarized in Table 4.10-1.

In Figure 4.10-6, the normal deflection at x =0 is plotted as a function of the circumferential coordinate θ for four different levels of applied load: 154, 300, 1689, and 2464 pounds (load steps 10, 20, 30 and 40 respectively). The final load step (step 40 at 2464 pounds) occurs just after collapse. The flat portions of the shell, from $45° \leq \theta \leq 90°$ and $157.5° \leq \theta \leq 180°$, show a rapid growth in normal deflections. Associated with this growth is a redistribution of the longitudinal stress indicating that the curved portions begin to take up a larger percentage of the total axial load. This type of behavior can be seen in Figure 4.10-7 which plots the longitudinal stress resultant as a function of the circumferential coordinate for load steps 10, 20, 30 and 40.

Table 4.10-1 Elastic collapse loads for pear-shaped cylinder.

Source	Element Name	Mesh(1)	Elastic Collapse Load, pounds
Hartung and Ball	(2)	4 × 40	2372
Almroth and Brogan	411	3 × 27	3586
		5 × 37	2731
		7 × 47	2586
	440	5 × 37	2657
		7 × 47	2530
STAGSC-1	410	3 × 26	3570
		5 × 37	2734
CSM Testbed	ES5/E410	3 × 26	3343
		5 × 37	2753
		7 × 51	2577
	ES1/EX97	5 × 37	2475
		7 × 51	2466
	ES1/EX47	3 × 26	3945
		5 × 37	2696
		7 × 51	2568

(1) Mesh description is $n \times m$ meaning n rows by m columns of nodes.
(2) Finite difference version of STAGS.

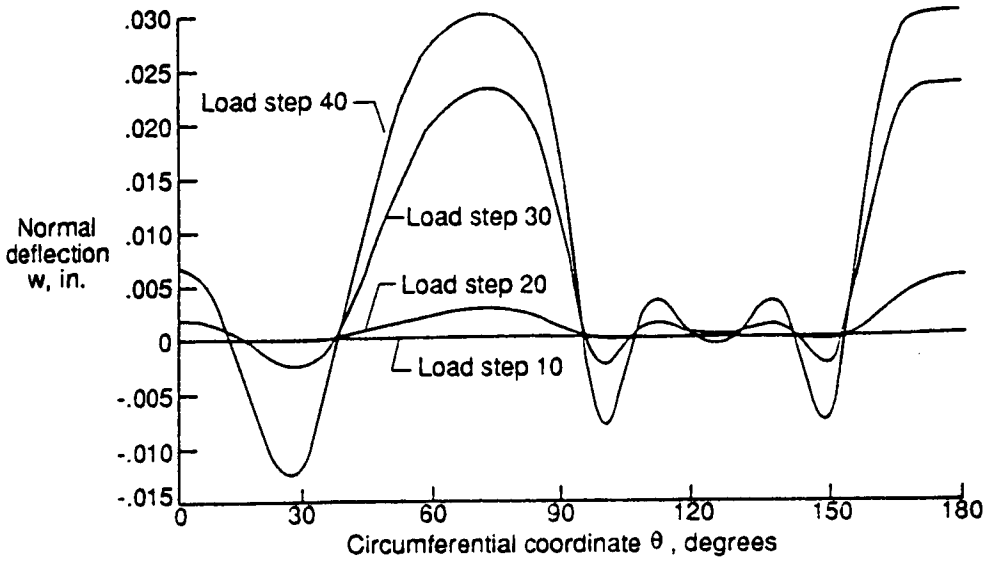


Figure 4.10-6 Normal deflection distribution at cylinder midlength for various load steps.

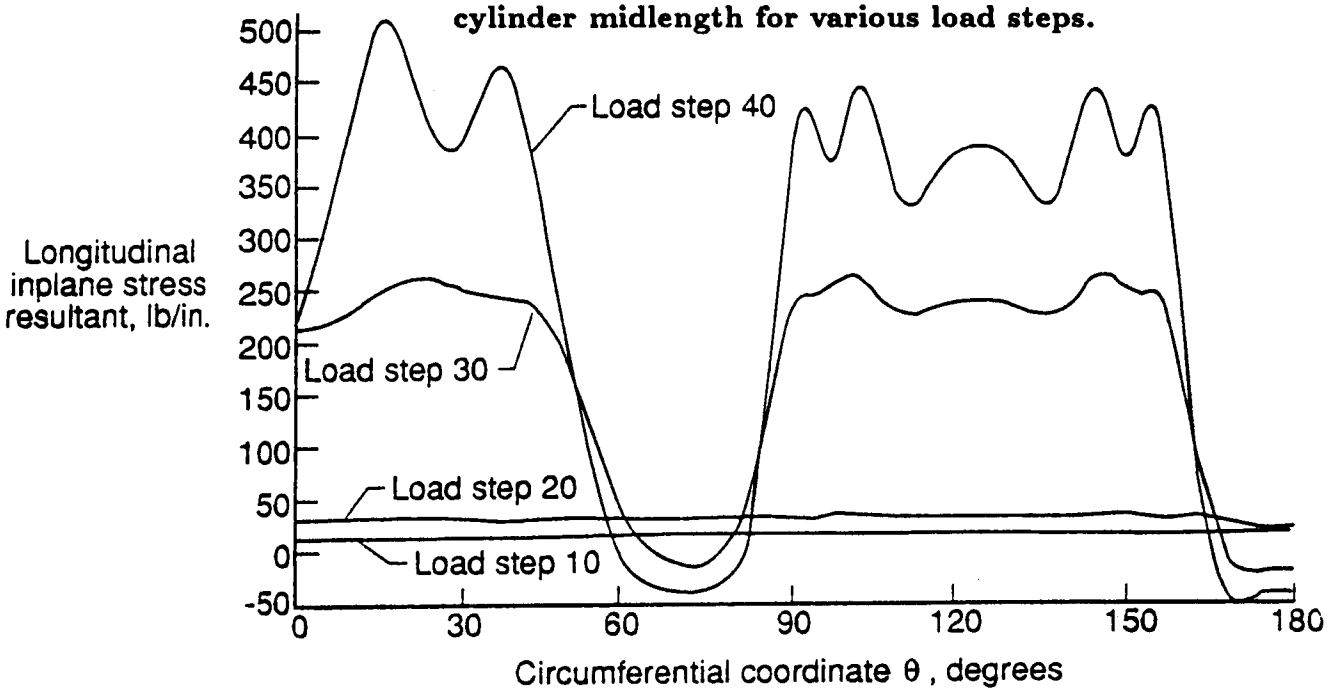


Figure 4.10-7 Longitudinal inplane stress resultant N_x distribution at cylinder midlength for various load steps.

4.10.2 PROCEDURE USAGE

Procedure **PEAR_CYL** may be used by preceding the procedure name by the ***call** directive, and following it by a list of arguments enclosed in parentheses. Procedure arguments are order-independent, and most have default values thus making them optional. The formal syntax is as follows:

```
*call PEAR_CYL ( arg1 = val1 ; arg2 = val2 ; ...)
```

where **arg1** and **arg2** represent argument names, and **val1** and **val2** represent their corresponding values. Note that semi-colons are required between arguments, and a double dash (--) may be used to continue the argument list on the next line.

The allowable arguments for procedure **PEAR_CYL** are summarized in the following table, along with their default values (if they exist). Note that arguments without defaults are generally *mandatory*, while those with defaults are generally optional. Exceptions to this rule are noted in the following section under detailed argument descriptions.

Argument	Default Value	Meaning
ES_PROC	ES1	Element Processor
ES_NAME	EX97	Element name
ES_PARS	0.	Element research parameters
NEL_ARC		Number of elements along 45° arc segment
NEL_DEPTH		Number of elements along depth of shell
NEL_FLAT		Number of elements along largest flat segment
LD_DIR		Direction to total load
LINEAR		Perform linear analysis
NSTEPS	30	Number of nonlinear load steps
BEG_STEP	1	Starting load step number
MAX_CUTS	3	Maximum number of step cuts (halvings)
BEG_LOAD	.1	Starting load factor
MAX_LOAD	1.0	Maximum load factor
DBC	PEAR.DBC	Computational database name
DBR	PEAR.DBR	Results database name
STABILITY	<true>	Perform linear stability (buckling) analysis
NONLINEAR	<true>	Perform nonlinear analysis
POST	<false>	Perform postprocessing (selected data archival)

4.10.3 ARGUMENT DESCRIPTIONS

4.10.3.1 BEG_LOAD

Starting load factor in nonlinear analysis (default=.1). This factor is multiplied times the reference load vector to obtain the starting load vector. For example, if specified forces are applied (which is the default option), then **BEG_LOAD** = .1 means that the first load step to

be computed in the nonlinear analysis will be at one-tenth of the classical buckling load level. Note that this argument is *irrelevant* for re-start runs. For more details, refer to the same argument name under solution procedure **NL_STATIC_1**.

4.10.3.2 BEG_STEP

Number of starting load step in nonlinear analysis (default=1). This is the number of the first step to be computed during an analysis interval. When starting a nonlinear analysis, the first step is obviously 1. When re-starting (*i.e.*, continuing in a subsequent run) a nonlinear analysis, **BEG_STEP** should be set to the number of the next step to be computed — not to the number of the last step computed. The solution procedure (**NL_STATIC_1**) will automatically use any previous step(s) required to continue the analysis — providing the necessary steps are available in the database. Currently, the number of consecutive preceding steps required for a restart is three. For more details, refer to the same argument name under solution procedure **NL_STATIC_1**.

4.10.3.3 DBC

Name of computational database file (default=**PEAR.DBC**). This file will contain all model definition data, element computational data, one copy of the assembled and factored stiffness matrices, the buckling eigensolution, and displacement and internal force vectors for every load step computed during the analysis.

4.10.3.4 DBR

Name of results database file (default=**PEAR.DBR**). This file will contain one dataset called **RESPONSE.HISTORY** generated during nonlinear analysis. The dataset will contain record groups — indexed by load step number — for a number of solution parameters, including the load factor and maximum axial displacement components. This database is valuable for obtaining load-displacement plots, and for evaluating the performance of the nonlinear solution strategy employed.

4.10.3.5 ES_NAME

Element name (default: **EX97**). This is the name of the specific shell-element type you wish to select, within the element processor defined by argument **ES_PROC**. The default shell-element type, **EX97**, is a 9-node quadrilateral element implemented in processor **ES1**, and described in The Computational Structural Mechanics Testbed User's Manual (see ref. 4.10-3).

4.10.3.6 ES_PARS

Element research parameters (default: `0., ...`). This is an optional list of element-dependent parameters that some elements provide, primarily when the element is still undergoing research and refinement.

4.10.3.7 ES_PROC

Element processor (default: ES1) This is the name of the structural element (ES) processor that contains the shell element type you wish to employ in the model. The default shell-element, processor ES1, is described in The Computational Structural Mechanics Testbed User's Manual (see ref. 4.10-3).

4.10.3.8 LD_DIR

Direction in which to total the load (default: 3). This argument specifies the direction in which the reaction forces will be summed in order to determine the total load applied.

4.10.3.9 LINEAR

Linear analysis flag (default: <false>). This flag should be turned on to perform only a linear static analysis.

4.10.3.10 MAX_CUTS

Maximum number of load step cuts in nonlinear analysis (default=3). For more details, refer to the same argument name under solution procedure NL_STATIC_1.

4.10.3.11 MAX_LOAD

Maximum load factor in nonlinear analysis (default=1.0). This sets an upper limit for the load level, which can be a convenient way of terminating the arc-length controlled solution algorithm employed within procedure NL_STATIC_1. Since the load factor is actually an unknown in this solution procedure, there is no way of knowing a-priori how many load steps will be required to attain a particular load level. Thus, the analysis will be terminated when either MAX_LOAD is exceeded or NSTEPS is exceeded — whichever comes first. For more details, refer to the same argument name under solution procedure NL_STATIC_1.

4.10.3.12 NEL_ARC

Number of elements along arc segment (default: 4). This argument specifies the number of elements along a 45° arc segment of the shell.

4.10.3.13 NEL_DEPTH

Number of elements along depth of shell (default: 1). This argument specifies the number of elements along the length (or depth) of the shell.

4.10.3.14 NEL_FLAT

Number of elements along flat segments (default: 4). This argument specifies the number of elements along the flat segments of the shell.

4.10.3.15 NONLINEAR

Nonlinear analysis flag (default=<true>). This flag should be turned on to perform non-linear analysis in the current run.

4.10.3.16 NSTEPS

Maximum number of load steps to be computed in the current nonlinear analysis run (default=30). For more details, refer to the same argument name under solution procedure NL_STATIC_1.

4.10.3.17 POST

Postprocessing flag (default=<false>). This flag should be turned on if you want selected response-history parameters to be added to the PEAR.DBR database. Note that it is not necessary to use this option in order to archive the basic load-displacement curve and solution parameters. It is only needed if you wish to archive special displacement and/or internal force component response histories post-facto.

4.10.3.18 STABILITY

Stability (buckling) analysis flag (default <true>). This flag should be turned on if you want the buckling eigenvalue analysis to be performed in the current run. Preprocessing is a pre-requisite for this option. If you are just performing a nonlinear analysis re-start run, then you should turn this flag off.

4.10.4 USAGE GUIDELINES AND EXAMPLES

Procedure PEAR_CYL may be used by preceding the procedure name by the *call directive. Procedure arguments may be changed from their default values by including any or all of the arguments and their new values when the procedure is called. A space or blank is required between the end of the procedure name and the left parenthesis. If the default values of the procedure arguments are to be used, then only the procedure name is required.

```
*call PEAR_CYL ( ES_PROC = ES1    ; -- . Element Processor
                 ES_NAME = EX97   ; -- . Element name
                 NEL_ARC  = 4     ; -- . Elements along 45 deg. arc
                 NEL_FLAT = 4     ; -- . Elements along flat segment
                 NEL_DEPTH= 2     ; -- . Elements through depth (z)
                 LINEAR     = <true> ; -- .
                 PREP       = <true> ; -- .
                 STABILITY  = <false> ; -- .
                 NONLINEAR  = <false> ; -- .
                 POST       = <false> ; -- .
                 DBC        = PEAR.DBC ; -- .
                 DBR        = PEAR.DBR ; -- .
                 NSTEPS     = 30  ; -- .
                 BEG_STEP   = 1   ; -- .
                 MAX_CUTS   = 3   ; -- .
                 BEG_LOAD   = .1  ; -- .
                 MAX_LOAD   = 1   ; -- .
                 LD_dir     = 3     -- . Direction of total load calculation
                               )
```

(E1) To perform an entire analysis using the default options, simply invoke the procedure without any arguments, *i.e.*,

```
*call PEAR_CYL
```

This will perform linear buckling eigenvalue analysis and 30 steps of nonlinear analysis with a 7×7 grid of **ES1/EX97** shell elements. The time required for this analysis is machine-dependent. Using the default values for the procedure arguments, the amount of CPU times required for a stability analysis and a nonlinear analysis on various computer systems are shown in Table 4.10-2.

Table 4.10-2 CPU TIMES Table

Computer System	Total CPU Time, sec.	
	Stability	Nonlinear
VAX 11/785 VMS 4.7		
MicroVAX ULTRIX 2.2		
SUN SUNVIEW 4.0		
CONVEX C220 VERSION 7.0		
CRAY-2 UNICOS 4.0		

(E2) Not suppressing the drilling rotational freedoms can cause strange behavior for some elements during nonlinear analysis. On the other hand, suppressing these freedoms explicitly using the **DRILLING_DOF** argument may cause some over-stiffening for coarse meshes with some elements. It is probably best to suppress the drilling freedoms explicitly unless the element actually has intrinsic drilling stiffness.

4.10.5 LIMITATIONS

None.

4.10.6 ERROR MESSAGES AND WARNINGS

None.

4.10.7 PROCEDURE FLOWCHART

PEAR_CYL	(main procedure)
PEAR_MODEL	(define model)
ES	(define elements)
PEAR_CON	(define boundary conditions)
MATDAT	(define material properties)
DEF_ELTS4	(define 4-node models)
DEF_ELTS9	(define 9-node models)
PEAR_AD	(define applied displacements)
L_STATIC	(linear static analysis procedure)
TOTAL_LOAD	(sum reaction forces to get total load)
L_STABIL_2	(linear stability analysis procedure)
TOTAL_LOAD	(sum reaction forces to get total load)
NL_STATIC_1	(nonlinear static analysis procedure)
HISTORY	(archive historical data)
TOTAL_LOAD	(sum reaction forces to get total load)

4.10.8 PROCEDURE LISTING

4.10.8.1 UNIX Script

pear_cyl.com

```
cd $SCR/$USER
cp $CSM_PRC/proclib.gal proclib.gal
chmod u+w proclib.gal
rm PEAR.*
time testbed <<\endinput
  *set echo off
  *set plib 28
  *open 28 proclib.gal /old
  *add '$CSM_APP/pear_cyl/pear_cyl.clp'
*def/i es_proj = 1
*def/i es_coro = 1
*def/i nl_geom = 2
*def/a solver_name =BAND
*def/a nl_solver =BAND
.
*call PEAR_CYL ( ES_PROC = ES1  ; -- . Element Processor
                 ES_NAME = EX97  ; -- . Element name
```

```
                    NEL_arc  = 4    ; -- . Elements along 45 deg. arc
                    NEL_flat = 4    ; -- . Elements along flat segment
                    NEL_depth= 3    ; -- . Elements through depth (z)
                    NSTEPS   = 60   ; -- .
                    DBC      = PEAR.DBC ; --
                    DBR      = PEAR.DBR ; --
                    PREP     = <true>   ; --
                    LINEAR   = <false>  ; --
                    STABILITY= <false>   ; --
                    NONLINEAR= <true>   ; --
                    NEWTON   = <false>   ; --
                    POST     = <true>  ; --
                    BEG_LOAD = .2   ; -- .
                    MAX_LOAD = 120.0      )
[xqt EXIT
\endinput
```

4.10.8.2 CLAMP Procedure

pear_cyl.clp

```
*procedure PEAR_CYL ( ES_PROC = ES1  ; -- . Element Processor
                      ES_NAME = EX97 ; -- . Element name
                      NEL_arc  = 4    ; -- . Elements in 45 deg. arc
                      NEL_flat = 4    ; -- . Elements along flat segment
                      NEL_depth= 1    ; -- . Elements through depth
                      LD_dir   = 3    ; -- . Direction of total load
                      NSTEPS   = 30     ; --
                      BEG_STEP =  1     ; --
                      BEG_LOAD = .01    ; --
                      MAX_LOAD =  1.0   ; --
                      MAX_CUTS = 3      ; --
                      DBC  = PEAR.DBC; DBR = PEAR.DBR; --
                      PREP = <true>; --
                      LINEAR = <true>; --
                      STABILITY = <false>; --
                      NONLINEAR = <false>; --
                      NEWTON = <false>; --
                      POST = <false> --
                    )
.
.          ------------------------------------------------------
.          CSM Testbed Procedure for Analysis of Pear-Shaped Cylinder
.                written by S. McCleary (PRC Kentron) -- 2/88
.          ------------------------------------------------------
.
*def/i NOM_ldi = 3
*open/new 1 [DBc]
.
.
*if [PREP] /then
    *call PEAR_MODEL ( ES_proc = [ES_PROC]; --
                       ES_name = [ES_NAME]; --
```

```
                    nel_a    = [NEL_arc] ; --
                    nel_f    = [NEL_flat]; --
                    nel_z    = [NEL_depth] )
*endif
.
.
*if < [LINEAR] >/then
.
*call L_STATIC
.
          *call TOTAL_LOAD ( EQUIL_CK= 0       ; --
                             NSTEPS = 0        ; --
                             LOAD_SET = 1      ; --
                             CONS_SET = 1      ; --
                             NAME1 = 'STAT'    ; --
                             NAME2 = 'REAC'    ; --
                             FAC = 1.0         ; --
                             LD_DIR = [Ld_dir]  --
                           )
.
*elseif < [STABILITY] > /then
.
*call L_STABIL_2 (function='all';n_modes=2;print=<true>;stress=<true> ; --
                  cons_set=1; bcon_set=1 )
.
          *call TOTAL_LOAD ( EQUIL_CK= 0       ; --
                             NSTEPS = 0        ; --
                             LOAD_SET = 1      ; --
                             CONS_SET = 1      ; --
                             NAME1 = 'STAT'    ; --
                             NAME2 = 'REAC'    ; --
                             FAC = 1.0         ; --
                             LD_DIR = [Ld_dir]  --
                           )
.
*endif
*open/new 3, [DBr]
.
.
*if < [NONLINEAR] > /then
.
    *def/i ns_overwrite == <true>
    *call NL_STATIC_1 ( NL_GEOM = <nl_geom>  ; --
                        COROTATION=<es_coro> ; --
                        NEWTON  = [NEWTON]   ; --
                        BEG_STEP  = [BEG_STEP]   ; --
                        MAX_STEPS = [NSTEPS]     ; --
                        BEG_LOAD  = [BEG_LOAD]   ; --
                        MAX_LOAD  = [MAX_LOAD]   ; --
                        MAX_CUTS  = [MAX_CUTS]   ; --
                        NOMINAL_DB = [DBr]       ; --
                        NOMINAL_DS = [ES_PROC].[ES_NAME].1.1 ; --
                        N_SELECT  = 2            ; --
```

```
                        SEL_NODES = 1,<<nntotal>-<nnod_z>+1>    ; --
                        SEL_DOFS  = 1,1              --
                      )
   *toc 1
*endif
.
*if < [POST] /gt 0 > /then
    *open 1, [DBc]
    *open 3, [DBr]
    *if < [POST] /eq 1 > /then
          *call HISTORY ( input_ds = TOT.DISP; output_ldi = 1         ; --
                          output_ds= [ES_PROC].[ES_NAME].1.1        ; --
                          output_rn= RADIAL_DISP ; LOCATION = NODES  ; --
                          num_steps = [NSTEPS]                      ; --
                          steps = [BEG_STEP]:<[BEG_STEP]+[NSTEPS]-1>; --
                          num_nodes= 1                               ; --
                          nodes = <<nntotal>-<nnod_z>+1>             ; --
                          component = 1                                --
                        )
          *call TOTAL_LOAD ( EQUIL_CK= 0       ; --
                             NSTEPS = [NSTEPS] ; --
                             LOAD_SET = 0      ; --
                             CONS_SET = 0      ; --
                             NAME1 = 'REAC'    ; --
                             NAME2 = 'FORC'    ; --
                             FAC = 1.0         ; --
                             LD_DIR = [Ld_dir]   --
                           )
            *print 3 1 TOTAL* /out=9
            *print 3 1 DISP_U_<<nntotal>-<nnod_z>+1>* /out=9
  *endif
*endif
.
*end
.
.
*procedure PEAR_MODEL ( ES_proc = ES1  ; --
                        ES_name = EX97 ; --
                        nel_a   = 4    ; --
                        nel_f   = 4    ; --
                        nel_z   = 2    )
.
.      ------------------------
.      Initialize element macros
.      ------------------------
.
*call ES ( function = 'DEFINE ELEMENTS' ; es_proc = [ES_PROC]; --
                                          es_name = [ES_NAME] )
.
.      ----------------------------------------
.      Set dimensions and number-of-nodes macros:
.      ----------------------------------------
.
```

```
*def/g dz  == 0.40                          . Depth (in z direction) of cylinder
*def/g r   == 1.0                           . Radius of curved portions of cylinder
*def/g sq2  = (2.0^0.5)
*def/g rs2  = < <r> * <sq2> >
*def/g rd2  = < <r>/<sq2> >
*def/g rr2  = < <r> + <rd2> >
*def/g rm   = < -<r> >
*def/g rr2m = < -<rr2> >
*def/g rd2m = < -<rd2> >
*def/i nel_b = <[nel_f]*if < <es_nen> /eq 4 > /then
    *def/i nnod_z == < [nel_z] + 1 >                    . Through z
    *def/i nnod_a == < [nel_a] + 1 >                    . Along arc
    *def/i nnod_f == < [nel_f] + 1 >                    . Along flat segment
    *def/i nnod_b == < <nel_b> + 1 >                    . Along bottom segment
    *def/i nnod_c == <4*[nel_a]+[nel_f]+<nel_b> + 1 >   . Circumferentially
*elseif < <es_nen> /eq 9 > /then
    *def/i nnod_z == < 2*[nel_z] + 1 >                  . Through z
    *def/i nnod_a == < 2*[nel_a] + 1 >                  . Along arc
    *def/i nnod_f == < 2*[nel_f] + 1 >                  . Along flat segment
    *def/i nnod_b == < 2*<nel_b> + 1 >                  . Along bottom segment
    *def/i nnod_c == < 2*<4*[nel_a]+[nel_f]+<nel_b>> + 1 > . Circumferentially
*endif
*def/i nnod_3a =  < 3*< <nnod_a> - 1 > + 1 >           . Around 135 deg. arc
*def/i bseg1   =  1                                    . Start segment 1
*def/i bseg2   =  < <bseg1> + <<nnod_a> -1>*<nnod_z> > . Start segment 2
*def/i bseg3   =  < <bseg2> + <<nnod_f> -1>*<nnod_z> > . Start segment 3
*def/i bseg4   =  < <bseg3> + <<nnod_3a>-1>*<nnod_z> > . Start segment 4
*def/i nntotal == < <nnod_c>*<nnod_z> >                . Total number of nodes
.
[xqt TAB
     START <nntotal>
.
.    ------------------------------------------------------------------------
.    Define alternate reference frames to be used in setting up joint locations:
.    ------------------------------------------------------------------------
.
     ALTERNATE REFERENCE FRAMES
       4   1,0.   2,0.   3,0.   0.0     <rm>    0.0
       5   1,0.   2,0.   3,0.   <rd2m>  <rr2m>  0.0
       6   1,0.   2,0.   3,0.   0.0     <rr2m>  0.0
       7   1,0.   2,0.   3,135.
       8   1,0.   2,0.   3,270.
.
.    ----------------------
.    Define joint locations:
.    ----------------------
.
     JOINT LOCATIONS
 NREF = 4              . 90 to 135 degree arc
 FORMAT = 2
 <bseg1>   <r>      90.0  0.0   <r>    135.0  0.0  <nnod_a>  <nnod_z> <nnod_z>
    1      <r>      90.0  0.4   <r>    135.0  0.4
```

```
FORMAT = 1              . Flat segment
<bseg2>   <rd2m>  <rd2>  0.0   <rs2m> 0.0    0.0  <nnod_f>  <nnod_z> <nnod_z>
   1      <rd2m>  <rd2>  0.4   <rs2m> 0.0    0.4
NREF = 5                . 135 degree arc
FORMAT = 2
<bseg3>   <r>     135.0  0.0   <r>    270.0  0.0  <nnod_3a> <nnod_z> <nnod_z>
   1      <r>     135.0  0.4   <r>    270.0  0.4
NREF = 6                . Bottom flat segment
FORMAT = 1
<bseg4>   <rd2m>  <rm>   0.0   0.0    <rm>   0.0  <nnod_b>  <nnod_z> <nnod_z>
   1      <rd2m>  <rm>   0.4   0.0    <rm>   0.4
.
.    ------------------------------
.    Assign Joint Reference Frames:
.    ------------------------------
.
     JOINT REFERENCE FRAME ASSIGNMENTS
       NREF = -4:   <bseg1>, <<bseg2> - 1>
       NREF =  7:   <bseg2>, <<bseg3> - 1>
       NREF = -5:   <bseg3>, <<bseg4> - 1>
       NREF =  8:   <bseg4>, <nntotal>
.
.    ----------------------
.    Constraint Definitions:
.    ----------------------
.
*call PEAR_CON ( nnod_z = <nnod_z> ; nntotal = <nntotal> )
.
.    --------------------
.    Material Properties:
.    --------------------
.
*call MATDAT
.
.    -------------------
.    Element Definitions:
.    -------------------
.
*if < <es_nen> /eq 4 > /then
    *call DEF_ELTS4 ( ES_name = [ES_name] ; --
                      nel_a    = [nel_a]    ; --
                      nel_f    = [nel_f]    ; --
                      nel_z    = [nel_z]    ; --
                      nel_b    = <nel_b>    )
*else
    *call DEF_ELTS9 ( ES_name = [ES_name] ; --
                      nel_a    = [nel_a]    ; --
                      nel_f    = [nel_f]    ; --
                      nel_z    = [nel_z]    ; --
                      nel_b    = <nel_b>   )
*endif
.
.    ---------------
```

```
.    Applied Loading:
.    ----------------
.
*call PEAR_AD ( nnod_z = <nnod_z> ; nntotal = <nntotal> )
.
*end
.
*procedure PEAR_CON ( nnod_z ; nntotal )
.
     CONSTRAINT DEFINITION 1            . Buckling; simply supported
       symm plane=1                     . Plane 2,3 plane of symmetry
       symm plane=3                     . Plane 1,2 plane of symmetry
       nonzero 3 : [nnod_z],[nntotal],[nnod_z]  . Apply displacement at z=dz
       zero 1,2,6: [nnod_z],[nntotal],[nnod_z]  . Edge z=dz simply supported
       *if <ifeqs(<es_name>;E410)> /then
           *Remark E410 drilling freedoms ON
       *else
           zero 4 :    1, [nntotal], 1  . Constrain in-plane rotations
       *endif
*end
.
*procedure DEF_ELTS4 ( ES_name ; --
                       nel_a    ; --
                       nel_f    ; --
                       nel_z    ; --
                       nel_b    )
.
*def/i nel_circum = <4*[nel_a] + [nel_f] + [nel_b]>
*def/i jinc = <[nel_z]+1>
*def/i j1 = 1
*def/i j2 = 2
*def/i j3 = <<j2> + <jinc>>
*def/i j4 = <<j1> + <jinc>>
.
[xqt ELD
    <es_expe_cmd>
.
    <j1> <j2> <j3> <j4>  1  [nel_z] <nel_circum>
.
    stop
.
*end
.
*procedure DEF_ELTS9 ( ES_name ; --
                       nel_a    ; --
                       nel_f    ; --
                       nel_z    ; --
                       nel_b    )
.
*def/i nel_circum = <4*[nel_a] + [nel_f] + [nel_b]>
*def/i jinc = <2*[nel_z]+1>
*def/i j1 = 1
*def/i j2 = <<j1>+2>
```

```
*def/i j3 = <<j2> + 2*<jinc>>
*def/i j4 = <<j1> + 2*<jinc>>
*def/i j5 = <<j1> + 1>
*def/i j6 = <<j2> + <jinc>>
*def/i j7 = <<j5> + 2*<jinc>>
*def/i j8 = <<j1> + <jinc>>
*def/i j9 = <<j5> + <jinc>>
.
[xqt ELD
   <es_expe_cmd>
.
    <j1> <j2> <j3> <j4> <j5> <j6> <j7> <j8> <j9> 1  [nel_z] <nel_circum>
.
    stop
.
*end
.
*procedure MATDAT
.
[xqt AUS
*def   g = 3.84615e+6
*def/g t == 0.01
.
.
. E11 NU12 E22 G12 G13 G23 ALPHA1 ALPHA2 WTDEN
.
     TABLE(NI=16,NJ=1): OMB DATA 1 1
     I=1,2,3,4,5,6,7,8,9
     J=1: 10.0E+6 .30 10.0E+6 <g> <g> <g> 0.0 0.0 .1
.
     TABLE (NI=3,NJ=1,itype=0):  LAM OMB  1  1
           J=1 : 1 <t> 0.00
.
[xqt lau
.
*end
.
*procedure PEAR_AD ( nnod_z ; nntotal )
.
[xqt AUS
     sysvec : appl moti
     i=3 : j=[nnod_z],[nntotal],[nnod_z] : -2.E-5
*end
```

4.10.9 REFERENCES

4.10-1 Hartung, R. F.; and Ball, R. E.: A Comparison of Several Computer Solutions to Three Structural Shell Analysis Problems. AFFDL TR-73-15, April 1973.

4.10-2 Almroth, B. O. and Brogan, F. A. (1981). "Computational Efficiency of Shell Elements," *Nonlinear Finite Element Analysis of Plates and Shells*, Hughes, T. J. R., Pifko, A., and Jay, A. (Editors), AMD Vol. 48, ASME, pp. 147-165.

4.10-3 Stewart, Caroline B.: *The Computational Structural Mechanics Testbed User's Manual.* NASA TM-100644, October 1989.

4.11 Procedure PINCHED_CYL

4.11.1 GENERAL DESCRIPTION

The pinched cylinder – a circular cylindrical shell subjected to two equal and opposite radial forces at 0 and 180 degrees along its mid-span circumference – is a classical problem that has been used extensively to check the ability of shell elements to represent inextensional bending deformation. Two versions of the problem are considered here: (i) with open ends, and (ii) with a rigid membrane diaphragm on each end to prevent cross-sectional distortion. The open-ended version leads to pure inextensional deformation (i.e., zero membrane strains) in the limit as the shell thickness to shell radius ratio h/R approaches zero. An exact solution for this limiting case may be found in reference 4.11-1. The pinched cylinder with closed ends does not exhibit pure inextensional deformation, but rather a complex combination of bending and membrane effects with a bending boundary layer in the immediate vicinity of the concentrated loads. A series solution for this latter case may be found in reference 4.11-2.

This section describes a procedure that solves another classical problem which is one of the MacNeal-Harder test cases. Two versions of this circular cylindrical shell subjected to equal and opposite radial forces at its mid-span circumference are provided: (i) with open ends and (ii) with a rigid membrane diaphragm on each end. The L/2 by 90-degree model used in each case is shown in figure 4.11-1.

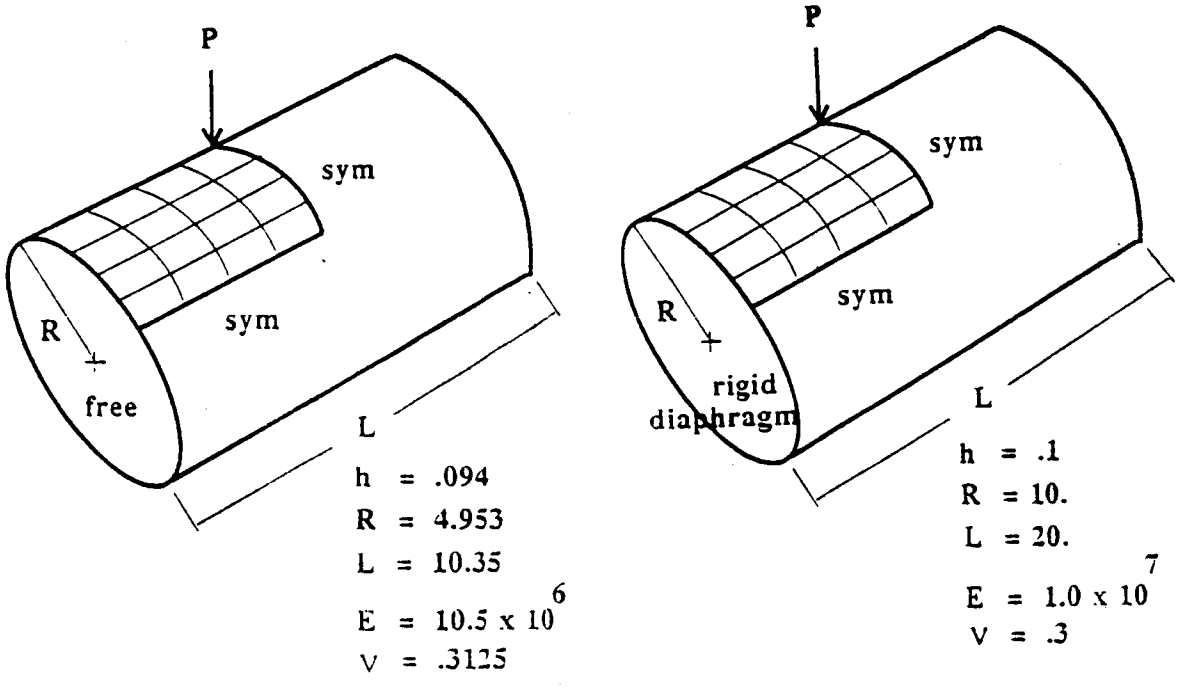


Figure 4.11-1 Pinched Cylinder Problem.

4.11.1.1 Model Description

4.11.1.2 Analysis Description

4.11.1.3 Available Solutions

4.11.2 PROCEDURE USAGE

Procedure PINCHED_CYL may be used by preceding the procedure name by the *call directive, and following it by a list of arguments enclosed in parentheses. Procedure arguments are order-independent, and most have default values thus making them optional. The formal syntax is as follows:

```
*call PINCHED_CYL ( arg1 = val1 ; arg2 = val2 ; ...)
```

where arg1 and arg2 represent argument names, and val1 and val2 represent their corresponding values. Note that semi-colons are required between arguments, and a double dash (--) may be used to continue the argument list on the next line.

The allowable arguments for procedure PINCHED_CYL are summarized in the following table, along with their default values (if they exist). Note that arguments without defaults are generally *mandatory*, while those with defaults are generally optional. Exceptions to this rule are noted in the following section under detailed argument descriptions.

Argument	Default Value	Meaning
ES_PROC	ES1	Element processor
ES_NAME	EX97	Element name
ES_PARS	0.	Element research parameters
NNODES_A	5	Number of axial nodes
NNODES_C	5	Number of circumferential nodes
AUTO_DOF_SUP	<true>	Automatic degrees of freedom suppression
DRILLING_DOF	<false>	Drilling (normal rotational) freedoms
DBC	PC.DBC	Computational database name
DBR	PC.DBR	Results database name
PREP	<true>	Perform preprocessing (model generation)
LINEAR	<true>	Perform linear static analysis
REACTION	<false>	Calculate internal forces or reactions
POST	<true>	Perform postprocessing (selected data archival)

4.11.3 ARGUMENT DESCRIPTIONS

4.11.3.1 AUTO_DOF_SUP

Automatic degree of freedom suppression flag (default: <true>). This option provides a convenient way of suppressing any freedoms that do not have any (or adequate) stiffness associated with them — for example, at nodes used to prescribe geometry only; or drilling freedoms in fine meshes composed of elements without normal rotational stiffnesses (see argument DRILLING_DOF).

4.11.3.2 DBC

Name of computational database file (default=PC.DBC). This file will contain all model definition data, element computational data, one copy of the assembled and factored stiffness matrices, the buckling eigensolution, and displacement and internal force vectors for every load step computed during the analysis.

4.11.3.3 DBR

Name of results database file (default=PC.DBR). This file will contain one dataset called RESPONSE.HISTORY generated during nonlinear analysis. The dataset will contain record groups — indexed by load step number — for a number of solution parameters, including the load factor and maximum axial displacement components. This database is valuable for obtaining load-displacement plots, and for evaluating the performance of the nonlinear solution strategy employed.

4.11.3.4 DRILLING_DOF

Drilling degree of freedom flag (default: <false>). Drilling freedoms are defined as rotations normal to the surface of the shell. Leaving this flag off forces all drilling freedoms in the model to be suppressed. Turning it on forces all drilling freedoms to be active — unless they are automatically suppressed using use of the AUTO_DOF_SUP argument. Note that while many shell elements do not have any rotational stiffness associated with their own surface-normal directions (at nodes), when shell elements are assembled as facets approximating an arbitrary shell surface, there is usually some misalignment between the element normal and the actual shell normal. This is especially true of "flat" (*e.g.*, 4-node) elements. Hence, *some* rotational stiffness about the *shell* normal is usually present in the model. (A clear exception to this is a flat plate, where element and shell normals are identical.) For a cylindrical shell, the misalignment diminishes only as the number of elements is increased. Most shell elements in the Testbed have their own misalignment tolerance parameter, which determines when the AUTO_DOF_SUP argument will automatically suppress the drilling freedom. Note that for elements which *have* drilling stiffness, the DRILLING_DOF argument should be set to <true> regardless of how AUTO_DOF_SUP is set.

4.11.3.5 ES_NAME

Element name (default: **EX97**). This argument is the name of the specific shell-element type you wish to select, within the element processor defined by argument **ES_PROC**. The default shell-element type, **EX97**, is a 9-noded quadrilateral element implemented in processor ES1, and described in The Computational Structural Mechanics Testbed User's Manual (see ref. 2.1-1).

4.11.3.6 ES_PARS

Element research parameters (default: 0., ...). This argument allows is an optional list of element-dependent parameters that some elements provide, primarily when the element is still undergoing research and refinement.

4.11.3.7 ES_PROC

Element processor (default: ES1) This argument is the name of the structural element (ES) processor that contains the shell element type you wish to employ in the model. The default shell-element, Processor ES1, is described in The Computational Structural Mechanics Testbed User's Manual.

4.11.3.8 LINEAR

Linear stress analysis flag (default: <true>).

4.11.3.9 NNODES_A

Number of axial nodes (default: 7). This argument is the number of nodes you wish to have along the axial direction of the cylindrical shell model, *i.e.*, along one-tenth of the full cylinder's length. Note that this number should be consistent with the number of nodes per element. For example, **NNODES_A** can be any number greater than 1 for 4-node quadrilateral elements, whereas it must be an odd number greater than 1 for 9-node quadrilateral elements.

4.11.3.10 NNODES_C

Number of circumferential nodes (default: 7). This argument is the number of nodes you wish to have along the circumferential direction of the cylindrical shell model, *i.e.*, along 15 degrees of circular arclength. Note that this number should be consistent with the number of nodes per element. For example, **NNODES_C** can be any number greater than 1 for 4-node quadrilateral elements, whereas it must be an odd number greater than 1 for 9-node quadrilateral elements.

4.11.3.11 POST

Postprocessing flag (default=<false>). This flag should be turned on if you want selected response-history parameters to be added to the CC.DBR database. Note that it is not necessary to use this option in order to archive the basic load-displacement curve and solution parameters. It is only needed if you wish to archive special displacement and/or internal force component response histories post-facto.

4.11.3.12 PREP

Pre-processing flag (default=<true>). This flag must be turned on the first time procedure PINCHED_CYL is run, as it causes the model to be generated. If subsequent runs are used to perform other stages of the analysis (*e.g.*, nonlinear restarts), then PREP must be set to <false> for those subsequent runs.

4.11.4 USAGE GUIDELINES AND EXAMPLES

Procedure PINCHED_CYL may be used by preceding the procedure name by the *call directive. Procedure arguments may be changed from their default values by including any or all of the arguments and their new values when the procedure is called. A space or blank is required between the end of the procedure name and the left parenthesis.

```
*call PINCHED_CYL ( ES_PROC   = ES1        ; ES_NAME = EX97 ; --
                    NNODES_A  =  7         ; NNODES_C =  7   ; --
                    SPEC_DIS  =    <false> ; --
                    DRILLING_DOF = <false> ; --
                    AUTO_DOF_SUP = <true>  ; --
                    PREP      =    <true>  ; --
                    STABILITY =    <true>  ; --
                    IMPERFECTION = <true> ; --
                    LINEAR =     <true> ; --
                    POST      =    <true> ; --
                    BEG_STEP  =  1         ; --
                    NSTEPS    = 10         ; --
                    BEG_LOAD  =   .1       ; --
                    MAX_LOAD  =  3.0       ; --
                    DBc       = PC.DBc     ; DBr     = PC.DBr )
```

(E1) To perform an entire analysis using the default options, simply invoke the procedure without any arguments, *i.e.*,

```
*call PINCHED_CYL
```

This will perform linear buckling eigenvalue analysis and 10 steps of nonlinear analysis with a 7×7 grid of ES1/EX97 shell elements. The time required for this analysis is machine-dependent. Using the default values for the procedure arguments, the amount of CPU time required for this analysis on various computer systems is shown in Table 4.11-1.

Table 4.11-1 CPU TIMES Table

Computer System	Total CPU Time, seconds.
VAX 11/785 VMS 4.7	
MicroVAX ULTRIX 2.2	
SUN SUNVIEW 4.0	
CONVEX C220 VERSION 7.0	
CRAY-2 UNICOS 4.0	

(E2) Not suppressing the drilling rotational freedoms can cause strange behavior for some elements during nonlinear analysis. On the other hand, suppressing these freedoms explicitly using the DRILLING_DOF argument may cause some over-stiffening for coarse meshes with some elements. It is probably best to suppress the drilling freedoms explicitly unless the element actually has intrinsic drilling stiffness.

4.11.5 LIMITATIONS

4.11.6 ERROR MESSAGES AND WARNINGS

None.

4.11.7 PROCEDURE FLOWCHART

```
PINCHED_CYL          (main procedure)
    GEN_SHELL        (generate model)
        PC_BC        (generate boundary conditions/loads)
        PDC_BC
    L_STATIC_1       (perform linear static analysis)
    POST
```

4.11.8 PROCEDURE LISTING

4.11.9 REFERENCES

4.11-1 Timoshenko, S. and Woinowsky-Krieger, S., *Theory of Plates and Shells*, McGraw-Hill. New York, 1969.

4.11-2 Lindberg, G.M., Olson, M.D. and Cowper, G.R. "New Developments in the Finite Element Analysis of Shells", *Q. Bull. Div. Mech. Eng. and the National Aeronautical Establishment, National Research Council of Canada, vol. 4, 1969*, pp. xxx.

4.11-3 Ashwell, D.G.: "Strain Elements, with Applications to Arches, Rings, and Cylindrical Shells," *Finite Elements for Thin Shells and Curved Members*, D.G. Ashwell and R.H. Gallagher (editors), John Wiley and Sons, New York, 1974, pp. 91-111.

4.11-4 Forsberg, Kevin: "An Evaluation of Finite Difference and Finite Element Techniques for Analysis of General Shells," In Proceedings of the Symposium on High Speed Computing of Elastic Structures, B. Fraeijs de Veubeke (editor), I.U.T.A.M., Liège, 1970, pp. 837-859.

4.11-5 Stewart, Caroline B.: *The Computational Structural Mechanics Testbed User's Manual.* NASA TM-100644, October 1989.

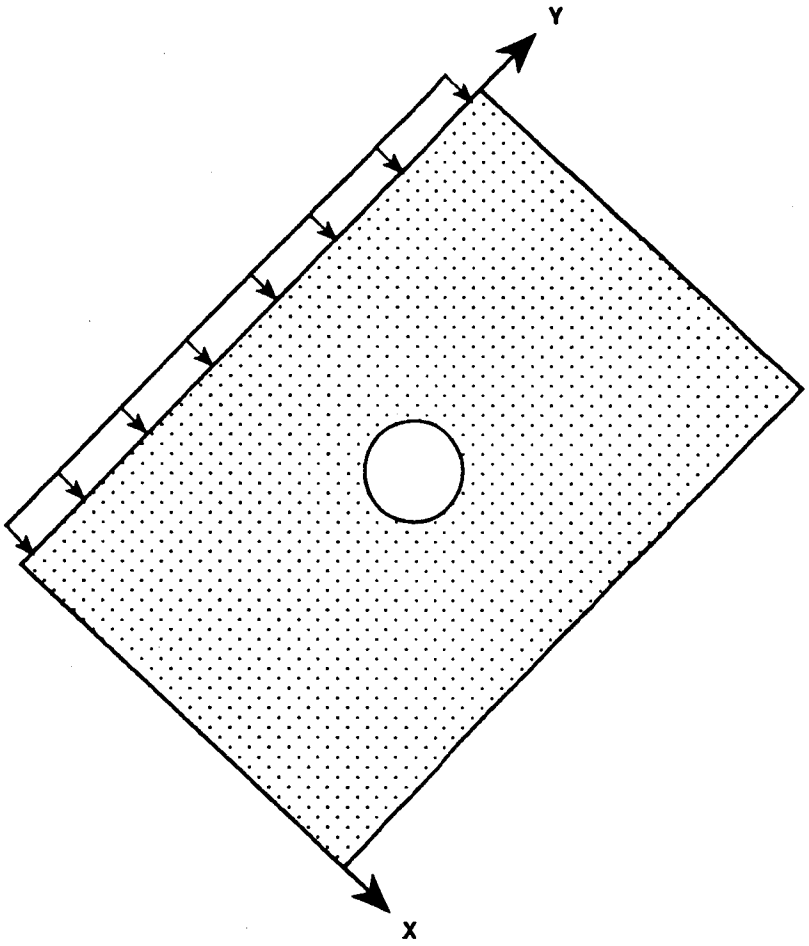
THIS PAGE LEFT BLANK INTENTIONALLY.

4.12 Procedure PWHOLE

4.12.1 GENERAL DESCRIPTION

4.12.1.1 Problem Description

This application problem involves a linear static stress analysis of a thin isotropic membrane with a central circular hole subjected to uniform compression (see Figure 4.12-1). This problem is important in that plate and shell elements are frequently used to model regions with a severe stress gradient.



Length = 20. inches
Width = 10. inches
Diameter of hole = 2. inches

Plate thickness = .1 inches
E = 10. x 10^6 psi
ν = .3
Uniform end-shortening, u_0=.1 inches

Figure 4.12-1 Plate with Circular Hole

4.12.1.2 Model Description

Procedure **PWHOLE** models an entire rectangular membrane using 2-D quadrilateral elements. The mesh is generated using processor CSM1. This processor generates a file named **PANEL.PRC** which is added to the procedure library. This file contains various procedures associated with model generation (**PANEL_BC**), and applied displacements (**PANEL_AD**).

4.12.1.3 Analysis Description

Procedure **PWHOLE** performs a single linear static stress analysis. The solution procedure **L_STATIC** described in Chapter 3 is used to perform the static stress.

4.12.1.4 Available Solutions

4.12.2 PROCEDURE USAGE

Procedure **PWHOLE** may be used by preceding the procedure name by the ***call** directive, and following it by a list of arguments enclosed in parentheses. Procedure arguments are order-independent, and most have default values thus making them optional. The formal syntax is as follows:

```
*call PWHOLE ( arg1 = val1 ; arg2 = val2 ; ...)
```

where **arg1** and **arg2** represent argument names, and **val1** and **val2** represent their corresponding values. Note that semi-colons are required between arguments, and a double dash (--) may be used to continue the argument list on the next line.

The allowable arguments for procedure **PWHOLE** are summarized in the following table, along with their default values (if they exist). Note that arguments without defaults are generally *mandatory*, while those with defaults are generally optional. Exceptions to this rule are noted in the following section under detailed argument descriptions.

Argument	Default Value	Meaning
ES_PROC	ES1	Element processor
ES_NAME	EX47	Element name
ES_PAR	0.0	Element research parameters
NRINGS	4	Number of rings of elements
NSPOKES	16	Number of radial spokes of nodes
NELX	6	Number of elements along x-direction
NELE	2	Number of elements across edge
NELBS	2	Number of elements between interior stiffeners
PRINT	<false>	Print flag
DIRECTION	1	Direction of stress reference frame
LOCATION	'NODES'	Location for stress recovery

4.12.3 ARGUMENT DESCRIPTIONS

4.12.3.1 DIRECTION

Direction for the element stress (stress resultant) output (default: 1). The element stress coordinate system will be used if DIRECTION=0. The material axes (x_m, y_m, z_m) will be used if DIRECTION=1; the material axes (y_m, z_m, x_m) will be used for DIRECTION=2; and the material axes (z_m, x_m, y_m) will be used for DIRECTION=3. For isotropic materials, the first material axis is replaced by the corresponding global axis (see Section 4.3.3.9 of the CSM Testbed User's Manual, ref. 4.12-3).

4.12.3.2 ES_NAME

Element name (default: EX47). This argument specifies the name of the specific shell-element type to select within the element processor defined by argument ES_PROC. The default shell-element type, EX47, is a 4-node quadrilateral element implemented in processor ES1, and described in the Computational Structural Mechanics Testbed User's Manual (see ref. 4.12-2).

4.12.3.3 ES_PARS

Element research parameters (default: 0.0, ...). This array allows an optional list of element-dependent parameters that some elements provide, primarily when the element is still undergoing research and refinement.

4.12.3.4 ES_PROC

Element processor (default: ES1) This is the name of the structural element (ES) processor that contains the shell-element type you wish to employ in the model. The default shell-element, processor ES1, is described in the Computational Structural Mechanics Testbed User's Manual (see refs. 4.12-2).

4.12.3.5 LOCATION

Location of the evaluation points for the element stresses or stress resultants (default: NODES). The element stresses or stress resultants are optionally computed by calling procedure STRESS (see Section 6.4). This argument may have four values. For LOCATION=INTEG_PTS, the element stresses are computed at the element integration points. For LOCATION=CENTROIDS, the element stresses are computed at the element centroid. For LOCATION=NODES, the element stresses are extrapolated from the integration points to be element nodes. These element nodal stresses will be discontinuous across interelement boundaries.

4.12.3.6 NELBS

Number of elements between interior stiffness (default: 2). Refer to processor CSM1 in reference 4.12-3 for a discussion of this argument.

4.12.3.7 NELE

Number of elements across edge (default: 2). Refer to processor CSM1 in reference 4.12-3 for a discussion of this argument.

4.12.3.8 NELX

Number of elements along x-direction (default: 6). Refer to processor CSM1 in reference 4.12-3 for a discussion of this argument.

4.12.3.9 NRINGS

Number of rings of elements (default: 4). This argument specifies the number of rings of elements, either 4-node or 9-node quadrilateral elements, around the hole.

4.12.3.10 NSPOKES

Number of radial spokes of nodes (default: 16). This argument specifies the number of radial spokes of nodes which must be a multiple of 8.

4.12.3.11 PRINT

Print flag (default: <false>). If the argument PRINT is defined to be <true>, then all computed results (displacements, stresses) will be printed.

4.12.4 USAGE GUIDELINES AND EXAMPLES

Procedure PWHOLE may be used by preceding the procedure name by the *call directive. Procedure arguments may be changed from their default values by including any or all of the arguments and their new values when the procedure is called. A space or blank is required between the end of the procedure name and the left parenthesis.

```
*call PWHOLE ( ES_PROC = ES1 ; -- . ES_NAME = EX47 ; -- .
      ES_PAR  = 0.0 ; -- .  Element research parameters
      NRINGS  = 4   ; -- .  Number of rings of elements
      NSPOKES = 16  ; -- .  Number of radial spokes of nodes
      NELX    = 6   ; -- .  Number of elements along x-direction
      NELE    = 2   ; -- .  Number of elements across edge
      NELBS   = 2   ; -- .  Number of elements between interior stiffeners
      DIRECTION = 1 ; -- .  Direction of stress reference frame
      LOCATION  = 'NODES'  -- .  Location for stress recovery
                         )
```

(E1) To perform an entire analysis using the default options, simply invoke the procedure without any arguments, that is,

```
*call PWHOLE
```

Using the default values for the procedure arguments, the amount of CPU time required for this analysis on various computer systems is shown in Table 4.12-1.

Table 4.12-1 CPU TIMES Table

Computer System	Total CPU Time, seconds.
VAX 11/785 VMS 4.7	
MicroVAX ULTRIX 2.2	
SUN SUNVIEW 4.0	
CONVEX C220 VERSION 7.0	
CRAY-2 UNICOS 4.0	

4.12.5 LIMITATIONS

None

4.12.6 ERROR MESSAGES AND WARNINGS

None.

4.12.7 PROCEDURE FLOWCHART

PWHOLE	(main procedure)
ES	(define elements)
PANEL_START	(define start card)
PANEL_JLOC	(define joint locations)
PANEL_BC	(define boundary conditions)
MATDAT	(define material properties)
PANEL_CONN	(define element connectivity)
ES	(define freedoms)
PANEL_AD	(define applied displacements)
L_STATIC	(linear static solution procedure)

4.12.8 PROCEDURE LISTING

4.12.9 REFERENCES

4.12-1 Timoshenko, S. P. and Goodier, J. N.: *Theory of Elasticity (Third Edition).* McGraw-Hill Book Company. New York, 1970, pp. 90-97.

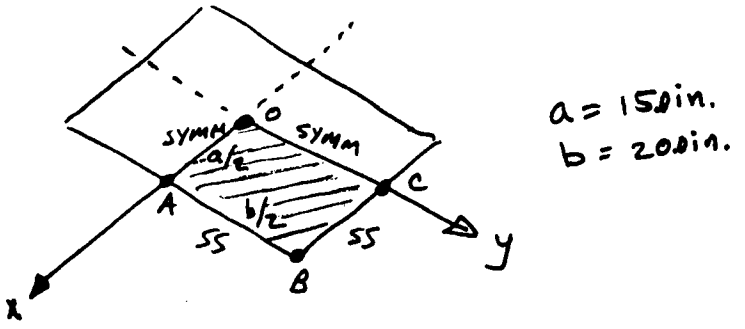
4.12-2 Peterson, R. E.: *Stress Concentration Design Factors.* John Wiley and Sons-International, New York, 1953, pp. 77-88.

4.12-3 Stewart, Caroline B.: *The Computational Structural Mechanics Testbed User's Manual.* NASA TM-100644, October 1989.

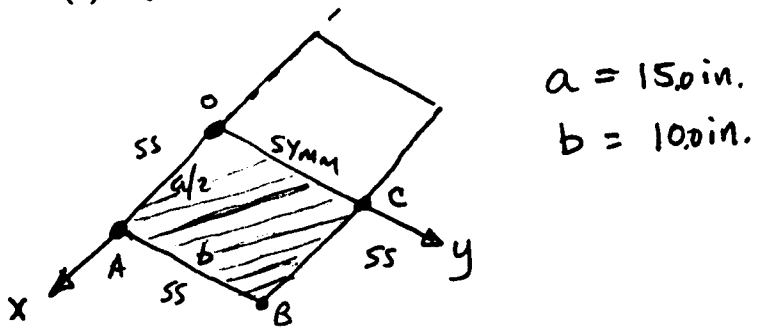
4.13 Procedure RECT_PLATE

4.13.1 GENERAL DESCRIPTION

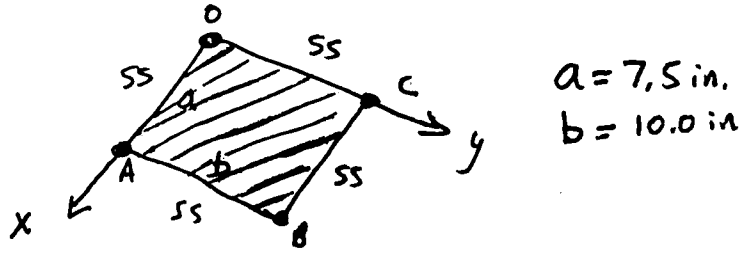
This application problem involves linear static stress analysis, linear buckling analysis and linear vibration analysis of three thin isotropic rectangular plates simply supported along their edges (see Figure 4.13-1). Various loading cases are considered including uniaxial and biaxial compression, uniform inplane shear, and inplane bending (see Figure 4.13-2).



(a) Quarter Model of Plate 1.

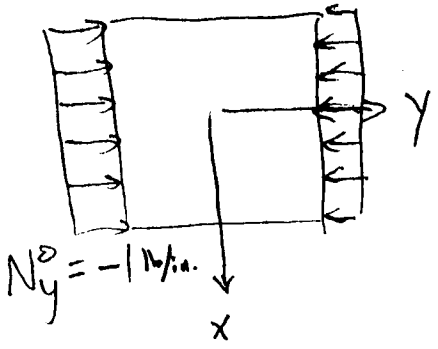


(b) Half Model of Plate 2.

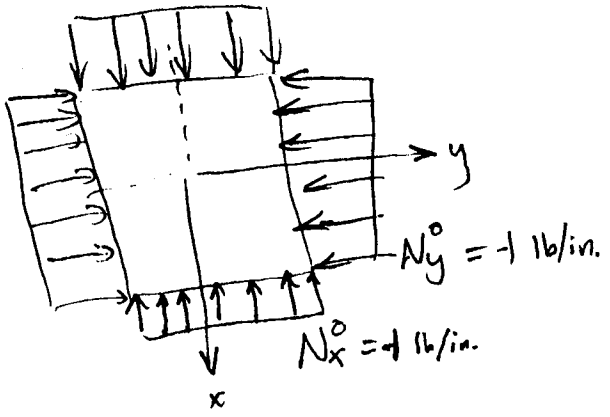


(c) Full Model of Plate 3.

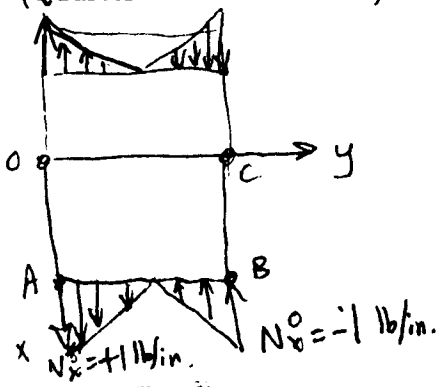
Figure 4.13-1 Rectangular Plate Problems.



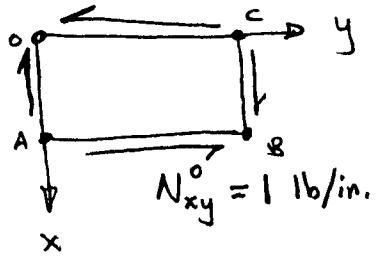
(a) Uniaxial Compression
(Quarter Model of Plate 1)



(b) Biaxial Compression
(Quarter Model of Plate 1)



(c) Inplane Bending
(Half Model of Plate 2)



(d) Inplane Shear
(Full Model of Plate 3)

Figure 4.13-2 Loading Conditions.

4.13.1.1 Model Description

Procedure **RECT_PLATE** uses a quarter model, a half model, and a full model of a plate depending on the loading case. The mesh topology is rectangular and restricted to quadrilateral shell elements with 4-nodes. For the quarter model shown in Figure 4.13-1a, two edges are simply supported and two edges have symmetry conditions imposed. For the half model shown in Figure 4.13-1b, three edges are simply supported and symmetry conditions are imposed on the remaining edge. For the full model shown in Figure 4.13-1c, all four edges are simply supported.

4.13.1.2 Analysis Description

Procedure **RECT_PLATE** performs six different analyses through successive procedure calls using the various solution procedures described in Chapter 3. The analyses performed are:

- Linear free vibration analysis about an unstressed state;
- Linear buckling analysis using a linearly-computed prestress state for uniform uniaxial compression;

- Linear buckling analysis using a prescribed prestress state for uniform biaxial compression;
- Linear free vibration analysis about a prescribed prestressed state for uniform biaxial tension;
- Linear buckling analysis using a linearly-computed prestress state for uniform inplane bending;
- Linear buckling analysis using a linearly-computed prestress state for uniform inplane shear.

4.13.1.3 Available Solution

Analytical solutions for the vibration analyses are available in reference 4.13-1 and for the buckling analyses in reference 4.13-2.

Case 1: Free vibration

For this analysis, the quarter model of plate 1 is used. From pages 492-494 of reference 4.13-1, the frequencies for vibration are given as

$$\omega_{mn} = \frac{\pi^2}{a^2}\sqrt{\frac{D}{\rho h}}\left(m^2 + n^2\ \frac{a^2}{b^2}\right) \qquad (4.13-1)$$

where

ω_{mn} = circular frequency (radians/second)

ρ = mass density

$D = \dfrac{Eh^3}{12(1-\nu^2)}$

h = plate thickness

E = Young's modulus

ν = Poisson's ratio

m, n = number of half-waves in the $x-$, $y-$ directions, respectively

and

$$f_{mn} = \frac{\omega_{mn}}{2\pi} \quad \text{(hertz)}$$

Using this formula, frequencies for this problem may be computed as follows:

m	n	ω_{mn}^2	f_{mn}
1	1	4.9867×10^5	112.4
1	3	75.0712×10^5	436.1
3	1	186.7724×10^5	687.8

Case 2: Uniform Uniaxial Compression

For this analysis, the quarter model of plate 1 is used and the loading corresponds to $N_y = -1.0$ lb/in. From pages 351-356 of reference 4.13-2, the buckling load is given as

$$(N_y)_{cr} = \frac{\pi^2 D}{b^2}\left(n + \frac{1}{n}\frac{b^2}{a^2}\right)^2 \qquad (4.13-2)$$

This expression implies that a plate buckles in such a way that there are n halfwaves in the direction of compression and only one halfwave in the perpendicular direction. The first term in equation (4.13-2) represents the Euler load for a strip of unit width and of length b. The second term represents the change in buckling load for a continuous plate from an isolated strip. For this plate,

n	$(N_y)_{cr}$, lb/in.
1	523.04
3	874.89
5	1944.23

Case 3: Uniform Biaxial Compression

For this analysis, the quarter model of plate 1 is used and the loading corresponds to uniform biaxial compression

$$N_x^o = N_y^o = -1.0 \text{ lb/in.}$$

From pages 356-360 of reference 4.13-2, the buckling load may be derived from

$$N_x \frac{m^2\pi^2}{a^2} + N_y \frac{n^2\pi^2}{b^2} = D\left(\frac{m^2\pi^2}{a^2} + \frac{n^2\pi^2}{b^2}\right)^2 \qquad (4.13-3)$$

If N_x and N_y are proportional, then

$$N_y = RN_x$$

where R is constant. Substituting this relation into equation (4.13-3) and solving for critical values of N_x gives

$$(N_x)_{cr} = \frac{\pi^2 D}{a^2}\frac{\left(m^2 + n^2\frac{a^2}{b^2}\right)^2}{\left(m^2 + Rn^2\frac{a^2}{b^2}\right)} \qquad (4.13-4)$$

The value of R is 1 for the loading case being considered. Thus

$$(N_x)_{cr} = \frac{\pi^2 D}{a^2}\left(m^2 + n^2\frac{a^2}{b^2}\right) \qquad (4.13-5)$$

The critical values are therefore

m	n	$(N_x)_{cr}$, $lb/in.$
1	1	188.29
1	3	730.58
3	1	1152.36
1	5	1815.15
5	1	3080.48

Case 4: Vibration with Prestress State

For this analysis, the quarter model of plate 1 is used and the prestress state corresponds to uniform biaxial tension.

$$N_x^o = N_y^o = S = 1000 \text{ lb/in.}$$

From pages 481-483 and 492-494 of reference 4.13-1, the vibration frequencies for a prestressed rectangular plate may be derived as

$$\omega_{mn}^2 = \frac{\pi^4 D}{a^4 \rho h}\left(m^2 + n^2\frac{a^2}{b^2}\right)^2 + \frac{\pi^2 S}{a^2 \rho h}\left(m^2 + n^2\frac{a^2}{b^2}\right) \qquad (4.13-6)$$

The first term of equation 4.13-6 represents the free vibration frequency. The second term represents the effect of uniform prestress on the vibration frequencies. Tensile prestress forces increases the vibration frequencies while compressive prestress forces decrease the vibration frequencies. For the plate considered herein, the vibration frequencies for a tensile prestress force S of 1000 lb/in. are as follows:

m	n	ω_{mn}^2	f_{mn}
1	1	3.1470×10^6	292.34
1	3	17.7827×10^6	671.15
3	1	34.8851×10^6	940.03

Case 5: Inplane Bending

For this analysis, the half model of plate 2 is used and the loading across the edge AB varies linearly from $N_x^o = 1.0$ lb/in. at point A to $N_x^o = -1.0$ lb/in. at point B. From pages 373-379 of reference 4.13-2, the buckling load is given as

$$(N_x)_{cr} = k\frac{\pi^2 D}{b^2} \qquad (4.13-7)$$

where the value of k depends on the ratio $\frac{a}{b}$ and on the linear distribution of the loading. Specific values of k are given in Table 9-6 on page 337 of reference 4.13-2. For the plate considered herein, the value of k is 24.1 which then gives a critical load of 6534.54 lb/in.

Uniform Inplane Shear

For this analysis, the full model of plate 3 is used and the loading corresponds to

$$N_{xy}^{o} = 1.0 \text{ lb/in.}$$

From pages 379-385 of reference 4.13-2, the buckling load is given as

$$(N_{xy})_{cr} = k\frac{\pi^2 D}{a^2} \qquad (4.13-8)$$

where the value of k depends on the ratio of $\frac{b}{a}$ and is given in Table 9-10 on page 382 of reference 4.13-2. For the plate considered herein, the value of k is 7.53 which then gives a critical load of 3629.69 lb/in.

4.13.2 PROCEDURE USAGE

Procedure **RECT_PLATE** may be used by preceding the procedure name by the ***call** directive, and following it by a list of arguments enclosed in parentheses. Procedure arguments are order-independent, and most have default values thus making them optional. The formal syntax is as follows:

```
*call RECT_PLATE ( arg1 = val1 ; arg2 = val2 ; ...)
```

where **arg1** and **arg2** represent argument names, and **val1** and **val2** represent their corresponding values. Note that semi-colons are required between arguments, and a double dash (--) may be used to continue the argument list on the next line.

The allowable arguments for procedure **RECT_PLATE** are summarized in the following table, along with their default values (if they exist). Note that arguments without defaults are generally *mandatory*, while those with defaults are generally optional. Exceptions to this rule are noted in the following section under detailed argument descriptions.

Argument	Default Value	Meaning
CASE	0	Select analysis case
E	30.E6	Young's Modulus
ES_PROC	ES1	Element Processor
ES_NAME	EX47	Element name
LX	7.5	Length in x-direction
LY	10.0	Length in y-direction
NU	0.3	Poisson's ratio
PRINT	<false>	Print flag
THICKNESS	0.1	Thickness
WTDEN	0.1	Weight density

4.13.3 ARGUMENT DESCRIPTIONS

4.13.3.1 CASE

Select analysis case (default: 0). This argument is used to select a specified analysis case number from one to six. The procedure will then perform only that specfic analysis. If all six analysis cases are desired, then the argument **CASE** should be set to zero.

4.13.3.2 E

Young's elastic modulus (default: 30.0×10^6 psi).

4.13.3.3 ES_NAME

Element name (default: **EX47**). This is the name of the specific shell-element type you wish to select, within the element processor defined by argument **ES_PROC**. The default shell-element type, **EX47**, is a 4-noded quadrilateral element implemented in Processor **ES1**, and described in The Computational Structural Mechanics Testbed User's Manual (see ref. 4.13-3).

4.13.3.4 ES_PROC

Element Processor (default: **ES1**) This is the name of the structural element (ES) Processor that contains the shell element type you wish to employ in the model. The default shell-element, Processor **ES1**, is described in The Computational Structural Mechanics Testbed User's Manual.

4.13.3.5 LX

Length of the plate model in the $x-$direction (default: 7.5 inches).

4.13.3.6 LY

Length of the plate model in the $y-$direction (default: 10.0 inches).

4.13.3.7 NU

Poisson's ratio (default: 0.3).

4.13.3.8 PRINT

Print flag (default: **<false>**). If the argument **PRINT** is defined to be **<true>**, then all computed results (displacements, modeshapes, stresses) will be printed.

4.13.3.9 THICKNESS

Thickness of the plate (default: 0.1 inches).

4.13.3.10 WTDEN

Weight density (default: 0.1 lb/in.^{3}). Processor LAU converts the weight density to mass density.

4.13.4 USAGE GUIDELINES AND EXAMPLES

Procedure **RECT_PLATE** may be used by preceding the procedure name by the ***call** directive. Procedure arguments may be changed from their default values by including any or all of the arguments and their new values when the procedure is called. A space or blank is required between the end of the procedure name and the left parenthesis.

```
*call RECT_PLATE ( ES_PROC  = ES1        ; ES_NAME = EX47 ; --
                   CASE = 0 ; --
                   LX = 7.5 ; --
                   LY = 10.0 ; --
                   E = 30.E6 ; --
                   NU = 0.3 ; --
                   PRINT = <false> ; --
                   THICKNESS = 0.1 ; --
                   WTDEN = 0.1 )
```

(E1) To perform an entire analysis using the default options, simply invoke the procedure without any arguments, that is,

```
*call RECT_PLATE
```

Using the default values for the procedure arguments, the amount of CPU time required for this analysis on various computer systems is shown in Table 4.13-1.

Table 4.13-1 Typical CPU Times for Various Computer Systems

Computer System	Total CPU Time, seconds.
VAX 11/785 VMS 4.7	
MicroVAX ULTRIX 2.2	
SUN SUNVIEW 4.0	
CONVEX C220 VERSION 7.0	
CRAY-2 UNICOS 4.0	

4.13.5 LIMITATIONS

The finite element model is restricted to models using 4-node quadrilateral elements. The model has a total of 54 nodes and 40 elements (5 elements in the x-direction and 8 elements in the y-direction).

4.13.6 ERROR MESSAGES AND WARNINGS

None.

4.13.7 PROCEDURE FLOWCHART

RECT_PLATE	(main procedure)
ES	(define elements)
L_VIBRAT_0	(free vibration analysis; CASE=1)
L_STABIL_2	(buckling analysis with uniaxial compression; CASE=2)
L_STABIL_1	(buckling analysis with biaxial compression; CASE=3)
L_VIBRAT_1	(vibration with prestress; CASE=4)
L_STABIL_2	(buckling with inplane bending; CASE=5)
L_STABIL_1	(buckling with inplane shear; CASE=6)

4.13.8 PROCEDURE LISTING

4.13.8.1 UNIX Script

rect_plate.com

```
cd /scr/$USER
cp $CSM_PRC/proclib.gal proclib.gal
chmod u+w proclib.gal
rm rect_plate.101
time testbed << \endinput
*set echo off
*set plib=28
*open 28 proclib.gal /old
*add '$CSM_APP/rect_plate/rect_plate.clp'
*open 1 rect_plate.101 /new
*def/a solver_name = BAND
*def/a eigensolver_name = LANZ
.
.
.  Eigenvalue Problems (Buckling and Vibration)
.       for an Isotropic Rectangular Plate
.
```

```
*call RECT_PLATE ( case=2; es_name=ex47;es_proc=es7;print=<true>)
*stop
*close 28 /delete
\endinput
```

4.13.8.2 CLAMP Procedure

rect_plate.clp

```
*PROCEDURE RECT_PLATE ( case=0; es_name='ex47';es_proc='es1'; --
                        lx=7.5;ly=10.0;e=30.0e+6;nu=0.3; --
                        thickness=0.1;wtden=0.1;print=<false> )
.
.  Rectangular plate
.
*def/e g       = < [e] / <2. * <1.+[nu]> > >
*def/e alpha   = .1-4
.
[XQT TAB
 START 54,6
TITLE'RECTANGULAR PLATE PROBLEM
TEXT
'THE FOLLOWING PLATE PROBLEMS ARE SOLVED IN THIS RUN:
' 1 FREE VIBRATION OF A RECTANGULAR PLATE.
' 2 BUCKLING OF A RECTANGULAR PLATE, COMPRESSED IN ONE DIRECTION.
' 3 BUCKLING OF A RECTANGULAR PLATE, UNIFORM COMPRESSION.
' 4 FREE VIBRATION OF A PRE-STRESSED RECTANGULAR PLATE.
' 5 BUCKLING OF A RECTANGULAR PLATE, BENDING LOAD.
' 6 BUCKLING OF A RECTANGULAR PLATE, SHEAR LOAD.
'
JOINT LOCATIONS
  1 0 0 0 [lx] 0 0 6 1 9
  6 0 [ly] 0 [lx] [ly] 0
.
.         CONSTRAINT CASE 1:
.
CON CASE 1
SYMMETRY PLANE=1
SYMMETRY PLANE=2
ZERO 3 4: 6,48,6
ZERO 3 5: 49,53
ZERO 3: 54
.
.         CONSTRAINT CASE 2:
.
CON CASE 2
SYMMETRY PLANE=1
ZERO 2: 25
ZERO 3: 1,6: 12,54,6: 49,53
ZERO 4: 12,48,6
ZERO 5: 1,5: 49,53
.
.         CONSTRAINT CASE 3:
```

```
.
CONSTRAINT CASE 3
ZERO 1: 1,6: 54
ZERO 2: 1,49,6
ZERO 3: 1,6: 12,54,6: 7,49,6: 50,53
ZERO 4: 12,48,6: 7,43,6
ZERO 5: 2,5: 50 53
.
[XQT AUS
     TABLE(NI=16,NJ=1): OMB DATA 1 1
        I=1,2,3,4,5,6,7,8,9
        J=1: [e] [nu] [e] <g> <g> <g> <alpha> <alpha> [wtden]
     TABLE (NI=3,NJ=1,ITYPE=0): LAM OMB 1 1
        J=1 : 1 [thickness] 0.
[XQT LAU
stop
.
*call ES ( function = 'DEFINE ELEMENTS' ; es_proc = [es_proc]  ; --
           es_name = [es_name]        ; es_pars = 0.0  )
[XQT ELD
   <ES_EXPE_CMD>
     NSECT = 1
       1 2 8 7 1 5 8
.
.         LOAD CASE 1:
.
[XQT AUS
ALPHA: CASE TITLE 1
1'compressive forces in y-direction applied at y=10.
SYSVEC: APPLIED FORCES 1
I=2
JOINTS=49: 54: -.75
JOINTS=50,53: -1.5
.
.         LOAD CASE 2:
.
ALPHA: CASE TITLE 2
1'compressive forces in x- and y-directions applied at y=10., x=7.5
SYSVEC: APPLIED FORCES 2
I=2
JOINTS=49: 54: -.75
JOINTS=50,53: -1.50
I=1
JOINTS=6:54:-.625
JOINTS=12,48,6: -1.25
.
.         LOAD CASE 3:
.
ALPHA: CASE TITLE 3
1'bending forces applied at x=7.5
SYSVEC: APPLIED FORCES 3
I=1
JOINTS=6: .57292
```

```
JOINTS=12,24,6: .9375 .6250 .3125
JOINTS=36,48,6: -.3125 -.6250 -.9375
JOINTS=54: -.57292
.
.         LOAD CASE 4:
.
ALPHA: CASE TITLE 4
1'shear forces applied at x=7.5 and y=10.
SYSVEC: APPLIED FORCES 4
I=2
JOINTS=6: 54: .625
JOINTS=12,48,6: 1.250
I=1
JOINTS=49: 54: .750
JOINTS=50,53: 1.500
.
.         LOAD CASE 5:
.
ALPHA: CASE TITLE 5
1'tensile forces in x-direction applied at x=7.5
SYSVEC: APPLIED FORCES 5
I=1
JOINTS=6: 54: 1250.
JOINTS=12,48,6: 2500.
stop
.
.        PROCESS
.
*if < [case] /eq 0 >/then
.
*call L_VIBRAT_0 ( function='all';N_modes=6;print=[print];vcon_set=1 ; --
                    mass_type='consistent' )
.
*call L_STABIL_2 ( function='fact_solv';N_modes=2; --
                   cons_set=1; bcon_set=1; --
                   shift=0.0; stress=<true>; --
                     reaction=<true>;print=[print] )
.
*call L_STABIL_1 ( function='EIGEN'; bcon_set=1; --
                   N_modes=2; ps_1=-1.0;ps_2=-1.0;print=[print] )
.
*call L_VIBRAT_1 ( function='all';vcon_set=1; --
                   ps_1=1000.0;ps_2=1000.0;N_modes=6;print=[print] ; --
                    mass_type='consistent' )
.
*call L_STABIL_2 ( function='fact_solv';N_modes=2; --
                    cons_set=2;bcon_set=2;load_set=3; --
                    shift=0.0; stress=<true>; --
                      reaction=<true>;print=[print] )
.
*call L_STABIL_1 ( function='all';N_modes=2; --
                    ps_3=1.0;bcon_set=3; --
                    shift=0.0;  --
```

```
                    print=[print] )
.
*else
*if < [case] /eq 1 >/then
.
*call L_VIBRAT_0 ( function='all';N_modes=6;print=[print];vcon_set=1 ; --
                    mass_type='consistent' )
.
*elseif < [case] /eq 2 >/then
.
*call L_STABIL_2 ( function='all';N_modes=2; --
                   location='all';direction=1; --
                   cons_set=1; bcon_set=1; --
                   shift=0.0; stress=<true>; --
                      reaction=<true>;print=[print] )
.
*elseif < [case] /eq 3 >/then
.
*call L_STABIL_1 ( function='all'; bcon_set=1; --
                   N_modes=2; ps_1=-1.0;ps_2=-1.0;print=[print] )
.
*elseif < [case] /eq 4 >/then
.
*call L_VIBRAT_1 ( function='all';vcon_set=1; --
                   ps_1=1000.0;ps_2=1000.0;N_modes=6;print=[print] ; --
                    mass_type='consistent' )
.
*elseif < [case] /eq 5 >/then
.
*call L_STABIL_2 ( function='all';N_modes=2; --
                    cons_set=2;bcon_set=2;load_set=3; --
                    shift=0.0; stress=<true>; --
                      reaction=<true>;print=[print] )
.
*elseif < [case] /eq 6 >/then
.
*call L_STABIL_1 ( function='all';N_modes=2; --
                    ps_3=1.0;bcon_set=3; --
                    shift=0.0;  --
                    print=[print] )
.
*endif
*endif
.
[xqt dcu
 toc 1
.
*end
```

4.13.9 REFERENCES

4.13-1 Timoshenko, S., Young, D. H. and Weaver, W., Jr.: *Vibration Problems in Engineering*, Fourth Edition, John Wiley and Sons, Inc., New York, 1974.

4.13-2 Timoshenko, S. P.; and Gere, J. M.: *Theory of Elastic Stability*, Second Edition, McGraw-Hill, New York, 1961.

4.13-3 Stewart, Caroline B.: *The Computational Structural Mechanics Testbed User's Manual.* NASA TM-100644, October 1989.

4.14 Processor RHOMBIC_PLATE

THIS SECTION UNDER PREPARATION

THIS PAGE LEFT BLANK INTENTIONALLY.

4.15 Processor TRUNCATED_CONE

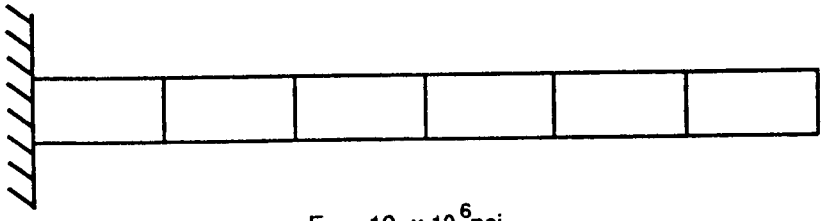
THIS SECTION UNDER PREPARATION

THIS PAGE LEFT BLANK INTENTIONALLY.

4.16 Procedure VIB_2D

4.16.1 GENERAL DESCRIPTION

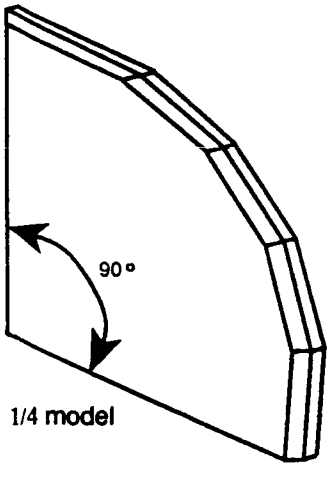
This application procedure involves linear free vibration analyses of a beam and a circular ring (see Figure 4.16-1). These one-dimensional structures are modeled using two-dimensional shell elements. Various boundary conditions are imposed to obtain vibration modes about different coordinate axes.



$E = 10. \times 10^6$ psi
$\nu = 0.$

Length = 10. inches
width = 1.0 inches
thickness = .1 inches

(a) Cantilevered Beam



$E = 10. \times 10^6$ psi
$\nu = 0.$

inner radius = 6.366 inches
outer radius = 7.366 inches
thickness = .1 inches

1/4 model

(b) Circular Ring

Figure 4.16-1 Free Vibration Problems

4.16.1.1 Model Description

Procedure VIB_2D uses a full model of the cantilevered beam and a quarter model of the circular ring. The mesh topology is rectangular and restricted to quadrilateral shell elements with 4- or 9-nodes. The mesh generation capability of processor ELD is used to generate the various meshes used in this application problem. The cantilevered beam shown in figure 4.16-1a is fixed at one end and free at the other. Only a quarter of the circular ring shown in figure 4.16-1b is modeled with symmetry conditions imposed at both ends.

4.16.1.2 Analysis Description

Procedure VIB_2D performs three different linear free vibration analyses through successive procedure calls using the solution procedure L_VIBRAT_0 described in Chapter 3.

4.16.1.3 Available Solution

Analytical solutions for the vibration analyses are available in reference 4.16-1.

Case 1: Free Inplane Vibration

For this analysis, all degrees of freedom in the z-direction have been constrained to zero so that only inplane vibration will occur. From page 369 of reference 4.16-1, the frequencies for longitudinal (axial) vibration are given as

$$\omega_m = \frac{m\pi}{2L}\sqrt{\frac{E}{\rho}} \qquad (4.16-1)$$

and the frequencies for inplane flexural vibrations are given on page 426 of references 4.16-1 as

$$\omega_m = k_m^2\sqrt{\frac{EI_{zz}}{\rho A}} \qquad (4.16-2)$$

where

ω_m = circular frequency (radians/second)

ρ = mass density

L = beam length

E = Young's modulus

I_{zz} = Moment of Inertia about z-axis, $\frac{1}{12}b^3h$

ν = Poisson's ratio

m = number of half-waves in the x-directions, respectively

and

$$f_m = \frac{\omega_m}{2\pi} \quad \text{(hertz)}$$

The constant k_m is obtained from the roots of the equation

$$cos\ k_m L\ cosh\ k_m L = -1 \qquad (4.16-3)$$

The consecutive roots of this equation are

m	$k_m L$
1	1.875
2	4.694
3	7.855
4	10.996

Using these formula, frequencies for this problem may be computed as follows:

m	ω_m^2	f_m	Type
1	4.776×10^5	109.990	Flexural
2	187.590×10^5	689.326	Flexural
1	1144.085×10^5	1702.351	Axial
3	1471.031×10^5	1930.328	Flexural
2	4576.338×10^5	3404.702	Axial
4	5649.057×10^5	3782.754	Flexural
3	10296.761×10^5	5107.054	Axial

Case 2: Free Lateral Vibration

For this analysis, only the degrees of freedom in the z-direction are unconstrained and hence lateral vibration mode shapes and frequencies will be obtained. From page 426 of reference 4.16-1, the frequencies are given as

$$\omega_m = k_m^2 \sqrt{\frac{EI_{yy}}{\rho A}} \qquad (4.16-4)$$

where

k_m = roots of equation 4.16-3

I_{yy} = moment of inertia about y-axis, $\frac{1}{12}bh^3$

Using this formula, the vibration frequencies for this problem may be computed as follows:

m	ω_m^2	f_m
1	4.776×10^3	10.999
2	187.590×10^3	68.933
3	1471.031×10^3	193.033
4	5649.057×10^3	378.275

Case 3: Free Vibration of Ring

For this analysis, a quarter model of the circular ring is used. The extensional vibration modes resemble the longitudinal vibration modes of prismatic beams. The simplest extensional mode of vibration corresponds to uniform radial motion of the ring (circles of periodically varying radius). From pages 476-477 of reference 4.16-1, the frequencies for the extensional vibration modes are given as

$$\omega_m = \sqrt{\frac{E(1+m^2)}{\rho R^2}} \qquad (4.16-5)$$

where R is the radius of centerline of ring.

Pure radial vibrations are obtained for the case of m = 0. Using this formula, the frequencies for extensional vibration may be computed as follows:

m	ω_m^2	f_m
0	1.1441×10^8	1702.351
1	2.2882×10^8	2407.488
2	5.7204×10^8	3806.573
3	11.4408×10^8	5383.308

The inplane flexural vibration of a circular ring corresponds to vibration modes in the plane of the ring. From pages 479-481 of reference 4.16-1, these frequencies are given as:

$$\omega_m = \sqrt{\frac{EI_{zz}m^2(1-m^2)^2}{\rho A\ R^4(1+m^2)}} \qquad (4.16-6)$$

For $m = 1$, the frequency is zero which implies that the ring moves as a rigid body. Using this formula, the frequencies for the inplane vibration modes may be computed as follows:

m	ω_m^2	f_m
1	0.0	0.0
2	169.3749	2.071
3	1354.9994	5.858
4	4981.6155	11.233
6	28038.4171	26.650
8	91931.4991	48.256
10	228278.8317	76.042

4.16.2 PROCEDURE USAGE

Procedure **VIB_2D** may be used by preceding the procedure name by the ***call** directive, and following it by a list of arguments enclosed in parentheses. Procedure arguments are order-independent, and most have default values thus making them optional. The formal syntax is as follows:

```
*call VIB_2D ( arg1 = val1 ; arg2 = val2 ; ...)
```

where **arg1** and **arg2** represent argument names, and **val1** and **val2** represent their corresponding values. Note that semi-colons are required between arguments, and a double dash (--) may be used to continue the argument list on the next line.

The allowable arguments for procedure **VIB_2D** are summarized in the following table, along with their default values (if they exist). Note that arguments without defaults are generally *mandatory*, while those with defaults are generally optional. Exceptions to this rule are noted in the following section under detailed argument descriptions.

Argument	Default Value	Meaning
CASE	BAR	Select vibration case
E	10.E6	Young's Modulus
ES_PROC	ES1	Element Processor
ES_NAME	EX97	Element name
GRIDS	1	Grid identification numbers
NGRIDS	1	Number of grids
NNX	3	Number of nodes for each grid
L	10.0	Length in x-direction
PRINT	<false>	Print flag
THICKNESS	0.1	Thickness
WTDEN	5/6	Weight density
B	1.0	Beam or ring depth
DBC	VIB_2D.DBC	Computational database name
DBR	VIB_2D.DBR	Results database name
SHIFT	0.0	Eigenvalue shift
MASS_TYPE	CONSISTENT	Type of mass matrix
N_MODES	5	Number of eigenvalues desired

4.16.3 ARGUMENT DESCRIPTIONS

4.16.3.1 CASE

Select vibration case (default: **BAR**). This argument selects the vibration analysis to perform. The options include **BAR**, **BEAM**, and **RING**. The **CASE=BAR**, the analysis corresponds to the axial vibration of a rod. For **CASE=BEAM**, the analysis corresponds to the lateral vibration of a beam. For **CASE=RING**, the analysis corresponds to extensional and flexural vibration of a circular ring.

4.16.3.2 DBC

Name of computational database file (default=VIB_2D.DBC). This file will contain all model definition data, element computational data, one copy of the assembled and factored stiffness matrices, the buckling eigensolution, and displacement and internal force vectors for every load step computed during the analysis.

4.16.3.3 DBR

Name of results database file (default=VIB_2D.DBR). This file will contain one dataset called RESPONSE.HISTORY generated during nonlinear analysis. The dataset will contain record groups — indexed by load step number — for a number of solution parameters, including the load factor and maximum axial displacement components. This database is valuable for obtaining load-displacement plots, and for evaluating the performance of the nonlinear solution strategy employed.

4.16.3.4 E

Young's elastic modulus (default: 10.0×10^6 psi).

4.16.3.5 ES_NAME

Element name (default: EX97). This argument gives the name of the specific shell-element type to select, within the element processor defined by argument ES_PROC. The default shell-element type, EX97, is a 9-node quadrilateral element implemented in processor ES1, and described in The Computational Structural Mechanics Testbed User's Manual (see ref. 4.16-2).

4.16.3.6 ES_PROC

Element processor (default: ES1) This argument gives the name of the structural element (ES) processor that contains the shell element type to employ in the model. The default shell-element, processor ES1, is described in The Computational Structural Mechanics Testbed User's Manual.

4.16.3.7 GRIDS

Grid identification numbers (default: 1). This array specifies the identification number for each discretization.

4.16.3.8 L

Length of the beam model in the x-direction (default: 10.0 inches). For the case of the circular ring, the radius is calculated as $2L/\pi$.

4.16.3.9 MASS_TYPE

Type of mass matrix (default: CONSISTENT). If MASS_TYPE = CONSISTENT, the element processor will generate consistent element mass matrices that will be assembled by processor K to form the system mass matrix. If MASS_TYPE = DIAGONAL, the element processor will generate a diagonal or lumped mass matrix.

4.16.3.10 NGRIDS

Number of grids (default: 1). This argument provides a mechanism to perform automatically a limited convergence study using several different discretizations.

4.16.3.11 N_MODES

Number of converged eigenvalues desired (default: 5). This argument specifies the number of eigenvalues to calculate to a convergence criterion of .0001.

4.16.3.12 NNX

Array of number of nodes for NGRIDS models (default: 3). This array is of length NGRIDS and represents the number of nodes along the length of the beam or along the circumference of the ring.

4.16.3.13 PRINT

Print flag (default: <false>). If the argument PRINT is defined to be <true>, then all computed results (displacements, modeshapes, stresses) will be printed.

4.16.3.14 SHIFT

Eigenvalue shift (default: 0.0). Converged eigenvalues will only be obtained for eigenvalues greater than SHIFT. The shift parameter refers to the frequency squared (ω^2) for vibration problems.

4.16.3.15 THICKNESS

Thickness of the beam (default: 0.1 inches). The ring thickness has a default value of one-tenth the beam thickness.

4.16.3.16 WTDEN

Weight density (default: 5/6 lb/in.^{3}). Processor LAU converts the weight density to mass density.

4.16.4 USAGE GUIDELINES AND EXAMPLES

Procedure VIB_2D may be used by preceding the procedure name by the *call directive. Procedure arguments may be changed from their default values by including any or all of the arguments and their new values when the procedure is called. A space or blank is required between the end of the procedure name and the left parenthesis.

```
*call VIB_2D ( ES_PROC  = ES1        ; ES_NAME = EX47 ; --
                     L = 10.0 ; --
                     E = 10.E6 ; --
                     B = 1.0 ; --
                     DBC = VIB_2D.DBC ; --
```

```
DBR = VIB_2D.DBR ; --
MASS_TYPE = CONSISTENT ; --
GRIDS = 1,2,3 ; --
NGRIDS = 3 ; --
NNX = 5,9,17 ; --
SHIFT = 0.0 ; --
N_MODES = 5 ; --
PRINT = <false> ; --
THICKNESS = 0.1 ; --
WTDEN = 0.1      )
```

(E1) To perform an entire analysis using the default options, simply invoke the procedure without any arguments, that is,

```
*call VIB_2D
```

Using the default values for the procedure arguments, the amount of CPU time required for this analysis on various computer systems is shown in Table 4.16-1.

Table 4.16-1 Typical CPU Times for Various Computer Systems

Computer System	Total CPU Time, seconds.
VAX 11/785 VMS 4.7	
MicroVAX ULTRIX 2.2	
SUN SUNVIEW 4.0	
CONVEX C220 VERSION 7.0	
CRAY-2 UNICOS 4.0	

4.16.5 LIMITATIONS

None.

4.16.6 ERROR MESSAGES AND WARNINGS

None.

4.16.7 PROCEDURE FLOWCHART

```
VIB_2D                  (main procedure)
    VIB_2D_CTL          (control procedure)
        VIB_2D_MOD      (define 2-D models of beam/ring)
            ES          (define elements)
            VIB_2D_CON  (define connectivities)
            VIB_2D_BCS  (define boundary conditions)
            ES          (define freedoms)
        L_VIBRAT_O      (linear free vibration solution procedure)
```

4.16.8 PROCEDURE LISTING

4.16.8.1 UNIX Script

vib_2d.com

```
cd /scr/$USER
cp $CSM_PRC/proclib.gal proclib.gal
chmod u+w proclib.gal
rm VIB*.*
time testbed << \endinput
*set echo off
*set plib=28
*open 28 proclib.gal /old
*add '$CSM_APP/vib_2d/vib_2d.clp'
.
*def/a solver_name = INV
*def/a eigensolver_name = EIG2
.
.  Axial Vibration of a Bar
.
*call VIB_2D ( case    = BAR; mass_type = CONSISTENT; --
                  ES_PROC=ES7       ; ES_NAME=EX97; --
                  L       = 10.0    ; thickness = .1 ; b = 1.0      ; --
                  E       = 1.e7    ; wtden   = <5./6.>     ; ---
                  N_modes = 5       ; print= <false>      ; --
                  shift=0.0     ; nnx=7,9,11; grids=1,2,3; ngrids=3; --
                  DBC=VIB_2D.DBC; DBR=VIB_2D.DBR )
.
.  Lateral Vibration of a Beam
.
*call VIB_2D ( case    = BEAM ; mass_type = CONSISTENT; --
                  ES_PROC=ES7       ; ES_NAME=EX97; --
                  L       = 10.0    ; thickness = .1 ; b = 1.0      ; --
                  E       = 1.e7    ; wtden   = <5./6.>     ; ---
                  N_modes = 5       ; print= <false>      ; --
                  shift=0.0     ; nnx=5,9,17; grids=1,2,3; ngrids=3; --
                  DBC=VIB_2D.DBC; DBR=VIB_2D.DBR )
.
.  Extensional and Flexural Vibrations of a Circular Ring
.
*call VIB_2D ( case    = RING ; mass_type = CONSISTENT; --
                  ES_PROC=ES7       ; ES_NAME=EX97; --
                  L       = 10.0    ; thickness = .1 ; b = 1.0      ; --
                  E       = 1.e7    ; wtden   = <5./6.>     ; ---
                  N_modes = 5       ; print= <false>      ; --
                  shift=0.0     ; nnx=5,9,17; grids=1,2,3; ngrids=3; --
                  DBC=VIB_2D.DBC; DBR=VIB_2D.DBR )
 *stop
 *close 28 /delete
\endinput
```

4.16.8.2 CLAMP Procedure

vib_2d.clp

```
*procedure VIB_2D_CTL ( case      = BAR        ; -- .  BAR | BEAM | RING
                  mass_type = CONSISTENT ; -- .  CONSISTENT | DIAGONAL
                  ES_PROC=ES1       ; ES_NAME=EX97; --
                  L       = 10.0    ; thickness = .1 ; b = 1.0     ; --
                  E       = 1.e7    ; wtden   = <5./6.>    ; ---
                  N_modes = 5       ; print= <false>     ; --
                  shift=0.0     ; nnx=5; grid=1; --
                  DBC=VIB_2D.DBC; DBR=VIB_2D.DBR )
      *open/new 1, [DBc]
      *call VIB_2D_MOD ( es_proc = [es_proc] ; es_name = [es_name]; --
                         nnx     = [nnx]     ; L = [L]; h=[thickness] ; --
                         E=[E]; b=[b]; rho=[wtden]; case=[case] )
. *show macros
. *set echo,on
      *call L_VIBRAT_0 ( N_modes=[N_modes]; print = [print]; --
                         shift=[shift]; --
                         mass_type=[mass_type] )
*end
*procedure VIB_2D_MOD ( es_proc=ES1; es_name=EX97; es_pars=0.; --
                        nnx      = 11   ; --
                        L        = 10.0 ; h = .1 ; --
                        E        = 1.e7 ; b = 1.0; --
                        rho      = <5./6.> ; case = BAR )
.
. ---------------------------------------------------
. Vibrations of a Cantilevered Beam via Plate Elements
. ---------------------------------------------------
.
.
. Define Model
.
   *call ES ( function = 'DEFINE ELEMENTS' ; es_proc = [es_proc]; --
              es_name  =  [es_name]        ; es_pars = [es_pars] )
   *def/i nny = < <ES_NEN>^.5 >
   [XQT TAB
   *def/i nnt = < [nnx]*<nny> >
    START <nnt>
    JLOC
.
. DEFINE NODAL COORDINATES
.
   *if < <IFELSE([case];RING;1;0)> > /then
. -----------------------------------------------------
. Define First Quadrant of a Cylinder of Unit Axial Length
. -----------------------------------------------------
      *def/e theta  = 90.
      *def/e radius = ([L]*2/<pi>)
      FORMAT = 2
      1,       <radius>, 0., 0.,   <radius>, <theta>, 0.,   [nnx], 1, <nny>
      [nnx],   <radius>, 0., 1.,   <radius>, <theta>, 1.
```

```
     JREF
        NREF = -1
           1, <nnt>
  *else
     1,       0.,   0., 0.,   [L],  0., 0.,        [nnx], 1, <nny>
     [nnx],   0.,  [b], 0.,   [L], [b], 0.
  *endif
.
. DEFINE MATERIAL/SECTION PROPERTIES
.
   [XQT AUS
.
. Build Table of Material Data
   TABLE(NI=16,NJ=1): OMB DATA 1 1
   *def/e12.4 NU = 0.0
   *def/g12.4 G  = < [E] / (2.*(1.+<NU>)) >
   I=1,2,3,4,5,6,7,8,9
   J=1: [E] <NU> [E]  <G> <G> <G>  0. 0. [rho]
.
. Build Laminate Data Tables
   TABLE(NI=3,NJ=1,ITYPE=0): LAM OMB 1 1
   I=1,2,3 . (material_type, layer_thickness, angle(deg.)
   J=1:          1             [h]            0.0
.
   [XQT LAU
   ONLINE=2
.
. GENERATE ELEMENTS
.
   [xqt ELD
     <ES_EXPE_CMD>
      NSECT = 1
.
.     Define element nodal connectivity
.
      *call VIB_2D_CONN ( nnx=[nnx]; nny=<nny>; nen=<es_nen> )
.
. DEFINE BOUNDARY CONDITIONS
.
      *call VIB_2D_BCS ( nnx = [nnx]     ; nny = <nny> ; --
                         nen = <es_nen> ; case = [case] )
      *call ES ( function = 'DEFINE FREEDOMS' )
.
*end
*procedure VIB_2D_CONN ( nnx; nny; nen )
.
.        ============================
. Define Element Connectivity Record for ELD Processor
.        ============================
.
   *if < [nen] /eq 4 > /then
      *do $iy = 1, <[nny]-1>
         *do $ix = 1, <[nnx]-1>
```

```
            *def/i n1 =  < (<$iy>-1)*[nnx] + <$ix> >
            *def/i n2 =  < <n1> +   1   >
            *def/i n3 =  < <n2> + [nnx] >
            *def/i n4 =  < <n3> -   1   >
.
.            ======================
             <n1>  <n2>  <n3>  <n4>
.            ======================
.
         *enddo
      *enddo
   *elseif < [nen] /eq 9 > /then
      *do $iy = 1, <[nny]-2>, 2
         *do $ix = 1, <[nnx]-2>, 2
            *def/i n1 =  < (<$iy>-1)*[nnx] + <$ix> >
            *def/i n2 =  < <n1> + 2 >
            *def/i n3 =  < <n2> + (2*[nnx]) >
            *def/i n4 =  < <n3> - 2 >
            *def/i n5 =  < <n1> + 1 >
            *def/i n6 =  < <n2> + [nnx] >
            *def/i n7 =  < <n4> + 1 >
            *def/i n8 =  < <n6> - 2 >
            *def/i n9 =  < <n8> + 1 >
.
.            ==========================================
             <n1> <n2> <n3> <n4> <n5> <n6> <n7> <n8> <n9>
.            ==========================================
.
         *enddo
      *enddo
   *endif
*end
*procedure VIB_2D_BCS ( nnx=3; nny=3; nen=9; case = BAR )
.
. Define Boundary Conditions for Beam/Plate Vibrations
.
*def/i nnt = < [nnx]*[nny] >
.
[xqt TAB
 CON 1
*if < <IFELSE([case];BAR;1;0)> > /then
. ------------------------------------------------------------
. DEFINE CLAMPED BOUNDARY CONDITIONS for LONGITUDINAL VIBRATIONS
. ------------------------------------------------------------
    ZERO 3, 4, 5  : 1, <nnt>       . Suppress Out-of-Plane DOFS Everywhere
    ZERO 1,2,3,4,5,6            . Clamp left boundary
   *do $n=1,<<nnt>-[nnx]+1>, [nnx]
      <$n>
   *enddo
*elseif < <IFELSE([case];BEAM;1;0)> > /then
. ------------------------------------------------------
. DEFINE CLAMPED BOUNDARY CONDITIONS for INPLANE VIBRATIONS
. ------------------------------------------------------
```

```
    ZERO 1, 2, 6: 1, <nnt>      . Suppress In-Plane DOFS Everywhere
    ZERO 1,2,3,4,5,6             . Clamp left boundary
   *do $n=1,<<nnt>-[nnx]+1>, [nnx]
       <$n>
   *enddo
*elseif < <IFELSE([case];RING;1;0)> > /then
. ----------------------------------------------------------------
. DEFINE BOUNDARY CONDITIONS for EXTENSIONAL and FLEXURAL VIBRATIONS
. ----------------------------------------------------------------
   ZERO 3, 4, 5:  1, <nnt>      . Suppress Out-of-Plane DOFS Everywhere
   SYMMETRY PLANE = 1           . Symmetry at theta = 90; 1/4 Model
   SYMMETRY PLANE = 2           . Symmetry at theta =  0
*endif
*end
*procedure VIB_2D ( case=BAR; mass_type=CONSISTENT;ngrids=1; grids=1; --
                    L       = 10.0    ; thickness = .1 ; b = 1.0      ; --
                    E       = 1.e7    ; wtden   = <5./6.>     ; ---
                    N_modes = 5       ; print= <false> ; nnx=3     ; --
                    shift=0.0     ; es_proc=ES1; es_name=EX97; --
                    DBC=VIB_2D.DBC; DBR=VIB_2D.DBR )
   *if < <IFELSE([CASE];BAR;1;0)> > /then
      *def/d thickness = [thickness]
   *elseif < <IFELSE([CASE];BEAM;1;0)> > /then
      *def/d thickness = [thickness]
   *elseif < <IFELSE([CASE];RING;1;0)> > /then
      *def/d thickness = < [thickness] / 10.0 >
   *endif
   *def/i grids[1:[ngrids]] = [grids]
   *def/i nnx[1:[ngrids]] = [nnx]
   *do $g = 1,[ngrids]
      *def/i grid = <grids[<$g>]>
      *def/i nnxs = <nnx[<$g>]>
      *call VIB_2D_CTL ( case     = [case]; mass_type = [mass_type] ; --
                         es_proc = [es_proc]; es_name = [es_name]  ; --
                         nnx = <nnxs> ; thickness = <thickness> ; --
                        L       = [L]    ; b= [b]  ; --
                        E       = [E]    ; wtden   = [wtden]     ; ---
                        N_modes = [N_modes]; print= [print]      ; --
                        shift=[shift]     ; --
                        DBC=[DBC]; DBR=[DBR] ; --
                         grid=<grid> )
   *enddo
*end
```

4.16.9 REFERENCES

4.16-1 Timoshenko, S., Young, D. H. and Weaver, W., Jr.: *Vibration Problems in Engineering*, Fourth Edition, John Wiley and Sons, Inc., New York, 1974.

4.16-2 Stewart, Caroline B.: *The Computational Structural Mechanics Testbed User's Manual.* NASA TM-100644, October 1989.

5.0 Element Assessment Procedures

The procedures documented in this chapter are representative of the types of procedures that may be written to solve specific application (structural analysis) problems. Many of these high-level procedures invoke other (lower-level) procedures to perform preprocessing, solution, and postprocessing functions; which are described elsewhere in this manual. The use of procedures to perform structural analysis applications can provide users flexibility for parameterizing geometric data (*e.g.*, stiffener spacing) as well as spatial discretization parameters (*e.g.*, number of elements). The problems represented here are also intended to serve as part of a standard series of test problems to assess new structural elements installed in the CSM Testbed.

A summary of the procedures found in this chapter is provided by Table 5.0-1.

Table 5.0-1. Summary of Application Procedures

Procedure Name	*Problem Description*
DISTORTED_PC	Distorted-mesh version of the pinched_cyl problem; used to study mesh distortion sensitivity of shell elements.
MH_BEAM	MacNeal-Harder Beam problems.
MH_CYL	MacNeal-Harder Thick-Walled Cylinder Problem.
MH_PLATE	MacNeal-Harder Plate Problems.
MH_SPHERE	Linear inextensional bending of a hemispherical shell loaded by point forces. Model features "warped" quadrilateral shell-element (facet) discretization.
PATCH_TEST	Patch Test Problems.
SKEWED_GRID	Plate Buckling Problem with Skewed Grid.

5.1 Processor DISTORTED_PC

THIS SECTION UNDER PREPARATION

THIS PAGE LEFT BLANK INTENTIONALLY.

5.2 Processor DISTORTED_PC_3D

THIS SECTION UNDER PREPARATION

THIS PAGE LEFT BLANK INTENTIONALLY.

5.3 Procedure MH_BEAMS

5.3.1 GENERAL DESCRIPTION

5.3.1.1 Problem Description

This application problem solves the three MacNeal/Harder cantilever beam problems – the twisted beam, the straight beam, and the curved beam. All three beams are solved with 4- and 9-node 2-D elements and 8-, 20-, and 32-node 3-D elements.

5.3.1.2 Model Description

The straight cantilever beam problem is solved using three element shapes – rectangular, trapezoidal, and parallelogram. This problem tests an element's ability to handle distorted meshes under various types of loading. It is clamped on one end and free at the other end where unit loads are applied in four directions – extension, torque, in-plane, and out-of-plane. The dimensions, mesh, and material properties of the beam are given in Figure 5.3-1.

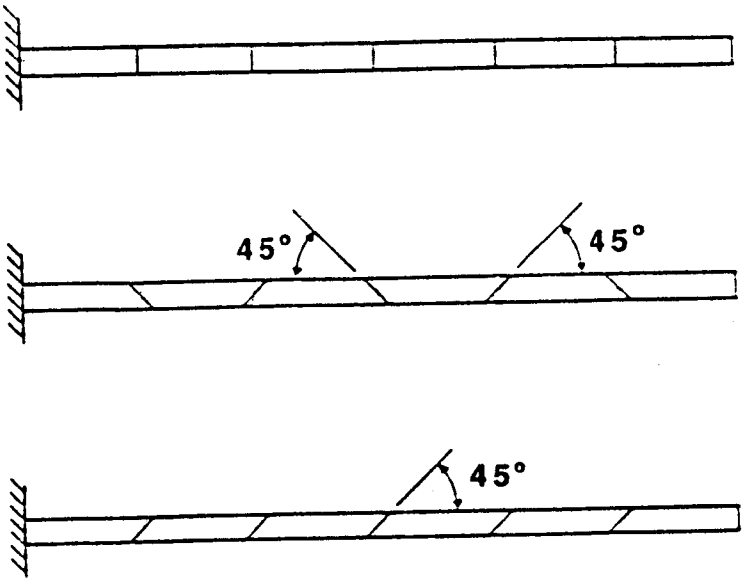


Figure 5.3-1 MacNeal/Harder Straight Cantilever Beam

The twisted cantilever beam is modeled with 24 elements. The beam is clamped on one end and free at the other. It is twisted through 90° root to tip. This problem tests an element's ability to handle warped elements (the four corners of the element do not lie in the same plane). Two different loading conditions are applied at the free end – in-plane unit load, and out-of-plane unit load. The dimensions, mesh, and material properties are shown in Figure 5.3-2.

E = 29.0 x 10^6 psi
ν = 0.22

length	=	12.0 inches
width	=	1.1 inches
depth	=	0.32 inches
twist	=	90° (root to tip)
mesh	=	12 X 2

Figure 5.3-2 MacNeal/Harder Twisted Cantilever Beam

The curved cantilever beam is modeled with six elements. It is clamped on one end and free at the other end, where unit loads are applied. The beam is curved, forming a 90° arc. Two loading conditions are used – in-plane (vertical) and out-of-plane. This problem tests an element's ability to model in-plane curvature. The dimensions, mesh, and material properties are shown in Figure 5.3-3.

E = 1.0 x 10^7 psi
ν = 0.25

inner radius	=	4.12 inches
outer radius	=	4.32 inches
arc	=	90°
thickness	=	0.1 inches
mesh	=	6 X 1

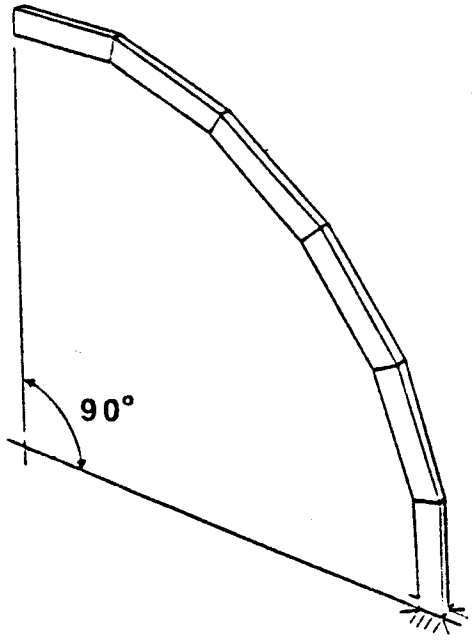


Figure 5.3-3 MacNeal/Harder Curved Cantilever Beam

5.3.1.3 Analysis Description

A static, linear elastic analysis is performed, for any or all of the three problems. If the user chooses, displacements at the point of, and in the direction of the load are calculated and normalized with the theoretical results, for each loading condition.

5.3.1.4 Available Solutions

From reference 5.3-1, the theoretical solutions for the tip displacements for each problem and each loading condition are shown in Table 5.3-1.

Load Condition	Tip displacement in direction of load (inches)		
	Straight	Curved	Twisted
Extension	3.0×10^{-5}	–	–
In-plane	.1081	.08734	.005424
Out-of-plane	.4321	.5022	.001754
Twist	.03208	–	–

Table 5.3-1 Theoretical tip displacements for each beam.

5.3.2 PROCEDURE USAGE

Procedure **MH_BEAMS** may be used by preceding the procedure name by the ***call** directive, and following it by a list of arguments enclosed in parentheses. Procedure arguments are order-independent, and most have default values thus making them optional. The formal syntax is as follows:

```
*call MH_BEAMS ( arg1 = val1 ; arg2 = val2 ; ...)
```

where **arg1** and **arg2** represent argument names, and **val1** and **val2** represent their corresponding values. Note that semi-colons are required between arguments, and a double dash (--) may be used to continue the argument list on the next line.

The allowable arguments for procedure **MH_BEAMS** are summarized in the following table, along with their default values (if they exist). Note that arguments without defaults are generally *mandatory*, while those with defaults are generally optional. Exceptions to this rule are noted in the following section under detailed argument descriptions.

Argument	Default Value	Meaning
ES_PARS	0.	Element research parameters
ES_PROC	ES1	Element Processor
ES_NAME	EX47	Element name
SPAR_DIM	none	2-D or 3-D SPAR element
DEL_LIB	<false>	Delete libraries
REPORT	<true>	Generate "report card" file
STRAIGHT	<true>	Solve straight beam
TWISTED	<true>	Solve twisted beam
CURVED	<true>	Solve curved beam

5.3.3 ARGUMENT DESCRIPTIONS

5.3.3.1 ES_PARS

Element research parameters (default: 0., ...). This is an optional list of element-dependent parameters that some elements provide, primarily when the element is still undergoing research and refinement.

5.3.3.2 ES_PROC

Element processor (default: ES1). This is the name of the element processor that contains the element you wish to evaluate in this procedure. If a SPAR element is used, ES_PROC should be set to SPAR. The element processors are described in The Computational Structural Mechanics Testbed User's Manual.

5.3.3.3 ES_NAME

Element name (default: EX47). This is the name of the element type you wish to select, within the element processor defined by argument ES_PROC. SPAR or ES elements may be used. The elements are described in the appropriate section in the The Computational Structural Mechanics Testbed User's Manual (reference 5.3-2).

5.3.3.4 SPAR_DIM

If a SPAR element is used, this parameter must be set to the number of dimensions of that element. If ES elements are used, this parameter need not be set at all.

5.3.3.5 DEL_LIB

Libraries are created for each model–stbm.l01 for the straight beam, cvbm.l01 for the curved beam, and twbm.l01 for the twisted beam. As the analysis proceeds, these libraries will automatically be deleted if DEL_LIB is <true>. This allows disk space to be conserved if the analyst does not need the libraries.

5.3.3.6 REPORT

Print report file (default=<true>). If REPORT is <true> an external file named BEAM_<es_name>.<es_proc> will be generated. This file contains the normalized radial displacements of the nodes at the inner radius of the cylinder.

5.3.3.7 STRAIGHT

Solve straight beam problem (default: <true>. This parameter simply indicates whether the straight cantilever beam problem is to be solved or not.

5.3.3.8 TWISTED

Solve twisted beam problem (default: <true>. This parameter simply indicates whether the twisted cantilever beam problem is to be solved or not.

5.3.3.9 CURVED

Solve curved beam problem (default: <true>. This parameter simply indicates whether the curved cantilever beam problem is to be solved or not.

5.3.4 USAGE GUIDELINES AND EXAMPLES

Procedure MH_BEAMS may be used by preceding the procedure name by the *call directive. Procedure arguments may be changed from their default values by including any or all of the arguments and their new values when the procedure is called. A space or blank is required between the end of the procedure name and the left parenthesis.

```
*procedure MH_BEAMS ( ES_NAME = EX47     ; --
                      ES_PROC = ES1      ; --
                      ES_PARS = 0        ; --
                      SPAR_DIM           ; --
                      DEL_LIB  = <false> ; --
                      STRAIGHT = <true>  ; --
                      TWISTED  = <true>  ; --
                      CURVED   = <true>  ; --
                      REPORT   = <true>  )
```

(E1) To perform an analysis using the default options, simply invoke the procedure without any arguments, *i.e.*,

```
*call MH_BEAMS
```

This will perform a static, linear elastic analysis of the MH_BEAM problems. Since STRAIGHT, CURVED, and TWISTED are <true> and all three will be solved. Since REPORT is <true> and report card file named BEAM_EX47.ES1 will be generated containing the normalized displacements of the corner nodes at the free end of the beams, Figure 5.3-4.

```
##########################################################################
          MACNEAL-HARDER 2-D CANTILEVER BEAM PROBLEMS
                ELEMENT NAME =    EX47
                ELEMENT PROCESSOR =    ES1
           ---------- NORMALIZED  RESULTS ----------
          *      1      *       2       *       3       *       4
 *************************************************************************
          *             *               *               *
          *  extension      in-plane       out-of-plane      torsion
          *
 *************************************************************************
          *
 STRAIGHT *
    rect  *    0.9955         0.9036          0.9801          0.9096
    rect  *    0.9955         0.9036          0.9801          0.9096
          *
    trap  *    0.7605         0.4693          0.7644          2.2199
    trap  *    0.7610         0.4693          0.7624          -0.4299
          *
    parl  *    0.9660         0.3306          0.9379          -0.1234
    parl  *    0.9656         0.3306          0.9393          1.8199
          *
          *
 TWISTED  *    -----          0.9849          0.9941          -----
          *    -----          0.9850          0.9941          -----
          *    -----          0.9849          0.9941          -----
          *
          *
 CURVED   *    -----          0.9446          0.9221          -----
          *    -----          0.9446          0.9484          -----
          *
          *
##########################################################################
```

Figure 5.3-3 Report file generated using default arguments.

(E2) The following call statement will solve only the twisted beam problem for the element EX08 in processor ES3.

```
*call MH_BEAMS ( ES_NAME = EX08       ; --
                 ES_PROC = ES3        ; --
                 DEL_LIB = <true>     ; --
                 STRAIGHT = <false>   ; --
                 CURVED   = <false>   ; --
                 REPORT   = <true>    )
```

The library twbm.l01 will automatically be deleted. The report file BEAM_EX08.ES3 is generated, Figure 5.3-5.

```
################################################################################
              MACNEAL-HARDER 3-D CANTILEVER BEAM PROBLEMS
                     ELEMENT NAME =    EX08
                     ELEMENT PROCESSOR =    ES3
                     ELEMENT PARAMETERS =   0.
                  ------------ NORMALIZED  RESULTS --------------
            *      1      *       2       *       3       *       4
  ***************************************************************************
            *             *               *               *
            *  extension      in-plane       out-of-plane      torsion
            *
  ***************************************************************************
            *
            *
  TWISTED   *    -----          0.9907          0.9880          -----
            *    -----          0.9907          0.9880          -----
            *    -----          0.9907          0.9881          -----
            *    -----          0.9907          0.9881          -----
            *    -----          0.9907          0.9880          -----
            *    -----          0.9907          0.9880          -----
            *
################################################################################
```

Figure 5.3-5 BEAM_EX08.ES3 resulting from the calling statement (E2).

(E3) The following call statement will solve the straight and twisted cantilever beam problems for the SPAR E43 element and generate the report file BEAM_E43.SPAR.

```
*call MH_BEAMS ( ES_NAME = E43       ; --
                 ES_PROC  = SPAR     ; --
                 SPAR_DIM = 2        ; --
                 CURVED   = <false>  ; --
                 REPORT   = <true>   )
```

5.3.5 LIMITATIONS

(L1) This procedure can only evaluate 4- and 9-node 2-D elements, and 8- and 20-node solid elements.

5.3.6 ERROR MESSAGES AND WARNINGS

If a SPAR element is used, and the argument SPAR_DIM is not set, the following error message will be printed and execution terminated.

```
The argument SPAR_DIM must be defined as 2 or 3 when
  using SPAR elements.
Please set this argument and rerun.
```

5.3.7 PROCEDURE FLOWCHART

5.3.8 PROCEDURE LISTING

5.3.9 REFERENCES

5.3-1 MacNeal, R. H.; and Harder, R. L.: "A Proposed Set of Problems to Test Finite Element Accuracy," *Finite Elements in Analysis and Design*, Vol. 1, 1985, pp. 3-20.

5.3-2 Stewart, Caroline B.: *The Computational Structural Mechanics Testbed User's Manual.* NASA TM-100644, October 1989.

5.4 Procedure MH_CYL

5.4.1 GENERAL DESCRIPTION

5.4.1.1 Problem Description

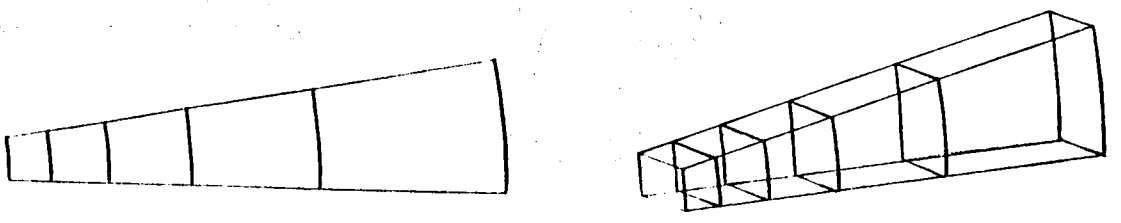
This application problem is another of the MacNeal/Harder Standard Test Problems. The thick-walled cylinder problem is used to test an element's ability to handle a nearly incompressible material. As Poisson's ratio approaches 0.5, singularities may arise in the stiffness matrix as a result of dividing by $(1\text{-}2\nu)$. Rubber, perhaps the most common nearly incompressible material, is sometimes used in structural components.

5.4.1.2 Model Description

A 10° of the thick-walled cylinder, inner radius = 6 inches, outer radius = 9 inches, is modeled with only five elements as is shown in Figure 5.4-1. The thickness of this cross section is unity. Constraints are applied to all nodes so as to allow only radial displacements.

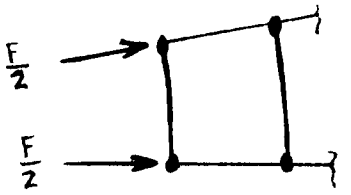
Either plate elements (2-D) or solid elements (3-D) may be used in this procedure. The 2-D elements may be either the 4- or 9-node variety and the 3-D elements may be either 8- or 20-node elements.

The cylinder is loaded with a unit pressure at the inner radius. This pressure is applied as a total force of .5236 lbs. distributed consistently on the nodes at the inner radius of the model, as is shown in Figure 5.4-2. Three sets of material properties are used in the analysis. Young's Modulus for all three is 1000.0 lb/in^2, but Poisson's ratio is allowed to approach 0.5 by assuming the three values .49, .499, and .4999.

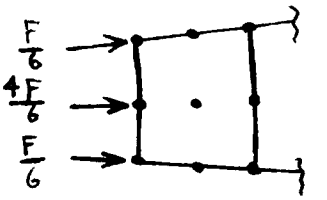


a) 2-D b) 3-D

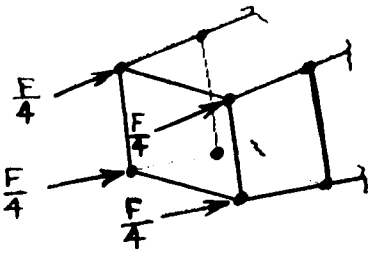
Figure 5.4-1 MacNeal/Harder Thick-Walled Cylinder



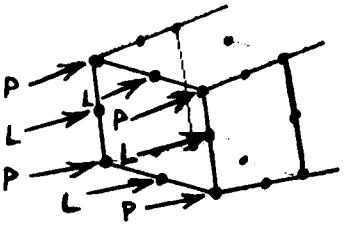
a) 4-node 2-D element



b) 9-node 2-D element



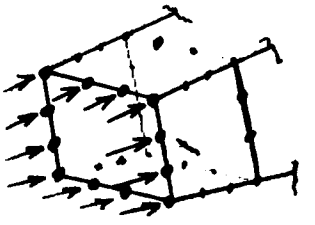
c) 8-node 3-D element



$$P = -\frac{F}{12}$$

$$L = \frac{4F}{12}$$

d) 20-node 3-D element



ON CORNER NODES = $-F/8$

ON MIDEDGE NODES = $3F/16$

e) 32-node 3-D element

Figure 5.4-2 Consistent loading, 1.0 psi pressure at inner radius. F = .5236 lb.

5.4.1.3 Analysis Description

A static, linear elastic analysis is performed. The radial displacements of the nodes at the inner radius are compared to theoretical displacements, and normalized results are printed.

5.4.1.4 Available Solutions

From reference 5.4-1, the exact solution for the radial displacement is:

$$u(r) = \frac{(1+\nu)PR_1^2}{E(R_2^2 - R_1^2)}\left[\frac{R_2^2}{r} + (1-2\nu)r\right]$$

where R_1 is the inner radius, R_2 is the outer radius, P is the applied pressure, and r is the variable radial position. Evaluating this expression at the inner radius ($r = 3.0$) for each value of ν yields the theoretical results in Table 5.4-1.

Table 4.5-1

Poisson's Ratio	Radial displacement at inner radius (inches)
0.49	5.0399×10^{-3}
0.499	5.0602×10^{-3}
0.4999	5.0623×10^{-3}

5.4.2 PROCEDURE USAGE

Procedure **MH_CYL** may be used by preceding the procedure name by the ***call** directive, and following it by a list of arguments enclosed in parentheses. Procedure arguments are order-independent, and most have default values thus making them optional. The formal syntax is as follows:

```
*call MH_CYL ( arg1 = val1 ; arg2 = val2 ; ...)
```

where **arg1** and **arg2** represent argument names, and **val1** and **val2** represent their corresponding values. Note that semi-colons are required between arguments, and a double dash (--) may be used to continue the argument list on the next line.

The allowable arguments for procedure **MH_CYL** are summarized in the following table, along with their default values (if they exist). Note that arguments without defaults are generally *mandatory*, while those with defaults are generally optional. Exceptions to this rule are noted in the following section under detailed argument descriptions.

Argument	Default Value	Meaning
ES_PROC	ES1	Element Processor
ES_NAME	EX47	Element name
ES_PARS	0.	Element research parameters
SPAR_DIM	none	2-D or 3-D SPAR element
REPORT	<true>	Generate "report card" file

5.4.3 ARGUMENT DESCRIPTIONS

5.4.3.1 ES_NAME

Element name (default: **EX47**). This is the name of the element type you wish to select, within the element processor defined by argument **ES_PROC**. SPAR or ES elements may be used. The elements are described in the appropriate section in the The Computational Structural Mechanics Testbed User's Manual (reference 5.4-2).

5.4.3.2 ES_PROC

Element processor (default: **ES1**). This is the name of the element processor that contains the element you wish to evaluate in this procedure. If a SPAR element is used, **ES_PROC** should be set to SPAR. The element processors are described in The Computational Structural Mechanics Testbed User's Manual.

5.4.3.3 ES_PARS

Element research parameters (default: `0., ...`). This is an optional list of element-dependent parameters that some elements provide, primarily when the element is still undergoing research and refinement.

5.4.3.4 SPAR_DIM

If a SPAR element is used, this parameter must be set to the number of dimensions of that element. If ES elements are used, this parameter need not be set at all.

5.4.3.5 REPORT

Print report file (default=**<true>**). If **REPORT** is **<true>** an external file named **CYL_<es_name>.<es_proc>** will be generated. This file contains the normalized radial displacements of the nodes at the inner radius of the cylinder.

5.4.4 USAGE GUIDELINES AND EXAMPLES

Procedure MH_CYL may be used by preceding the procedure name by the *call directive. Procedure arguments may be changed from their default values by including any or all of the arguments and their new values when the procedure is called. A space or blank is required between the end of the procedure name and the left parenthesis.

```
*procedure MH_CYL ( ES_NAME = EX47   ; --
                    ES_PROC = ES1    ; --
                    ES_PARS = 0      ; --
                    SPAR_DIM         ; --
                    REPORT   = <true> )
```

(E1) To perform an entire analysis using the default options, simply invoke the procedure without any arguments, *i.e.*,

```
*call MH_CYL
```

This will perform a static, linear elastic analysis of the MacNeal/Harder thick-walled cylinder problem using the EX47 element in processor ES1. Results will be given for all three values of Poisson's Ratio. The datasets will be written into a library called MH_CYL.L01, and the "report card" file named CYL_EX47.ES1 will be generated, Figure 5.4-3.

```
################################################################
################################################################
          MACNEAL-HARDER 2-D THICK-WALLED CYLINDER
       ELEMENT NAME =    EX47
       ELEMENT PROCESSOR =    ES1
 ***************************************************************
                     *
    Poisson's        *          Normalized displacements
      ratio          *             at inner radius
                     *
 ***************************************************************
                     *
      .49            *                1.0054
                     *                1.0054
                     *
      .499           *                1.0063
                     *                1.0063
                     *
      .4999          *                1.0063
                     *                1.0063
                     *
################################################################
################################################################
```

Figure 5.4-3 "Report Card" using default arguments.

(E2) To perform an entire analysis using an ES element, invoke the procedure with the appropriate arguments, *i.e.*

```
*call MH_CYL ( ES_NAME = BR20    ; --
      ES_PROC = ES10             ; --
      ES_PARS = 3, 3             ; --
      REPORT = <true> )
```

This will perform a static linear elastic analysis using the BR20 element, a 20-node brick element located in processor ES10. A report file will be generated named `CYL_BR20.ES10`.

(E3) To perform the analysis using a SPAR element, invoke the procedure using the appropriate arguments as follows:

```
*call MH_CYL ( ES_NAME = E43     ; --
      ES_PROC = SPAR             ; --
      SPAR_DIM = 2               ; --
      REPORT = <true> )
```

This will perform the analysis using the SPAR E43 element and generate the "report card" file named `CYL_E43.SPAR`. Note that the argument `SPAR_DIM` must be defined when using a SPAR element.

5.4.5 LIMITATIONS

(L1) This procedure can only evaluate 4- and 9-node 2-D elements, and 8-, 20-, and 32-node solid elements.

5.4.6 ERROR MESSAGES AND WARNINGS

If a SPAR element is used, and the argument `SPAR_DIM` is not set, the following error message will be printed and execution terminated.

```
The argument SPAR_DIM must be defined as 2 or 3 when
  using SPAR elements.
Please set this argument and rerun.
```

5.4.7 PROCEDURE FLOWCHART

5.4.8 PROCEDURE LISTING

5.4.9 REFERENCES

5.4-1 MacNeal, R. H.; and Harder, R. L.: "A Proposed Set of Problems to Test Finite Element Accuracy," *Finite Elements in Analysis and Design*, Vol. 1, 1985, pp. 3-20.

5.4-2 Stewart, Caroline B.: *The Computational Structural Mechanics Testbed User's Manual.* NASA TM-100644, October 1989.

THIS PAGE LEFT BLANK INTENTIONALLY.

5.5 Procedure MH_PATCH

5.5.1 GENERAL DESCRIPTION

5.5.1.1 Problem Description

This application problem is another of the MacNeal/Harder Standard Test Problems. The patch test is used to test an element's ability to handle a state of constant strain with a distorted grid. This is a necessary condition to guarantee convergence of the solution.

5.5.1.2 Model Description

The patch test is modeled for both 2-D and 3-D elements. The 2-D model consists of 5 distorted quadrilateral elements forming a .24 by .12 inch rectangular plate of thickness one inch, Figure 5.5-1. The 3-D model consists of seven distorted hexagonal elements forming a one inch cube, Figure 5.5-2. Element corner node locations for both the 2-D and 3-D model are shown in Table 5.5-1. For all higher order elements (2-D - 9-node and 3-D 20- and 32-node) the midedge nodes are placed equidistant from the corner nodes. However, for the 2-D 9-node model, a parameter called TWEEK has been added to allow the elements to have curved sides. See section 5.5.3 for details. The 2-D patch test is actually two separate tests, the membrane patch test, and the bending patch test. However, the bending patch test is not currently implemented.

In the membrane patch test, displacements are applied according to the following formulas:

$$u = 10^{-3}(x + y/2) \text{ in.}$$
$$v = 10^{-3}(y + x/2) \text{ in.}$$

yielding a state of constant strain.

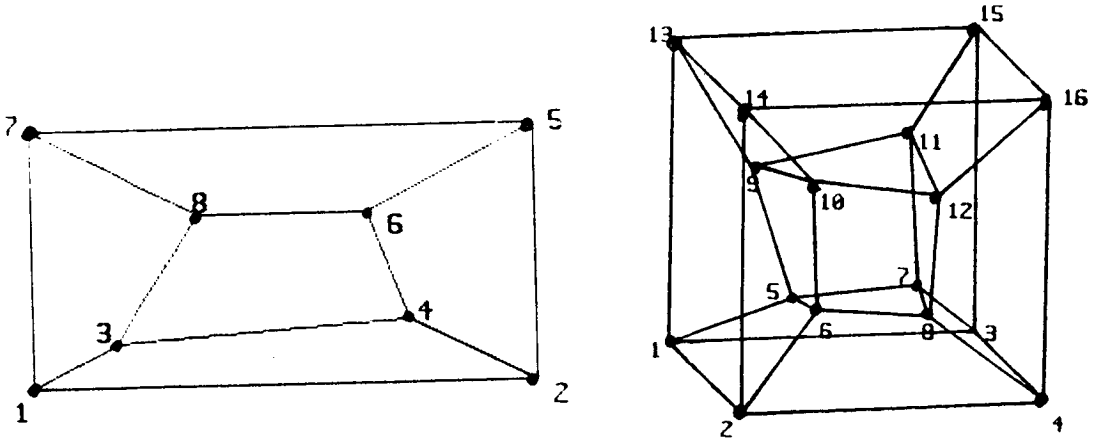
For the bending patch test:

$$w = 10^{-3}(x^2 + xy + y^2)/2 \text{ in.}$$
$$\theta_x = \frac{\partial w}{\partial y} = 10^{-3}(y + x/2)$$
$$\theta_y = -\frac{\partial w}{\partial x} = 10^{-3}(-x - y/2)$$

yielding a state of constant curvature.

A state of constant strain is applied to the 3-D model according to the formulas:

$$u = 10^{-3}(2x + y + z)/2 \text{ in.}$$
$$v = 10^{-3}(x + 2y + z)/2 \text{ in.}$$
$$w = 10^{-3}(x + y + 2z)/2 \text{ in.}$$



a) 2-D model b) 3-D model

Figure 5.5-1 MacNeal/Harder Patch Test

NODE	3-D Model			2-D Model	
	X	Y	Z	X	Y
1	0.000	0.000	0.000	0.00	0.00
2	1.000	0.000	0.000	0.24	0.00
3	0.000	1.000	0.000	0.04	0.02
4	1.000	1.000	0.000	0.18	0.03
5	0.249	0.342	0.192	0.24	0.12
6	0.826	0.288	0.288	0.16	0.08
7	0.273	0.750	0.230	0.00	0.12
8	0.850	0.649	0.263	0.08	0.08
9	0.320	0.186	0.643		
10	0.677	0.305	0.683		
11	0.165	0.745	0.702		
12	0.788	0.693	0.644		
13	0.000	0.000	1.000		
14	0.000	0.000	1.000		
15	0.000	1.000	1.000		
16	1.000	1.000	1.000		

Table 5.5-1 Element Corner Node Locations

5.5.1.3 Analysis Description

A static, linear elastic analysis is performed. Stresses are calculated and printed, as is specified by the user, at the integration points, centroids, and/or nodes. See section 5.5.2 for details.

5.5.1.4 Available Solutions

From reference 5.5-1, the exact solutions are as follows:

2-D Membrane Patch Test

$$\varepsilon_x = \varepsilon_y = \gamma_{xy} = 10^{-3}$$
$$\sigma_x = \sigma_y = 1333\ psi$$
$$\tau_{xy} = 400\ psi$$

2-D Bending Patch Test

$$M_x = M_y = 1.111 \times 10^{-7}\ in.lb.$$
$$M_{xy} = 10^{-7}\ in.lb.$$
$$\sigma_x = \sigma_y = \pm .667\ psi$$
$$\tau_{xy} = \pm .200\ psi$$

3-D Solid Patch Test

$$\varepsilon_x = \varepsilon_y = \varepsilon_z = \gamma_{xy} = \gamma_{yz} = \gamma_{xz} = 10^{-3}$$
$$\sigma_x = \sigma_y = \sigma_z = 2000\ psi$$
$$\tau_{xy} = \tau_{yz} = \tau_{xz} = 400\ psi$$

5.5.2 PROCEDURE USAGE

Procedure **MH_PATCH** may be used by preceding the procedure name by the ***call** directive, and following it by a list of arguments enclosed in parentheses. Procedure arguments are order-independent, and most have default values thus making them optional. The formal syntax is as follows:

```
*call MH_PATCH ( arg1 = val1 ; arg2 = val2 ; ...)
```

where **arg1** and **arg2** represent argument names, and **val1** and **val2** represent their corresponding values. Note that semi-colons are required between arguments, and a double dash (--) may be used to continue the argument list on the next line.

The allowable arguments for procedure **MH_PATCH** are summarized in the following table, along with their default values (if they exist). Note that arguments without defaults are generally *mandatory*, while those with defaults are generally optional. Exceptions to this rule are noted in the following section under detailed argument descriptions.

Argument	Default Value	Meaning
ES_PROC	ES1	Element Processor
ES_NAME	EX47	Element name
ES_PARS	0.	Element research parameters
SPAR_DIM	none	2-D or 3-D SPAR element
REPORT	<true>	Generate "report card" file
STR_ALL	<true>	Compute stresses at nodes, integration points, and centroids
STR_NODES	<false>	Compute stress at nodes
STR_INTPTS	<false>	Compute stress at integration points
STR_CENTS	<false>	Compute stress at centroids
TWEEK	0,0,0,0	Move midside nodes print stresses
DISPLAY	1	Print stresses
NODES	1	Stresses at nodes and centroids
CROSS	1	Stresses at top, mid, and bottom surfaces

5.5.3 ARGUMENT DESCRIPTIONS

5.5.3.1 ES_NAME

Element name (default=EX47). This is the name of the element type you wish to select, within the element processor defined by argument ES_PROC. SPAR or ES elements may be used. The elements are described in the appropriate section in the The Computational Structural Mechanics Testbed User's Manual (reference 5.5-2).

5.5.3.2 ES_PROC

Element processor (default=ES1). This is the name of the element processor that contains the element you wish to evaluate in this procedure. If a SPAR element is used, ES_PROC should be set to SPAR. The element processors are described in The Computational Structural Mechanics Testbed User's Manual.

5.5.3.3 ES_PARS

Element research parameters (default=0., ...). This is an optional list of element-dependent parameters that some elements provide, primarily when the element is still undergoing research and refinement.

5.5.3.4 SPAR_DIM

Number of dimensions of SPAR element (default=none). If a SPAR element is used, this parameter must be set to the number of dimensions of that element. If ES elements are used, this parameter need not be set at all.

5.5.3.5 REPORT

Print report file (default=<true>). If REPORT is <true> an external file named CYL_<es_name>.<es_proc> will be generated. This file contains the normalized radial displacements of the nodes at the inner radius of the cylinder.

5.5.3.6 STR_ALL

(default=<true>). If <true> stresses will be calculated at the element nodes, centroids, and integration points. If REPORT is <true>, stress values at all three locations will be printed in the report file.

5.5.3.7 STR_NODES

(default=<false>). By setting this argument to <true>, STR_ALL will be automatically reset to <false>, and stresses will be calculated only at the nodes. If REPORT is <true>, stress values at the nodes will be printed in the report file.

5.5.3.8 STR_INTPTS

(default=<false>). By setting this argument to <true>, STR_ALL will be automatically reset to <false>, and stresses will be calculated only at the integration points. If REPORT is <true>, stress values at the integration points will be printed in the report file.

5.5.3.9 STR_CENTS

(default=<false>). By setting this argument to <true>, STR_ALL will be automatically reset to <false>, and stresses will be calculated only at the centroids. If REPORT is <true>, stress values at the centroids will be printed in the report file.

5.5.3.10 TWEEK

(default=0., 0., 0., 0.). The TWEEK parameters are used to offset the midside nodes on 9-node 2-D elements. This allows the user to test an element's sensitivity to curved edges. There are four TWEEK parameters that one separated by commas, with the location and positive direction shown in Figure 5.5-3. =

5.5.3.11 DISPLAY

(default=1). For SPAR elements only. For the various values of DISPLAY, the following will be printed in the report file:

If DISPLAY

For two-dimensional elements,
= 1, stresses
= 2, membrane stress resultants
= 3, bending stress resultants

For three-dimensional solids (S41, S61, S81),
= 1, x, y, z stress components
relative to the element reference frame
= 2, principal stresses
= 3, octahedral normal stress (ONS),
octahedral shear stress (OSS),
stress intensity (SI),
and yield stress ratio (YSR)
= 4, display of $S_{xx}/Y_{xx}, S_{yy}/Y_{yy}, S_{zz}/Y_{zz}$

5.5.3.12 NODES

(default=1). For SPAR elements only. This parameter controls whether results will be printed for the element centroids or the nodes as follows:

If NODES = 0, restrict printout to element center
= 1, printout for each node and element center

5.5.3.13 CROSS

(default=1). For SPAR elements only. This parameter controls whether results are printed for the element mid-surface of top, mid and bottom surfaces as follows:

If CROSS = 0, restrict printout to mid-surface stresses
= 1, printout for top, mid, and bottom surface stresses

Note: The parameters DISPLAY, NODES and CROSS are actually resets to processor PSF. See section 12.2 in reference 5.5-2 for details.

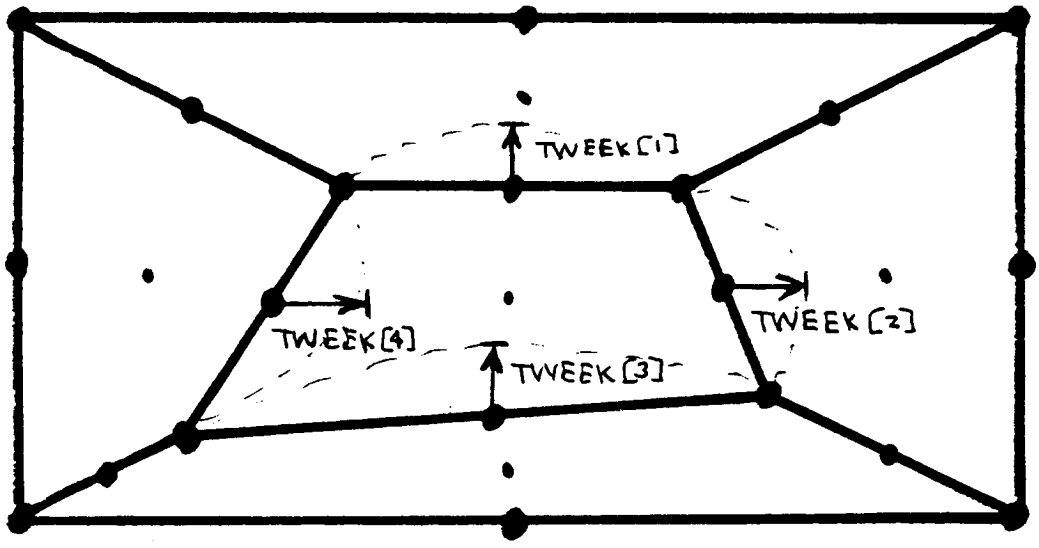


Figure 5.5-2 Location and positive direction of TWEEK parameters

5.5.4 USAGE GUIDELINES AND EXAMPLES

Procedure **MH_PATCH** may be used by preceding the procedure name by the *call directive. Procedure arguments may be changed from their default values by including any or all of the arguments and their new values when the procedure is called. A space or blank is required between the end of the procedure name and the left parenthesis.

```
*procedure MH_PATCH ( ES_NAME = EX47        ; --
                      ES_PROC = ES1         ; --
                      ES_PARS = 0           ; --
                      SPAR_DIM              ; --
                      STR_ALL = <true>      ; --
                      STR_NODES = <false>   ; --
                      STR_INTPTS = <false>  ; --
                      STR_CENTS = <false>   ; --
                      DISPLAY = 1            ; --
                      NODES = 1              ; --
                      CROSS = 1              ; --
                      TWEEK = 0., 0., 0., 0. ; --
                      REPORT   = <true>      )
```

(E1) To perform an entire analysis using the default options, simply invoke the procedure without any arguments, *i.e.*,

```
*call MH_PATCH
```

This will perform a static, linear elastic analysis of the MacNeal/Harder patch test using the EX47 element in processor ES1. Stresses will be calculated at nodes, centroids, and integration points and the results will be printed to a file named PATCH_EX47.ES1. The datasets will be written to a library named patch.101.

(E2) To perform an analysis using the BRO8 element in processor ES10 and generating stresses only at the integration points:

```
*call MH_PATCH ( ES_NAME = EX43     ; --
      ES_PROC = ES4                 ; --
      STR_NODES = <true>            ; --
      REPORT = <true>               )
```

The report file named PATCH_BR08.ES10 will be generated containing the stresses at the element integration points only, Figure 5.5-3.

(E3) To perform an analysis using a SPAR element, invoke the procedure using the appropriate arguments as follows:

```
*call MH_PATCH ( ES_NAME = E43   ; --
      ES_PROC = SPAR             ; --
      SPAR_DIM = 2               ; --
      DISPLAY = 1                 ; --
      NODES = 0                   ; --
      CROSS = 0                   ; --
      REPORT = <true>             )
```

The report file PATCH_E43.SPAR will be generated containing stresses (display=1) at the element centroid midsurface (nodes=0, cross=0). See reference 5.5-2, section 12.2 for further information about the parameters display, nodes, and cross (they are actually resets to processor PSF). Note that SPAR_DIM must be defined when using a SPAR element.

```
   ################################################################
   ################################################################
              MACNEAL-HARDER 2-D PATCH TEST
         ELEMENT NAME =     EX43
         ELEMENT PROCESSOR =     ES4
         ELEMENT PARAMATERS=     0
    ************************************************************
 ** BEGIN PESR **    DATA SPACE=  2000000 WORDS
1
 ELEMENT EX43/     LOAD CASE     1/     GLOBAL REFERENCE FRAME
 GROUP ELEM LOCATION        NX             NY             NXY
 ----- ---- -------- ------------ ------------ ------------ --
0    1    1       1N    1.333333E+00  1.333333E+00  4.000000E-01
                  2N    1.333333E+00  1.333333E+00  4.000000E-01
                  4N    1.333333E+00  1.333333E+00  4.000000E-01
                  3N    1.333333E+00  1.333333E+00  4.000000E-01
0    1    2       2N    1.333333E+00  1.333333E+00  4.000000E-01
                  5N    1.333333E+00  1.333333E+00  4.000000E-01
                  6N    1.333333E+00  1.333333E+00  4.000000E-01
                  4N    1.333333E+00  1.333333E+00  4.000000E-01
0    1    3       5N    1.333333E+00  1.333333E+00  4.000000E-01
                  7N    1.333333E+00  1.333333E+00  4.000000E-01
                  8N    1.333333E+00  1.333333E+00  4.000000E-01
                  6N    1.333333E+00  1.333333E+00  4.000000E-01
0    1    4       7N    1.333333E+00  1.333333E+00  4.000000E-01
                  1N    1.333333E+00  1.333333E+00  4.000000E-01
                  3N    1.333333E+00  1.333333E+00  4.000000E-01
                  8N    1.333333E+00  1.333333E+00  4.000000E-01
0    1    5       3N    1.333333E+00  1.333333E+00  4.000000E-01
                  4N    1.333333E+00  1.333333E+00  4.000000E-01
                  6N    1.333333E+00  1.333333E+00  4.000000E-01
                  8N    1.333333E+00  1.333333E+00  4.000000E-01
 EXIT PESR CPUTIME=     0.6 I/O(DIR,BUF)=       0       0
       ################################################################
       ################################################################
```

Figure 5.5-3 "Report Card" using EX43 in ES4.

5.5.5 LIMITATIONS

(L1) This procedure can only evaluate 4- and 9-node 2-D elements, and 8-, 20-, and 32-node solid elements.

(L2) Currently only the 2-D membrane patch test and the 3-D test are operational.

5.5.6 ERROR MESSAGES AND WARNINGS

If a SPAR element is used, and the argument **SPAR_DIM** is not set, the following error message will be printed and execution terminated.

```
The argument SPAR_DIM must be defined as 2 or 3 when
  using SPAR elements.
Please set this argument and rerun.
```

5.5.7 PROCEDURE FLOWCHART

5.5.8 PROCEDURE LISTING

5.5.9 REFERENCES

5.5-1 MacNeal, R. H.; and Harder, R. L.: "A Proposed Set of Problems to Test Finite Element Accuracy," *Finite Elements in Analysis and Design*, Vol. 1, 1985, pp. 3-20.

5.5-2 Stewart, Caroline B.: *The Computational Structural Mechanics Testbed User's Manual.* NASA TM-100644, October 1989.

THIS PAGE LEFT BLANK INTENTIONALLY.

5.6 Procedure MH_PLATE

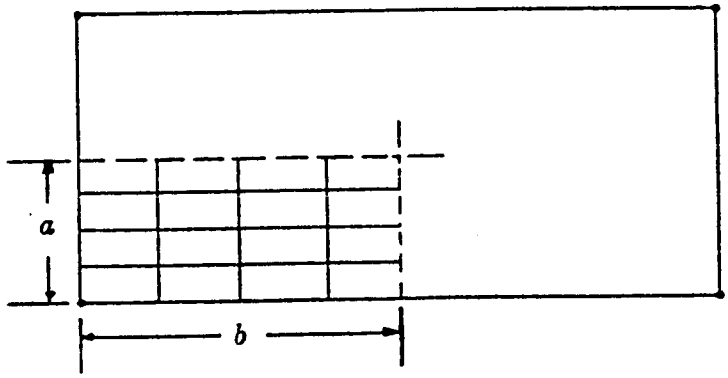
5.6.1 GENERAL DESCRIPTION

5.6.1.1 Problem Description

This application problem is another of the MacNeal/Harder Standard Test Problems. The plate problem tests an element's ability to handle aspect ratio, thinness, and various types of loading and boundary conditions. The problem can be solved with 4- or 9-node 2-D elements or 8-node 3-D elements.

5.6.1.2 Model Description

Due to the symmetry of the physical problem a quarter model is used. Symmetry boundary conditions are applied at x=0 and y=0. At x=a and y=b two different boundary conditions are used – simply supported and clamped. Two different types of loading are applied – a central point load of $\mathbf{P} = 4.0 \times 10^{-4}$ lbs. and a uniform pressure of $\mathbf{q} = 10^{-4}$ psi. The value of a is 1.0 in., but, in order to test an element's sensitivity to aspect ratio, b has two values, 1.0 in. and 5.0 in. The quarter plate is modeled with $N \times N$ elements. A mesh convergence study is done, with N taking the value in Table 5.6-1. The two values of b gives element aspect ratios of 1 and 5. A typical model with $N = 4$, material properties, and dimensions are shown in Figure 5.6-1.



E = 1.7472 x 10^7 psi
ν = 0.30

a = 1.0 inches
b = 1.0 or 5.0 inches
thickness = 0.0001 inches (2-D elements)
= 0.01 inches (3-D elements)
mesh = N X N (on ¼ of plate)

Figure 5.6-1 Typical plate model, $N = 4$

	2-D Model		3-D Model	
	4-node	9-node	8-node	20-node
N =	2	2	2	2
	4	4	4	4
	6	6	6	6
	8		8	

Table 5.6-1 Values of N for different element types

5.6.1.3 Analysis Description

A static, linear elastic analysis is performed for each of the eight possible combinations of aspect ratio, loading, and boundary condition. Libraries named pl<aspect>_<grid>.l01 are created, where <aspect> is aspect ratio and <grid> is N, the mesh size. As the analysis proceeds, these libraries may be deleted by setting the argument `DEL.LIB` is <true>. If the argument `REPORT` is <true>, the normalized midplate displacements will be printed in an external "report file" named PLATE_<es_name>.<es_proc>, where <es_name> is the element name and <es_proc> is the element processor.

5.6.1.4 Available Solutions

From reference 5.6-1, the exact solutions for the mid-plate displacement in the direction of the load for each boundary condition, load type, and aspect ratio are shown in Table 5.6-2.

Boundary Condition	Aspect Ratio b/a	Mid-plate displacement (inches)	
		Uniform Pressure	Point Load
Simple	1.0	4.062	11.60
	5.0	12.97	16.96
Clamped	1.0	1.26	5.60
	5.0	2.56	7.23

Table 5.6-2 Theoretical tip displacements for each beam.

5.6.2 PROCEDURE USAGE

Procedure `MH_PLATE` may be used by preceding the procedure name by the `*call` directive, and following it by a list of arguments enclosed in parentheses. Procedure arguments are order-independent, and most have default values thus making them optional. The formal syntax is as follows:

```
*call MH_PLATE ( arg1 = val1 ; arg2 = val2 ; ...)
```

where **arg1** and **arg2** represent argument names, and **val1** and **val2** represent their corresponding values. Note that semi-colons are required between arguments, and a double dash (--) may be used to continue the argument list on the next line.

The allowable arguments for procedure **MH_PLATE** are summarized in the following table, along with their default values (if they exist). Note that arguments without defaults are generally *mandatory*, while those with defaults are generally optional. Exceptions to this rule are noted in the following section under detailed argument descriptions.

Argument	Default Value	Meaning
ES_PARS	0.	Element research parameters
ES_PROC	ES1	Element Processor
ES_NAME	EX47	Element name
SPAR_DIM	none	2-D or 3-D SPAR element
DEL_LIB	<false>	Delete libraries
REPORT	<true>	Generate "report card" file

5.6.3 ARGUMENT DESCRIPTIONS

5.6.3.1 ES_PARS

Element research parameters (default: 0., ...). This is an optional list of element-dependent parameters that some elements provide, primarily when the element is still undergoing research and refinement.

5.6.3.2 ES_PROC

Element processor (default: ES1). This is the name of the element processor that contains the element you wish to evaluate in this procedure. If a SPAR element is used, ES_PROC should be set to SPAR. The element processors are described in The Computational Structural Mechanics Testbed User's Manual.

5.6.3.3 ES_NAME

Element name (default: EX47). This is the name of the element type you wish to select, within the element processor defined by argument ES_PROC. SPAR or ES elements may be used. The elements are described in the appropriate section in the The Computational Structural Mechanics Testbed User's Manual (reference 5.6-2).

5.6.3.4 SPAR_DIM

If a SPAR element is used, this parameter must be set to the number of dimensions of that element. If ES elements are used, this parameter need not be set at all.

5.6.3.5 DEL_LIB

As the mesh convergence study proceeds, DEL_LIB determines whether or not the individual libraries for each N will be deleted. This allows disk space to be conserved if the analyst does not need the libraries.

5.6.3.6 REPORT

Print report file (default=<true>). If REPORT is <true> an external file named PLATE_<es_name>.<es_proc> will be generated. This file contains the normalized radial displacements of the nodes at the inner radius of the cylinder.

5.6.4 USAGE GUIDELINES AND EXAMPLES

Procedure MH_PLATE may be used by preceding the procedure name by the *call directive. Procedure arguments may be changed from their default values by including any or all of the arguments and their new values when the procedure is called. A space or blank is required between the end of the procedure name and the left parenthesis.

```
*procedure MH_PLATE ( ES_NAME = EX47     ; --
                      ES_PROC = ES1      ; --
                      ES_PARS = 0        ; --
                      SPAR_DIM           ; --
                      DEL_LIB  = <false> ; --
                      REPORT   = <true>  )
```

(E1) To perform an analysis using the default options, simply invoke the procedure without any arguments, *i.e.*,

```
*call MH_PLATE
```

This will perform a static, linear elastic, analysis of the MH_PLATE problem. The normalized mid-plate displacements for all combinations of loading, boundary conditions, and aspect ratio will be calculated and printed in the report file named PLATE_EX47.ES1, Figure 5.6-2. Eight libraries named pl1_2.l01, pl1_4.l01, pl1_6.l01, pl1_8.l01, pl5_2.l01, pl5_4.l01, pl5_6.l01, and pl5_8.l01 will be generated. In a library named pl$A$_N.l01, A is the aspect ratio and N is the mesh. These eight libraries will not be deleted as the analysis proceeds, because the default value of DEL_LIB is <false>.

```
###############################################################################
              MACNEAL-HARDER 2-D RECTANGULAR PLATE PROBLEM
                     ELEMENT NAME =     EX47
                     ELEMENT PROCESSOR =    ES1
                     ELEMENT PARAMATERS =   0
                  ------------ NORMALIZED  RESULTS ------------
 ******************************************************************************
         *
         *                          ASPECT RATIO = 1
         *
 ******************************************************************************
         *                                  *
         *       CENTRAL POINT LOAD         *          DISTRIBUTED LOAD
 ******************************************************************************
         *                  *                *                  *                *
         *      clamped     *     simply     *   clamped        *     simply     *
```

```
        *                *    supported  *                *    supported  *
*************************************************************************
        *                                                                 *
MESH=2  *     0.8811          1.0042         0.9790          0.9866      *
        *                                                                 *
MESH=4  *     0.9706          0.9979         0.9969          0.9972      *
        *                                                                 *
MESH=6  *     0.9876          0.9986         1.0009          0.9988      *
        *                                                                 *
MESH=8  *     0.9936          0.9991         1.0024          0.9994      *
        *                                                                 *
*************************************************************************
        *
        *                       ASPECT RATIO = 5
        *
*************************************************************************
        *                                  *
        *      CENTRAL POINT LOAD          *         DISTRIBUTED LOAD
*************************************************************************
        *                *                 *                *              *
        *     clamped    *     simply      *   clamped      *     simply   *
        *                *    supported    *                *    supported *
*************************************************************************
        *                                                                 *
MESH=2  *     0.3189          0.8904         1.1257          0.9543      *
        *                                                                 *
MESH=4  *     0.8414          0.9417         0.9777          0.9779      *
        *                                                                 *
MESH=6  *     0.9222          0.9701         0.9890          0.9903      *
        *                                                                 *
MESH=8  *     0.9537          0.9817         1.0012          0.9946      *
        *                                                                 *
#########################################################################
```

Figure 5.6-2 Report file generated using default arguments.

(E2) To solve the MacNeal/Harder Plate problem with element E410 in processor ES5, deleting the libraries as the analysis proceeds, use the following call statement.

```
*call MH_PLATE ( ES_NAME = E410      ; --
                 ES_PROC = ES5       ; --
                 DEL_LIB = <true>    ; --
                 REPORT  = <true>    )
```

(E3) To perform the analysis using a SPAR element, invoke the procedure with the appropriate element name and SPAR for the processor. The argument SPAR_DIM must be defined when using a SPAR element. The libraries will be deleted as the analysis proceeds.

```
*call MH_PLATE ( ES_NAME  = E43      ; --
                 ES_PROC  = SPAR     ; --
                 SPAR_DIM = 2        ; --
                 DEL_LIB  = <true>   ; --
```

```
REPORT    = <true>    )
```

5.6.5 LIMITATIONS

(L1) This procedure can only evaluate 4- and 9-node 2-D elements, and 8-node solid elements.

5.6.6 ERROR MESSAGES AND WARNINGS

If a SPAR element is used, and the argument **SPAR_DIM** is not set, the following error message will be printed and execution terminated.

```
The argument SPAR_DIM must be defined as 2 or 3 when
  using SPAR elements.
Please set this argument and rerun.
```

5.6.7 PROCEDURE FLOWCHART

5.6.8 PROCEDURE LISTING

5.6.9 REFERENCES

5.6-1 MacNeal, R. H.; and Harder, R. L.: "A Proposed Set of Problems to Test Finite Element Accuracy," *Finite Elements in Analysis and Design*, Vol. 1, 1985, pp. 3-20.

5.6-2 Stewart, Caroline B.: *The Computational Structural Mechanics Testbed User's Manual.* NASA TM-100644, October 1989.

5.7 Procedure MH_SPHERE

5.7.1 GENERAL DESCRIPTION

5.7.1.1 Problem Description

This application problem is another of the MacNeal/Harder Standard Test Problems. The spherical shell problem tests an element's ability to handle double curvature.

5.7.1.2 Model Description

The spherical shell is modeled for both 2-D and 3-D elements. As is shown in figure 5.7-1, it is a one-eighth model of a spherical shell with a 12° hole in the top. The hole in the top of the sphere allows triangular (2-D) and wedge-shaped (3-D) elements to be avoided. The sphere is loaded by two point loads (1 lb. each) applied at centric points as is shown in figure 5.7-1. The sphere is modeled with $N \times N$ elements. A mesh convergence study is done with N taking the values in table 5.7-1.

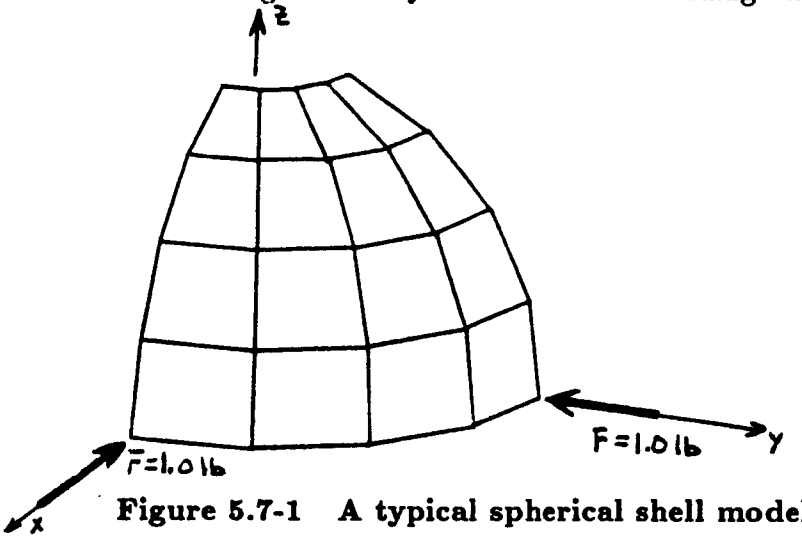


Figure 5.7-1 A typical spherical shell model, $N = 4$.

An example of a 2-D and a 3-D model are shown in figure 5.7-2.

5.7.1.3 Analysis Description

A static, linear elastic analysis is performed. Displacements at the point of, and in the direction of the loads are calculated, normalized with theoretical results, and printed for each value of N (table 5.7-1).

5.7.1.4 Available Solutions

From reference 5.7-1, the exact theoretical solution for the displacement at the point of, and in the direction of the load is .0940 inches.

5.7.2 PROCEDURE USAGE

Procedure **MH_SPHERE** may be used by preceding the procedure name by the ***call** directive, and following it by a list of arguments enclosed in parentheses. Procedure arguments are order-independent, and most have default values thus making them optional. The formal syntax is as follows:

```
*call MH_SPHERE ( arg1 = val1 ; arg2 = val2 ; ...)
```

where **arg1** and **arg2** represent argument names, and **val1** and **val2** represent their corresponding values. Note that semi-colons are required between arguments, and a double dash (--) may be used to continue the argument list on the next line.

The allowable arguments for procedure **MH_SPHERE** are summarized in the following table, along with their default values (if they exist). Note that arguments without defaults are generally *mandatory*, while those with defaults are generally optional. Exceptions to this rule are noted in the following section under detailed argument descriptions.

Argument	Default Value	Meaning
ES_PROC	ES1	Element Processor
ES_NAME	EX47	Element name
ES_PARS	0.	Element research parameters
SPAR_DIM	none	2-D or 3-D SPAR element
REPORT	<true>	Generate "report card" file
DEL_LIB	<false>	Delete libraries

5.7.3 ARGUMENT DESCRIPTIONS

5.7.3.1 ES_NAME

Element name (default: **EX47**). This is the name of the element type you wish to select, within the element processor defined by argument **ES_PROC**. SPAR or ES elements may be used. The elements are described in the appropriate section in The Computational Structural Mechanics Testbed User's Manual (reference 5.7-2).

5.7.3.2 ES_PROC

Element processor (default: **ES1**). This is the name of the element processor that contains the element you wish to evaluate in this procedure. If a SPAR element is used, **ES_PROC** should be set to SPAR. The element processors are described in The Computational Structural Mechanics Testbed User's Manual.

5.7.3.3 ES_PARS

Element research parameters (default: 0., ...). This is an optional list of element-dependent parameters that some elements provide, primarily when the element is still undergoing research and refinement.

5.7.3.4 SPAR_DIM

If a SPAR element is used, this parameter must be set to the number of dimensions of that element. If ES elements are used, this parameter need not be set at all.

5.7.3.5 REPORT

Print report file (default=<true>). If REPORT is <true> an external file named CYL_<es_name>.<es_proc> will be generated. This file contains the normalized radial displacements of the nodes at the inner radius of the cylinder.

5.7.3.6 DEL_LIB

Delete libraries (default=<false>). As the mesh convergence study progresses, libraries named SPL*N*.L01 are generated, where *N* is the number of elements along each edge of model. If DEL_LIB is set to <true>, these libraries will be deleted during the analysis. This allows disk space to be conserved if the analyst does not need the libraries.

	2-D Model		3-D Model	
	4-node	9-node	8-node	20-node
N =	2	1	4	2
	4	2	8	4
	6	3	12	6
	8	4		
	10	5		
	12	6		

Table 5.7-1 ***N* = number of elements for convergence study. Model is *N* × *N*.**

5.7.4 USAGE GUIDELINES AND EXAMPLES

Procedure MH_SPHERE may be used by preceding the procedure name by the *call directive. Procedure arguments may be changed from their default values by including any or all of the arguments and their new values when the procedure is called. A space or blank is required between the end of the procedure name and the left parenthesis.

```
*procedure MH_SPHERE ( ES_NAME = EX47   ; --
                       ES_PROC = ES1    ; --
                       ES_PARS = 0      ; --
                       SPAR_DIM         ; --
                       DEL_LIB = <false> ; --
                       REPORT   = <true>  )
```

(E1) To perform an entire analysis using the default options, simply invoke the procedure without any arguments, *i.e.*,

```
*call MH_SPHERE
```

This will perform a static, linear elastic analysis of the MacNeal/Harder spherical shell using the EX47 element in processor ES1. The one-eighth sphere, modeled by $N \times N$ elements, will be analyzed with N = 2, 4, 6, 8, 10, and 12. Six libraries will be created, SPHN.L01, one for each value of N. If the argument DEL_LIB is <true>, these libraries will be automatically deleted as the analysis proceeds. If the argument REPORT is <true>, the normalized displacements at the point of, and in the direction of, the load will be printed in a report file named CYL_EX47.ES1 for each value of N, figure 5.7-3.

```
######################################################################
######################################################################
          MACNEAL-HARDER 2-D SPHERICAL SHELL PROBLEM
       ELEMENT NAME =    EX47
       ELEMENT PROCESSOR =    ES1
       ELEMENT PARAMATERS =    0
------------------- NORMALIZED  RESULTS ----------------------
######################################################################
GRID = 2                0.9676
                        0.9676
GRID = 4                1.0176
                        1.0176
GRID = 6                1.0012
                        1.0012
GRID = 8                0.9950
                        0.9950
GRID = 10               0.9923
                        0.9923
GRID = 12               0.9919
                        0.9919
######################################################################
```

Figure 5.7-3 Report file generated using default arguments.

(E2) The following call statement will solve the spherical shell problem for the solid element EX20 in processor ES3. The libraries for each value of N will be deleted.

```
*call MH_SPHERE ( ES_NAME = EX20       ; --
                  ES_PROC = ES3        ; --
                  DEL_LIB = <true>     ; --
                  REPORT  = <true>      )
```

(E3) To perform the analysis using a SPAR element, invoke the procedure using the appropriate 4-node 2-D element or 8-node 3-D element and SPAR for ES_PROC. Note that the argument SPAR_DIM must be defined when using a SPAR element.

```
*call MH_SPHERE ( ES_NAME = S81      ; --
                  ES_PROC = SPAR     ; --
                  SPAR_DIM = 3       ; --
                  REPORT = <true>     )
```

5.7.5 LIMITATIONS

(L1) This procedure can only evaluate 4- and 9-node 2-D elements, and 8- and 20-node solid elements.

5.7.6 ERROR MESSAGES AND WARNINGS

If a SPAR element is used, and the argument SPAR_DIM is not set, the following error message will be printed and execution terminated.

```
The argument SPAR_DIM must be defined as 2 or 3 when
  using SPAR elements.
Please set this argument and rerun.
```

5.7.7 PROCEDURE FLOWCHART

5.7.8 PROCEDURE LISTING

5.7.9 REFERENCES

5.7-1 MacNeal, R. H.; and Harder, R. L.: "A Proposed Set of Problems to Test Finite Element Accuracy," *Finite Elements in Analysis and Design*, Vol. 1, 1985, pp. 3-20.

5.7-2 Stewart, Caroline B.: *The Computational Structural Mechanics Testbed User's Manual.* NASA TM-100644, October 1989.

THIS PAGE LEFT BLANK INTENTIONALLY.

5.8 Procedure MH_ROOF

5.8.1 GENERAL DESCRIPTION

5.8.1.1 Problem Description

This application problem is another of the MacNeal/Harder Standard Test Problems. The Scordelis-Lo Roof problem tests an element's ability to handle single curvature and distributed gravity loading.

5.8.1.2 Model Description

The Scordelis-Lo Roof is modeled for both 2-D and 3-D elements. Due to symmetry, only one-fourth of the roof is modeled. Loading is applied to the roof in the form of a distributed gravity load equal to 90.0 pounds per square foot of surface area. For higher-order elements the load is applied using consistent loading concepts. The roof is modeled with $N \times N$ elements on the quarter model. A mesh convergence study is done with N taking the values in Table 5.8-1. An example of a 2-D model, N=4, is shown in Figure 5.8-1. Material properties, boundary conditions, and dimensions are also shown in Figure 5.8-1.

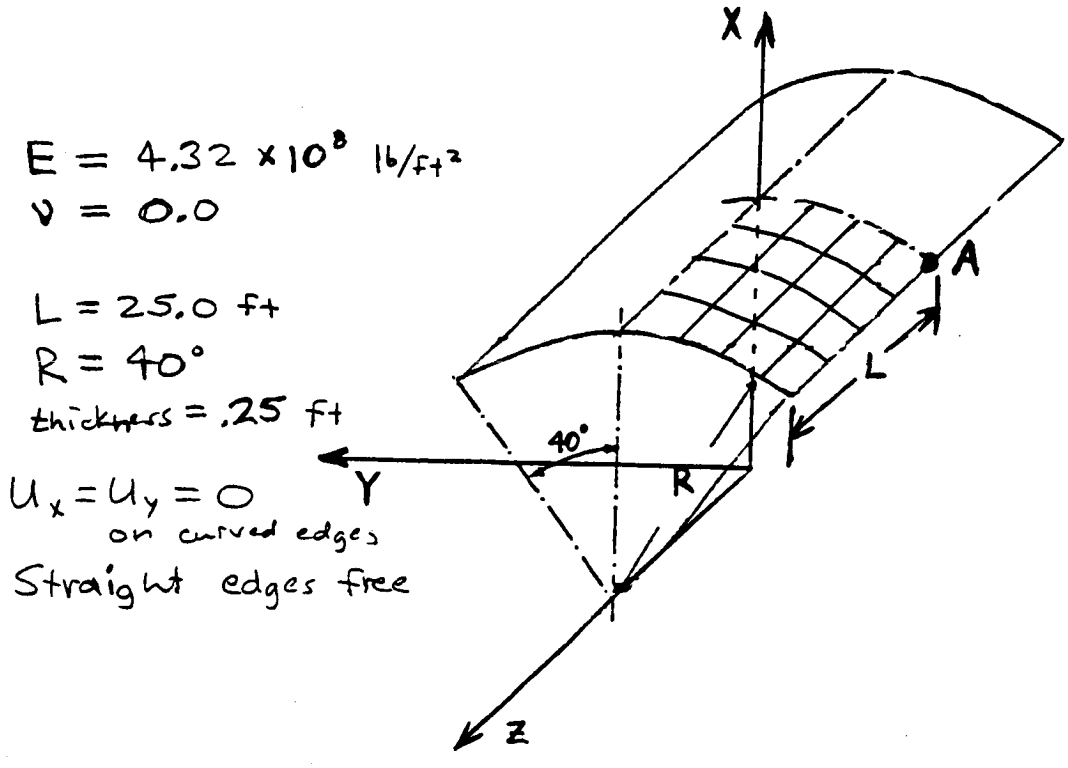


Figure 5.8-1 Example Scordelis-Lo Roof Model, $N = 4$.

5.8.1.3 Analysis Description

A static, linear elastic analysis is performed. The x-direction displacement at the midside of free edge, point A in Figure 5.8-1, is calculated, normalized with the theoretical result, and printed for each value of N (Table 5.8-1).

5.8.1.4 Available Solutions

From reference 5.8-1, the theoretical solution for the displacement in the direction of the load at the midside of the free edge is .3024 inches.

5.8.2 PROCEDURE USAGE

Procedure **MH_ROOF** may be used by preceding the procedure name by the ***call** directive, and following it by a list of arguments enclosed in parentheses. Procedure arguments are order-independent, and most have default values thus making them optional. The formal syntax is as follows:

```
*call MH_ROOF ( arg1 = val1 ; arg2 = val2 ; ...)
```

where **arg1** and **arg2** represent argument names, and **val1** and **val2** represent their corresponding values. Note that semi-colons are required between arguments, and a double dash (--) may be used to continue the argument list on the next line.

The allowable arguments for procedure **MH_ROOF** are summarized in the following table, along with their default values (if they exist). Note that arguments without defaults are generally *mandatory*, while those with defaults are generally optional. Exceptions to this rule are noted in the following section under detailed argument descriptions.

Argument	Default Value	Meaning
ES_PROC	**ES1**	Element Processor
ES_NAME	**EX47**	Element name
ES_PARS	**0.**	Element research parameters
SPAR_DIM	**none**	2-D or 3-D SPAR element
REPORT	**<true>**	Generate "report card" file
DEL_LIB	**<false>**	Delete libraries

5.8.3 ARGUMENT DESCRIPTIONS

5.8.3.1 ES_NAME

Element name (default: **EX47**). This is the name of the element type you wish to select, within the element processor defined by argument **ES_PROC**. SPAR or ES elements may be used. The elements are described in the appropriate section in The Computational Structural Mechanics Testbed User's Manual (reference 5.8-2).

5.8.3.2 ES_PROC

Element processor (default: ES1). This is the name of the element processor that contains the element you wish to evaluate in this procedure. If a SPAR element is used, ES_PROC should be set to SPAR. The element processors are described in The Computational Structural Mechanics Testbed User's Manual.

5.8.3.3 ES_PARS

Element research parameters (default: 0., ...). This is an optional list of element-dependent parameters that some elements provide, primarily when the element is still undergoing research and refinement.

5.8.3.4 SPAR_DIM

If a SPAR element is used, this parameter must be set to the number of dimensions of that element. If ES elements are used, this parameter need not be set at all.

5.8.3.5 REPORT

Print report file (default=<true>). If REPORT is <true> an external file named ROOF_<es_name>.<es_proc> will be generated. This file contains the normalized vertical displacement of the node at the midside of the free edge, point A in Figure 5.8-1.

5.8.3.6 DEL_LIB

Delete libraries (default=<false>). As the mesh convergence study progresses, libraries named ROOFN.L01 are generated, where N is the number of elements along each edge of model. If DEL_LIB is set to <true>, these libraries will be deleted during the analysis. This allows disk space to be conserved if the analyst does not need the libraries.

	2-D Model		3-D Model	
	4-node	9-node	8-node	20-node
N =	2	1	4	2
	4	2	8	4
	6	3	12	6
	8	4		
	10	5		
	12	6		

Table 5.8-1 N = number of elements for convergence study. Model is $N \times N$.

5.8.4 USAGE GUIDELINES AND EXAMPLES

Procedure **MH_ROOF** may be used by preceding the procedure name by the ***call** directive. Procedure arguments may be changed from their default values by including any or all of the arguments and their new values when the procedure is called. A space or blank is required between the end of the procedure name and the left parenthesis.

```
*procedure MH_ROOF ( ES_NAME = EX47    ; --
                     ES_PROC = ES1     ; --
                     ES_PARS = 0       ; --
                     SPAR_DIM          ; --
                     DEL_LIB = <false> ; --
                     REPORT  = <true>  )
```

(E1) To perform an entire analysis using the default options, simply invoke the procedure without any arguments, *i.e.*,

```
*call MH_ROOF
```

This will perform a static, linear elastic analysis of the MacNeal/Harder Scordelis-Lo roof using the EX47 element in processor ES1. The one-fourth roof, modeled by $N \times N$ elements, will be analyzed with N = 2, 4, 6, 8, 10, and 12. Six libraries will be created, ROOFN.L01, one for each value of N. If the argument **DEL_LIB** is **<true>**, these libraries will be automatically deleted as the analysis proceeds. If the argument **REPORT** is **<true>**, the normalized displacements at the point of, and in the direction of, the load will be printed in a report file named ROOF_EX47.ES1 for each value of N, Figure 5.8-3.

```
################################################################
################################################################
       MACNEAL-HARDER 2-D SCORDELIS-LO ROOF PROBLEM
       ELEMENT NAME =     EX47
       ELEMENT PROCESSOR =     ES1
       ELEMENT PARAMATERS =     0.
----------------- NORMALIZED  RESULTS -----------------
################################################################
          GRID = 2                  1.1225
          GRID = 4                  0.9703
          GRID = 6                  0.9884
          GRID = 8                  0.9941
          GRID = 10                 0.9971
          GRID = 12                 1.0000
################################################################
```

Figure 5.8-3 Report file generated using default arguments.

(E2) The following call statement will solve the Scordelis-Lo roof problem for the solid element EX08 in processor ES3. The libraries for each value of N will be deleted.

```
*call MH_ROOF ( ES_NAME = EX08       ; --
                ES_PROC = ES3        ; --
                DEL_LIB = <true>     ; --
```

```
                    REPORT  = <true>     )
```

(E3) To perform the analysis using a SPAR element, invoke the procedure using the appropriate 4-node 2-D element or 8-node 3-D element and SPAR for ES_PROC. Note that the argument SPAR_DIM must be defined when using a SPAR element.

```
*call MH_ROOF ( ES_NAME = S81       ; --
                ES_PROC = SPAR      ; --
                SPAR_DIM = 3        ; --
                REPORT = <true>     )
```

5.8.5 LIMITATIONS

(L1) This procedure can only evaluate 4- and 9-node 2-D elements, and 8- and 20-node solid elements.

5.8.6 ERROR MESSAGES AND WARNINGS

If a SPAR element is used, and the argument SPAR_DIM is not set, the following error message will be printed and execution terminated.

```
The argument SPAR_DIM must be defined as 2 or 3 when
  using SPAR elements.
Please set this argument and rerun.
```

5.8.7 PROCEDURE FLOWCHART

5.8.8 PROCEDURE LISTING

5.8.9 REFERENCES

5.8-1 MacNeal, R. H.; and Harder, R. L.: "A Proposed Set of Problems to Test Finite Element Accuracy," *Finite Elements in Analysis and Design*, Vol. 1, 1985, pp. 3-20.

5.8-2 Stewart, Caroline B.: *The Computational Structural Mechanics Testbed User's Manual.* NASA TM-100644, October 1989.

THIS PAGE LEFT BLANK INTENTIONALLY.

5.9 Processor SKEWED_GRID

THIS SECTION UNDER PREPARATION

THIS PAGE LEFT BLANK INTENTIONALLY.

6.0 Postprocessing Procedures

The procedures documented in this chapter are for postprocessing the results of a given computation. These procedures provide examples of stress recovery and internal force calculations as well as examples for extracting selected data from the computational database.

A summary of the procedures found in this chapter is provided in Table 6.0-1.

Table 6.0-1 Solution Procedures

Procedure Name	*Description*
HISTORY	Tabulate Response History in Database
POST	Tabulate Selected Results in Database
STRESS	Compute Stresses and/or Strains from Displacements
TOTAL_LOAD	Compute Total Load for Applied Displacement Problems

THIS PAGE LEFT BLANK INTENTIONALLY.

6.1 Procedure HISTORY

6.1.1 GENERAL DESCRIPTION

Procedure **HISTORY** is typically used to consolidate the response history of one or more nodes or elements into a database other than the database used for the analysis. The procedure allows for the extraction of a single nodal degree of freedom from one or more nodes. It also provides the means for extracting a single stress component from one or more elements.

6.1.2 PROCEDURE USAGE

Procedure **HISTORY** may be used by preceding the procedure name by the ***call** directive, and following it by a list of arguments enclosed in parentheses. Procedure arguments are order-independent, and most have default values thus making them optional. The formal syntax is as follows:

```
*call HISTORY ( arg1 = val1 ; arg2 = val2 ; ...)
```

where **arg1** and **arg2** represent argument names, and **val1** and **val2** represent their corresponding values. Note that semi-colons are required between arguments, and a double dash (--) may be used to continue the argument list on the next line.

The allowable arguments for procedure **HISTORY** are summarized in the following table, along with their default values (if they exist). Note that arguments without defaults are generally *mandatory*, while those with defaults are generally optional. Exceptions to this rule are noted in the following section under detailed argument descriptions.

For procedure **HISTORY**, the following table lists each argument, its default value and meaning.

Argument	Default Value	Meaning
COMPONENT	--	nodal dof or element stress component
ELEMENT_DS	--	name of element-list dataset
ELEMENT_RN	--	name of element-list record
ELEMENTS	--	element numbers
INPUT_DS	--	root name of input dataset
OUTPUT_DS	--	full name of output dataset
OUTPUT_RN	--	root name of output record
NODE_DS	--	name of node-list dataset
NODE_RN	--	name of node-list record
NODES	--	node numbers
NUM_ELEMENTS	--	number of elements to process
AVERAGE	<false>	average flag
INPUT_LDI	1	logical device index of input database
LOCATION	NODES	post nodal quantities
NUM_NODES	1	number of nodes to process
NUM_STEPS	1	number of steps
OUTPUT_LDI	2	logical device index of output database
STEPS	0:0	step numbers
SUM	<false>	summation flag (over nodes)

6.1.3 ARGUMENT DESCRIPTIONS

6.1.3.1 AVERAGE

Average flag (default: <false>). If <true>, an average value for specified nodes or elements will be computed and saved in the database; otherwise individual nodal or elemental values are computed and saved.

6.1.3.2 INPUT_LDI

Logical device index of input database (default: 1).

6.1.3.3 LOCATION

Identifies location of data to be processed; either NODES or ELEMENTS (default: NODES).

6.1.3.4 NUM_NODES

Number of nodes to process (default: 1). This argument is a required argument only if LOCATION=NODES.

6.1.3.5 NUM_STEPS

Number of steps for historical data recovery (default: 1).

6.1.3.6 OUTPUT_LDI

Logical device index of output database (default: 2).

6.1.3.7 STEPS

Step numbers for historical data recovery (default: 0). This argument is an array of length NUM_STEPS. If the argument STEPS=0, the linear solution is used.

6.1.3.8 SUM

Flag to perform summation (default: <false>). If <true> the quantity of interest wil be summed over either the elements or nodes as specified by LOCATIONS. It is the summed (resultant) value is then saved in the database.

6.1.3.9 COMPONENT

Column of data to be extracted. When LOCATION=NODES (*e.g.*, COMPONENT=1 corresponds to u-direction displacement). When LOCATION=ELEMENT, COMPONENT is typically a stress resultant (*e.g.*, COMPONENT=1 corresponds to N_x). This argument is always required.

6.1.3.10 ELEMENT_DS

Name of dataset containing list of elements to be processed. This argument is an active argument only if LOCATION=ELEMENTS.

6.1.3.11 ELEMENT_RN

Name of element-list record. This argument is an active argument only if LOCATION=ELEMENTS.

6.1.3.12 ELEMENTS

Element numbers to extract historical data. The argument ELEMENTS is an array of length NUM_ELEMENTS that lists the element numbers for which historical data are extracted. This is an active argument only if LOCATION=ELEMENTS. Element numbers may be input either through the ELEMENTS argument or through a dataset containing the list (*i.e.*, using the ELEMENTS_DS and ELEMENT_RN arguments.

6.1.3.13 INPUT_DS

Root name (*i.e.*, first two words) of source dataset. This argument is always required.

6.1.3.14 OUTPUT_DS

Root name (*i.e.*, first two words) of destination dataset. This argument is always required. The dataset is written one record per step per element for LOCATION=ELEMENTS or one record per step per node for LOCATION=NODES.

6.1.3.15 OUTPUT_RN

Root name (*i.e.*, first word) of destination record. This argument is always required.

6.1.3.16 NODE_DS

Name of dataset containing list of nodes to be processed. This argument is an active argument only if **LOCATION=NODES**.

6.1.3.17 NODE_RN

Name of node-list record. This argument is an active argument only if **LOCATION=NODES**.

6.1.3.18 NODES

Node numbers to extract historical data. The argument **NODES** is an array of length **NUM_NODES** that lists the node numbers for which historical data are extracted. This is an active argument only if **LOCATION=NODES**. Node numbers may be input either through the **NODES** argument or through a dataset containing the list (*i.e.*, using the **NODE_DS** and **NODE_RN** arguments).

6.1.3.19 NUM_ELEMENTS

Number of elements to extract historical data. The argument **NUM_ELEMENTS** specifies the number of elements from which historical data are extracted. This is a required argument only if **LOCATION=ELEMENTS**.

6.1.4 USAGE GUIDELINES AND EXAMPLES

Procedure **HISTORY** may be used by preceding the procedure name by the ***call** directive. Procedure arguments may be changed from their default values by including any or all of the arguments and their new values when the procedure is called. A space or blank is required between the end of the procedure name and the left parenthesis. If the default values of the procedure arguments are to be used, then only the procedure name is required.

```
*procedure HISTORY ( input_ldi = 1 ; -- . ldi of input database
                     input_ds      ; -- . root name of input dataset
                     output_ldi = 2 ; -- . ldi of output database
                     output_ds     ; -- . full name of output dataset
                     output_rn     ; -- . root name of output record
                     location = NODES ; -- . post nodal quantities
                     num_steps  = 1 ; -- . number of steps
                     steps     = 0:0; -- . step numbers
                     num_nodes  = 1 ; -- . number of nodes to process
                     nodes         ; -- . node numbers
                     component     ; -- . nodal dof or element stress
                     sum = <false>  ; -- . summation flag (over nodes)
                     average = <false>  ; -- . average flag
                     node_ds       ; -- . name of node-list dataset
```

```
node_rn          ; -- . name of node-list record
element_ds       ; -- . name of element-list dataset
element_rn       ; -- . name of element-list record
num_elements     ; -- . number of elements to process
elements           -- . element numbers)
```

6.1.5 LIMITATIONS

Due to a restriction on the number of active macrosymbols, no more than 700 nodes or elements may be processed using NODE_DS or ELEMENT_DS.

Procedure HISTORY should be used only to process limited amounts of data due to the procedure's inherent inefficiency. Except for small problems, the SUM argument should not be used to perform summation over the nodes for multiple load steps. In most cases this operation should be performed using the TOTAL_LOAD procedure (see Section 6.5).

6.1.6 ERROR MESSAGES AND WARNINGS

None.

6.1.7 PROCEDURE FLOWCHART

6.1.8 PROCEDURE LISTING

```
*procedure HISTORY ( input_ldi  = 1 ; -- . ldi of input database
                     input_ds       ; -- . root name of input dataset
                     output_ldi = 2 ; -- . ldi of output database
                     output_ds      ; -- . full name of output dataset
                     output_rn      ; -- . root name of output record
                     location = NODES ; -- . post nodal quantities
                     num_steps  = 1 ; -- . number of steps
                     steps      =0:0; -- . step numbers
                     num_nodes  = 1 ; -- . number of nodes to process
                     nodes          ; -- . node numbers
                     component      ; -- . nodal dof or element stress
                     sum = <false>  ; -- . summation flag (over nodes)
                     average = <false>  ; -- . average flag
                     node_ds        ; -- . name of node-list dataset
                     node_rn        ; -- . name of node-list record
                     element_ds     ; -- . name of element-list dataset
                     element_rn     ; -- . name of element-list record
                     num_elements   ; -- . number of elements to process
                     elements         -- . element numbers
                   )
.
. ------------------------------------------------------
. Procedure to Postprocess Response History from/to Database
```

```
. ------------------------------------------------------
.
  *def/i steps[1:[num_steps]] = [steps]
  *if < <IFELSE([LOCATION];NODES;1;0)> > /then
     *if < <IFELSE([node_ds]; ;1;0)> > /then
        *def/i num_nodes = [num_nodes]
        *def/i nodes[1:<num_nodes>] == [nodes]
     *else
        *find record [input_ldi], [node_ds], [node_rn] /nor=num_nodes
        *g2m /name==nodes /type=i /maxn=<num_nodes> [input_ldi] [node_ds] --
             [node_rn].1:<num_nodes>
     *endif
    [IQT VEC
     *find dataset [output_ldi] [output_ds] /seq=ids
     *if <<ids> /le 0 > /then
        *put dataset [output_ldi] [output_ds] /mrat=2000 /seq=ids
     *endif
     *do $is = 1, [num_steps]
        *def/i step_num = <steps[<$is>]>
        *find dataset [input_ldi] [input_ds].<step_num> /seq=hist_ids
        *if < <hist_ids> /le 0 > /then
           *remark
           *remark Dataset [input_ds].<step_num> not found by HISTORY
           *remark
           *return
        *endif
        *if < [sum] /or [average] > /then
           *def/e resultant = 0.0
        *endif
        *do $in = 1, <num_nodes>
           *def/i node_num = <nodes[<$in>]>
        COMPONENT <node_num> [component] [input_ds].<step_num> -> val
           *if < [sum] /or [average] > /then
              *def/e resultant = < <resultant> + <val> >
           *else
              *m2g /name=val /type=d [output_ldi] <ids>  --
                   [output_rn].<step_num>
           *endif
        *enddo
        *if < [sum] /or [average] > /then
           *if < [average] > /then
              *def/e resultant = < <resultant> / <num_nodes> >
           *endif
           *m2g /name=resultant /type=d [output_ldi] <ids>  --
                [output_rn].<step_num>
        *endif
     *enddo
  *elseif < <IFELSE([LOCATION];ELEMENTS;1;0)> > /then
     *if < <IFELSE([element_ds]; ;1;0)> > /then
        *def/i num_elements = [num_elements]
        *def/i elements[1:<num_elements>] == [elements]
     *else
        *find record [input_ldi], [element_ds], [element_rn] --
```

```
                                                /nor=num_elements
        *g2m /name=elements /type=i /maxn=<num_elements> [input_ldi] --
             [element_ds] [element_rn].1:<num_elements>
     *endif
     [XQT VEC
     *do $is = 1, [num_steps]
        *def/i step_num = <steps[<$is>]>
        *find dataset [input_ldi] [input_ds].<step_num> /seq=hist_ids
        *if < <hist_ids> /le 0 > /then
           *remark
           *remark Dataset [input_ds].<step_num> not found by HISTORY
           *remark
           *return
        *endif
        *if < [sum] /or [average] > /then
           *def/e resultant = 0.0
        *endif
        *do $in = 1, <num_elements>
           *def/i element_num = <elements[<$in>]>
        COMPONENT <element_num> [component] [input_ds].<step_num> -> val
           *if < [sum] /or [average] > /then
              *def/e resultant = < <resultant> + <val> >
           *else
              *m2g /name=val /type=d [output_ldi] [output_ds] --
                   [output_rn].<step_num>
           *endif
        *enddo
        *if < [sum] /or [average] > /then
           *if < [average] > /then
              *def/e resultant = < <resultant> / <num_elements> >
           *endif
           *m2g /name=resultant /type=d [output_ldi] [output_ds] --
                [output_rn].<step_num>
        *endif
     *enddo
  *endif
*end
```

6.1.9 REFERENCES

6.1-1 Stewart, Caroline B.: *The Computational Structural Mechanics Testbed User's Manual.* NASA TM-100644, October 1989.

6.1-2 Felippa, Carlos A.: *The Computational Structural Mechanics Testbed Architecture: Volume II - Directives.* NASA CR 178385, February 1989.

THIS PAGE LEFT BLANK INTENTIONALLY.

6.2 Procedure POST

6.2.1 GENERAL DESCRIPTION

Procedure POST is used to extract the response history of one or more nodes from one database and to consolidate it into another database. This procedure differs from procedure HISTORY in that POST **cannot** operate on element data (*e.g.*, element stress resultants).

6.2.2 PROCEDURE USAGE

Procedure POST may be used by preceding the procedure name by the *call directive, and following it by a list of arguments enclosed in parentheses. Procedure arguments are order-independent, and most have default values thus making them optional. The formal syntax is as follows:

```
*call POST ( arg1 = val1 ; arg2 = val2 ; ...)
```

where arg1 and arg2 represent argument names, and val1 and val2 represent their corresponding values. Note that semi-colons are required between arguments, and a double dash (--) may be used to continue the argument list on the next line.

The allowable arguments for procedure POST are summarized in the following table, along with their default values (if they exist). Note that arguments without defaults are generally *mandatory*, while those with defaults are generally optional. Exceptions to this rule are noted in the following section under detailed argument descriptions.

For procedure POST, the following table lists each argument, its default value and meaning.

Parameter	Default Value	Meaning
DOF	--	Nodal degree of freedom index
INPUT_DS	--	Root name of input dataset
NODES	--	Node numbers
NODES_DS	--	Name of node-list input dataset
NODES_RN	--	Name of node-list input record
OUTPUT_DS	--	Full name of output dataset
OUTPUT_RN	--	Root name of output record
INPUT_LDI	1	Logical device index of input database
OUTPUT_LDI	1	Logical device index of output database
NSTEPS	1	Number of steps
STEPS	0:0	Step numbers
NNODES	1	Number of nodes to process
SUM	<false>	Summation flag (over node list)

6.2.3 ARGUMENT DESCRIPTIONS

6.2.3.1 INPUT_LDI

Logical device index of input database (default: 1).

6.2.3.2 OUTPUT_LDI

Logical device index of output database (default: 1).

6.2.3.3 NSTEPS

Number of steps for historical data recovery (default: 1).

6.2.3.4 STEPS

Step numbers for historical data recovery (default: 0). This argument is an array of length NUM_STEPS. If the argument STEPS=0, the linear solution is used.

6.2.3.5 NNODES

Number of nodes to process (default: 1).

6.2.3.6 SUM

Flag to sum over nodes (default: <false>). If the argument SUM=<true>, the summation and not the nodal values will be saved in the database.

6.2.3.7 DOF

Nodal degree of freedom index (*i.e.*, 1, 2, 3, 4, 5, or 6).

6.2.3.8 INPUT_DS

Root name (first two words) of source dataset. This argument is mandatory.

6.2.3.9 NODES

Node numbers for which historical data are extracted. The argument NODES is an array of length NODES. Node numbers may be input either through the NODES argument or through a dataset containing the list (*i.e.*, using the NODES_DS and NODES_RN arguments).

6.2.3.10 NODES_DS

Name of dataset containing list of nodes to be processed.

6.2.3.11 NODES_RN

Name of node-list record.

6.2.3.12 OUTPUT_DS

Root name (*i.e.*, first two words) of destination dataset. This argument is mandatory. The dataset is written one record per step per node.

6.2.3.13 OUTPUT_RN

Root name (*i.e.*, first word) of destination record. This argument is mandatory.

6.2.4 USAGE GUIDELINES AND EXAMPLES

Procedure POST may be used by preceding the procedure name by the *call directive. Procedure arguments may be changed from their default values by including any or all of the arguments and their new values when the procedure is called. A space or blank is required between the end of the procedure name and the left parenthesis. If the default values of the procedure arguments are to be used, then only the procedure name is required.

```
*call POST ( input_ldi   = 1 ; -- . ldi of input database
             input_ds        ; -- . root name of input dataset
             output_ldi = 1  ; -- . ldi of output database
             output_ds       ; -- . full name of output dataset
             output_rn       ; -- . root name of output record
             nsteps     = 1  ; -- . number of steps
             steps      =0:0 ; -- . step numbers
             nnodes     = 1  ; -- . number of nodes to process
             nodes           ; -- . node numbers
             dof             ; -- . nodal degree of freedom index
             sum = <false>   ; -- . summation flag (over node list)
             nodes_ds        ; -- . [name of node-list input dataset]
             nodes_rn          -- . [name of node-list input record])
```

6.2.5 LIMITATIONS

Due to a restriction on the number of active macrosymbols, no more than 700 nodes should be processed using NODE_DS and NODE_RN.

6.2.6 ERROR MESSAGES AND WARNINGS

None.

6.2.7 PROCEDURE FLOWCHART

Procedure POST is self-contained and has no subprocedures.

6.2.8 PROCEDURE LISTING

```
*procedure POST ( input_ldi  = 1 ; -- . ldi of input database
                  input_ds       ; -- . root name of input dataset
                  output_ldi = 1 ; -- . ldi of output database
                  output_ds      ; -- . full name of output dataset
                  output_rn      ; -- . root name of output record
                  nsteps     = 1 ; -- . number of steps
                  steps      =0:0; -- . step numbers
                  nnodes     = 1 ; -- . number of nodes to process
                  nodes          ; -- . node numbers
                  dof            ; -- . nodal degree of freedom index
                  sum = <false>  ; -- . summation flag (over node list)
                  nodes_ds       ; -- . [name of node-list input dataset]
                  nodes_rn         -- . [name of node-list input record]
                )
.
. ------------------------------------------------------
. CLAMP Procedure to Post Response History in a Database
. ------------------------------------------------------
.
   *def/i steps[1:[nsteps]] = [steps]
   *if < <IFELSE([nodes_ds]; ;1;0)> > /then
      *def/i nnodes = [nnodes]
      *def/i nodes[1:<nnodes>] = [nodes]
   *else
      *find record [input_ldi], [nodes_ds], [nodes_rn] /nor=nnodes
      *do $in = 1, <nnodes>
         *g2m /name=nodes[<$in>] /type=i [input_ldi] [nodes_ds] --
              [nodes_rn].<$in>
      *enddo
   *endif
  [XQT VEC
  *find dataset [output_ldi] [output_ds] /seq=ids
  *if < <ids> /le 0 > /then
     *put dataset [output_ldi] [output_ds] /mrat=2000 /seq=ids
  *endif
   *do $is = 1, [nsteps]
      *def/i step_num = <steps[<$is>]>
      *if < [sum] > /then
         *def/e resultant = 0.0
      *endif
      *do $in = 1, <nnodes>
         *def/i node_num = <nodes[<$in>]>
          COMPONENT <node_num> [dof] [input_ds].<step_num>  ->  value
         *if < [sum] > /then
            *def/e resultant = < <resultant> + <value> >
         *else
            *m2g /name=value /type=d [output_ldi] [output_ds] --
                 [output_rn]_<node_num>.<step_num>
         *endif
      *enddo
      *if < [sum] > /then
```

```
        *m2g /name=resultant /type=d [output_ldi] [output_ds] --
             [output_rn].<step_num>
      *endif
   *enddo
*end
```

6.2.9 REFERENCES

6.2-1 Stewart, Caroline B.: *The Computational Structural Mechanics Testbed User's Manual.* NASA TM-100644, October 1989.

6.2-2 Felippa, Carlos A.: *The Computational Structural Mechanics Testbed Architecture: Volume II - Directives.* NASA CR 178385, February 1989.

THIS PAGE LEFT BLANK INTENTIONALLY.

6.3 Procedure STRESS

6.3.1 GENERAL DESCRIPTION

Procedure STRESS drives the calculation of stress resultant and strain component datasets for either linear or nonlinear analyses. The user may select recovery of stress resultants or strain components at element centroids, element nodes, element integration points or at all three locations.

6.3.2 PROCEDURE USAGE

Procedure STRESS may be used by preceding the procedure name by the *call directive, and following it by a list of arguments separated by semicolons (;) and enclosed in parentheses. Procedure arguments are order-independent, and most have default values thus making them optional. The formal syntax is as follows:

```
*call STRESS ( arg1 = val1 ; arg2 = val2 ; ...)
```

For procedure STRESS, the following table lists each argument, its default value and meaning.

Argument	Default Value	Meaning
CONS_SET	1	Constraint set number
COROTATION	1	Corotational flag (0=OFF, 1=ON)
DIM	2	Intrinsic element spatial dimension (2=2D, 3=3D)
DIRECTION	1	Direction for element stress and strain recovery
DISP_DS	STAT.DISP	First two words of displacement dataset name
LOAD_SET	1	Load set number
LOCATION	NODES	Location of the evaluation points for element stresses
NL_GEOM	0	Geometric nonlinearity flag
NUM_STEPS	1	Total number of steps requiring stress calculations
NVAL_METH	3	Method to be used for global smoothing
PRINT	<false>	Flag to print computed results
ROTN_DS	STAT.ROTN	First two words of rotation-vector dataset name
SMOOTH	<false>	Flag to compute smoothed global stresses
STEPS	0	Step numbers
STRAIN	<false>	Flag to compute element strains
STRESS	<true>	Flag to compute element stresses

Tables 6.3-1, 6.3-2, and 6.3-3 list the datasets used or created by procedure STRESS, the procedures invoked by procedure STRESS, and the processors invoked by procedure STRESS, respectively.

Table 6.3-1 Datasets Input/Output by procedure STRESS				
Dataset	*Description*	*Lib*	*Input*	*Output*
ES.SUMMARY	ES Processor Status	1	√	√
PROP.BTAB.*	Material/Section Properties	1	√	
STAT.DISP.*i.j*†	System Displacement Vector	1		√
STRS.<ES_NAME>.*i.j*† =	Element Stresses	1		√

† i = <load_set> and j = <cons_set>

Table 6.3-2 Sub-Procedures Invoked by procedure STRESS		
Procedure	*Type*	*Function*
ES	External	Element utility procedure
STRESS	Internal	Computes element stresses

Table 6.3-3 Processors Invoked by procedure STRESS		
Procedure	*Type*	*Function*
ES*i*	External	Element processors based on GEP
NVAL	Internal	Perform global smoothing
PESR	Internal	Print element stresses
PNSR	Internal	Print nodal stresses

6.3.3 ARGUMENT DESCRIPTIONS

6.3.3.1 CONS_SET

Constraint set number (default: 1). This argument selects which constraint set to use in solving the linear system of equations.

6.3.3.2 COROTATION

Corotational update switch for large-rotation problems (default: <true>). This switch should be set to <true> when the model involves finite elements that require corotation for geometric nonlinearity. This is true of most beam and shell elements, and may be true for some solid (3D) elements used to model shell structures. Consult the appropriate element processor (ES*i*) section in the CSM Testbed User's Manual (see ref. 6.3-1) for specific guidelines.

6.3.3.3 DIM

Intrinsic element spatial dimension (default: 2). This argument selects the element spatial dimension: 2 for a two-dimensional element or 3 for a three-dimensional solid element.

6.3.3.4 DIRECTION

Direction for the element stress (stress resultant) output (default: 1). The element stress coordinate system will be used if DIRECTION=0. The material axes (x_m, y_m, z_m) will be used if DIRECTION=1; the material axes (y_m, z_m, x_m) will be used for DIRECTION=2; and the material axes (z_m, x_m, y_m) will be used for DIRECTION=3. For isotropic materials, the first material axis is replaced by the corresponding global axis (see Section 4.3.3.9 of the CSM Testbed User's Manual, ref. 6.3-1).

6.3.3.5 DISP_DS

First two words of the dataset name for the displacement solution (default: STAT.DISP).

6.3.3.6 LOAD_SET

Load set number (default: 1). This argument selects which load set to use as a right-hand side vector.

6.3.3.7 LOCATION

Location of the evaluation points for the element stresses or stress resultants (default: NODES). The element stresses or stress resultants are optionally computed. This argument may have four values. For LOCATION=INTEG_PTS, the element stresses are computed at the element integration points. For LOCATION=CENTROIDS, the element stresses are computed at the element centroid. For LOCATION=NODES, the element stresses are extrapolated from the integration points to be element nodes. These element nodal stresses will be discontinuous across interelement boundaries. For LOCATION=ALL, the element stresses are computed at the element integration points, element centroid, and element nodes.

6.3.3.8 NL_GEOM

Geometric nonlinearity level: 0, 1, or 2 (default =2). A value of zero means that the problem is geometrically linear; a value of one means that the geometric nonlinearity will be handled globally (*i.e.*, using corotational updates only); and a value of two means that

the nonlinear element strain-displacement relations will be used in addition to any global treatment of geometric nonlinearity. If COROTATION = <true>, options 1 and 2 refer to first-order and second-order corotation, respectively. The latter option can be significantly more accurate than the former for a given finite element model — depending on which element types are involved.

6.3.3.9 NUM_STEPS

Number of steps for stress recovery (default: 1).

6.3.3.10 NVAL_METH

Select method to be used for computing the smoothed global stresses (default: 3). Processor NVAL is used to compute the smoothed global stresses using the method defined by the argument NVAL_METH (see Section 12.5 of reference 6.3-1). If NVAL_METH=1, a topological interpolation of the element centroidal stresses is performed, and the element stresses must have been computed using LOCATION=CENTROIDS. If NVAL_METH=2, a projected least-squares interpolation of the element centroidal stresses is performed, and the stresses must have been computed using LOCATION=CENTROIDS. If NVAL_METH=3, the element nodal stresses (discontinuous across interelement boundaries) are averaged, and the element stresses must have been computed using LOCATION=NODES. Using LOCATION=ALL will generate element stresses at the element centroids, element nodes, and element gauss points. Acceptable values of location for specific values of NVAL_METH are as follows:

NVAL_METH	Location
1	CENTROIDS, ALL
2	CENTROIDS, ALL
3	NODES, ALL

6.3.3.11 PRINT

Flag to print computed results (default: <false>). If the argument PRINT=<true>, then the element stresses and/or nodal stresses will be printed.

6.3.3.12 ROTN_DS

First two words of the dataset name for the rotation solution (default: STAT.ROTN).

6.3.3.13 SMOOTH

Flag to compute smoothed global stresses (default: <false>). If the argument SMOOTH=<true>, then smoothed global stresses will be computed by processor NVAL using the method defined by the argument NVAL_METH.

6.3.3.14 STEPS

Step numbers for stress recovery (default: 0). This argument is an array of length NUM_STEPS. If the argument STEPS=0, the linear solution is used.

6.3.3.15 STRAIN

Flag to compute element stresses or stress resultants (default: <false>). If the argument STRAIN=<true>, then the element stresses will be computed at the location and in the direction specified by the arguments LOCATION and DIRECTION.

6.3.3.16 STRESS

Flag to compute element stresses or stress resultants (default: <false>). If the argument STRESS=<true>, then the element stresses will be computed at the location and in the direction specified by the arguments LOCATION and DIRECTION.

6.3.4 USAGE GUIDELINES AND EXAMPLES

Procedure STRESS may be used by preceding the procedure name by the *call directive. Procedure arguments may be changed from their default values by including any or all of the arguments and their new values when the procedure is called. A space or blank is required between the end of the procedure name and the left parenthesis. If the default values of the procedure arguments are to be used, then only the procedure name is required.

```
*call STRESS ( STRESS    = <true>       ; --
               STRAIN    = <false>      ; --
               LOCATION  = CENTROIDS ; --
               DIRECTION = 0 ; --
               NL_GEOM   = 0 ; --
               CORO      = 1 ; --
               LOAD_SET  = 1 ; --
               CONS_SET  = 1 ; --
               NUM_STEPS = 1 ; --
               PRINT     = <false> ; --
               NVAL_METH = 3 ; --
               SMOOTH    = <false> ; --
               DIM       = 2 ; --
               STEPS     = 0 ; --
               DISP_DS   = STAT.DISP ; --
               ROTN_DS   = STAT.ROTN )
```

6.3.5 LIMITATIONS

Applicable only to elements implemented using the generic element processor template. Procedure STRESS assumes that all datasets either required or generated reside on library one (LDI = 1).

6.3.6 ERROR MESSAGES AND WARNINGS

An error message is printed and global stress smoothing will not be performed if the values of NVAL_METH and LOCATION are inconsistent. Processing of element data will continue. See Section 12.5 of reference 6.3-1 for usage of processor NVAL.

6.3.7 PROCEDURE FLOWCHART

STRESS	(main procedure)
ES	(calculate element stresses and/or strains)

6.3.8 PROCEDURE LISTING

```
*procedure STRESS ( STRESS = <true>       ; --
                    STRAIN = <false>      ; --
                    LOCATION  = CENTROIDS ; --
                    DIRECTION = 0 ; --
                    NL_GEOM   = 0 ; --
                    CORO      = 1 ;print=<false>; --
                    LOAD_SET  = 1 ; --
                    CONS_SET  = 1 ; --
                    NUM_STEPS = 1 ; --
                    SMOOTH    = <false>; NVAL_METH = 3; --
                    DIM       = 2 ; --
                    STEPS     = 0 ; --
                    DISP_DS   = STAT.DISP ; --
                    ROTN_DS   = STAT.ROTN   --
                  )
.
*remark ***************
*remark STRESS RECOVERY
*remark ***************
.
.  ----------------------
.  Loop on Solution Steps
.  ----------------------
   *def/i steps[1:[num_steps]]  =  [steps]
.
   *do $is = 1, [num_steps]
      *def/i istep = <steps[<$is>]>
      *if <<istep> /ne 0> /then
       *def/a DS_SFX = <istep>
       *else
       *def/a DS_SFX = [load_set].[cons_set]
      *endif
 *if < <[STRAIN] /ne <false>> /or <[STRESS] /ne <false>> > /then
.     ------------------------------------------------
.     Define Element Stress/Strain Option and Dataset Names
.     ------------------------------------------------
```

```
      *if < <[STRAIN] /eq <true>> /and <[STRESS] /eq <false>> > /then
         *def/p function  = 'FORM STRAIN'
         *def/a STRAIN_DS = <DS_SFX>
         *def/a STRESS_DS = ' '
      *elseif < <[STRAIN] /eq <true>> /and <[STRESS] /eq <true>> > /then
         *def/p function  = 'FORM STRESS'
         *def/a STRAIN_DS = <DS_SFX>
         *def/a STRESS_DS = <DS_SFX>
      *elseif < <[STRAIN] /eq <false>> /and <[STRESS] /eq <true>> > /then
         *def/p function = 'FORM STRESS'
         *def/a STRAIN_DS = ' '
         *def/a STRESS_DS = <DS_SFX>
      *endif
.     ------------------------------------------------
.     Invoke Element Processors to Form Stress/Strain
.     ------------------------------------------------
      *if < ifeqs( [location];ALL) > /then
.
.      Calculate the stresses/strains at element nodes, centroids,
.      and gauss points
.
 *do $iloc = 1,3
     *if < ifeqs( <$iloc>;1) > /then
      *def/a location = CENTROIDS
      *elseif < ifeqs( <$iloc>;2) > /then
      *def/a location = NODES
      *elseif < ifeqs( <$iloc>;3) > /then
      *def/a location = INTEG_PTS
     *endif
          *call ES ( function = <function>          ; --
                     es_nl_geom = [nl_geom]         ; --
                     es_coro    = [coro]            ; --
                     es_dis_ds = [DISP_DS].<DS_SFX> ; --
                     es_rot_ds = [ROTN_DS].<DS_SFX> ; --
                     es_str_dir = [DIRECTION]       ; --
                     es_str_loc = <location>        ; --
                     es_strain_ds = <strain_ds>     ; --
                     es_stress_ds = <stress_ds>         )
 *enddo
       *else
.
.     Calculate the stresses/strains only at [LOCATION]
.
      *call ES ( function = <function>          ; --
                 es_nl_geom = [nl_geom]         ; --
                 es_coro    = [coro]            ; --
                 es_dis_ds = [DISP_DS].<DS_SFX> ; --
                 es_rot_ds = [ROTN_DS].<DS_SFX> ; --
                 es_str_dir = [DIRECTION]       ; --
                 es_str_loc = [LOCATION]        ; --
                 es_strain_ds = <strain_ds>     ; --
                 es_stress_ds = <stress_ds>         )
     *endif
```

```
.
.     Print the Element Stresses
.
   *if < [print] > /then
     *if < ifeqs([location];ALL) > /then
      *def/i stress_loc = 0
     *elseif < ifeqs([location];CENTROIDS) > /then
      *def/i stress_loc = 1
     *elseif < ifeqs([location];NODES) > /then
      *def/i stress_loc = 2
     *elseif < ifeqs([location];INTEG_PTS) > /then
      *def/i stress_loc = 3
     *endif
     [xqt pesr
      SELECT /global=[direction] /LOCATION=<stress_loc>
      *if < <istep> /eq 0> /then
      PRINT /LOAD_SET=[load_set] /CONSTRAINT=[cons_set]
      *else
      PRINT /LOAD_SET=<istep> /CONSTRAINT=0
      *endif
      STOP
   *endif
*endif
.
.     Calculate nodal stresses using processor NVAL
.
*if < [SMOOTH] > /then
      *if < ifeqs( [LOCATION];ALL) > /then
      *elseif /then < <[NVAL_METH] /eq 3> /and --
                    <ifeqs([LOCATION];NODES)> >
      *elseif /then < <[NVAL_METH] /ne 3> /and --
                    <ifeqs([LOCATION];CENTROIDS)> >
      *else
      *remark ************ERROR MESSAGE******************
      *remark Stress evaluation point not consistent with
      *remark stress smoothing method selected.
      *remark ******************************************
      *jump :EXIT
      *endif
      *def/a STRS = STRS.<es_name>.<stress_ds>
      [xqt NVAL
       reset meth=[NVAL_METH], DIM=[DIM], DS=<STRS>
       reset first=1, last=<es_nstr>
       *if < [NVAL_METH] /eq 3 > /then
       *def/a REC = NODES_S[direction]
       *else
       *def/a REC = CENTROIDS_S[direction]
       *endif
       reset RECNAME=<REC>
      STOP
.
.     Print the Nodal Stresses
.
```

```
   *if < [print] > /then
     [xqt pnsr
      *if < <istep> /eq 0 > /then
      PRINT /load=[load_set] /constraint=[cons_set] /method=[nval_meth]
      *else
      PRINT /load=<istep> /constraint=0 /method=[nval_meth]
      *endif
      STOP
      :EXIT
   *endif
*endif
.
*enddo
*end
```

6.3.9 REFERENCES

6.3-1 Stewart, Caroline B.: *The Computational Structural Mechanics Testbed User's Manual.* NASA TM-100644, October 1989.

THIS PAGE LEFT BLANK INTENTIONALLY.

6.4 Procedure TOTAL_LOAD

6.4.1 GENERAL DESCRIPTION

Procedure TOTAL_LOAD sums the reaction forces to determine the total applied load for problems with applied displacements for either linear or nonlinear analyses. The computed total load may be divided by a constant and scaled by a second constant.

6.4.2 PROCEDURE USAGE

Procedure TOTAL_LOAD may be used by preceding the procedure name by the *call directive, and following it by a list of arguments separated by semicolons (;) and enclosed in parentheses. Procedure arguments are order-independent, and most have default values thus making them optional. The formal syntax is as follows (see example usage in Section 6.4.4):

```
*call TOTAL_LOAD ( arg1 = val1 ; arg2 = val2 ; ...)
```

For procedure TOTAL_LOAD, the following table lists each argument, its default value and meaning.

Argument	Default Value	Meaning
INPUT_LDI	1	Input library number
OUTPUT_LDI	1	Output library number
NL_RES_LDI	3	Library containing nonlinear results dataset
LOAD_SET	1	Load set number
CONS_SET	1	Constraint set number
NSTEPS	0	Total number of load steps to be processed
BEGIN_STEP	1	First load step to be processed
END_STEP	-1	Last load step to be processed
LD_DIR	1	Load direction
EQUIL_CK	0	Flag enabling an equilibrium check
EA	1.0	Scaling constant
FAC	2.0	Scaling constant
NAME1	'STAT'	First word of input dataset name
NAME2	'REAC'	Second word of input dataset
INPUT_DS	' '	Full name of input dataset
OUTPUT_DS	' '	Full name of output dataset
NL_RES_DS	' '	Full name of dataset containing nonlinear results
INPUT_REC	'DATA'	Input dataset record key
OUTPUT_REC	' '	Output record key
OUTPUT_FILE	' '	Output file name

Table 6.4-1 lists the datasets used by procedure TOTAL_LOAD.

Table 6.4-1 Datasets Input/Output by procedure TOTAL_LOAD				
Dataset	*Description*	*Lib*	*Input*	*Output*
STAT.REAC.*i.j*†	Reaction Force Vector	INPUT_LDI	√	
JDF1.BTAB.1.8	Number of nodes and active d.o.f.	INPUT_LDI	√	

† *i* = <load_set> and *j* = <cons_set> for linear analyses
i = <step> and j=0 for nonlinear analyses

6.4.3 ARGUMENT DESCRIPTIONS

6.4.3.1 INPUT_LDI

Input library number (default: 1).

6.4.3.2 OUTPUT_LDI

Output library number (default: 1). This argument sets the number of the output data library if an output dataset name (OUTPUT_DS) has been set. If an output dataset name has not been specified, this argument is ignored.

6.4.3.3 NL_RES_LDI

Library containing nonlinear results dataset (default: 3). This argument is used only if the number of nonlinear load steps is to be calculated automatically (*i.e.*, NSTEPS = ALL). In this case, the number of load steps is calculated based on the number of records found in the results dataset contained in the NL_RES_LDI library.

6.4.3.4 LOAD_SET

Load set number (default: 1). This argument selects the load set to be used as a right-hand side vector. LOAD_SET is an active argument only if NSTEPS = 0 and END_STEP = -1 (the default values).

6.4.3.5 CONS_SET

Load set number (default: 1). This argument selects the constraint set to be used in the solution of the linear system of equations. CONS_SET is an active argument only if NSTEPS = 0 and END_STEP = -1 (the default values).

6.4.3.6 NSTEPS

Number of steps to be processed (default: 0). The number of load steps may be set to an integer or to the character string ALL. If NSTEPS = ALL, the number of load steps to be processed is calculated based on the information contained in the NL_RES_DS in library NL_RES_LDI. For use in linear static analyses, NSTEPS must be set to zero.

6.4.3.7 BEGIN_STEP

First load step to be processed (default: 1). This argument is active only if END_STEP is greater than zero in which case, load steps BEGIN_STEP through END_STEP will be processed.

6.4.3.8 END_STEP

Last load step to be processed (default: -1). This argument is active only if it is greater than zero in which case, load steps BEGIN_STEP through END_STEP will be processed.

6.4.3.9 LD_DIR

Load direction (default: 1). This argument defines the direction in which to perform the summation. The value of LD_DIR must be an active degree of freedom (*i.e.*, 1, 2, 3, 4, 5, or 6 and not specified as globally constrained in TAB/START).

6.4.3.10 EQUIL_CK

Flag enabling an equilibrium check (default: 0). The total load is calculated by summing the absolute value of the nodal contributions in the direction of LD_DIR. When EQUIL_CK is set to one (1), the absolute value is not used when performing the summation.

6.4.3.11 EA

Scaling factor (default: 1.0). Frequently the total load may be normalized by the extensional stiffness or critical buckling load. The total will be divided by the scaling factor EA before being saved or written out.

6.4.3.12 FAC

Scaling factor (default: 2.0). The default value of 2.0 will be used in most applications where the total load is being calculated. This is due to the use of the absolute value in the summation which essentially doubles the applied load. Frequently for finite element models with symmetry conditions imposed, the total load is only half the value of the summed reaction forces. The total load will be divided by the scaling factor FAC before being saved or written out.

6.4.3.13 NAME1

First word of reaction force dataset name (default: STAT). This is an active argument only if INPUT_DS = ' ' (the default value).

6.4.3.14 NAME2

Second word of reaction force dataset name (default: REAC). This is an active argument only if INPUT_DS = ' ' (the default value).

```
*call TOTAL_LOAD ( --
                   nsteps      = ALL    ; -- . process all load steps
                   ld_dir      = 3      ; -- . direction of load is 3
                   name1       = 'REAC' ; -- . first word of input dataset
                   name2       = 'FORC' ; -- . second word of input dataset
                   output_file = 'sum_load.dat' )    . output file name
```

6.4.5 LIMITATIONS

None.

6.4.6 ERROR MESSAGES AND WARNINGS

None.

6.4.7 PROCEDURE FLOWCHART

Procedure TOTAL_LOAD is self-contained and calls no other procedures.

6.4.8 PROCEDURE LISTING

```
*procedure TOTAL_LOAD ( --
          input_ldi   = 1      ; -- . Input library
          output_ldi  = 1      ; -- . Output library
          nl_res_ldi  = 3      ; -- . Library containing nonlinear results ds
          load_set    = 1      ; -- . load set (linear only)
          cons_set    = 1      ; -- . constraint set (linear only)
          nsteps      = 0      ; -- . number of load steps to process
          begin_step  = 1      ; -- . first load step to process
          end_step    = -1     ; -- . last load step to process
          ld_dir      = 1      ; -- . direction of load
          equil_ck    = 0      ; -- . if = 0, total load; if = 1, check equil.
          EA          = 1.0    ; -- . EA (to get P/EA)
          fac         = 2.0    ; -- . divide final load
          name1       = 'STAT' ; -- . first word of input dataset
          name2       = 'REAC' ; -- . second word of input dataset
          input_ds    = ''' '''; -- . full name of input dataset
          output_ds   = ''' '''; -- . output dataset
          nl_res_ds   = 'ES*'  ; -- . nonlinear results dataset name
          input_rec   = 'DATA' ; -- . input dataset record key
          output_rec  = 'TOTAL_LOAD'; -- . output record
          output_file = ''' ''')    . output file name

 [xqt LOAD
      reset inlib=[input_ldi],   outlib=[output_ldi],  reslib=[nl_res_ldi]
      reset iset=[cons_set],     ncon=[cons_set]
      reset nsteps=[nsteps],     strt=[begin_step],    stop=[end_step]
      reset idir=[ld_dir],       eqck=[equil_ck]
```

```
      reset ea=[EA],              lfac=[fac]
      reset n1=[name1],           n2=[name2]
      reset dsin=[input_ds],      dsout=[output_ds],    dsres=[nl_res_ds]
      reset rnin=[input_rec],     rnout=[output_rec]
      reset outfil=[output_file]
      stop
*end
```

6.4.9 REFERENCES

None.

7.0 Utility Procedures

The procedures documented in this chapter are convenient utility procedures that are useful in many analysis tasks.

A summary of the procedures found in this chapter is provided in Table 7.0-1.

Table 7.0-1 Utility Procedures

Procedure Name	*Utility Description*
CONSTRAIN	Impose Scaled Applied Displacements
COPY_DS	Copy a Dataset and Rename
EIGEN	Perform Eigenvalue Analysis
ES	Generic Element Processor Control
FACTOR	Factor (Decompose) System Stiffness Matrix
FORCE	Form Force Vectors
IMPERFECTION	Superpose Initial Geometric Imperfection
INITIALIZE	Model initialization
MASS	Form Mass Matrix
MODEL_SUMMARY	Model Summary Information
PRINT_EFIL	Print Selected Segments of EFIL Dataset
RESEQUENCE	Resequence Nodal Equations
SOLVE	Solve System of Equations
STIFFNESS	Form Stiffness Matrix
SWITCH_DS	Switch the Names of Two Datasets

THIS PAGE LEFT BLANK INTENTIONALLY.

7.1 Procedure CONSTRAIN

7.1.1 GENERAL DESCRIPTION

Procedure **CONSTRAIN** scales the applied motion vector. This procedure is generally used for nonlinear analyses with applied displacements.

7.1.2 PROCEDURE USAGE

Procedure **CONSTRAIN** is used by preceding the procedure name by the ***call** directive, and following it by a list of arguments enclosed in parentheses. Procedure arguments are order-independent, and most have default values thus making them optional. The formal syntax is as follows:

```
*call CONSTRAIN ( arg1 = val1 ; arg2 = val2 ; ...)
```

where **arg1** and **arg2** represent argument names, and **val1** and **val2** represent their corresponding values. Note that semi-colons are required between arguments, and a double dash (--) may be used to continue the argument list on the next line.

The allowable arguments for procedure **CONSTRAIN** are summarized in the following table, along with their default values (if they exist). Note that arguments without defaults are generally *mandatory*, while those with defaults are generally optional. Exceptions to this rule are noted in the following section under detailed argument descriptions.

For procedure **CONSTRAIN**, the following table lists each argument, its default value and meaning.

Argument	Default Value	Meaning
APPLIED_MOTION		Name of applied motion dataset
CONSTRAINT_SET	1	Constraint set number
DISPLACEMENT		Name of scaled applied motion dataset
LDI	1	Logical device index
LOAD_FACTOR	1.0	Load factor

7.1.3 ARGUMENT DESCRIPTIONS

7.1.3.1 APPLIED_MOTION

Full dataset name of the basic applied displacements (*e.g.*, APPL.MOTI.1.1).

7.1.3.2 CONSTRAINT_SET

Constraint set number (default: 1). This argument selects which constraint set to be used in solving the linear system of equations.

7.1.3.3 DISPLACEMENT

Full dataset name of the scaled applied displacements (*e.g.*, TOT.DISP.1).

7.1.3.4 LDI

Logical device index (default: 1).

7.1.3.5 LOAD_FACTOR

Load factor (default: 1.0). This argument defines the load factor or scaling constant to be used to scale the basic applied motions.

7.1.4 USAGE GUIDELINES AND EXAMPLES

Procedure **CONSTRAIN** is used by preceding the procedure name by the ***call** directive. Procedure arguments may be changed from their default values by including any or all of the arguments and their new values when the procedure is called. A space or blank is required between the end of the procedure name and the left parenthesis.

```
*call CONSTRAIN  ( LDI = 1 ; CONSTRAINT_SET = 1 ;
                   LOAD_FACTOR = 1.0 ;
                   APPLIED_MOTION = APPL.MOTI.1.1;
                   DISPLACEMENT = TOT.DISP.1 )
```

7.1.5 LIMITATIONS

Procedure **CONSTRAIN** assumes that processor VEC is executing.

7.1.6 ERROR MESSAGES AND WARNINGS

None.

7.1.7 PROCEDURE FLOWCHART

Procedure **CONSTRAIN** is self-contained with no subprocedures.

7.1.8 PROCEDURE LISTING

```
*procedure CONSTRAIN ( ldi            = 1   ; --
                       constraint_set = 1   ; --
                       load_factor    = 1.0 ; --
                       applied_motion       ; --
                       displacement           )
. ----------------------------------------
. Impose Specified Displacement Constraints (Single Point)
. ----------------------------------------
   *if < <spec_disp_flag> > /then
      SPECIFY  [load_factor] [applied_motion] -> [displacement]
   *endif
*end
```

7.1.9 REFERENCES

7.1-1 Stewart, Caroline B.: *The Computational Structural Mechanics Testbed User's Manual.* NASA TM-100644, October 1989.

THIS PAGE LEFT BLANK INTENTIONALLY.

7.2 Procedure COPY_DS

7.2.1 GENERAL DESCRIPTION

Procedure COPY_DS copies a dataset in a given library to another dataset in possibly a different library. This procedure only uses directives from the command language (see reference 7.1-1).

7.2.2 PROCEDURE USAGE

Procedure COPY_DS may be used by preceding the procedure name by the *call directive, and following it by a list of arguments enclosed in parentheses. Procedure arguments are order-independent, and most have default values thus making them optional. The formal syntax is as follows:

```
*call COPY_DS ( arg1 = val1 ; arg2 = val2 ; ...)
```

where arg1 and arg2 represent argument names, and val1 and val2 represent their corresponding values. Note that semi-colons are required between arguments, and a double dash (--) may be used to continue the argument list on the next line.

The allowable arguments for procedure COPY_DS are summarized in the following table, along with their default values (if they exist). Note that arguments without defaults are generally *mandatory*, while those with defaults are generally optional. Exceptions to this rule are noted in the following section under detailed argument descriptions.

For procedure COPY_DS, the following table lists each argument, its default value and meaning.

Parameter	Default Value	Meaning
FROM_DS	--	Source dataset name
TO_DS	--	Target dataset name
FROM_LDI	1	Source logical device index
TO_LDI	1	Target logical device index

7.2.3 ARGUMENT DESCRIPTIONS

7.2.3.1 FROM_LDI

Source logical device index (default: 1).

7.2.3.2 FROM_DS

Source dataset to be copied.

7.2.3.3 TO_LDI

Target logical device index (default: 1).

7.2.3.4 TO_DS

Target dataset for copying source dataset into.

7.2.4 USAGE GUIDELINES AND EXAMPLES

Procedure COPY_DS may be used by preceding the procedure name by the *call directive. Procedure arguments may be changed from their default values by including any or all of the arguments and their new values when the procedure is called. A space or blank is required between the end of the procedure name and the left parenthesis. If the default values of the procedure arguments are to be used, then only the procedure name is required.

```
*procedure COPY_DS ( from_ldi=1; from_ds; to_ldi=1; to_ds )
```

7.2.5 LIMITATIONS

None.

7.2.6 ERROR MESSAGES AND WARNINGS

None.

7.2.7 PROCEDURE FLOWCHART

Procedure COPY_DS is self contained with no subprocedures.

7.2.8 PROCEDURE LISTING

```
*procedure COPY_DS ( from_ldi=1; from_ds; to_ldi=1; to_ds )
. ----------------------------------------
. CLAMP Procedure to copy dataset and rename
. ----------------------------------------
   *if < [from_ldi] /eq [to_ldi] > /then
      *def/i ldix = 30
      *open/scratch <ldix>
      *copy   <ldix>   =  [from_ldi], [from_ds]
      *rename <ldix>, [from_ds] = [to_ds]
      *copy   [to_ldi] =  <ldix>    , [to_ds]
      *close  <ldix>
   *else
      *copy   [to_ldi] = [from_ldi], [from_ds]
      *rename [to_ldi],  [from_ds] = [to_ds]
```

```
   *endif
*end
```

7.2.9 REFERENCES

7.1-1 Felippa, Carlos A.: *The Computational Structural Mechanics Testbed Architecture: Volume II - Directives.* NASA CR 178385, February 1989.

THIS PAGE LEFT BLANK INTENTIONALLY.

7.3 Procedure EIGEN

7.3.1 GENERAL DESCRIPTION

Procedure **EIGEN** solves a linear eigenvalue problem using an eigensolver defined by the global macrosymbol **eigensolver_name**. If **eigensolver_name** is **EIG** or **EIG2**, a matrix iteration procedure with subspace iteration is used as implemented in processors EIG and EIG2, respectively (see Section 9.1 of reference 7.3-1). If **eigensolver_name** is defined to be **LAN**, a Lanczos algorithm implemented in processor LAN will be used (see Section 9.2 of reference 7.3-1). If **eigensolver_name** is defined to be **LANZ**, the alternate Lanczos method implemented in processor LANZ will be used (see Section 9.3 of reference 7.3-1).

7.3.2 PROCEDURE USAGE

Procedure **EIGEN** is used by preceding the procedure name by the ***call** directive, and following it by a list of arguments enclosed in parentheses. Procedure arguments are order-independent, and most have default values thus making them optional. The formal syntax is as follows:

```
*call EIGEN ( arg1 = val1 ; arg2 = val2 ; ...)
```

where **arg1** and **arg2** represent argument names, and **val1** and **val2** represent their corresponding values. Note that semi-colons are required between arguments, and a double dash (--) may be used to continue the argument list on the next line.

The allowable arguments for procedure **EIGEN** are summarized in the following table, along with their default values (if they exist). Note that arguments without defaults are generally *mandatory*, while those with defaults are generally optional. Exceptions to this rule are noted in the following section under detailed argument descriptions.

For procedure **EIGEN**, the following table lists each argument, its default value and meaning.

Argument	Default Value	Meaning
CONSTRAINT_SET	1	Constraint set number
LDI	1	Logical device index
LOAD_SET	1	Load set number
KNAME	K	First word of the name of the dataset containing the assembled stiffness matrix.
PROBLEM_TYPE	'BUCK'	Define problem type for either buckling or vibration.
MASS_TYPE	'CONSISTENT'	Form of mass matrix (consistent or diagonal)
ERROR_TOL	1.0E-4	Error tolerance for eigenvalue convergence
N_MODES	1	Number of converged eigenmodes desired
INIT_VECTORS	0	Number of iteration vectors
SHIFT	0.0	Eigenvalue shift
MAX_ITERS	20	Maximum number of iterations

Tables 7.3-1 and 7.3-2 list the datasets used or created by procedure **EIGEN** and the processors invoked by procedure **EIGEN**, respectively.

Table 7.3-1 Datasets Input/Output by procedure EIGEN			
Dataset	*Description*	*Input*	*Output*
AMAP..ic2.isize	Factorization Map for INV	√	
CON..*j*	Constraints	√	
INV.<KNAME>.*j*	Factored Stiffness Matrix	√	
JDF1.BTAB.1.8	Model Summary	√	
JSEQ.BTAB.2.17	Nodal Elimination Sequence	√	
KMAP..nsubs.ksize	Model Connectivity Map	√	
<KNAME>.SPAR.*jdf2*	Assembled Stiffness Matrix	√	
<KGNAME>.SPAR.*jdf2*	Assembled Geometric Stiffness Matrix	√	
<MNAME>.SPAR.*jdf2*	Assembled Consistent Mass Matrix	√	
VIBR.MODE.*i.j*†	Vibration eigenmodes		√
VIBR.EVAL.*i.j*†	Vibration eigenvalues		√
BUCK.MODE.*i.j*†	Buckling eigenmodes		√
BUCK.EVAL.*i.j*†	Buckling eigenvalues		√

† i = <load_set> and j = <cons_set>

Table 7.3-2 Processors Invoked by procedure EIGEN		
Procedure	*Type*	*Function*
EIG/EIG2	Internal	Solves eigenproblem using subspace iteration
LAN	External	Solves eigenproblem uing Lanczos method
LANZ	External	Solves eigenproblem using Lanczos method
GETK	Internal	Converts nodal block matrix format to compact sparse format

7.3.3 ARGUMENT DESCRIPTIONS

7.3.3.1 PROBLEM_TYPE

Defines problem type (default: 'BUCK'). This argument selects the type of eigenproblem to solve either buckling (BUCK) or vibration (VIBR).

7.3.3.2 ECON_SET

Constraint set number for eigenproblem (default: 1). This argument selects which constraint set to be used in solving the eigenproblem which may be different from that used for determining the prestress state.

7.3.3.3 LDI

Logical device index (default: 1).

7.3.3.4 LOAD_SET

Load set number (default: 1). This argument selects which load set to be used as a right-hand side vector.

7.3.3.5 KNAME

First word of the dataset name containing the assembled stiffness matrix (default: K).

7.3.3.6 KGNAME

First word of the dataset name containing the assembled geometric stiffness matrix (default: KG).

7.3.3.7 MNAME

First word of the dataset name containing the assembled mass matrix (default: CEM).

7.3.3.8 MASS_TYPE

Form of mass matrix (default: 'CONSISTENT'. This argument defines the form of the mass matrix to be either consistent or diagonal.

7.3.3.9 ERROR_TOL

Error tolerance for convergence of eigenvalues (default: 1.0E-4).

7.3.3.10 N_MODES

Number of converged eigenmodes desired (default: 1).

7.3.3.11 INIT_VECTORS

Number of initial iteration vectors (default: 0). If the default value is used, then the number of initial iteration vectors will be the minimum of <2* <N_MODES>> and <<N_MODES>+8>.

7.3.3.12 SHIFT

Eigenvalue shift (default: 0.0).

7.3.3.13 MAX_ITER

Maximum number of iterations in solving the eigenproblem (default: 20).

7.3.4 USAGE GUIDELINES AND EXAMPLES

Procedure **EIGEN** is used by preceding the procedure name by the ***call** directive. Procedure arguments may be changed from their default values by including any or all of the arguments and their new values when the procedure is called. A space or blank is required between the end of the procedure name and the left parenthesis.

```
*call EIGEN ( KNAME = K ; ECON_SET = 1 ;
              LOAD_SET = 1 ; KGNAME = KG )
```

Before procedure **EIGEN** is called the global macrosymbol **eigensolver_name** should be defined as described in Section 7.3.1 and should be consistent with the value used when procedure **FACTOR** was called. If it is not specified, then the default value **EIG2** will be used.

7.3.5 LIMITATIONS

None.

7.3.6 ERROR MESSAGES AND WARNINGS

None.

7.3.7 PROCEDURE FLOWCHART

Procedure **EIGEN** is self-contained with no subprocedures.

7.3.8 PROCEDURE LISTING

```
*procedure EIGEN ( PROBLEM_TYPE = 'BUCK' ; -- . Problem type: BUCK or VIBR
                   ldi = 1 ; -- . library number
                   kname = 'K' ; -- . First name of material stiffness matrix
                   kgname = 'KG' ; -- . First name of geometric stiffness matrix
                   mname = 'CEM' ; -- . First name of mass matrix
                   error_tol = 1.0e-4 ; -- . Error tolerance on eigenevalues
                   load_set = 1 ; -- . Load set number
                   mass_type = 'CONSISTENT' ; -- . Form of mass matrix
                   N_modes = 1; -- . Number of modes desired
                   INIT_vectors = 0 ; --  . Number of initial vectors
                   SHIFT =0.0 ; -- . Eigenvalue shift
                   max_iters = 20 ; -- . Maximum number of iterations
                   econ_set = 1 ) . Constraint set for eigenproblem
.
. Procedure to solve the eigenvalue problem using different
. eigensolvers.  The eigensolver is selected using the
. global macrosymbol "eigensolver_name".
.
.
.  ---------------------------------------------------------------
.  Perform Eigenvalue Analysis (with [kname].SPAR and [kgname].SPAR)
.  ---------------------------------------------------------------
.
   *if < [INIT_vectors] /eq 0 > /then
    *def/i N_vectors = < MIN( 2*[N_modes]; [N_modes]+8) >
    *else
    *def/i N_vectors = [INIT_vectors]
   *endif
.
.  ------------------
.  Select eigensolver
.  ------------------
.
.  Use processor EIG
.
*if < ifeqs(<eigensolver_name>;EIG) > /then
.
   *def NS_OVERWRITE = <false>
   [XQT EIG
   RESET PROB=[PROBLEM_TYPE]
   RESET INIT=<N_vectors>, SHIFT=[shift], NREQ=[N_modes]
   RESET CONV=[error_tol], NDYN=[max_iters], con=[econ_set],NEWSET=[load_set]
   RESET K=[kname]
   RESET KG=[kgname]
   RESET M=[mname]
   RESET KLIB=[LDI],KILIB=[LDI],KGLIB=[LDI],MLIB=[LDI],OUTLIB=[LDI]
   RESET INLIB=[LDI],OLDSET=[LDI]
   STOP
.
.  Use processor EIG2
.
```

```
*elseif < ifeqs(<eigensolver_name>;EIG2) > /then
.
   *def NS_OVERWRITE = <false>
   [XQT EIG2
   RESET PROB=[PROBLEM_TYPE]
   RESET INIT=<N_vectors>, SHIFT=[shift], NREQ=[N_modes]
   RESET CONV=[error_tol], NDYN=[max_iters], con=[econ_set],NEWSET=[load_set]
   RESET K=[kname]
   RESET KG=[kgname]
   RESET M=[mname]
   RESET KLIB=[LDI],KILIB=[LDI],KGLIB=[LDI],MLIB=[LDI],OUTLIB=[LDI]
   RESET INLIB=[LDI],OLDSET=[LDI]
   STOP
*elseif < ifeqs(<eigensolver_name>;LAN) > /then
.
.  Use processor LAN
.
   *def NS_OVERWRITE = <false>
   [xqt LAN
     RESET PROB=[PROBLEM_TYPE],LDI=[LDI]
     RESET KNAM=[kname],KGNA=[kgname],MNAM=[mname]
     RESET NK=[max_iters], shift=[shift], ncon=[econ_set]
     STOP
*elseif < ifeqs(<eigensolver_name>;LANZ) > /then
   *def NS_OVERWRITE = <false>
 *if < ifeqs([PROBLEM_TYPE];BUCK) > /then
   [xqt GETK
     reset sipr=0, form=2, save=1
     reset k=k, redo=0, kred=k, reac=0
   [xqt GETK
     reset sipr=0, form=2, save=1
     reset k=kg, redo=1, kred=k, reac=0
   [xqt LANZ
   RESET PROB=2, plvl=3,check=1, con=[econ_set]
   RESET nreq=[N_modes], conv=[error_tol]
   RESET kg=[kgname]
 *elseif < ifeqs([PROBLEM_TYPE];VIBR) > /then
   [xqt GETK
     reset sipr=0, form=2, save=1
     reset k=k, redo=0, kred=k, reac=0
   *if < ifeqs([mass_type];CONSISTENT) > /then
   *def/i mcase=1
   [xqt GETK
     reset sipr=0, form=2, save=1
     reset k=[mname], redo=1, kred=k, reac=0
   *else
   *def/i mcase=0
   *endif
   [xqt LANZ
   RESET mcase=<mcase>, plvl=3,check=1,con=[econ_set]
   RESET nreq=[N_modes], conv=[error_tol]
   RESET m=[mname]
 *endif
```

```
*endif
.
*end
```

7.3.9 REFERENCES

7.3-1 Stewart, Caroline B.: *The Computational Structural Mechanics Testbed User's Manual.* NASA TM-100644, October 1989.

7.4 Procedure ES

7.4.1 GENERAL DESCRIPTION

Procedure **ES** is a high-level procedure that controls the structural element processors (ES*i*). The Generic Element Processor (GEP) provides a *template* with which many individual finite-element processors may be developed and coexist as independent modules in the CSM Testbed (see ref. 7.4-1). The GEP template for *structural* elements is referred to as ES, and all element processors built with this template have names that begin with ES (*e.g.*, ES1, ES2, ...). Each of these ES*i* processors performs *all* operations for all of the elements implemented within the processor — including element definition, stiffness, force, mass (etc.) generation, and various pre/post-processing functions.*

Since ES*i* processors are typically built by different developers, you might expect a wide variety of User-interface characteristics. However, because of the generic template employed, all ES*i* processors share the same command language and global datasets. This means that once the user has learned to invoke one ES*i* processor, the user has learned them all.

The main difference between ES*i* processors will be in the specific elements implemented within them. Some ES*i* processors may have only one element type inside. Others may have a family of elements of a certain class (*e.g.*, 4-, 9- and 16-node shell elements). Still others may embed an entire library of structural elements, containing various members of each class (*e.g.*, beam, shell, solid). Each of these specific element types is given a corresponding name within each ES*i* processor; so that the combination of ES*i* processor name and element type is unique. Thus, to employ a particular ES*i* processor correctly, the user will have to consult specific documentation on the individual elements contained within that processor. Such documentation is provided in the CSM Testbed User's Manual (see ref. 7.4-2).

In the following sections, the generic features of ES*i* processors are described in detail.

* Exception: Element connectivity for all element processors is currently performed by Testbed processor ELD.

The description includes ES*i* processor commands and macrosymbols, and the datasets required or produced by these commands (Section 2.2), the high-level procedure interface, which makes it possible to write analysis procedures that invoke a single, generic procedure (called **ES**) to generate element arrays automatically for all ES*i* processors required in a given problem; a glossary of ES macrosymbols, which gives more detailed definitions for the macrosymbols and procedure arguments; and some explicit examples of how to use ES*i* processors, interactively, or using the procedure **ES** interface.

7.4.2 PROCEDURE USAGE

Procedure **ES** may be invoked by the ***call** directive, and following it by a list of arguments separated by semicolons(;) and enclosed in parentheses. Procedure arguments are order-independent, and most have default values thus making them optional. The formal syntax is as follows:

```
*call ES ( arg1 = val1; arg2 = val2; ...)
```

where **arg***i* are argument names and **val***i* are the corresponding values. The following are valid arguments for procedure **ES**; note that those arguments without default values are mandatory, while the others are optional.

Argument	Default Value	Meaning
ES_NAME	--	Element-type name
ES_NL_GEOM	--	Geometric nonlinearity
ES_NL_MATL	--	Material nonlinearity
ES_NL_LOAD	--	Load nonlinearity
ES_PARS	--	Research parameters
ES_PROC	--	Processor name
FUNCTION	--	Processor command
ES_CORO	1	Corotational option
ES_COUNT	0	Create or accumulate ES.SUMMARY
ES_DIS_DS	STAT.DISP.1.1	Displacement dataset
ES_DOF_DS	ES.DOFS	Active-freedom dataset
ES_ECC_DS	WALL.PROP	Eccentricity dataset
ES_FRC_DS	INT.FORC.1.1	Force vector dataset
ES_ROT_DS	STAT_ROTA.1.1	Rotation vector dataset
ES_STR_LOC	CENTROIDS	Stress locations
ES_STR_DIR	0	Stress directions
ES_STRAIN_DS	' '	Element strain dataset
ES_STRESS_DS	' '	Element stress dataset
ES_SUM_DS	ES.SUMMARY	ES summmary dataset
LDI	1	Logical device index
NUM_CON_DS	1	Number of constraint datasets

Table 7.4-1 Datasets Input/Output by Procedure ES				
Dataset	*Description*	*Lib*	*Input*	*Output*
ES.SUMMARY	ES Processor Status	1		√
ES.DOFS	1			√
<ES_NAME>.EFIL.0.*nnod*	Element Computational Data	1		√
DIR.<ES_NAME>.0.*nnod*	Element EFIL Directory	1	√	
DEF.<ES_NAME>.0.*nnod*	Element Definition (Connectivity)	1	√	
PROP.BTAB.2.*	Material/Section Properties	1	√	
STAT.DISP.*i.j*	System Displacement Vector	1	√	
STAT.ROTA.*i.j*	System Rotation Vector	1	√	
QJJT.BTAB.	Nodal Transformations	1	√	
INT.FORC.*i.j*	System Internal Force Vector	1		√
DEM.DIAG	Diagonal (Lumped) Mass Matrix	1		√
CEM.SPAR.*jdf2*	Assembled Consistent Mass Matrix	1		√
STRN.<ES_NAME>.<ES_STRAIN_DS>	Element Strains	1		√
STRS.<ES_NAME>.<ES_STRESS_DS>	Element Stresses	1		√
K.SPAR.*jdf2*	Assembled material stiffness matrix	1		√
KG.SPAR.*jdf2*	Assembled geometric stiffness matrix	1		√

† i = <load_set> and j = <cons_set>

Table 7.4-2 Sub-Procedures Invoked by Procedure ES		
Procedure	*Type*	*Function*
ES	Internal	Main procedure
ES_DOFS	Internal	Automatic DOF Suppression for ES*i*

Table 7.4-3 Sub-Procedures Invoked by Procedure ES		
Processor	*Type*	*Function*
ES*i*	External	Various element processors
VEC	Internal	Vector functions

Table 7.4-4 ES PROCESSOR MACROSYMBOL GLOSSARY

Macrosymbol	*Type*	*Definition*
ES_C	I	Continuity of interelement "displacement" field, *e.g.*, $0 \Rightarrow C^0$ (displacement continuity only) $1 \Rightarrow C^1$ (displacement and slope continuity)
ES_CLAS	A	Element class. Currently valid classes: BEAM, SHELL, SOLID, WILD (See Section 3.4 for examples.)
ES_CNS	I	Constitutive interface option (see Chapter 5). 0 ⇒elements use the standard constitutive interface for stress and tangent-modulus calculations; 1 ⇒elements compute their own stresses, but use standard constitutive interface for tangent-modulus calculations; ≥2 ⇒elements compute their own stresses and tangent-moduli; standard constitutive interface is not used.
ES_CORO	I	Corotation switch; employed by ES processor for automatic treatment of geometric nonlinearity due to large rotations. Relevant only if problem is geometrically nonlinear. (See Chapter 4 for an explanation of these options.) 0 ⇒ Off: corotational operations will be skipped; 1 ⇒ Low-Order Option: basic corotational transformations will be employed to enable large rotations; 2 ⇒ High-Order Option: a more accurate (and expensive) treatment of large rotations and consistent linearization than option 1 will be employed.
ES_COUNT	I	Element processor count. Relevant only for DEFINE ELEMENTS command. 0 ⇒ First element processor to be defined; create ES.SUMMARY dataset. > 0 ⇒ Not first element processor to be defined; use existing ES.SUMMARY dataset and Increment element processor number by one.
ES_DIM	I	Number of intrinsic element spatial dimensions, *e.g.*, 1 if ES_CLAS = BEAM 2 if ES_CLAS = SHELL 2 or 3 if ES_CLAS = SOLID

Table 7.4-4 ES PROCESSOR MACROSYMBOL GLOSSARY (continued)

Macrosymbol	*Type*	*Definition*
ES_DIS_DS	A	Name of system displacement-vector dataset (SYSVEC format). Relevant for most FORM commands and some POST commands. (Default: STAT.DISP.1.1)
ES_DOF_DS	A	Name of element freedom-table dataset (SYSVEC format). Relevant only for DEFINE FREEDOMS command. When using procedure ES, this dataset will automatically be created, initialized, and updated cumulatively with contributions from all pertinent ES processors. (Default: ES.DOFS)
ES_ECC_DS	A	First two words of dataset name that contains section property data including reference surface eccentricities. (Default: WALL.PROP).
ES_EXPE_CMD	A	Complete EXPE command line appropriate for defining elements with processor ELD. This macrosymbol is constructed only in response to a call to procedure ES with FUNCTION='DEFINE ELEMENTS'.
ES_FRC_DS	A	Name of system force-vector dataset (SYSVEC format), where element distributed forces (internal and/or external) are to be assembled. Relevant only for FORM FORCE command, in which case this vector must be created and initialized before issuing the command (or calling procedure ES). Contributions from all pertinent ES processors will be assembled into this vector if procedure ES is called with argument FUNCTION = 'FORM FORCE ...'. (Default: INT.FORC.1.1)
ES_LOAD_FACTOR	F	Load factor. (Default: 1.0)
ES_LOAD_SET	I	Load set number. (Default: 1)

Table 7.4-4 ES PROCESSOR MACROSYMBOL GLOSSARY (continued)

Macrosymbol	*Type*	*Definition*
ES_MASS_DIAG	A	First word of the dataset name for a diagonal mass matrix. (Default: DEM).
ES_MASS_DS	A	First two words of the dataset name for a diagonal mass matrix. (Default: <ES_MASS_DIAG>.DIAG).
ES_NAME	A	Name of element type, within current ES processor, to be processed by subsequent commands. (For example EX97 would be a valid element-type name within processor ES1.)
ES_NDOF	I	Number of freedoms per element node. Currently valid options (2, 3 or 6), *e.g.*,
		2 for 2-D solid elements (u, v)
		3 for 3-D solid elements (u, v, w)
		6 for beam, plate or shell elements $(u, v, w, \theta_x, \theta_y, \theta_z)$
ES_NEE	I	Number of element equations; = ES_NEN × ES_NDOF.
ES_NEN	I	Number of element nodes.
ES_NIP	I	Number of element integration points; *i.e.*, points at which stresses (continuum or resultants, depending on element type) are stored.
ES_NL_GEOM	I	Geometric nonlinearity switch; 0 ⇒ Off: problem is geometrically linear (small displacements/rotations);
		1 ⇒ Low-Order Option: problem is geometrically nonlinear, but elements should use linear strain-displacement relations. Meaningful only if ES_CORO > 0, so that large rotations can be handled automatically by the corotational algorithm;

Table 7.4-4 ES PROCESSOR MACROSYMBOL GLOSSARY (continued)

Macrosymbol	*Type*	*Definition*
		2 ⇒ High-Order Option: problem is geometrically nonlinear and elements should use *nonlinear* element strain-displacement relations. May be used in conjunction with ES_CORO > 0 to obtain higher-order accuracy for beam and shell elements that employ moderate-rotation strain-displacement relations;
		3 ⇒ same as 2 plus finite strains are expected.
ES_NL_LOAD	I	Load nonlinearity switch; 1 ⇒ process displacement-dependent element loads only; 0 ⇒ process displacement-independent element loads only.
ES_NL_MATL	I	Material nonlinearity switch; 1 ⇒ element is materially nonlinear; 0 ⇒ element materially linear.
ES_NORO	I	Element normal-rotation parameter. Relevant only for automatic freedom suppression of plate/shell elements. Indicates minimum angle (in degrees) between shell element normal vector and any computational basis vector — at each element node — below which the corresponding rotational freedom should be suppressed if no other elements are attached;
		=0 ⇒Element has normal-rotation ("drilling") stiffness; normal rotational freedoms will automatically be suppressed.
		>0 ⇒Element does not have normal-rotation ("drilling") stiffness; it is assumed that rotational stiffness exists about any computational axis that makes an angle of at least <ES_NORO> degrees with the element normal vector at an element node.

Table 7.4-4 ES PROCESSOR MACROSYMBOL GLOSSARY (continued)

Macrosymbol	*Type*	*Definition*
ES_NPAR	I	Number of element research parameters in array ES_PARS.
ES_NSTR	I	Number of stress components per integration point. Currently valid options compatible with standard constitutive interface: 8 for ES_CLAS = SHELL and ES_C = 0; 6 for ES_CLAS = SOLID and ES_DIM = 3; 6 for ES_CLAS = SHELL and ES_C = 1; 6 for ES_CLAS = BEAM and ES_C = 0; 4 for ES_CLAS = BEAM and ES_C = 1; 3 for ES_CLAS = SOLID and ES_DIM = 2.
ES_OPT	I	Element-type option number. Meaningful only to the element developer; it is the developer's numerical equivalent of ES_NAME.
ES_PARS	D	Array of element research parameters. Meaning depends on specific element type. (Consult appropriate section in CSM Testbed User's Manual, ref. 4).
ES_PROC	A	Name of element (ES) processor to be executed (*e.g.*, ES1, ES2, ...); currently relevant only as an argument for procedure ES, and only when argument FUNCTION = 'DEFINE ELEMENTS'.

Table 7.4-4 ES PROCESSOR MACROSYMBOL GLOSSARY (continued)

Macrosymbol	*Type*	*Definition*
ES_PROJ	I	Rigid-body *projection* option. Used to automatically remove (most of) the spurious energy generated by some elements during infinitesimal rigid-body motion. This is performed by operating on the element stiffness and force arrays with a projection matrix (or *projector*). The projector, and its derivative, can have a beneficial effect on element accuracy in both linear and geometrically nonlinear regimes. 0 ⇒Off: projection will be omitted; 1 ⇒Low-order Option: basic projection will be included; 2 ⇒High-order Option: basic projection plus a differential correction to the geometric stiffness will be included.
ES_ROT_DS	A	Name of system rotation pseudo-vector dataset (SYSVEC format). Relevant for most FORM commands during *geometrically nonlinear* analysis, but only if elements with rotational freedoms are involved. (Default: STAT.ROTA.1.1)
ES_SHAP	A	Shape of element surface used to define coordinate triad; currently recognized options: LINE, TRIA or QUAD.
ES_STOR	I	Number of "private" variables to be stored/retrieved for the element developer in dataset EFIL.<ES_NAME>.
ES_STR_DIR	I	Element stress/strain direction option. Indicates in which coordinate system element stresses or strains in datasets defined by <ES_STRESS_DS> or <ES_STRAIN_DS> (respectively) will be computed. Relevant only for command = 'FORM STRAIN' or 'FORM STRESS'. Valid options: 0 ⇒ element stress coordinate system 1 ⇒ material axes $\{x_m, y_m, z_m\} = \{x_g, y_g, z_g\}$ 2 ⇒ material axes $\{y_m, z_m, x_m\} = \{y_g, z_g, x_g\}$ 3 ⇒ material axes $\{z_m, x_m, y_m\} = \{z_g, x_g, y_g\}$ (Note: For isotropic materials, the first material axis is replaced by the corresponding global axis; see the SREF command under processor ELD in the Testbed User's Manual (ref. 4) for details. Default: 0)

Table 7.4-4 ES PROCESSOR MACROSYMBOL GLOSSARY (continued)

Macrosymbol	*Type*	*Definition*
ES_STR_LOC	A	Element stress/strain evaluation point option. Indicates where stresses or strains in datasets defined by <ES_STRESS_DS> or <ES_STRAIN_DS> (respectively) will be computed. Relevant only for command = 'FORM STRAIN' or 'FORM STRESS'. Currently valid options: CENTROIDS ⇒ element centroids; creates record group: CENTROIDS_<ES_STR_DIR>.1:nel NODES ⇒ element nodes; creates record group: NODES_<ES_STR_DIR>.1:nel INTEG_POINTS ⇒ element integration points; creates: INTEG_PTS_<ES_STR_DIR>.1:nel where "nel" is the number of elements in the dataset, and where macrosymbol ES_STR_DIR designates the directions of the stress/strain components, and is defined elsewhere in this Glossary. (Default: 'CENTROIDS')
ES_STRAIN_DS	A	Third part of element strain dataset name. The first name is always STRN; the second name is always the element-type name, *i.e.*, <ES_NAME>; and the third name, <ES_STRAIN_DS> must be a string of integers separated by periods. For example, if <ES_STRAIN_DS> = <step>.<iter> where <step> = 20 and <iter> = 3, and if the element-type name were EX97, then the full dataset name would be: STRN.EX97.20.3 Relevant only for command = 'FORM STRAIN'. (No default; absence means that strains will be stored (embedded) within dataset EFIL.<ES_NAME> only — *currently not implemented.*)

Table 7.4-4 ES PROCESSOR MACROSYMBOL GLOSSARY (concluded)

Macrosymbol	*Type*	*Definition*
ES_STRESS_DS	A	Third part of element stress dataset name. The first name is always STRS; the second name is always the element-type name, *i.e.*, <ES_NAME>; and the third name, <ES_STRESS_DS> must be a string of integers separated by periods. For example, if <ES_STRESS_DS> = <step>.<iter>, where <step> = 20 and <iter> = 3, and if the element-type name were EX97, then the full dataset name would be: STRS.EX97.20.3 Relevant only for command = 'FORM STRESS'. (No default; absence means that stresses will be stored (embedded) within dataset EFIL.<ES_NAME> only.)
ES_SUM_DS	A	Name of ES summary dataset, which contains nominal records corresponding to most of the macrosymbol parameters appearing in this Glossary — for each ES processor/element defined in the model. Relevant for all ES commands. (Default: ES.SUMMARY)
ES_TWIS	I	Sign of twisting curvature for shell elements. +1 ⇒twist based on continuum definition of shear strain; −1 ⇒twist based on negative of continuum definition. (Note: The default convention for constitutive matrices output from processor LAU corresponds to the -1 option. Hence processor ES compensates for this reversal if ES_TWIS = +1.)
ES_TGC_DS	I	Name of nodal transformation dataset (QJJT.BTAB.*.*)
ES_XYZ_DS	I	Name of nodal coordinate dataset (JLOC.BTAB.*.*)
LDI	I	Logical device index or library number for archiving and retrieving data. (Default: 1)

7.4.3 ARGUMENT DESCRIPTION

7.4.3.1 ES_CORO

Corotation switch; employed by ES*i* processors for automatic treatment of geometric nonlinearity due to large rotations. Relevant only if problem is geometrically nonlinear. (See Chapter 4 of reference 7.4-1 for an explanation of these options.) If **ES_CORO** equals zero, then the corotational operations will be skipped. If **ES_CORO** equals one (low-order option), then the basic corotational transformations will be employed to enable large rotations. If **ES_CORO** equals two (high-order option), then a more accurate (and expensive) treatment of large rotations and consistent linearization than option 1 will be employed.

7.4.3.2 ES_COUNT

Element processor count (default: 0). This argument is used to create and/or accumulate data in the ES.SUMMARY dataset for finite element models using multiple element processors.

7.4.3.3 ES_DIS_DS

Name of system displacement-vector dataset in **SYSVEC** format (default: **STAT.DISP.1.1**). Relevant for most **FORM** commands.

7.4.3.4 ES_DOF_DS

Name of element freedom-table dataset (**SYSVEC** format). Relevant only for **DEFINE FREEDOMS** command. When using procedure **ES**, this dataset will automatically be created, initialized, and updated cumulatively with contributions from all pertinent ES*i* processors. (Default: **ES.DOFS**)

7.4.3.5 ES_ECC_DS

Name of element section property eccentricity dataset (default: **WALL.PROP**). Relevant for all **FORM** commands.

7.4.3.6 ES_FRC_DS

Name of system force-vector dataset in SYSVEC format, where element distributed forces (internal and/or external) are to be assembled (default: STAT.REAC.1.1). Relevant only for FORM FORCE command, in which case this vector must be created and initialized before issuing the command (or calling procedure ES). Contributions from all pertinent ES*i* processors will be assembled into this vector if procedure ES is called with argument FUNCTION = 'FORM FORCE.

7.4.3.7 ES_NAME

Name of element type, within current ES*i* processor, to be processed by subsequent commands. (For example, EX97 would be a valid element-type name within processor ES1.)

7.4.3.8 ES_NL_GEOM

Geometric nonlinearity switch. If ES_NL_GEOM equals zero (no corotational frames), then the problem is geometrically linear (small displacements/rotations). If ES_NL_GEOM equals one (low-order option), then the problem is geometrically nonlinear, but elements should use linear strain-displacement relations. Meaningful only if ES_CORO > 0, so that large rotations can be handled automatically by the corotational algorithm. If ES_NL_GEOM equals two (high-order option), then the problem is geometrically nonlinear and elements should use *nonlinear* element strain-displacement relations. May be used in conjunction with ES_CORO > 0 to obtain higher-order accuracy for beam and shell elements that employ moderate-rotation strain-displacement relations.

7.4.3.9 ES_NL_MATL

Material nonlinearity switch. If ES_NL_MATL is greater than zero, then the element is materially nonlinear. If ES_NL_MATL is zero, then the element is materially linear.

7.4.3.10 ES_NL_LOAD

Load nonlinearity switch. If ES_NL_LOAD is greater than zero, then the element loads are displacement dependent. If ES_NL_LOAD equals zero, then the element loads are not displacement dependent.

7.4.3.11 ES_PARS

Array of element research parameters. Meaning depends on specific element type. Consult appropriate section in CSM Testbed User's Manual, ref. 7.4-2.

7.4.3.12 ES_PROC

Name of element (ES)i processor to be executed (*e.g.*, ES1, ES2, ...); currently relevant only as an argument for procedure ES, and only when argument `FUNCTION = 'DEFINE ELEMENTS'`.

7.4.3.13 ES_ROT_DS

Name of system rotation pseudo-vector dataset in SYSVEC format (default: `STAT.ROTA.1.1`). Relevant for most FORM commands during *geometrically nonlinear* analysis; but only if elements with rotational freedoms are involved.

7.4.3.14 ES_STR_LOC

Element stress/strain location option (default: `'CENTROIDS'`). Indicates where stresses or strains in datasets defined by `<ES_STRESS_DS>` or `<ES_STRAIN_DS>` (respectively) will be computed. Relevant only for `FUNCTION = 'FORM STRAIN'` or `'FORM STRESS'`. Currently valid options:

- `CENTROIDS` ⇒ element centroids; creates record group:
  `CENTROIDS_<ES_STR_DIR>.1:nel`
- `NODES` ⇒ element nodes; creates record group:
  `NODES_<ES_STR_DIR>.1:nel`
- `INTEG_POINTS` ⇒ element integration points; creates:
  `INTEG_PTS_<ES_STR_DIR>.1:nel`

where "`nel`" is the number of elements in the dataset, and where macrosymbol `ES_STR_DIR` designates the directions of the stress/strain components, and is defined elsewhere in this glossary.

7.4.3.15 ES_STR_DIR

Element stress/strain direction option (default: E). Indicates in which coordinate system element stresses or strains in datasets defined by <ES_STRESS_DS> or <ES_STRAIN_DS> (respectively) will be computed. Relevant only for FUNCTION = 'FORM STRAIN' or 'FORM STRESS'. Currently valid options include E for the element stress coordinate system, M for the material coordinate system, and G*i* for the coordinate system whose x axis corresponds to global coordinate axis x_i.

7.4.3.16 ES_STRAIN_DS

Third part of element strain dataset name. The first name is always STRN; the second name is always the element-type name, *i.e.*, <ES_NAME>; and the third name, <ES_STRAIN_DS> must be a string of integers separated by periods (see CSM Testbed Data Library Description, reference 7.4-3). For example, if <ES_STRAIN_DS> = <step>.<iter> where <step> = 20 and <iter> = 3, and if the element-type name were EX97, then the full dataset name would be: STRN.EX97.20.3. Relevant only for FUNCTION = 'FORM STRAIN'. No default; absence means that strains will be stored (embedded) within dataset EFIL.<ES_NAME> only — *currently not implemented.*

7.4.3.17 ES_STRESS_DS

Third part of element stress dataset name. The first name is always STRS; the second name is always the element-type name, *i.e.*, <ES_NAME>; and the third name, <ES_STRESS_DS> must be a string of integers separated by periods same as previously stated. For example, if <ES_STRESS_DS> = <step>.<iter>, where <step> = 20 and <iter> = 3, and if the element-type name were EX97, then the full dataset name would be: STRS.EX97.20.3. Relevant only for FUNCTION = 'FORM STRESS'. No default; absence means that stresses will be stored (embedded) within dataset EFIL.<ES_NAME> only.

7.4.3.18 ES_SUM_DS

Name of ES summary dataset, which contains nominal records corresponding to most of the macrosymbol parameters appearing in this glossary — for each ES*i* processor/element defined in the model (default: ES.SUMMARY). Relevant for all ES commands.

7.4.3.19 FUNCTION

All structural element (ES*i*) processors based on the generic element processor template, share the same processor commands. This makes it easier for the User, and enables the construction of a generic command procedure to handle any combination of ES*i* processors. The commands recognized by ES*i* processors fall into three categories: **INITIALIZE**, **DEFINE**, and **FORM**, which roughly correspond to preprocessing and computation phases of analysis. The **INITIALIZE** command initializes the element data. The **DEFINE** commands are used to prepare or re-format model definition datasets, such as element connectivity, freedom activity, loads, etc.* The **FORM** commands are used to form element computational data, such as stiffness matrices, force vectors, etc. Table 7.4-5 provides a summary of ES processor commands and their respective functions.

* Again, element connectivity is currently performed using Testbed processor ELD.

Table 7.4-5 Summary of Generic ES Processor Commands		
Command		*Description*
INITIALIZE		Initialize element data
DEFINE	ELEMENTS	Define element parameters
	FREEDOMS	Perform automatic DOF suppression
FORM	STIFFNESS [/MATL/GEOM/LOAD/TANG]	Form element stiffness
	FORCE [/INT/EXT/RES/DYN]	Form element force
	MASS [/CONS/DIAG]	Form element mass
	STRAIN	Form element strains
	STRESS	Form element stress

In addition to these commands, ES*i* processors can be controlled with a number of built-in *macrosymbols*, all of which begin with ES_. The macrosymbols typically set logical switches and/or control parameters, and allow re-assignment of database names from their default values.

Table 7.4-6 ANALYSIS/COMMAND CORRESPONDENCE ES Processor Commands vs Analysis Type	
Analysis Type	*Processor Commands*
All Preprocessing	**INITIALIZE** **DEFINE ELEMENTS** **DEFINE FREEDOMS**
Linear Statics	**FORM STIFFNESS/MATL** **FORM FORCE/EXT**
Linear Dynamics	**FORM STIFFNESS/MATL** **FORM MASS/CONS** **FORM FORCE/EXT**
Buckling Eigenvalue	**FORM STIFFNESS/MATL** **FORM STIFFNESS/GEOM** **FORM FORCE/EXT**
Vibration Eigenvalue	**FORM STIFFNESS/MATL** **FORM MASS/{CONS | DIAG}**
Nonlinear Statics	**FORM STIFFNESS/TANG** **FORM FORCE/INT** **FORM FORCE/EXT** **FORM FORCE/RES**
Nonlinear Dynamics	**FORM STIFFNESS/TANG** **FORM MASS/{CONS | DIAG}** **FORM FORCE/INT** **FORM FORCE/EXT** **FORM FORCE/DYN**

7.4.4 PROCEDURE FLOWCHART

ES (main procedure)

ES_DOFS (automatic DOF suppression for ES*i*)

7.4.5 LIMITATIONS

Procedure **ES** assumes that all datasets either required or generated will reside on library one (LDI=1).

7.4.6 ERROR MESSAGES AND WARNINGS

None.

7.4.7 USAGE GUIDELINES AND EXAMPLES

Procedure **ES** may be used by preceding the procedure name by the ***call** directive. Procedure arguments may be changed from their default values by including any or all the arguments and their new values when the procedure is called. A space or blank is required between the end of the procedure name and the left parenthesis.

```
*call ES (  FUNCTION      =                 ; -- Processor command
            ES_PROC       =                 ; -- . Processor name
            ES_NAME       =                 ; -- . Element-type name
            ES_PARS       =  0.0            ; -- . Research parameters
            ES_CORO       =  1              ; -- . Corotational option
            ES_NL_GEOM    =  <false>        ; -- . Geom. nonlinearity
            ES_NL_MATL    =  <false>        ; -- . Matl. nonlinearity
            ES_NL_LOAD    =  <false>        ; -- . Load   nonlinearity
            ES_DIS_DS     =  STAT.DISP      ; -- . Displacement dataset
```

```
ES_DOF_DS    =  ES.DOFS          ; -- . Active-freedom dataset
ES_ECC_DS    =  ES.ECCEN         ; -- . Eccentricity dataset
ES_FRC_DS    =  STAT.REAC        ; -- . Force-vector dataset
ES_ROT_DS    =  STAT.ROTA        ; -- . Rotation-vector dataset
ES_SUM_DS    =  ES.SUMMARY       ; -- . ES summary dataset
ES_STRAIN_DS    =  <false>       ; -- . Element strain dataset
ES_STRESS_DS    =  <false>       ; -- . Element stress dataset
ES_STR_LOC      =  'CENTROIDS'   ; -- . Stress locations
ES_STR_DIR      =  'E'           ; -- . Stress directions
NUM_CON_DS      =  1             ; -- . No. of constraint datasets
LDI               =  1             -- . Logical device index
        )
```

The following examples illustrate how element processors based on the generic structural-element (ES) processor template can be used to perform structural analysis with the CSM Testbed. For simplicity, we consider preprocessing (*i.e.*, model generation), linear analysis, nonlinear analysis, and postprocessing (*i.e.*, stress recovery, etc.) examples separately. The differences between employing individual ES*i* processors directly versus accessing them using the generic ES procedure interface will be stressed, with an intended bias towards the latter.

7.4.7.1 Preprocessing Examples

For clarity, a very simple problem will be considered, and the use of the generic structural-element processor illustrated by showing all of the steps involved in generating a finite element (FE) model for this problem. Both the physical problem and the discrete model to be used is given in Figure 7.4-1. The problem is a rectangular plate (10 in. by 5 in.), cantilevered on one edge, and loaded on the other by a concentrated lateral force. For the model, a 3 by 3 grid is used which corresponds to a 2 by 2 mesh of 4-node plate/shell elements. The Testbed procedure for this model is given in Figure 7.4-2. The interpretation of each line of the procedure will now be described.

The ***procedure** statement in Figure 7.4-3 shows the arguments (parameters) for the procedure and sets default values for each of them (x-length, y-length, thickness, elastic modulus, Poisson's ratio, and precision). Thus the dimensions and properties of the plate model are parameterized, but the finite element discretization is fixed.† The next two lines (**[xqt TAB** and **START 9**) run the processor TAB and reserve space for a total of 9 nodes. Then, the JLOC command and subsequent data define the global coordinates for these 9 nodes in a rectangular 3 by 3 grid.

Material and section properties are then defined by first executing processor AUS to tabulate the raw data, and then executing processor LAU to generate an integrated constitutive matrix for shell elements (see the CSM Testbed User's Manual for details on processor LAU).

The key portion of the example begins with the ***call** to procedure ES with **FUNCTION = 'DEFINE ELEMENTS'**, in which element type EX42 of processor ES1 is registered for participation in the model. This call also causes a number of element type-oriented macrosymbols, all beginning with **ES_**, to be defined — for example, **ES_NEN** (number of element nodes), **ES_NIP** (nuber of element integration points), etc. These macrosymbol values are automatically built into the character string macrosymbol **ES_EXPE_CMD** (by procedure **ES**), which serves as the **EXPE** command for processor ELD. Note that in the **PLATE_MODEL** example, **<ES_EXPE_CMD>** appears immediately after **[XQT ELD** (execute processor ELD). This sequence causes the **ES_EXPE_CMD** macrosymbol to expand internally into the following commmand line:

```
EXPE  <ES_NAME>  <ES_NEN>  <ES_OPT>  <ES_NEN>  <ES_NDOF>  --
                                  <ES_STOR>  1  10<ES_DIM>  <CSM_PRECISION>
```

which would eventually decode into:

```
EXPE  EX42  4  2  4  6  0  1  102  2
```

The execution of processor ELD with the above **EXPE** command is necessary for generation of various element datasets such as **DEF.EX42.*** and **DIR.EX42.*** (see the CSM Testbed

† In practice, it is often the other way around: The model properties and dimensions are fixed, while the discretization is varied. We have fixed the discretization here merely to simplify the example.

User's and Dataset Manuals, references 7.4.2 and 7.4-3, respectively). Note that the ES_ macrosymbols referenced in this example are all described in the Macrosymbol Glossary in Table 7.4-4.

Next, the NSECT command is used as a section property pointer. NSECT=1 means that the integrated constitutive matrix stored in the first column of dataset PROP.BTAB.2.101 will be employed by all elements whose nodal connectivity is defined on the following lines. The element nodal connectivity is then defined for four 4-node elements. Note that the numbering convention is counter-clockwise within each element (see Fig. 2.2-1). Boundary conditions and loads are then defined using the CON command of processor TAB, and the SYSVEC command of processor AUS, respectively (see Figure 7.4-2). The CON command suppresses all freedoms along $x = 0$ (the built-in edge), and the SYSVEC command distributes transverse nodal forces along the other edge, which add up to a unit load. Note that both boundary conditions and loads have been defined with respect to nodes rather than elements. Automatic consistent element nodal load generation will be available once the DEFINE LOADS command has been implemented within the ES processor shell.

Finally, procedure ES is called again to perform *automatic degree of freedom suppression*, using the DEFINE FREEDOMS command. In this case, the effect will be for processor ES1 to suppress drilling rotational freedoms (*i.e.*, the 6th degree of freedom) at all nodes, since element type EX42 has no stiffness associated with these freedoms. (Note: If the plate had blade stiffeners which were also modeled with EX42 elements, all 6 degrees of freedoms at nodes along the plate/stiffener intersection lines would be retained, since at least one of the intersecting elements at those nodes would possess the necessary stiffness.)

```
*Procedure PLATE_MODEL ( Lx=10.; Ly=5.; h=.1; E=1.e7; PR=.3; prec=2 )

     [xqt TAB
      START 9
.
. GENERATE NODES  ( rectangular grid, 3 x 3 nodes )
.
      JLOC
         1    0.,  0. , 0.   [Lx],  0. , 0.  3, 1, 3
         3    0., [Ly], 0.   [Lx], [Ly], 0.
.
. TABULATE MATERIAL AND SECTION PROPERTIES
.
     [xqt AUS.

         TABLE(NI=16,NJ=1): OMB DATA 1 1..
          I=1,2,3,4,5,6  : J=1 : [E] [PR] [E] <G> <G> <G>

         TABLE(NI=3,NJ=1,ITYPE=0): LAM OMB 1 1
          I=1,2,3 .      : J=1:   1 [h] 0.0.
.
. RUN CONSTITUTIVE PRE-PROCESSOR
.
     [xqt LAU
.
. GENERATE ELEMENTS
.
. ------------------------------------------------------------------
    *call ES ( function='DEFINE ELEMENTS'; ES_PROC=ES1; ES_NAME=EX42 )
. ------------------------------------------------------------------
     [xqt ELD
         <ES_EXPE_CMD>
          NSECT = 1
.          elt 1        elt 2          elt 3          elt 4
         1 2 5 4   :  2 3 6 5   :    4 5 8 7    :    5 6 9 8
```

Figure 7.4-1 Sample Preprocessing Procedure.

```
.
. IMPOSE BOUNDARY CONDITIONS
.
   [xqt TAB
    CON
       ZERO 1:6 : 1 : 4 : 9
.
. APPLY LOADS
.
   [xqt AUS
    SYSVEC : APPL FORC 1
   i=3 : j=3 : .25    :    i=3 : j=6 : .50     :    i=3 : j=9 : .25
.
. ------------------------------------------------------------------
     *call ES ( function = 'DEFINE FREEDOMS' )
. ------------------------------------------------------------------

*end
```

Figure 7.4-2 Sample Preprocessing Procedure.

7.4.8 Linear Analysis Examples

A sample linear static analysis procedure, which employs the procedure ES to invoke the appropriate ES*i* processors is shown in Figure 7.4-4. For purposes of illustration, the problem has been kept simple (notice that there are no procedure arguments), but keep in mind that many analysis procedures may involve more sophisticated features. The procedure in Figure 7.4-4 can be read as follows. The `[xqt E` directive causes processor E to construct `*.EFIL.*` datasets for all participating element types. Note that, while space for these datasets is reserved in the database, meaningful data has not yet been deposited there.

The first call to procedure ES then causes `INITIALIZE` to be stored in the `EFIL` datasets by all participating ES*i* processors — as prescribed by previous calls to procedure ES with `function = 'DEFINE ELEMENTS'`. For example, if the preprocessing example given in Figure 7.4-3 had preceded the call to `L_STATIC`, then only processor ES1, element type `EX42`, would be invoked to `INITIALIZE`. More information on the effect of the `INITIALIZE` command may be found in Section 2.2. The second call to procedure ES is to form the element material (linear) stiffness matrices for all elements in the model using `*call ES (function='FORM STIFFNESS/MATL')`. The element matrices are deposited in Segment 5 of the `EFIL` dataset. Since no other arguments are employed in this call to the procedure ES, the default values are implied. Thus, for example, the problem is assumed to be linear (`ES_NL_GEOM = 0`), and there is no need for a displacement dataset (whose name is given by argument `ES_DIS_DS`) to be input by the ES Processors.

Next, assembly of the element stiffness matrices into a system matrix is performed by processor K. Note that the element matrices have already been transformed to the computational (nodal degrees of freedom) bases, so that the function of processor K is merely to add appropriate submatrices.

Finally, processors INV and SSOL are executed in order to factor and solve the assembled system of equations, respectively. The displacement solution will be stored, as indicated by the `RESET` command for processor SSOL, in dataset `STAT.DISP.1.1`.

```
*procedure L_STATIC
..
.  ---------------------------
.  Initialize Element Datasets
.  ---------------------------
   [xqt E
.
.  -------------------------------------
.  Initialize Element Computational Data
.  -------------------------------------
   *call ES ( function = 'INITIALIZE' )
.
.  ---------------------------------------
.  Form Element Material Stiffness Matrices
.  ---------------------------------------
   *call ES ( function = 'FORM STIFFNESS/MATL' )
.
.  ---------------------------------
.  Assemble Material Stiffness Matrix
.  ---------------------------------
   [xqt K
.
.  ----------------------
.  Factor Stiffness Matrix
.  ----------------------
   [xqt INV
.
.  ----------------------
.  Solve for Displacements
.  ----------------------
   [XQT SSOL
    RESET SET=1, CON=1
.
*end
```

Figure 7.4-3 Sample Linear Static Analysis Procedure

7.4.9 Nonlinear Analysis Examples

A brief example of how to employ ESi processors in nonlinear analysis procedures, by including selected excerpts from an actual nonlinear static analysis procedure that will

hopefully convey the essential aspects. Typically, engineering-oriented users will invoke an existing nonlinear analysis procedure rather than writing their own, so this example is intended more for researchers involved in algorithm development.

The "skeleton" of a nonlinear static analysis procedure is shown in Figure 7.4-5, with only those aspects involving ES*i* processors shown. There is very little difference in the use of the procedure ES to invoke ES processors from what was employed in the *linear* static analysis procedure compare with Figure 7.4-4, except that some additional arguments must be explicitly defined.

First, the usual call to INITIALIZE for all ES*i* processors is present. This call enables the participating element processors to generate, and store, any data that will be used repeatedly during the analysis — rather than having to recompute it at each iteration of every analysis load step.

Next, the nested load-step and iteration loops, typical of most incremental/iterative nonlinear solution algorithms for structural analysis are encountered. Within these loops, it is necessary to compute residual force vectors (right-hand sides) at each iteration and tangent stiffness matrices at selected load steps/iterations.

The assembled residual force vector is computed by first initializing a system internal force vector (using processor VEC), calling the procedure ES to form/assemble all element contributions to the internal force vector, and finally subtracting the assembled internal force vector from a load-step scaled external force vector, which is assumed to have been generated elsewhere.

Notice that in the FORM FORCE/INT call to procedure ES, arguments ES_NL_GEOM, ES_CORO, ES_DIS_DS, ES_ROT_DS and ES_FRC_DS are each explicitly defined. The reader is urged to look up these arguments in the Macrosymbol/Argument Glossary, Table 7.4-4. These arguments let the ES*i* processors know that the problem is geometrically nonlinear (both globally and at the element level). The corotational algorithm is to be employed to make the rotational motion "appear" small at the element level, but allow it to be arbitrarily large globally. The current displacement dataset is called TOT.DISP.<$step>, where <$step> is the load-step number; the current rotation (pseudo-vector) dataset is called

TOT.ROTA.<$step>; and the current system internal force vector into which the element contributions are to be assembled is called INT.FORC.<$step>.

Finally, the formation/factorization of the tangent stiffness matrix is shown in Figure 7.4-5 which involves first the formation/transformation of the element tangent stiffness matrices by ES*i* processors, which are deposited in the *.EFIL.* datasets; then the assembly of the element matrices into the system tangent stiffness matrix by processor K; and finally the factorization of the assembled (system) tangent stiffness matrix by processor INV. Note that both material and geometric stiffness contributions have been superimposed at the element level.

```
*procedure NL_STATIC ( NUM_STEPS = 1; NUM_ITERS = 10 )

. ------------------------------------
. Initialize Element Computational Data
. ------------------------------------
  *call ES ( function = 'INITIALIZE' )
.   :
  *do $step = 1, [NUM_STEPS]

     *do $iter = 1, [NUM_ITERS]

.       -------------------------
.       FORM RESIDUAL FORCE VECTOR
.       -------------------------
.
        [xqt VEC
         INT.FORC <- 0.
..
        *call ES ( function    = 'FORM FORCE/INT'; --
                   es_nl_geom =  2; es_coro = 1; --
                   es_dis_ds  = TOT.DISP.<$step>; --
                   es_rot_ds  = TOT.ROTA.<$step>; --
                   es_frc_ds  = INT.FORC.<$step> )

       [xqt VEC
        RES.FORC  <-  <load_factor> * EXT.FORC  -  INT.FORC.<$step>
.   :
.       ------------------------------------
.       FORM/FACTOR TANGENT STIFFNESS MATRIX
.       ------------------------------------

       *call ES ( function    = 'FORM STIFFNESS/TANG' ; --
                  es_nl_geom =  2; es_coro = 1; --
                  es_dis_ds  = TOT.DISP.<$step>; --
                  es_rot_ds  = TOT.ROTA.<$step>; --
                  es_frc_ds  = INT.FORC.<$step> )

       [xqt K
       [xqt INV
.   :
      *enddo

   *enddo
```

Figure 7.4-4 Sample Nonlinear Static Analysis Procedure Excerpts

7.4.10 Postprocessing Examples

By postprocessing refers to functions which can be performed after the displacement solution has been obtained for a linear or nonlinear structural analysis. For example, in a nonlinear static (or transient) analysis, the user may choose not to archive the stresses and strains which were used as intermediate variables during the process of obtaining a displacement solution history. The user may then compute stresses and/or strains at selected load (or time) steps and save these in the database for perusal. The end-user phase of postprocessing is of course the actual printing or display of the results (displacements, stresses, strains, etc.); however, the main interest here is in the prerequisite functions that are performed by ES*i* processors.

An example of a postprocessing procedure that employs ES*i* processors to form, and archive in the database, both stresses and strains after the mainstream analysis has already been performed. Procedure `STRESS_STRAIN` contains arguments to select the stress/strain locations (the default is at element centroids), component directions (the default is in the element local stress/strain coordinate system), existing displacement and rotation (for nonlinear analysis) datasets to be employed for strain computation.

Note that there is a step loop in the procedure, and that both stresses and strains for all participating elements are formed by just a single call to procedure `ES` per step. This is because the `FORM STRESS` command automatically causes strains to be formed as well as stresses, and when both `ES_STRAIN_DS` and `ES_STRESS_DS` are explicitly defined, then both of these quantities are also output to the database.

```
*procedure STRESS_STRAIN ( LOCATION  = CENTROIDS ; DIRECTION = 0    ; --
                           NL_GEOM   = 0         ; CORO      = 0    ; --
                           NUM_STEPS = 1         ; STEPS     = 1:1  ; --
                           DIS_DS    = TOT.DISP  ; ROT_DS    = TOT.ROTA )

.  ----------------------
.  Loop on Solution Steps
.  ----------------------
   *def/i steps[1:[num_steps]]  =  [steps]

   *do \$is = 1, [num_steps]

      *def/i step = <steps[<\$is>]>

.     ------------------------------------------------
.     Invoke Element Processors to Form Stress/Strain
.     ------------------------------------------------
      *call ES ( function    = 'FORM STRESS'   ; --
                 es_nl_geom = [nl_geom]        ; --
                 es_coro    = [coro]           ; --
                 es_dis_ds  = [DIS_DS].<step>  ; --
                 es_rot_ds  = [ROT_DS].<step>  ; --
                 es_str_dir = [DIRECTION]      ; --
                 es_str_loc = [LOCATION]       ; --
                 es_strain_ds = <step>         ; --
                 es_stress_ds = <step>  )

   *enddo

*end
```

Figure 7.4-5 Sample Postprocessing Procedure

7.4.11 PROCEDURE LISTING

```
*procedure ES ( function    ; --

               es_proc     ; --

               es_name     ; --

               es_pars     ; --
```

```
. beg_update 06_MAY_1989 [GMS] ES_COUNT
                es_count = 0 ; -- . 0/1=>create/accum ES.SUMMARY
. end_update
                es_coro = 1; --
                es_nl_geom = 0 ; --
                es_nl_matl = 0 ; --
                es_nl_load = 0 ; --
                es_dis_ds = STAT.DISP.1.1        ; --
                es_dof_ds = ES.DOFS              ; --
                es_ecc_ds = WALL.PROP            ; --
                es_frc_ds = INT.FORC.1.1         ; --
                es_rot_ds = STAT.ROTA.1.1        ; --
                es_sum_ds = ES.SUMMARY           ; --
                es_mass_diag = DEM               ; --
                es_strain_ds = ' '               ; --
                es_stress_ds = ' '               ; --
                es_str_loc   = CENTROIDS         ; --
                es_str_dir   = 0                 ; --
                num_con_ds = 1                   ; --
. beg_update 26_APR_1989 [GMS] CONSISTENT LOADS
                es_load_factor = 1.0             ; --
                es_load_set    = 1               ; --
. end_update 26_APR_1989 [GMS] CONSISTENT LOADS
                ldi        = 1 )
.  ------------------------------------------------------
.  Generic Procedure for Structural Element (ES) Processors
.  ------------------------------------------------------
```

```
*if < <IFELSE([function];DEFINE ELEMENTS;1;0)> > /then
   *def/a es_proc       = [es_proc]
   *def/a es_name       = [es_name]
   *def/d es_pars[1:10] = [es_pars]
   *def/a es_sum_ds     = [es_sum_ds]
.     ----------------------------
.     Initialize ES SUMMary Dataset
.     ----------------------------
      *find dataset [ldi] <es_sum_ds> /seq=es_ids
. beg_update 06_MAY_1989 [GMS] ES_COUNT
      *if < < <es_ids> /le 0 >  /or  < [es_count] /eq 0 > > /then
         *put dataset [ldi] <es_sum_ds> /mrat=2000 /seq=es_ids
         *def/i es_num = 1
      *else
         *find record [ldi], <es_sum_ds>, ES_PROC /nor=es_num
         *def/i es_num = < <es_num> + 1 >
      *endif
. end_update
      *m2g /name==es_proc /type=a          [ldi] <es_ids> ES_PROC.<es_num>
      *m2g /name==es_name /type=a          [ldi] <es_ids> ES_NAME.<es_num>
      *m2g /name==es_pars /type=d /maxn=10 [ldi] <es_ids> ES_PARS.<es_num>
.
.  ----------------
.  Run ES Processor
.  ----------------
   [IQT <es_proc>
       [function]
```

```
.
.  ------------------------------------
.  Define macrosymbol for ELD/EXPE command
.  ------------------------------------
.
      *if < <es_stor> /gt 0 > /then
         *def/i es_nst = < ((<es_stor>-1)/<es_nen>) + 1 >
      *else
         *def/i es_nst = 1
      *endif
. BEG_UPDATE MAY_5_1989 [SNO] EXPE CHANGE
.     *def/a ES_EXPE_CMD=='EXPE '[es_name]' '<es_nen>' '--
      *def/a ES_EXPE_CMD=='EXPE '[es_name]' '4' '--
. END_UPDATE
<es_opt>' '<es_nen>' '<es_ndof>' '<es_nst>' 1 10'<es_dim>' '<csm_precision>
*else
  *find dataset [ldi] [es_sum_ds] /seq=es_ids
.
  *def/a es_mass_diag == [es_mass_diag]
  *def/a es_mass_ds   == [es_mass_diag].DIAG
.
.  ------------------------------------------------
.  Find Number of Element Processors from Database
.  ------------------------------------------------
   *find record [ldi], <es_ids>, ES_PROC /nor=num_es_proc
   *if < <num_es_proc> /le 0 > /then
      *remark Cannot find any ES Processor names in dataset [es_sum_ds]
```

```
      e
   *endif
   *if < <IFELSE([function];DEFINE FREEDOMS;1;0)> > /then
      *call ES_DOFS ( function=INITIALIZE; es_dof_ds=[es_dof_ds] )
   *endif
.  ----------------------------------
.  Process all Element Processors Found
.  ----------------------------------
   *do $ies = 1, <num_es_proc>
      *g2m /name==ES_PROC /type=a [ldi] <es_ids> ES_PROC.<$ies>
      *g2m /name==ES_NAME /type=a [ldi] <es_ids> ES_NAME.<$ies>
      *g2m /name==ES_PARS /type=d [ldi] <es_ids> ES_PARS.<$ies>
      *g2m /name==ES_NSTR /type=d [ldi] <es_ids> ES_NSTR.<$ies>
      *def/i ES_CORO    = [es_coro]
      *def/a ES_DIS_DS  = [es_dis_ds]
      *def/a ES_DOF_DS  = [es_dof_ds]
      *def/a ES_ECC_DS  = [es_ecc_ds]
      *def/i ES_FIXI    = [es_fixi]
      *def/a ES_FRC_DS  = [es_frc_ds]
      *def/i ES_NL_GEOM = [es_nl_geom]
      *def/i ES_NL_LOAD = [es_nl_load]
      *def/i ES_NL_MATL = [es_nl_matl]
      *def/a ES_ROT_DS  = [es_rot_ds]
. beg_update 26_APR_1989 [GMS] CONSISTENT LOADS
      *def/g ES_LOAD_FACTOR = [es_load_factor]
      *def/i ES_LOAD_SET    = [es_load_set]
. end_update 26_APR_1989 [GMS] CONSISTENT LOADS
```

```
      *def/a ES_STR_DIR = [es_str_dir]
      *def/a ES_STR_LOC = [es_str_loc]
      *if < <IFELSE([es_strain_ds]; ;0;1)> > /then        . (if not blank)
         *def/a ES_STRAIN_DS = [es_strain_ds]
      *endif
      *if < <IFELSE([es_stress_ds]; ;0;1)> > /then        . (if not blank)
         *def/a ES_STRESS_ds  = [es_stress_ds]
      *endif
      [XQT <es_proc>
       [function]    . DEFINE FREEDOMS | FORM  FORCE | STIFFNESS | ...
   *enddo
   *if < <IFELSE([function];DEFINE FREEDOMS;1;0)> > /then
      *call ES_DOFS ( function=FINALIZE; es_dof_ds=[es_dof_ds]; --
                                        num_con_ds=[num_con_ds] )
   *endif
*endif
STOP
*end
. =DECK ES_DOFS
*procedure ES_DOFS ( function=INITIALIZE; es_dof_ds=ES.DOFS; --
                                          num_con_ds=1; ldi=1 )
   *if < <IFELSE([function];INITIALIZE;1;0)> > /then
      *remark
      *remark ES_DOFS: Performing Automatic DOF Suppression for ES Elts
      *def/i ns_overwrite == <true>
.     ----------------------------
.     Initialize Element DOF Dataset
```

```
.     ------------------------------
      *g2m /name=parameters /type=i /maxn=18 [ldi] JDF1.BTAB.1.8 DATA.1
      *def/i num_nodes = <parameters[1]>
      *def/i num_dofs  = <parameters[2]>
      [XQT VEC
       INIT_TAB [es_dof_ds] <num_dofs> BY <num_nodes>
   *elseif < <IFELSE([function];FINALIZE;1;0)> > /then
.     ------------------------------------------------
.     Merge Element DOF Dataset with Constraint Datasets
.     ------------------------------------------------
      *def/i num_cds = 0
      *do $cds = 1, [num_con_ds]
         *def/a con_ds = CON..<$cds>
         *find dataset [ldi], <con_ds> /seq=id_cds
         *if < <$cds> /eq 1 > /then
          [XQT VEC
         *endif
         *if < <id_cds> /gt 0 > /then
            *remark ES_DOFS Processing Constraint Dataset: <con_ds>
             MERGE_DOF [es_dof_ds] -> <con_ds>
            *def/i num_cds = < <num_cds> + 1 >
         *endif
      *enddo
      *remark
      *remark ES_DOFS Processed <num_cds> Constraint Datasets
      *remark
      *def/i ns_overwrite == <false>
```

```
   *endif

*end
```

7.4.12 REFERENCES

7.4-1 Stanley, G. M.; and Nour-Omid, S.: *The Computational Structural Mechanics Testbed Generic Structural-Element Processor Manual.* NASA CR-181728, March 1990.

7.4-2 Stewart, Caroline B.: *The Computational Structural Mechanics Testbed Users Manual.* NASA TM-100644, October 1989.

7.4-3 Stewart, Caroline B.: *The Computational Structural Mechanics Testbed Data Library Description.* NASA TM-100645, October 1988.

7.5 Procedure FACTOR

7.5.1 GENERAL DESCRIPTION

Procedure **FACTOR** decomposes or factors a system matrix using a solver defined by the global macrosymbol **solver_name**. If **solver_name** is defined to be **INV**, the original nodal-block sparse solver implemented in processor INV will be used (see Section 6.7 of reference 7.5-1). If **solver_name** is defined to be **BAND**, a variable-bandwidth direct solver implemented in processor BAND will be used (see Section 8.5 of reference 7.5-1).

7.5.2 PROCEDURE USAGE

Procedure **FACTOR** is used by preceding the procedure name by the ***call** directive, and following it by a list of arguments enclosed in parentheses. Procedure arguments are order-independent, and most have default values thus making them optional. The formal syntax is as follows:

```
*call FACTOR ( arg1 = val1 ; arg2 = val2 ; ...)
```

where **arg1** and **arg2** represent argument names, and **val1** and **val2** represent their corresponding values. Note that semi-colons are required between arguments, and a double dash (--) may be used to continue the argument list on the next line.

The allowable arguments for procedure **FACTOR** are summarized in the following table, along with their default values (if they exist). Note that arguments without defaults are generally *mandatory*, while those with defaults are generally optional. Exceptions to this rule are noted in the following section under detailed argument descriptions.

For procedure **FACTOR**, the following table lists each argument, its default value and meaning.

Argument	Default Value	Meaning
CONSTRAINT_SET	1	Constraint set number
INPUT_MATRIX	K	First word of the name of the dataset containing the assembled stiffness matrix
OUTPUT_MATRIX	K	Second word of the name of the dataset containing the factored system stiffness matrix
INPUT_LDI	1	Logical device index for input_matrix
OUTPUT_LDI	1	Logical device index for output_matrix

Tables 7.5-1 and 7.5-2 list the datasets used or created by procedure **FACTOR** and the processors invoked by procedure **FACTOR**, respectively.

Table 7.5-1 Datasets Input/Output by procedure FACTOR			
Dataset	*Description*	*Input*	*Output*
AMAP..ic2.isize	Factorization Map for INV	√	
CON..*j*†	Constraints	√	
INV.*O_M.j*†*	Factored Stiffness Matrix		√
JDF1.BTAB.1.8	Model Summary	√	
JSEQ.BTAB.2.17	Nodal Elimination Sequence	√	
KMAP..nsubs.ksize	Model Connectivity Map	√	
I_M*.SPAR.*jdf2**	Assembled Stiffness Matrix	√	

† *j* = <constraint_set> **O_M* = <output_matrix> ***I_M* = <input_matrix>

Table 7.5-2 Processors Invoked by procedure FACTOR		
Procedure	*Type*	*Function*
INV	Internal	Factor stiffness matrix in nodal-block-sparse format
BAND	External	Factor stiffness matrix in variable-bandwidth format

7.5.3 ARGUMENT DESCRIPTIONS

7.5.3.1 CONSTRAINT_SET

Constraint set number (default: 1). This argument selects which constraint set to be used in solving the linear system of equations.

7.5.3.2 INPUT_MATRIX

First word of the dataset name containing the assembled stiffness matrix (default: K).

7.5.3.3 OUTPUT_MATRIX

Second word of the dataset name containing the factored system stiffness matrix (default: K). This notation is only valid if the macrosymbol **solver_name** is set to **INV**.

7.5.3.4 INPUT_LDI

Input logical device index containing the input matrix (default: 1).

7.5.3.5 OUTPUT_LDI

Output logical device index containing the output matrix (default: 1).

7.5.4 USAGE GUIDELINES AND EXAMPLES

Procedure **FACTOR** may be used by preceding the procedure name by the ***call** directive. Procedure arguments may be changed from their default values by including any or all of the arguments and their new values when the procedure is called. A space or blank is required between the end of the procedure name and the left parenthesis.

```
*call FACTOR ( INPUT_MATRIX = K ; CONSTRAINT_SET = 1 )
```

Before procedure **FACTOR** is called, the global macrosymbol **solver_name** should be defined as described in Section 7.5.1; otherwise, the default value of **INV** will be used.

7.5.5 LIMITATIONS

None.

7.5.6 ERROR MESSAGES AND WARNINGS

None.

7.5.7 PROCEDURE FLOWCHART

Procedure **FACTOR** is self-contained with no subprocedures.

7.5.8 PROCEDURE LISTING

```
*procedure FACTOR ( input_matrix   = K ; output_matrix = K ; --
                    input_ldi      = 1 ; output_ldi    = 1 ; --
                    constraint_set = 1 )
.
. Procedure to factor a system matrix using different solvers.
. The solver is selected using the global macrosymbol "solver_name".
.
*if < ifeqs( <solver_name>; ) > /then
*def/a solver_name == INV
*endif
.
```

```
*if < ifeqs( <solver_name>;INV) > /then
.
. Use original nodal block sparse solver originally in SPAR
.
[XQT INV
     reset K     = [input_matrix]
     reset KLIB  = [input_ldi]
     reset KILIB = [output_ldi]
     reset CON   = [constraint_set]
     reset LRA   = 40000
     reset DZERO = 1.E-20
     reset spdp  = <csm_precision>
.
*elseif < ifeqs( <solver_name>;BAND) > /then
.
. Use Banded solver
.
[XQT BAND
    reset meth=28, ldl=1, klib=[input_ldi]
    reset ncon=[constraint_set], dzero=1.0e-20, k=[input_matrix]
    reset opt=1, res=-1
*endif
STOP
.
. Define factored-matrix parameters as global macrosymbols
.
*if < ifeqs( <solver_name>;ITER) > /then
   *def/e coef_det == 1.0E+01
   *def/i exp10_det == 1
   *def/i num_neg == 0
   *def/i sign_det == 1
*else
   *remark Determinant of [input_matrix] = <coef_det> * 10 ** <exp10_det>
   *remark Number of negative roots = <num_neg>
   *def/i sign_det == <SIGN(1.;<coef_det>)>
*endif
*end
```

7.5.9 REFERENCES

7.5-1 Stewart, Caroline B.: *The Computational Structural Mechanics Testbed User's Manual.* NASA TM-100644, October 1989.

7.6 Procedure FORCE

7.6.1 GENERAL DESCRIPTION

Procedure **FORCE** drives the calculation of force vectors for either linear or nonlinear analyses.

7.6.2 PROCEDURE USAGE

Procedure **FORCE** may be used by preceding the procedure name by the ***call** directive, and following it by a list of arguments separated by semicolons (;) and enclosed in parentheses. Procedure arguments are order-independent, and most have default values thus making them optional. The formal syntax is as follows:

```
*call FORCE ( arg1 = val1 ; arg2 = val2 ; ...)
```

For procedure **FORCE**, the following table lists each argument, its default value and meaning.

Argument	Default Value	Meaning
COROTATION	1	Corotational flag (0, 1, 2)
DISPLACEMENT		First two words of displacement dataset name
LDI	1	Logical device index
LOAD_SET	1	Load set number
LOAD_FACTOR	1.0	Load factor
NL_GEOM	0	Geometric nonlinearity flag
NL_LOAD	0	Nonlinear load terms (live terms)
NL_MATL	0	Material nonlnearity flag
ROTATION		First two words of rotation-vector dataset name
TYPE	RESIDUAL	Type of force vector (residual, external, internal)
INPUT_FORCE		Input force vector dataset name
OUTPUT_FORCE	SYS.FORCE	Output force vector dataset name

Tables 7.6-1, 7.6-2, and 7.6-3 list the datasets used or created by procedure **FORCE**, the procedures invoked by procedure **FORCE**, and the processors invoked by procedure **FORCE**, respectively.

Table 7.6-1 Datasets Input/Output by procedure FORCE			
Dataset	*Description*	*Input*	*Output*
ES.SUMMARY	ES Processor Status	√	√
INT.FORC.*i.j*†	System Internal Force Vector		√
STAT.DISP.*i.j*†	System Displacement Vector		√
STAT.REAC.*i.j*†	System Reaction Force Vector		√

† i = <load_set> and j = <cons_set>

Table 7.6-2 Sub-procedures Invoked by procedure FORCE		
Procedure	*Type*	*Function*
ES	External	Element utility procedure
FORCE		Main Procedure

Table 7.6-3 Processors Invoked by procedure FORCE		
Procedure	*Type*	*Function*
ES*i*	External	Element processors based on GEP
VPRT	Internal	Print SYSVEC-format vectors

7.6.3 ARGUMENT DESCRIPTIONS

7.6.3.1 COROTATION

Corotational update switch for large-rotation problems (default: 1). This switch should be set to 1 when the model involves finite elements that require corotation for geometric nonlinearity. This is true of most beam and shell elements, and may be true for some solid (3D) elements used to model shell structures. Consult the appropriate element processor (ES*i*) section in the CSM Testbed User's Manual (see ref. 7.6-1) for specific guidelines.

7.6.3.2 DISPLACEMENT

First two words of the dataset name for the displacement solution.

7.6.3.3 INPUT_FORCE

Full dataset name for the input or basic force vector.

7.6.3.4 OUTPUT_FORCE

Full dataset name for the output force vector (default: SYS.FORCE).

7.6.3.5 LDI

Logical device index (default: 1).

7.6.3.6 LOAD_FACTOR

Load factor used to scale the basic applied loads or displacements (default: 1.0).

7.6.3.7 NL_GEOM

Geometric nonlinearity level: 0, 1, or 2 (default: 2). A value of zero means that the problem is geometrically linear; a value of one means that the geometric nonlinearity will be handled globally (*i.e.*, using corotational updates only); and a value of two means that the nonlinear element strain-displacement relations will be used in addition to any global treatment of geometric nonlinearity.

7.6.3.8 ES_NL_MATL

Material nonlinearity switch. If ES_NL_MATL is greater than zero, then the element is materially nonlinear. If ES_NL_MATL is zero, then the element is materially linear.

7.6.3.9 NL_LOAD

Nonlinear loading index for live loads (default: 0).

7.6.3.10 ROTATION

First two words of the dataset name for the rotation solution.

7.6.3.11 TYPE

Type of force vector to form (default: RESIDUAL. This argument selects the type of force vector to form (RESIDUAL, INTERNAL, or EXTERNAL).

7.6.4 USAGE GUIDELINES AND EXAMPLES

Procedure FORCE may be used by preceding the procedure name by the *call directive. Procedure arguments may be changed from their default values by including any or all of the arguments and their new values when the procedure is called. A space or blank is required between the end of the procedure name and the left parenthesis. If the default values of the procedure arguments are to be used, then only the procedure name is required.

```
*call FORCE ( LOAD_SET = 1 ; INPUT_FORCE = APPL.FORC.[load_set].1;
              OUTPUT_FORCE = EXT.FORC; TYPE = EXTERNAL;
              LOAD_FACTOR = 2.5; LDI = 1 )
```

7.6.5 LIMITATIONS

Applicable only to elements implemented using the generic element processor template.

7.6.6 ERROR MESSAGES AND WARNINGS

None.

7.6.7 PROCEDURE FLOWCHART

FORCE	(main procedure)
ES	(internal force calculation)

7.6.8 PROCEDURE LISTING

```
*procedure FORCE ( type = RESIDUAL ; --  .  RESIDUAL | EXTERNAL | INTERNAL
                   input_force             ; --
                   output_force = SYS.FORCE ; --
                   load_set     = 1        ; --
                   load_factor  = 1.0      ; --
                   nl_geom = 0             ; --
                   nl_matl = 0             ; --
                   nl_load = 0             ; --
                   displacement            ; --
                   rotation                ; --
                   corotation   = 0        ; --
                   ldi      = 1 )
      [xqt VEC
.
.      ------------------------------
.      Initialize Output Force Vector
.      ------------------------------
       *find [ldi] [output_force] /seq=ids_oforce
       *if < <ids_oforce> /le 0 > /then
```

```
        *g2m /name=num_nodes /type=i [ldi] MODEL.SUMMARY NUM_NODES
        *g2m /name=num_dofs  /type=i [ldi] MODEL.SUMMARY NUM_DOFS
         INIT_VEC [output_force] <num_dofs> BY <num_nodes>
      *else
         [output_force] <- 0.
      *endif
.     ----------------------------
.     Add Scaled Input Force Vector
.     ----------------------------
      *if < <IFELSE([input_force]; ;0;1)> > /then
         *find [ldi] [input_force] /seq=ids_iforce
         *if < <ids_iforce> /gt 0 > /then
          [output_force] <- [load_factor] [input_force]
         *endif
      *endif
.     ------------------
.     Add Element Forces
.     ------------------
      *call ES ( function = 'FORM FORCE/[type]'; --
                 ldi        = [ldi]             ; --
                 es_nl_geom = [nl_geom]         ; --
                 es_nl_load = [nl_load]         ; --
                 es_coro    = [corotation]      ; --
                 es_dis_ds  = [displacement]    ; --
                 es_rot_ds  = [rotation]        ; --
                 es_load_factor = [load_factor]; --
                 es_load_set    = [load_set]    ; --
                 es_frc_ds  = [output_force] )
*end
```

7.6.9 REFERENCES

7.6-1 Stewart, Caroline B.: *The Computational Structural Mechanics Testbed User's Manual.* NASA TM-100644, October 1989.

THIS PAGE LEFT BLANK INTENTIONALLY.

7.7 Procedure IMPERFECTION

7.7.1 GENERAL DESCRIPTION

Procedure **IMPERFECTION** to superpose a set of normalized buckling modes to form an initial geometric imperfection shape. This imperfection shape is then added to the nodal coordinates of the structure. The original nodal coordinates are first copied into another dataset and saved for subsequent use if desired.

7.7.2 PROCEDURE USAGE

Procedure **IMPERFECTION** is used by preceding the procedure name by the ***call** directive, and following it by a list of arguments enclosed in parentheses. Procedure arguments are order-independent, and most have default values thus making them optional. The formal syntax is as follows:

```
*call IMPERFECTION ( arg1 = val1 ; arg2 = val2 ; ...)
```

where **arg1** and **arg2** represent argument names, and **val1** and **val2** represent their corresponding values. Note that semi-colons are required between arguments, and a double dash (--) may be used to continue the argument list on the next line.

The allowable arguments for procedure **IMPERFECTION** are summarized in the following table, along with their default values (if they exist). Note that arguments without defaults are generally *mandatory*, while those with defaults are generally optional. Exceptions to this rule are noted in the following section under detailed argument descriptions.

For procedure **IMPERFECTION**, the following table lists each argument, its default value and meaning.

Argument	Default Value	Meaning
AMPS	1.0	Amplitude of each buckling mode
MODE	1	Mode numbers
N_MODES	1	Number of buckling modes used in computing imperfection shape
LDI	1	Logical device index

Processor IMP is used to perform the modal superposition (see reference 7.7-1).

7.7.3 ARGUMENT DESCRIPTIONS

7.7.3.1 AMPS

Amplitudes of buckling mode shapes (default: 1.0). This argument gives an array of amplitudes for the buckling mode shapes. The array **amps** has **N_MODES** entries.

7.7.3.2 LDI

Logical device index (default: 1).

7.7.3.3 MODE

Buckling mode shape numbers (default: 1). This argument gives an array of mode numbers to be used in generating the initial geometric imperfection shape. The array MODE has N_MODES entries.

7.7.3.4 N_MODES

Number of buckling mode shapes to use (default: 1). This argument gives the total number of buckling mode shapes to be superposed to form the initial geometric imperfection shape.

7.7.4 USAGE GUIDELINES AND EXAMPLES

Procedure IMPERFECTION may be used by preceding the procedure name by the *call directive. Procedure arguments may be changed from their default values by including any or all of the arguments and their new values when the procedure is called. A space or blank is required between the end of the procedure name and the left parenthesis. If the default values of the procedure arguments are to be used, then only the procedure name is required.

```
*call IMPERFECTION ( n_modes=1; modes=1; amps=1.0 )
```

7.7.5 LIMITATIONS

Procedure IMPERFECTION only considers normalized buckling modeshapes in forming the initial geometric imperfection shape.

7.7.6 ERROR MESSAGES AND WARNINGS

None.

7.7.7 PROCEDURE FLOWCHART

IMPERFECTION	(main procedure)
COPY_DS	(copy datasets)
SWITCH_DS	(switch datasets)

7.7.8 PROCEDURE LISTING

```
*procedure IMPERFECTION ( n_modes=1; modes=1; amps=1.0; ldi=1 )
.
. ----------------------------------------------------------------
. CLAMP Procedure to Superpose Buckling Modes as Initial Imperfections
. ----------------------------------------------------------------
.
  *remark
  *remark Imposing Geometric Imperfections
  *remark
  *remark - - -  Saving old nodal coordinates in JLOC.OLD.2.5
   *call COPY_DS   ( from_ldi=[ldi]; from_ds = JLOC.BTAB.2.5 ; --
     to_ldi = [ldi]; to_ds = JLOC.OLD.2.5 )
   *call SWITCH_DS ( ldi=[ldi]; --
     ds_1 =  JLOC.BTAB.2.5 ; ds_2  = JLOC.OLD.2.5 )
  *remark
  *remark - - -  Adding linear combination of buckling modes to coords
  *remark
   [XQT IMP
     *def/i     modes[1:[n_modes]] = [modes]
     *def/e12.4 amps[1:[n_modes]]  = [amps]
     *do $im = 1, [n_modes]
        SUPERPOSE <amps[<$im>]> BUCK.MODE.1.1 /mode=<modes[<$im>]> /NORM
     *remark     - - -  Mode = <modes[<$im>]> , Amplitude = <amps[<$im>]>
     *enddo
*end
```

7.7.9 REFERENCES

7.7-1 Stewart, Caroline B.: *The Computational Structural Mechanics Testbed User's Manual.* NASA TM-100644, October 1989.

THIS PAGE LEFT BLANK INTENTIONALLY.

7.8 Procedure INITIALIZE

7.8.1 GENERAL DESCRIPTION

Procedure **INITIALIZE** performs standard initialization functions and resequencing of the joint.

7.8.2 PROCEDURE USAGE

Procedure **INITIALIZE** is used by preceding the procedure name by the ***call** directive, and following it by a list of arguments enclosed in parentheses. Procedure arguments are order-independent, and most have default values thus making them optional. The formal syntax is as follows:

```
*call INITIALIZE ( arg1 = val1 ; arg2 = val2 ; ...)
```

where **arg1** and **arg2** represent argument names, and **val1** and **val2** represent their corresponding values. Note that semi-colons are required between arguments, and a double dash (--) may be used to continue the argument list on the next line.

The allowable arguments for procedure **INITIALIZE** are summarized in the following table, along with their default values (if they exist). Note that arguments without defaults are generally *mandatory*, while those with defaults are generally optional. Exceptions to this rule are noted in the following section under detailed argument descriptions.

For procedure **INITIALIZE**, the following table lists each argument, its default value and meaning.

Argument	Default Value	Meaning
CONSTRAINT_SET	1	Constraint set number
LDI	1	Logical device index
MAXCON	35	Maximum number of joints connected to any one joint
RENUMBER	<false>	Flag to resequence the joints internally
RSEQ_METHOD	-1	Resequencing method

7.8.3 ARGUMENT DESCRIPTIONS

7.8.3.1 CONSTRAINT_SET

Constraint set number (default: 1). This argument selects which constraint set to be used in solving the linear system of equations.

7.8.3.2 MAXCON

Maximum number of joints connected to any one joint (default: 35).

7.8.3.3 RENUMBER

Flag to resequence the joints internally (default: <false>).

7.8.3.4 RSEQ_METHOD

Resequence method (default: -1). This argument selects which resequencing method to use. The procedure **RESEQUENCE** is called for this task (see Section 7.12).

7.8.3.5 LDI

Logical device index (default: 1).

7.8.4 USAGE GUIDELINES AND EXAMPLES

Procedure **INITIALIZE** is used by preceding the procedure name by the *call directive. Procedure arguments may be changed from their default values by including any or all of the arguments and their new values when the procedure is called. A space or blank is required between the end of the procedure name and the left parenthesis.

```
*call INITIALIZE ( LDI = 1 ; CONSTRAINT_SET = 1 ; RENUMBER = <true> )
```

7.8.5 LIMITATIONS

None.

7.8.6 ERROR MESSAGES AND WARNINGS

None.

7.8.7 PROCEDURE FLOWCHART

INITIALIZE	(main procedure)
ES	(initialize element processor)
RESEQUENCE	(resequence the joints)
MODEL_SUMMARY	(create MODEL_SUMMARY dataset)

7.8.8 PROCEDURE LISTING

```
*procedure INITIALIZE  ( renumber=<false>; constraint_set = 1; ldi=1; --
                         rseq_method = -1; maxcon = 35 )
.  ---------------------------------------
.  Perform Standard Initialization Functions
.  ---------------------------------------
.  --------------------------------
.  INITIALIZE ELEMENT CONFIGURATION
.  --------------------------------
   [XQT E
   *call ES ( function = 'INITIALIZE'; ldi=[ldi] )
   *Remark    Element configuration initialized.
.  ------------------------------------
.  RESEQUENCE NODES FOR SOLVER EFFICIENCY
.  ------------------------------------
   *if < [renumber] > /then
      *call RESEQUENCE ( rseq_method=[rseq_method]; --
                         ldi=[ldi]; maxcon=[maxcon] )
   *endif
.  ----------------------------
.  CONSTRUCT MATRIX TOPOLOGY MAPS
.  ----------------------------
   [XQT TOPO
          reset maxsub = 100000, lram=40000, lrkm=100000
   RESET BLIB=[ldi]
   RESET HLIB=[ldi]
   RESET ILIB=[ldi]
   *if < ifeqs( <solver_name>;INV ) > /then
   *else
   reset AMAP=0
   *endif
.  ------------------
.  CONSTRUCT DOF TABLE
.  ------------------
   [XQT VEC
      INIT_DOF CON..[constraint_set] -> DOF.TABL
```

```
    *Remark    DOF TABLE initialized.
.  ---------------------------
.  CREATE MODEL SUMMARY DATASET
.  ---------------------------
    *call MODEL_SUMMARY ( ldi=[ldi] )
  STOP
*end
```

7.8.9 REFERENCES

7.8-1 Stewart, Caroline B.: *The Computational Structural Mechanics Testbed User's Manual.* NASA TM-100644, October 1989.

7.9 Procedure MASS

7.9.1 GENERAL DESCRIPTION

Procedure **MASS** forms the assembled unconstrained mass matrix.

7.9.2 PROCEDURE USAGE

Procedure **MASS** is used by preceding the procedure name by the ***call** directive, and following it by a list of arguments enclosed in parentheses. Procedure arguments are order-independent, and most have default values thus making them optional. The formal syntax is as follows:

```
*call MASS ( arg1 = val1 ; arg2 = val2 ; ...)
```

where **arg1** and **arg2** represent argument names, and **val1** and **val2** represent their corresponding values. Note that semi-colons are required between arguments, and a double dash (--) may be used to continue the argument list on the next line.

The allowable arguments for procedure **MASS** are summarized in the following table, along with their default values (if they exist). Note that arguments without defaults are generally *mandatory*, while those with defaults are generally optional. Exceptions to this rule are noted in the following section under detailed argument descriptions.

For procedure **MASS**, the following table lists each argument, its default value and meaning.

Argument	Default Value	Meaning
TYPE	CONSISTENT	Type of mass matrix
MASS	CEM	Output dataset name
LDI	1	Logical device index

Tables 7.9-1 and 7.9-2 list the datasets used or created by procedure **MASS** and the processors invoked by procedure **MASS**, respectively.

Table 7.9-1 Datasets Input/Output by procedure MASS

Dataset	*Description*	*Input*	*Output*
JDF1.BTAB.1.8	Model Summary	√	
JSEQ.BTAB.2.17	Nodal Elimination Sequence	√	
KMAP..nsubs.ksize	Model Connectivity Map	√	
<MASS>.SPAR.*jdf2*	Assembled Mass Matrix		√
DEM.DIAG	Diagonal Mass Matrix		√

Table 7.9-2 Processors Invoked by procedure MASS

Procedure	*Type*	*Function*
VEC	Internal	Vector algebra utilities
K	Internal	Assembles unconstrained system matrices
ESu	External	Various element processors based on the GEP

7.9.3 ARGUMENT DESCRIPTIONS

7.9.3.1 LDI

Logical device index (default: 1).

7.9.3.2 MASS

Name of the mass matrix dataset (default: CEM). For a consistent mass matrix, the dataset name is [MASS].SPAR.*jdf2*. For a diagonal mass matrix, the dataset name will be <es_mass_diag> or DEM.DIAG.

7.9.3.3 TYPE

Type of mass matrix (default: CONSISTENT). This argument defines the form of the mass matrix to be either consistent or diagonal.

7.9.4 USAGE GUIDELINES AND EXAMPLES

Procedure MASS is used by preceding the procedure name by the *call directive. Procedure arguments may be changed from their default values by including any or all of the arguments and their new values when the procedure is called. A space or blank is required between the end of the procedure name and the left parenthesis.

```
*call MASS ( TYPE = DIAGONAL; MASS = <es_mass_diag>; ldi = 1)
```

7.9.5 LIMITATIONS

None.

7.9.6 ERROR MESSAGES AND WARNINGS

None.

7.9.7 PROCEDURE FLOWCHART

MASS (main procedure)
  ES (form element mass matrices)

7.9.8 PROCEDURE LISTING

```
*procedure MASS  ( type = CONSISTENT  ; -- .  CONSISTENT | DIAGONAL
                   ldi=1; mass = CEM )              . Output dataset
   *remark
   *Remark              Forming [type] Mass Matrix
   *remark
   *if < <IFELSE([TYPE];DIAGONAL;1;0)> > /then
.     ---------------------------------------
.     Initialize Diagonal (Lumped) Mass Matrix
.     ---------------------------------------
      *g2m /name=num_dofs  /type=i  [ldi] MODEL.SUMMARY NUM_DOFS
      *g2m /name=num_nodes /type=i  [ldi] MODEL.SUMMARY NUM_NODES
     [XQT VEC
      INIT_VEC <es_mass_ds>  <NUM_DOFS> BY <NUM_NODES>
   *endif
.  ---------------------------------------------
.  Form Element Mass Matrices (in EFIL Dataset)
```

```
.  ------------------------------------------------
   *call ES ( function = 'FORM MASS/[type]'; ldi=[ldi] )
.  ------------------------------------------------------
.  Assemble Element Consistent Mass Matrix into System Matrix
.  ------------------------------------------------------
   *if < <IFELSE([TYPE];CONSISTENT;1;0)> > /then
      [XQT K
       reset name = [mass]
       RESET BLIB=[ldi]
       RESET ELIB=[ldi]
       RESET HLIB=[ldi]
       RESET OUTLIB=[ldi]
       RESET SPDP=<csm_precision>
   *endif
*end
```

7.9.9 REFERENCES

7.9-1 Stewart, Caroline B.: *The Computational Structural Mechanics Testbed User's Manual.* NASA TM-100644, October 1989.

7.10 Procedure MODEL_SUMMARY

7.10.1 GENERAL DESCRIPTION

Procedure **MODEL_SUMMARY** generates a summary dataset which contains parameters about the finite element model. These parameters include the number of element processors used (<num_es_types>), the total number of nodes (<num_nodes>), the number of unconstrained degrees of freedom (<num_dofs>), and the number of equations (<num_eqns>). These parameters are written to the MODEL.SUMMARY dataset.

7.10.2 PROCEDURE USAGE

Procedure **MODEL_SUMMARY** is used by preceding the procedure name by the ***call** directive, and following it by a list of arguments enclosed in parentheses. Procedure arguments are order-independent, and most have default values thus making them optional. The formal syntax is as follows:

```
*call MODEL_SUMMARY ( arg1 = val1 ; arg2 = val2 ; ...)
```

where **arg1** and **arg2** represent argument names, and **val1** and **val2** represent their corresponding values. Note that semi-colons are required between arguments, and a double dash (--) may be used to continue the argument list on the next line.

The allowable arguments for procedure **MODEL_SUMMARY** are summarized in the following table, along with their default values (if they exist). Note that arguments without defaults are generally *mandatory*, while those with defaults are generally optional. Exceptions to this rule are noted in the following section under detailed argument descriptions.

For procedure **MODEL_SUMMARY**, the following table lists each argument, its default value and meaning.

Argument	Default Value	Meaning
CONSTRAINT_SET	1	Constraint set number
LDI	1	Logical device index

Table 7.10-1 lists the datasets used or created by procedure **MODEL_SUMMARY**.

Table 7.10-1 Datasets Input/Output by procedure MODEL_SUMMARY

Dataset	*Description*	*Lib*	*Input*	*Output*
JDF1.BTAB.1.8	Model Summary	√		
MODEL.SUMMARY	Model Summary Parameters		√	

7.10.3 ARGUMENT DESCRIPTIONS

7.10.3.1 CONSTRAINT_SET

Constraint set number (default: 1). This argument selects which constraint set to be used in solving the linear system of equations.

7.10.3.2 LDI

Logical device index (default: 1).

7.10.4 USAGE GUIDELINES AND EXAMPLES

Procedure **MODEL_SUMMARY** is used by preceding the procedure name by the ***call** directive. Procedure arguments may be changed from their default values by including any or all of the arguments and their new values when the procedure is called. A space or blank is required between the end of the procedure name and the left parenthesis.

```
*call MODEL_SUMMARY ( LDI = 1 ; CONSTRAINT_SET = 1 )
```

7.10.5 LIMITATIONS

None.

7.10.6 ERROR MESSAGES AND WARNINGS

None.

7.10.7 PROCEDURE FLOWCHART

Procedure **MODEL_SUMMARY** is self-contained with no subprocedures.

7.10.8 PROCEDURE LISTING

```
*procedure MODEL_SUMMARY ( ldi=1 ; constraint_set=1 )
.
. ----------------------------------------
. Get model parameters from various datasets
. ----------------------------------------
 *find record [ldi] ES.SUMMARY, ES_PROC /nor=num_es_types
 *g2m /name=parameters /type=i /maxn=18 [ldi] JDF1.BTAB.1.8 DATA.1
 *def/i num_nodes = <parameters[1]>
 *def/i num_dofs  = <parameters[2]>
.
. ------------------------------------------
. Install the MODEL.SUMMARY dataset, if necessary
. ------------------------------------------
 *find dataset [ldi] MODEL.SUMMARY /seq=ids_MS
 *if < <ids_MS> /le 0 > /then
     *put dataset [ldi] MODEL.SUMMARY /mrat=64 /seq=ids_MS
 *endif
.
. ------------------------------------------
. Store parameters in the MODEL.SUMMARY dataset
. ------------------------------------------
 *m2g /name=num_es_types /type=i [ldi] MODEL.SUMMARY NUM_ES_TYPES
 *m2g /name=num_nodes    /type=i [ldi] MODEL.SUMMARY NUM_NODES
 *m2g /name=num_dofs     /type=i [ldi] MODEL.SUMMARY NUM_DOFS
 *m2g /name=num_eqns     /type=i [ldi] MODEL.SUMMARY NUM_EQNS
*end
```

7.10.9 REFERENCES

7.10-1 Stewart, Caroline B.: *The Computational Structural Mechanics Testbed User's Manual.* NASA TM-100644, October 1989.

THIS PAGE LEFT BLANK INTENTIONALLY.

7.11 Procedure PRINT_EFIL

7.11.1 GENERAL DESCRIPTION

Procedure **PRINT_EFIL** prints one or all segments of the *es_name*.**EFIL**.*itype.nnod* dataset where *es_name* is the element name (*e.g.*, EX97). The procedure processes all elements of the specified element type.

7.11.2 PROCEDURE USAGE

Procedure **PRINT_EFIL** is used by preceding the procedure name by the ***call** directive, and following it by a list of arguments enclosed in parentheses. Procedure arguments are order-independent, and most have default values thus making them optional. The formal syntax is as follows:

```
*call PRINT_EFIL ( arg1 = val1 ; arg2 = val2 )
```

where **arg1** and **arg2** represent argument names, and **val1** and **val2** represent their corresponding values. Note that semi-colons are required between arguments, and a double dash (--) may be used to continue the argument list on the next line.

The allowable arguments for procedure **PRINT_EFIL** are summarized in the following table, along with their default values (if they exist). Note that arguments without defaults are generally *mandatory*, while those with defaults are generally optional. Exceptions to this rule are noted in the following section under detailed argument descriptions.

For procedure **PRINT_EFIL**, the following table lists each argument, its default value and meaning.

Argument	Default Value	Meaning
ELEMENT	**EX97**	Element name
SEGMENT	**ALL**	EFIL segment

The segments of the **EFIL** are printed using processor PRTE (see reference 7.11-1).

7.11.3 ARGUMENT DESCRIPTIONS

7.11.3.1 ELEMENT

Element name (default: **EX97**). The argument specifies the element name for which the **EFIL** segments are to be printed.

7.11.3.2 SEGMENT

EFIL segment to be printed (default: ALL). The EFIL (see reference 7.11-2) contains nine segments. Any one or all of the segments may be selected for printing using the argument SEGMENT. The EFIL segments are as follows:

Segment		Item	Length	Type	Description
1	Definition			Integ	Same as dataset DEF.<ES_NAME>.
2	Material			Real	(currently unused)
3	Geometry			Real	Element geometric parameters.
		XE0	(3,*nen*)		Initial element nodal coordinates in element basis.
		TEG	(3,3)		Transf. from global to current element basis.
		TEC	(3,3,*nen*)		Transf. from computational to element basis at element nodes.
		XG0	(3,*nen*)		Initial elt. nodal coords in global basis.
		TEG0	(3,3)		Transf. from global to initial element basis.
		DE	(*nee*)		Deformational displacements (d_e^{def}).
4	Property			Real	(currently unused)
5	Matrix	KM	*nmt*	Real*	Element matrix (stiffness/mass); only upper triangle of nodal blocks.
6	Aux. Storage			Real	Auxiliary storage for element developer.
7	Stress			Real	(currently unused)
8	Therm. Force			Real	(currently unused)
9	Therm. Stress			Real	(currently unused)

*The element stiffness/mass matrix, item KM in Segment 5, may be stored in either single or double precision, as specified in dataset DIR.*es_name.itype.nnod* (entry 15). However, all of the other REAL data in the *es_name*.**EFIL**.*itype.nnod* dataset are stored exclusively in single precision.

7.11.4 USAGE GUIDELINES AND EXAMPLES

Procedure PRINT_EFIL may be used by preceding the procedure name by the *call directive. Procedure arguments may be changed from their default values by including any

or all of the arguments and their new values when the procedure is called. A space or blank is required between the end of the procedure name and the left parenthesis. If the default values of the procedure arguments are to be used, then only the procedure name is required.

```
*call PRINT_EFIL ( element = EX97; segment = ALL )
```

7.11.5 LIMITATIONS

None.

7.11.6 ERROR MESSAGES AND WARNINGS

None.

7.11.7 PROCEDURE FLOWCHART

Procedure PRINT_EFIL is self-contained with no subprocessors.

7.11.8 PROCEDURE LISTING

```
*procedure PRINT_EFIL ( element = EX97; segment = ALL )
. -------------------------------
. Print Segment(s) of EFIL Dataset
. -------------------------------
*if < <ifelse([segment];ALL;1;0)> > /then
   *def/i seg1 = 1
   *def/i seg2 = 9
*else
   *if < <ifelse([segment];DEFINITION;1;0)> > /then
      *def/i seg1 = 1
   *elseif < <ifelse([segment];MATERIAL;1;0)> > /then
      *def/i seg1 = 2
   *elseif < <ifelse([segment];GEOMETRY;1;0)> > /then
      *def/i seg1 = 3
   *elseif < <ifelse([segment];PROPERTY;1;0)> > /then
      *def/i seg1 = 4
   *elseif < <ifelse([segment];STIFFNESS;1;0)> > /then
      *def/i seg1 = 5
   *elseif < <ifelse([segment];STRESS_RECOVERY;1;0)> > /then
      *def/i seg1 = 6
   *elseif < <ifelse([segment];STORE;1;0)> > /then
      *def/i seg1 = 6
   *elseif < <ifelse([segment];STRESS;1;0)> > /then
      *def/i seg1 = 7
   *elseif < <ifelse([segment];THERMAL_FORCE;1;0)> > /then
      *def/i seg1 = 8
```

```
   *elseif < <ifelse([segment];THERMAL_RECOVERY;1;0)> > /then
      *def/i seg1 = 9
   *endif
   *def/i seg2 = <seg1>
*endif
[XQT PRTE
 RESET SEG1 = <seg1>
 RESET SEG2 = <seg2>
 [element]
STOP
*end
```

7.11.9 REFERENCES

7.11-1 Stewart, Caroline B.: *The Computational Structural Mechanics Testbed User's Manual.* NASA TM-100644, October 1989.

7.11-2 Stewart, Caroline B.: *The Computational Structural Mechanics Testbed Data Library Description.* NASA TM-100645, October 1988.

7.12 Procedure RESEQUENCE

7.12.1 GENERAL DESCRIPTION

Procedure **RESEQUENCE** resequences the node numbers internally for a solver defined by the global macrosymbol **solver_name**. If **rseq_method** is -1, then a resequencing method implemented in processor RSEQ (see Section 6.1 of reference 7.12-1) is selected which is appropriate for the equation solver defined by the global macrosymbol **solver_name**. That is,

solver_name	rseq_method
INV	1
BAND	2
ITER	1
SPK	none

If **rseq_method** is 10, then the profile-front minimization resequencing method implemented in processor PFM (see Section 6.2 of reference 7.12-1). If the global macrosymbol **solver_name** is SPK, no resequencing is performed since processor SPK performs its own resequencing (see Section 8.6 of reference 7.12-1). Other values of **rseq_method** will select a specific resequencing method requested by a user.

7.12.2 PROCEDURE USAGE

Procedure **RESEQUENCE** is used by preceding the procedure name by the ***call** directive, and following it by a list of arguments enclosed in parentheses. Procedure arguments are order-independent, and most have default values thus making them optional. The formal syntax is as follows:

```
*call RESEQUENCE ( arg1 = val1 ; arg2 = val2 ; ...)
```

where **arg1** and **arg2** represent argument names, and **val1** and **val2** represent their corresponding values. Note that semi-colons are required between arguments, and a double dash (--) may be used to continue the argument list on the next line.

The allowable arguments for procedure **RESEQUENCE** are summarized in the following table, along with their default values (if they exist). Note that arguments without defaults are generally *mandatory*, while those with defaults are generally optional. Exceptions to this rule are noted in the following section under detailed argument descriptions.
For procedure **RESEQUENCE**, the following table lists each argument, its default value and meaning.

Argument	Default Value	Meaning
LDI	1	Logical device index
MAXCON	35	Maximum number of joints connected to any one joint
RSEQ_METHOD	-1	Resequencing method

Tables 7.12-1 and 7.12-2 list the datasets used or created by procedure **RESEQUENCE** and the processors invoked by procedure **RESEQUENCE**, respectively.

Table 7.12-1 Datasets Input/Output by procedure RESEQUENCE

Dataset	*Description*	*Input*	*Output*
JSEQ.BTAB.2.17	Nodal Elimination Sequence		√
KMAP..nsubs.ksize	Model Connectivity Map	√	

Table 7.12-2 Processors Invoked by procedure RESEQUENCE

Procedure	*Type*	*Function*
RSEQ	Internal	Joint resequencing
PFM	Internal	Joint resequencing using profile-front minimization

7.12.3 ARGUMENT DESCRIPTIONS

7.12.3.1 LDI

Logical device index (default: 1).

7.12.3.2 MAXCON

Maximum number of joints connected to any one joint (default: 35).

7.12.3.3 RSEQ_METHOD

Resequencing method (default: -1). If **rseq_method** is -1, then a resequencing method implemented in processor RSEQ (see Section 6.1 of reference 7.12-1) is selected which is appropriate for the equation solver defined by the global macrosymbol **solver_name**.

solver_name	**rseq_method**
INV	1
BAND	2
ITER	1
SPK	none

If **rseq_method** is 10, then the profile-front minimization resequencing method implemented in processor PFM (see Section 6.2 of reference 7.12-1). If the global macrosymbol

solver_name is SPK, no resequencing is performed since processor SPK performs its own resequencing (see Section 8.6 of reference 7.12-1). Other values of **rseq_method** will select a specific resequencing method requested by a user.

7.12.4 USAGE GUIDELINES AND EXAMPLES

Procedure **RESEQUENCE** is used by preceding the procedure name by the ***call** directive. Procedure arguments may be changed from their default values by including any or all of the arguments and their new values when the procedure is called. A space or blank is required between the end of the procedure name and the left parenthesis.

```
*call RESEQUENCE ( RSEQ_METHOD = 2 )
```

7.12.5 LIMITATIONS

None.

7.12.6 ERROR MESSAGES AND WARNINGS

None.

7.12.7 PROCEDURE FLOWCHART

Procedure **RESEQUENCE** is self-contained with no subprocedures.

7.12.8 PROCEDURE LISTING

```
*procedure RESEQUENCE ( rseq_method=-1 ; maxcon=35; ldi=1 )
.
*if < ifeqs( <solver_name>;SPK) > /then
  *remark ******************************************
  *remark * Processor SPK does its own resequencing *
  *remark ******************************************
  *else
    *if < [rseq_method] /eq 10 > /then
        [xqt PFM
             RESET BLIB=[ldi]
             reset maxcon = [maxcon]
    *elseif < [rseq_method] /ne -1 > /then
        [XQT RSEQ
             RESET BLIB=[ldi]
             reset method=[rseq_method], maxcon=[maxcon]
    *else
       *if < ifeqs( <solver_name>;INV) > /then
          *def/i rseq_method = 1
```

```
      *elseif < ifeqs( <solver_name>;BAND) > /then
         *def/i rseq_method = 2
      *else
         *def/i rseq_method = 1
      *endif
      [XQT RSEQ
           RESET BLIB=[ldi]
           reset method=<rseq_method>, maxcon=[maxcon]
   *endif
 *endif
.
*end
```

7.12.9 REFERENCES

7.12-1 Stewart, Caroline B.: *The Computational Structural Mechanics Testbed User's Manual.* NASA TM-100644, October 1989.

7.13 Procedure SOLVE

7.13.1 GENERAL DESCRIPTION

Procedure **SOLVE** solves a linear system of equations using a solver defined by the global macrosymbol **solver_name**. If **solver_name** is **INV**, the forward-reduction-back-substitution phase of the original nodal-block sparse solver implemented in processor SSOL will be used (see Section 8.3 of reference 7.13-1). If **solver_name** is defined to be **BAND**, a variable-bandwidth direct solver implemented in processor BAND will be used (see Section 8.5 of reference 7.13-1). If **solver_name** is defined to be **ITER**, an iterative method implemented in processor ITER will be used (see Section 8.4 of reference 7.13-1). If **solver_name** is defined to be **SPK**, the SPARSPAK-A direct solver will be used to factor and solve the system of equations (see Section 8.6 of reference 7.13-1).

7.13.2 PROCEDURE USAGE

Procedure **SOLVE** is used by preceding the procedure name by the ***call** directive, and following it by a list of arguments enclosed in parentheses. Procedure arguments are order-independent, and most have default values thus making them optional. The formal syntax is as follows:

```
*call SOLVE ( arg1 = val1 ; arg2 = val2 ; ...)
```

where **arg1** and **arg2** represent argument names, and **val1** and **val2** represent their corresponding values. Note that semi-colons are required between arguments, and a double dash (--) may be used to continue the argument list on the next line.

The allowable arguments for procedure **SOLVE** are summarized in the following table, along with their default values (if they exist). Note that arguments without defaults are generally *mandatory*, while those with defaults are generally optional. Exceptions to this rule are noted in the following section under detailed argument descriptions.

For procedure **SOLVE**, the following table lists each argument, its default value and meaning.

Argument	Default Value	Meaning
APPLIED_MOTION	1	Flag for applied displacements
CONSTRAINT_SET	1	Constraint set number
LDI	1	Logical device index
LOAD_SET	1	Load set number
MATRIX	K	First word of the name of the dataset containing the assembled stiffness matrix.
RHS	APPL.FORC	First two words of the dataset name for the right-hand side system vector.
SOLN	STAT.DISP	First two words of the dataset name for the displacement solution

Tables 7.13-1 and 7.13-2 list the datasets used or created by procedure **SOLVE** and the processors invoked by procedure **SOLVE**, respectively.

Table 7.13-1 Datasets Input/Output by procedure SOLVE

Dataset	*Description*	*Lib*	*Input*	*Output*
AMAP..ic2.isize	Factorization Map for INV	1	√	
APPL.FORC.*i*.1†	Applied force vector	1	√	
APPL.MOTI.*i*.1†	Specified displacement vector	1	√	
CON..*j*	Constraints	1	√	
EQNF.FORC.iset.1	Equivalent nodal forces	1	√	
INV.<KNAME>.*j*	Factored Stiffness Matrix	1	√	
JDF1.BTAB.1.8	Model Summary	1	√	
JSEQ.BTAB.2.17	Nodal Elimination Sequence	1	√	
KMAP..nsubs.ksize	Model Connectivity Map	1	√	
<KNAME>.SPAR.*jdf2*	Assembled Stiffness Matrix	1	√	
STAT.DISP.*i*.*j*†	System Displacement Vector	1		√
STAT.REAC.*i*.*j*†	System Reaction Force Vector	1		√

† i = <load_set> and j = <cons_set>

Table 7.13-2 Processors Invoked by procedure SOLVE		
Procedure	*Type*	*Function*
BAND	External	Solves system of equations using stiffness matrix in variable-bandwidth solver
ITER	External	Solves system of equations using iterative method
SPK	External	Solves system of equations using SPARSPAK-A solve
SSOL	Internal	Solves system of equations using nodal-block-sparse solver
VEC	Internal	Vector algebra utilities

7.13.3 ARGUMENT DESCRIPTIONS

7.13.3.1 APPLIED_MOTION

Flag indicating that applied displacements are involved in the loading (default: `0` or `FALSE`).

7.13.3.2 CONSTRAINT_SET

Constraint set number (default: `1`). This argument selects which constraint set to be used in solving the linear system of equations.

7.13.3.3 LDI

Logical device index (default: `1`).

7.13.3.4 LOAD_SET

Load set number (default: `1`). This argument selects which load set to be used as a right-hand side vector.

7.13.3.5 MATRIX

First word of the dataset name containing the assembled stiffness matrix (default: `K`).

7.13.3.6 RHS

First two words of the dataset name for the right-hand side system vector (default: `APPL.FORC`).

7.13.3.7 SOLN

First two words of the dataset name for the displacement solution (default: `STAT.DISP`).

7.13.4 USAGE GUIDELINES AND EXAMPLES

Procedure **SOLVE** is used by preceding the procedure name by the ***call** directive. Procedure arguments may be changed from their default values by including any or all of the arguments and their new values when the procedure is called. A space or blank is required between the end of the procedure name and the left parenthesis.

```
*call SOLVE ( MATRIX = K ; CONSTRAINT_SET = 1 ;
              LOAD_SET = 'APPL.FORC' ; SOLN = 'STAT.DISP' )
```

Before procedure **SOLVE** is called the global macrosymbol **solver_name** should be defined as described in Section 7.12.1 and should be consistent with the value used when procedure **FACTOR** was called. If it is not specified, then the default value **INV** will be used.

7.13.5 LIMITATIONS

None.

7.13.6 ERROR MESSAGES AND WARNINGS

None.

7.13.7 PROCEDURE FLOWCHART

Procedure **SOLVE** is self-contained with no subprocedures.

7.13.8 PROCEDURE LISTING

```
*procedure SOLVE ( rhs='appl.forc'; soln='stat.disp' ; matrix = K    ; --
                   load_set=1; constraint_set=1; applied_motion; --
                   ldi=1 )
.
. Procedure to solve a linear system of equations using different solvers.
. The solver is selected using the global macrosymbol "solver_name".
.
*if < ifeqs( <solver_name>; ) > /then
*def/a solver_name == INV
*endif
.
.  ---------------------------------------------------------
.  Define Unique Pseudo Load Set Number for SSOL Vector Datasets
.  ---------------------------------------------------------
  *def/i iunique = 9999
.  -----------------------------------------------
.  Copy Right-Hand-Side Vector to SSOL Force Dataset
.  -----------------------------------------------
   [xqt VEC
```

```
   APPL.FORC.<iunique>.1  <-  [rhs]  /single_precision
.  ------------------------------------------------
.  Copy Applied Motions to SSOL Applied Motion Dataset
.  ------------------------------------------------
  *find [ldi] APPL.MOTI.<iunique>.1 /seq=ids_AMU
  *if < <ids_AMU> /gt 0 > /then
     APPL.MOTI.<iunique>.1  <-  0.
  *else
     *g2m /name=num_nodes /type=i [ldi] MODEL.SUMMARY NUM_NODES
     *g2m /name=num_dofs  /type=i [ldi] MODEL.SUMMARY NUM_DOFS
     INIT_VEC APPL.MOTI.<iunique>.1  <num_dofs> BY <num_nodes>
  *endif
  *if < <IFELSE([applied_motion]; ;0;1)> >/then
     *find [ldi] [applied_motion] /seq=ids_AM
     *if < <ids_AM> /gt 0 > /then
        APPL.MOTI.<iunique>.1  <-  [applied_motion]
     *endif
  *endif
*if < ifeqs( <solver_name>;INV) > /then
.
. Use original nodal block sparse solver originally in SPAR
.
[XQT SSOL
     RESET K     = [matrix]
     RESET KLIB  = [ldi]
     RESET KILIB = [ldi]
     RESET QLIB  = [ldi]
     RESET SET   = <iunique>
     RESET CON   = [constraint_set]
*elseif < ifeqs( <solver_name>;BAND) > /then
.
. Use Banded solver
.
[XQT BAND
    reset meth=28, ldl=1
    reset ncon=[constraint_set], dzero=1.0e-20, k=[matrix]
    reset opt=2, res=1, iset=<iunique>
*elseif < ifeqs( <solver_name>;ITER) > /then
.
. Use Iterative solver
.
[XQT ITER
    reset ep=1,init=11,xnit=-.1,meth=1,time=1,itol=15
    reset ncon=[constraint_set], iset=<iunique>; k=[matrix]
    reset amax=0.5
*elseif < ifeqs( <solver_name>;SPK) > /then
.
. Use SPARSPAK solver
.
[XQT SPK
    select /const=[constraint_set] /kname=[matrix]
    select /msglva=2 /msglvl=2 /load=<iunique>
*endif
```

```
   [xqt VEC
.   ---------------------------------------------
.   Copy Solution Vector to Required Output Dataset
.   ---------------------------------------------
    *def/a std_name = STAT.DISP.[load_set].[constraint_set]
    *if < <IFELSE([soln];<std_name>;1;0)> > /then
       *rename [ldi] STAT.DISP.<iunique>.[constraint_set]  = --
                     STAT.DISP.[load_set].[constraint_set]
      *rename [ldi] STAT.REAC.<iunique>.[constraint_set]  = --
                    STAT.REAC.[load_set].[constraint_set]
    *else
       [soln]  <-  STAT.DISP.<iunique>.[constraint_set]
    *endif
*end
```

7.13.9 REFERENCES

7.13-1 Stewart, Caroline B.: *The Computational Structural Mechanics Testbed User's Manual.* NASA TM-100644, October 1989.

7.14 Procedure STIFFNESS

7.14.1 GENERAL DESCRIPTION

Procedure **STIFFNESS** forms the assembled unconstrained stiffness matrix.

7.14.2 PROCEDURE USAGE

Procedure **STIFFNESS** is used by preceding the procedure name by the ***call** directive, and following it by a list of arguments enclosed in parentheses. Procedure arguments are order-independent, and most have default values thus making them optional. The formal syntax is as follows:

```
*call STIFFNESS ( arg1 = val1 ; arg2 = val2 ; ...)
```

where **arg1** and **arg2** represent argument names, and **val1** and **val2** represent their corresponding values. Note that semi-colons are required between arguments, and a double dash (--) may be used to continue the argument list on the next line.

The allowable arguments for procedure **STIFFNESS** are summarized in the following table, along with their default values (if they exist). Note that arguments without defaults are generally *mandatory*, while those with defaults are generally optional. Exceptions to this rule are noted in the following section under detailed argument descriptions.

For procedure **STIFFNESS**, the following table lists each argument, its default value and meaning.

Argument	Default Value	Meaning
TYPE	**TANG**	Type of stiffness matrix to form (default: **TANG**). This argument defines the type of stiffness to form linear material stiffness matrix (**MATL**), geometric stiffness matrix (**GEOM**), or the tangent stiffness matrix (**TANG**).
CONSTRAINT_SET	1	Constraint set number (default: tt 1). This argument selects which constraint set to be used in solving the linear system of equations.
COROTATION	1	Corotation flag
NL_GEOM	0	Geometric nonlinearity flag
NL_LOAD	0	Load nonlinearity flag
NL_MATL	0	Material nonlinearity flag
INPUT_LDI	1	Logical device index with the element stiffness matrices (default: 1).
OUTPUT_LDI	1	Logical device index for the assembled unconstrained stiffness matrix (default: 1).
LOAD_SET	1	Load set number (default: 1). This argument selects which load set to be used as a right-hand-side vector.
LOAD_FACTOR	1.0	Load factor for the basic load system (default: 1.0).
STIFFNESS	**K**	First word of the dataset containing the assembled unconstrained stiffness matrix (default: **K**).
DISPLACEMENT		Full dataset name for the displacement vector.
ROTATION		Full dataset name for the rotation vector.

Tables 7.14-1 and 7.14-2 list the datasets used or created by procedure **STIFFNESS** and the processors invoked by procedure **STIFFNESS**, respectively.

Table 7.14-1 Datasets Input/Output by procedure STIFFNESS

Dataset	*Description*	*Input*	*Output*
JDF1.BTAB.1.8	Model Summary	√	
JSEQ.BTAB.2.17	Nodal Elimination Sequence	√	
KMAP..nsubs.ksize	Model Connectivity Map	√	
<STIFFNESS>.SPAR.*jdf2*	Assembled Stiffness Matrix		√

Table 7.14-2 Processors Invoked by procedure STIFFNESS		
Procedure	*Type*	*Function*
VEC	Internal	Vector algebra utilities
ESi	External	Various element processors based on the GEP
K	Internal	Assembles unconstrained stiffness matrix

7.14.3 ARGUMENT DESCRIPTIONS

7.14.3.1 TYPE

Type of stiffness matrix to form (default: **TANG**). This argument defines the type of stiffness to form linear material stiffness matrix (**MATL**), geometric stiffness matrix (**GEOM**), or the tangent stiffness matrix (**TANG**).

7.14.3.2 CONSTRAINT_SET

Constraint set number (default: 1). This argument selects which constraint set to be used in solving the linear system of equations.

7.14.3.3 INPUT_LDI

Logical device index with the element stiffness matrices (default: 1).

7.14.3.4 OUTPUT_LDI

Logical device index for the assembled unconstrained stiffness matrix (default: 1).

7.14.3.5 LOAD_SET

Load set number (default: 1). This argument selects which load set to be used as a right-hand side vector.

7.14.3.6 LOAD_FACTOR

Load factor for the basic load system (default: 1.0).

7.14.3.7 STIFFNESS

First word of the dataset name containing the assembled unconstrained stiffness matrix (default: **K**).

7.14.3.8 DISPLACEMENT

Full dataset name for the displacement vector.

7.14.3.9 ROTATION

Full dataset name for the rotation vector.

7.14.3.10 COROTATION

Corotation switch; employed by ES*i* processors for automatic treatment of geometric nonlinearity due to large rotations (default: 1). Relevant only if problem is geometrically nonlinear. (See Chapter 4 of reference 7.2-1 for an explanation of these options.) If COROTATION equals zero, then the corotational operations will be skipped. If COROTATION equals one (low-order option), then the basic corotational transformations will be employed to enable large rotations. If COROTATION equals two (high-order option), then a more accurate (and expensive) treatment of large rotations and consistent linearization than option 1 will be employed.

7.14.3.11 ES_NL_GEOM

Geometric nonlinearity switch (default: 0). If ES_NL_GEOM equals zero (no corotational frames), then the problem is geometrically linear (small displacements/rotations). If NL_GEOM equals one (low-order option), then the problem is geometrically nonlinear, but elements should use linear strain-displacement relations. Meaningful only if COROTATION > 0, so that large rotations can be handled automatically by the corotational algorithm. If NL_GEOM equals two (high-order option), then the problem is geometrically nonlinear and elements should use *nonlinear* element strain-displacement relations. May be used in conjunction with COROTATION > 0 to obtain higher-order accuracy for beam and shell elements that employ moderate-rotation strain-displacement relations.

7.14.3.12 NL_MATL

Material nonlinearity switch (default: 0). If NL_MATL is greater than zero, then the element is materially nonlinear. If L_MATL is zero, then the element is materially linear.

7.14.3.13 NL_LOAD

Load nonlinearity switch (default: 0). If NL_LOAD is greater than zero, then the element loads are displacement dependent. If NL_LOAD equals zero, then the element loads are not displacement dependent.

7.14.4 USAGE GUIDELINES AND EXAMPLES

Procedure **STIFFNESS** is used by preceding the procedure name by the ***call** directive. Procedure arguments may be changed from their default values by including any or all of the arguments and their new values when the procedure is called. A space or blank is required between the end of the procedure name and the left parenthesis.

```
*call STIFFNESS ( TYPE = MATL ; CONSTRAINT_SET = 1 ; STIFFNESS = K )
```

7.14.5 LIMITATIONS

None.

7.14.6 ERROR MESSAGES AND WARNINGS

None.

7.14.7 PROCEDURE FLOWCHART

STIFFNESS (main procedure)
  ES (form element stiffness matrices)

7.14.8 PROCEDURE LISTING

```
*procedure STIFFNESS ( type     = TANG       ; -- .  MATL | GEOM | TANG
                       nl_geom  = 0          ; --
                       nl_matl  = 0          ; --
                       nl_load  = 0          ; --
                       displacement          ; -- . Input dataset
                       rotation              ; -- . Input dataset
                       corotation = 1        ; --
                       load_factor = 1.0     ; --
                       load_set    = 1       ; --
                       stiffness   = K       ; -- . Output dataset
                       input_ldi   = 1       ; --
                       output_ldi  = 1       ; --
                       constraint_set = 1       )
   *remark
   *Remark             Forming [type] Stiffness Matrix
   *remark
.  ------------------------------------------------
.  Form Element Stiffness Matrices (in EFIL Dataset)
.  ------------------------------------------------
   *call ES ( function       = 'FORM STIFFNESS/[type]' ; --
              ldi            = [input_ldi]             ; --
              es_nl_geom     = [nl_geom]               ; --
```

```
            es_nl_matl     = [nl_matl]              ; --
            es_nl_load     = [nl_load]              ; --
            es_dis_ds      = [displacement]         ; --
            es_rot_ds      = [rotation]             ; --
            es_coro        = [corotation]           ; --
            es_load_factor = [load_factor]          ; --
            es_load_set    = [load_set]  )
.  ------------------------------------------------
.  Assemble Element Stiffness Matrix into System Matrix
.  ------------------------------------------------
   [XQT K
    RESET NAME = [stiffness]
    RESET BLIB = [input_ldi]
    RESET ELIB = [input_ldi]
    RESET HLIB = [input_ldi]
    RESET OUTLIB = [output_ldi]
    RESET SPDP = <csm_precision>
*end
```

7.14.9 REFERENCES

7.14-1 Stewart, Caroline B.: *The Computational Structural Mechanics Testbed User's Manual.* NASA TM-100644, October 1989.

7.15 Procedure SWITCH_DS

7.15.1 GENERAL DESCRIPTION

Procedure SWITCH_DS switches two datasets within the same data library. This procedure only uses directives from the command language (see reference 7.7-1).

7.15.2 PROCEDURE USAGE

Procedure SWITCH_DS is used by preceding the procedure name by the *call directive, and following it by a list of arguments enclosed in parentheses. Procedure arguments are order-independent, and most have default values thus making them optional. The formal syntax is as follows:

```
*call SWITCH_DS ( arg1 = val1 ; arg2 = val2 ; ...)
```

where arg1 and arg2 represent argument names, and val1 and val2 represent their corresponding values. Note that semi-colons are required between arguments, and a double dash (--) may be used to continue the argument list on the next line.

The allowable arguments for procedure SWITCH_DS are summarized in the following table, along with their default values (if they exist). Note that arguments without defaults are generally *mandatory*, while those with defaults are generally optional. Exceptions to this rule are noted in the following section under detailed argument descriptions.

For procedure SWITCH_DS, the following table lists each argument, its default value and meaning.

Argument	Default Value	Meaning
DS_1	--	Source dataset name
DS_2	--	Target dataset name
LDI	1	Logical device index

7.15.3 ARGUMENT DESCRIPTIONS

7.15.3.1 LDI

Source logical device index (default: 1). Data library number containing the two datasets DS_1 and DS_2.

7.15.3.2 DS_1

Source dataset to be copied from within the data library with a logical device index of LDI to the target dataset named DS_2.

7.15.3.3 DS_2

Target dataset for copying the source dataset named DS_1 from within the data library with a logical device index of LDI.

7.15.4 USAGE GUIDELINES AND EXAMPLES

Procedure **SWITCH_DS** is used by preceding the procedure name by the ***call** directive. Procedure arguments may be changed from their default values by including any or all of the arguments and their new values when the procedure is called. A space or blank is required between the end of the procedure name and the left parenthesis. If the default values of the procedure arguments are to be used, then only the procedure name is required.

```
*call SWITCH_DS ( ldi=1; ds_1; ds_2 )
```

7.15.5 LIMITATIONS

None.

7.15.6 ERROR MESSAGES AND WARNINGS

None.

7.15.7 PROCEDURE FLOWCHART

Procedure **SWITCH_DS** is self contained with no subprocedures.

7.15.8 PROCEDURE LISTING

```
*procedure SWITCH_DS ( ldi=1; ds_1; ds_2 )
. CLAMP Procedure to switch two datasets within a data library
   *def/a  tmp_ds = xxxx.xxxx.9999.9999.9999
   *rename [ldi], [ds_1]   = <tmp_ds>
   *rename [ldi], [ds_2]   = [ds_1]
   *rename [ldi], <tmp_ds> = [ds_2]
*end
```

7.15.9 REFERENCES

7.7-1 Felippa, Carlos A.: *The Computational Structural Mechanics Testbed Architecture: Volume II - Directives.* NASA CR 178385, February 1989.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	December 1991	Technical Memorandum

4. TITLE AND SUBTITLE
The Computational Structural Mechanics Testbed Procedures Manual

5. FUNDING NUMBERS
505-63-53-01

6. AUTHOR(S)
Caroline B. Stewart, Compiler

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
NASA Langley Research Center
Hampton, VA 23665-5225

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)
National Aeronautics and Space Administration
Washington, DC 20546-0001

10. SPONSORING / MONITORING AGENCY REPORT NUMBER
NASA TM-100646

11. SUPPLEMENTARY NOTES
Caroline B. Stewart was working for Analytical Services and Materials Inc., Hampton, VA.; presently with Continuous Electronic Beam Accelerator Facility (CEBAF), Newport News, VA.

12a. DISTRIBUTION / AVAILABILITY STATEMENT
Unclassified - Unlimited

Subject Category 39

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

The purpose of this manual is to document the standard high-level command language procedures of the Computational Structural Mechanics (CSM) Testbed software system. A description of each procedure including its function,commands, data interface, and use is presented. This manual is designed to assist users in defining and using command procedures to perform structural analyses in the CSM Testbed User's Manual (NASA TM-100644) and The CSM Testbed Data Library Description (NASA TM-100645)

14. SUBJECT TERMS
Computational Structural Mechanics
Structural Analysis

15. NUMBER OF PAGES
597

16. PRICE CODE
A25

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	