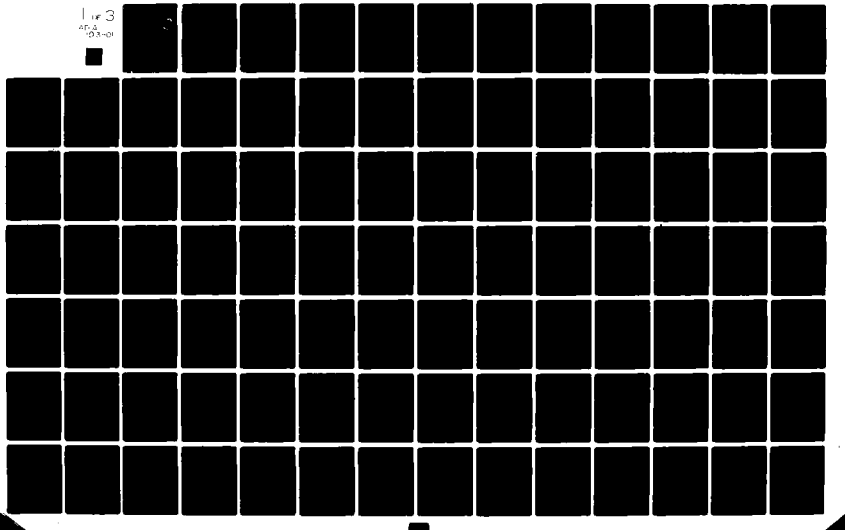
AD-A103 801 LOCKHEED MISSILES AND SPACE CO INC PALO ALTO CA PALO --ETC F/G 13/13
INTERACTIVE NONLINEAR STRUCTURAL ANALYSIS: ENHANCEMENT.(U)
JUL 81 6 M STANLEY N00014-80-C-0831
UNCLASSIFIED LMSC-D811535 NL



INTERACTIVE NONLINEAR STRUCTURAL ANALYSIS: ENHANCEMENT

Final Report,
Contract N00014-80-C-0831

LMSC-D811535 31 July 1981

Prepared by:

G. M. Stanley
Applied Mechanics Laboratory
Department 52-33, Building 255
LOCKHEED PALO ALTO RESEARCH LABORATORY
3251 Hanover Street
Palo Alto, California 94304

G I S T

GRAPHICS-INTERACTIVE STRUCTURAL ANALYSIS

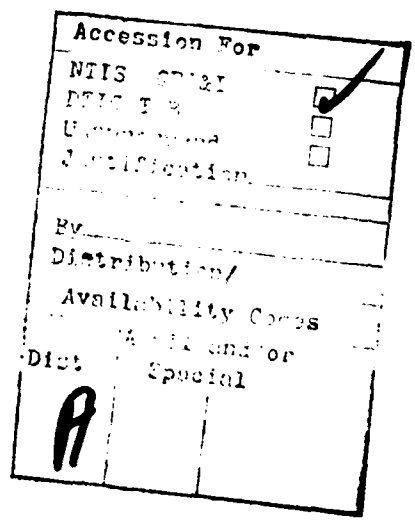
VIA THE GIFTS/STAGS SOFTWARE ASSEMBLY

[TUTORIAL]

G. M. Stanley
Lockheed Palo Alto

H. A. Kamel
University of Arizona

MAY 1981



C O N T E N T S

T H E G I S T T U T O R I A L

PART 0:

OVERVIEW; THE GIST SYSTEM

WHAT IS THE GIST OF IT?

GIST is a system for performing NONLINEAR FINITE-ELEMENT STRUCTURAL ANALYSIS in an INTERACTIVE, GRAPHICS-ORIENTED computational environment. The system constitutes a network of integrated software processors, i.e., independently executable computer programs, collectively referred to as the GIFTS/STAGS Software Assembly, or more concisely as GIST.

This network is analogous to a computer operating system, which enables the user to perform a wide variety of operations accessible through a common "command" (or "control") language. Many of these operations are actually performed by independent processors which are linked through a "global database" to behave as an integrated system.

Thus, GIST may be regarded as a special-purpose (or "virtual") operating system for structural analysis, where instead of text editors, compilers and file handlers, GIST employs model editors, structural analyzers and solution post-processors as basic components. Furthermore, the user addresses the GIST "operating system" in a PROBLEM-ORIENTED COMMAND LANGUAGE, and is free (by virtue of the global database) to exercise the system in a "reasonably" arbritrary fashion, with frequent stops and restarts.

WHAT ARE THE BASIC COMPONENTS OF GIST?

To clarify the above notions, we now introduce the user to the principal components of the GIFTS/STAGS Software Assembly:

GIFTS (Graphics-Interactive Finite element Timesharing System) is a collection of software processors primarily designed for convenient definition and evaluation of general (three-dimensional) finite element structural models [G1]. Featuring extensive graphics capabilities driven by both command language and optically digitized input, and a global database utilized for archival by all of its processors, GIFTS provides both a user- and developer- oriented approach to pre- and post- processing. Other members of the GIFTS processor family perform linear analysis functions and include bandwidth optimization and reduced substructuring. However, of particular importance here will be the pre- and post- processing functions which, as a virtue of the GIFTS architecture, are interfacable to a variety of more general analyzers [G1].

STAGS (STructural Analysis of General Shells) is a finite element structural analyzer primarily intended for "moderately" nonlinear analysis of generally configured shell structures [S1]. Its capabilities include static, dynamic and eigenvalue (buckling/vibration) analysis and permit both geometric and material nonlinearities. Solution strategies are partially automated and partially user-controlled with an emphasis towards computational efficiency. Architecturally, STAGS consists of two general-purpose processors: a "pre-analyzer" which defines the mathematical model, and an "analyzer" which performs the actual solution. Like GIFTS,

STAGS also relies on a global database for incremental operation (e.g., restarts) and analysis archival, but employs an independent data-management system (DAL). Unlike GIFTS, STAGS is not an inherently interactive system, but when accessed within the GIST framework this distinction disappears.

To enable the coordinated utilization of GIFTS pre- and post-processing and STAGS analysis capabilities within the same application, three additional "architectural" components have been introduced. This triad of special processors, which is derived from the NICE [N1] utility library, performs most of the work which is NOT specifically related to structural analysis -- permitting the user to concentrate on the problem at hand.

The G2S (GIFTS->STAGS) and S2G (STAGS->GIFTS) "adaptor" modules establish the necessary link between the independent GIFTS and STAGS global databases. They are thus introduced at the pre-processing -> analysis, and analysis -> post-processng interfaces, respectively. In addition to moving data around, the adaptors are necessary to ensure data compatibility and should alert the user regarding "unadaptable" situations (e.g., a GIFTS option which is not recognized by STAGS).

The GIST Control Module completes the triad and is the most visible of the architectural components. It is this processor which intermediates between the user and the computer operating system to produce an analysis oriented setting; through the GIST Command Language, the user interacts with the Control Module and thereby invokes the capabilities of GIFTS/STAGS. For example, by issuing the 'EDITM' command from the GIST Control Module, control is transferred to the GIFTS model editor, and automatically returned upon completion of the model editing session. Similarly, various GIST analysis commands allow the user to interactively prepare solution strategy and engage the STAGS analyzer in either on- or off- line computation. Additional functions of the Control Module include monitoring problem status (through the global database), interjecting "adaptor" modules as necessary, and providing extended analysis REVIEW, HELP and data-management capabilities via command.

WHAT ARE THE REQUIREMENTS FOR UTILIZING GIST?

The prospective GIST user will be relieved to know that the system can be applied with relatively little effort. However, this will depend strongly on the user's previous experience with computational structural analysis in general, and with GIFTS or STAGS in particular. Starting with a fundamental background in finite-element structural analysis, the initial task reduces to: (1) learning some basic GIST Commands, (2) developing experience with the interactive GIFTS pre- and post-processors, and (3) understanding the key assumptions and limitations engendered by the STAGS analyzer. An effective approach is to perform a casual reading of the User's Manuals (GIST, GIFTS and STAGS) followed by an immediate application of the system to a simple, "test-case"

problem. Due to the availability of on-line documentation (via HELP), the attempt should prove educational, if not successful.

For those who find the on-line documentation insufficient, there is a substantial base of reference material available to complement, refresh and expand the user's knowledge of the system. Nevertheless, the user is reminded that "nonlinear" structural analysis, at the present time, is not a uniformly predictable process. Thus computational strategies will often require a combination of preliminary studies, numerical experimentation and physical insight to obtain meaningful solutions. It is hoped that the interactive features of the GIST System will both facilitate and enrich this process.

WHAT IS THE SCOPE OF THE GIST TUTORIAL?

The GIST Tutorial is intended as both a primer to the GIFT/STAGS Software Assembly, and as a reference manual for the GIST Command Language. Consequently, the user should find enough here to get started with appropriate references made to more specific documentation. This is not unlike the documentation structure of some computer operating systems. It allows the user to branch-out according to need and not be over- whelmed with all of the details at once.

The tutorial is arranged in layers, starting with the user-interface, progressing to the logical application of the system, and finally to the physical organization, or architecture, of the software network.

Part 1 introduces the GIST Command Language, which, as mentioned above, provides the most convenient means for accessing processors and coordinating operations. This part covers everything from "logging-on" to a comprehensive description of every command in the GIST language.

In Part 2, the application and coordination of the Command Language to perform general types of structural analysis is first outlined, and then demonstrated by comprehensive interactive examples. The correspondence of certain pre/post-processing (i.e., GIFTS) terminology to certain analysis-phase (i.e., STAGS) terminology is also noted in this part. Finally, a section on data management is included, which indicates how the user may play an active role in maintaining the "database" as the need arises.

Part 3 is for software enthusiasts, prospective co-developers and those interested in more than a superficial understanding of the system. It deals with software architecture, i.e., how the system is put together and how it may be extended. Included here are full source code listings of the "architectural components", i.e., the GIFTS-STAGS Adaptor Modules and the GIST Control Module.

Several Appendices have been included to cover material which is expected to change rapidly in the near future, but which is nevertheless important for present utilization: Appendix A summarizes various GIFTS-STAGS interface conventions which should eventually disappear (or become transparent) as the system evolves. Appendix B itemizes the relatively new STAGS Global Database, which is also scheduled for restructuring. Finally, Appendix C is devoted to "computer implementation", and is thus aimed at those individuals who are blessed with the responsibility of installing and/or maintaining GIST on a particular operating system. General guidelines are presented and followed by specific instructions for dealing with two important systems: VAX/VMS and CDC/NOS.

The following is a summary of recommended supplementary reading:

(1) The GIFTS User's Manual
(2) The GIFTS Primer
(3) The GIFTS System Manual
(4) The GIFTS Pocket Guide
(5) The STAGS User's Manual (Section 6 ...)
(6) The STAGS Theory Manual
(7) The STAGS Example Manual (due to be released)

T H E G I S T T U T O R I A L

PART 1:

ACCESS; THE GIST COMMAND LANGUAGE

1.0 INTRODUCTION

The user, i.e., the structural analyst, accesses the GIST System via the GIST Command Language. This is the language used to communicate with the GIST Control Module, an interactive vantage point from which all of the basic system components -- pre-processors, analyzers, post-processors, etc. -- may be invoked. By issuing GIST commands, the analyst may systematically direct and monitor the course of an analysis, performing many tasks on-line, and dispatching others, e.g., "number-crunchers" for off-line (batch) processing. Guidance from the Control Module is provided only in case of logical errors or if explicitly requested, e.g., through the HELP or REVIEW commands.

In addition to the GIST Command Language, which may be viewed as the "global" command language, certain processors participating in the GIST network may have a language of their own, i.e, a "local" command language. This, however, is presently confined to the interactive GIFTS pre- and post-processors, which are driven in the GIFTS command language. When the user exits from one of these processors, control is automatically returned to the Control Module and communciation resumes in the GIST language.

The "computational" tasks of analysis, which are performed by the STAGS processors, are interactively prepared, invoked and monitored from the GIST Control Module, and are therefore entirely within the domain of the GIST Command Language.

In Part 1, we present the basics of this problem-oriented Command Language. (The reader is referred to the GIFTS User's Manual [G1] for an equivalent treatment of the special command language required for pre- and post-processing.) The part of the tutorial opens with instructions for getting on and off of the system, general rules for entering commands and some examples illustrating the HELP command (through which much of the present manual may be abstracted on-line). This is followed by a brief summary of available GIST commands and their basic functions. Part 1 concludes by expanding the brief summary into a comprehensive set of command descriptions arranged in alphabetical order.

The basic information presented in Part 1 will be collected and integrated in Part 2, where we concentrate on the application of the Command Language to perform interactive nonlinear structural analysis.

1.1 "LOGGING-ON/OFF" AND THE 'JOBNAME'

Suppose GIST has been implemented on your favorite computing system and you are wondering where to begin. First, look around for an experienced user; if none is available then continue reading. After performing your customary log-on procedure, i.e., identifying yourself to the operating system, you will need to perform a similar procedure to begin accessing the GIST network. The opening line will depend on the actual computer operating system and the GIST implementation (implementors: see Appendix C; users: see your implementor). For example, on a VAX/VMS computer, one might enter the following operating system command:

```
$ @GIST
```

where $ is the VAX command prompt; while on a CDC/NOS computer, an equivalent control statement is:

```
/ -GIST
```

In any case, the net effect is to summon the GIST Control Module which should respond with the following "splash":

```
================================================================

<<>>  G I S T   <GRAPHICS-INTERACTIVE STRUCTURAL ANALYSIS>

================================================================
              "Welcome to the GIFTS/STAGS Network"

Jobname:
```

To complete the "log-on" procedure, simply enter the 'Jobname'. The GIST 'Jobname' is analogous to the operating system's 'Username' except that it identifies a particular problem (or "Job") rather than an individual. Also, it is more arbitrary; any alphanumeric string not exceeding 8 characters (4 on CDC/NOS) is acceptable.

The important thing to remember is that the Jobname enables both you, the analyst, and the Control Module to keep track of the problem from session to session. Thus, by selecting unique Jobnames, you may process multiple GIST jobs concurrently. (See Section 2.4 for the relationship of the Jobname to the GIST database, i.e., file names.)

Now suppose you have selected a Jobname, e.g.:

```
Jobname: DRYRUN

<> New Job

COMMAND | HELP | QUIT
G I S T >
```

In the above dialogue, the Control Module has responded to the user's entry, DRYRUN, with the message 'New Job' and is prompting for a GIST command.

Congratulations, you are now "logged-on".

The standard 2-line prompt:

```
COMMAND | HELP | QUIT
G I S T >
```

will appear frequently. It conveys a double message: First, it indicates that you are talking to the GIST Control Module; and second, it reminds you of perhaps the two most important commands in the language: HELP and QUIT. When in doubt, type HELP (see Section 1.3); when exhausted, exasperated or to excuse yourself, type QUIT.

The QUIT command will "log-you-off" of the GIST network and return you to the computer operating system. [This is to be distinguished from the GIFTS pre- and post-processing QUIT command which is used to exit from any of the GIFTS modules and return to the Control Module.]

To continue the Job at a later time, just repeat the above "log-on" procedure remembering to use the same Jobname. By default, everything needed will automatically be saved in the database and restored at the proper time.

Finally, to permanently end a given Job, and thus clear all disk space occupied by its database, you may wish to employ the CLEAR command (discussed in Section 2.4).

1.2 RULES FOR ENTERING COMMANDS

All GIST commands share the following general format:

```
+-------------------------------------------------------------------+
|                                                                   |
|     Command-name [/Qualifiers...]  Keyword-phrases  [...]         |
|                                                                   |
+-------------------------------------------------------------------+
```

where the 'command name' and optional 'qualifiers' are typically character strings and a 'keyword phrase' is a simple expresion of the form:

Keyword = data

where 'keyword' represents a character string and 'data' may be just about anything depending on the context of the expression. The keyword phrases define mandatory parameters and/or options associated with the particular combination of 'command name' and 'qualifiers'.

For example:

STATIC/NONLINEAR INTERVAL=1.,2. STEPSIZE=.1 NEWTON=TRUE

is a valid command expression. Comparing this with the general format, we can identify the basic command components as:

```
Command name  .......  STATIC
Qualifier  ..........  NONLINEAR
Keyword phrase 2  ...  INTERVAL=1.,2.
Keyword phrase 2  ...  STEPSIZE=.1
Keyword phrase 3  ...  NEWTON=TRUE
```

Some commands require much less information. For example:

REVIEW STRATEGY

is also a valid command expression. In this case, the command name is REVIEW, no qualifiers have been selected and the single, data-less keyword, STRATEGY, specifies the command option.

Thus, each command has its own particular variation, or subset, of the general command format. Actual command requirements are presented in later sections. The purpose of this section is to

present some general rules and a compact notation to facilitate the "interpretation" of command requirements.

The following is a list of rules for entering GIST commands from either an interactive terminal or a peripheral text file:

GRAMMAR RULES

(R1) All command names and keywords may be abbreviated by truncating to a unique shortened form.

For example, the following two statements are equivalent:

```
REVIEW SOLUTION
REV SOL
```

(R2) A command expression may always be continued on a subsequent line by appending a double dash (--) to the current line, e.g.:

For example:

```
G I S T > STATIC/NONLINEAR INTERVAL=1.,2. STEP --
G I S T > = .1 NEWTON=TRUE
```

(R3) Commands which require at least one mandatory keyword will automatically prompt for keywords if none are detected.

For example:

```
COMMAND | HELP | QUIT
G I S T > STATIC/NONLINEAR

Keywords: INTERVAL STEPSIZE NEWTON [...]
G I S T STATIC > INTERVAL=1.,2. STEPSIZE=.1

Keywords: INTERVAL STEPSIZE NEWTON [...]
G I S T STATIC >
```

and so on.

A null response, e.g., pressing <return>, pops the user out of the keyword prompt mode and back to the command level prompt:

```
COMMAND | HELP | QUIT
G I S T >
```

(R4) Data is always entered in "keyword phrases", i.e., expressions of the form: 'keyword = data'. The order of keyword phrases in a given command expression is arbitrary.

For example, the following two expressions are equivalent:

```
DYNAMIC  INTERVAL=0.,1.  METHOD=TRAPEZOID
DYNAMIC  METHOD=TRAPEZOID  INTERVAL=0.,1.
```

(R5) Equal signs (=) between keywords and their data are optional; a space may be used instead.

For example, the following two expressions are equivalent:

```
RESULT SOLUTION=1,4
RESULT SOLUTION 1,4
```

(R6) Commas (,) must be used to separate the individual items in a numeric data list.

For example, the comma in the preceeding example (R5) is required.

(R7) The slash (/) used to separate a command name and its qualifier(s) is mandatory. Conversely, if no qualifers are used then no slashes should appear.

For example, the following expressions are equivalent:

```
STATIC  INTERVAL=100.,101.
STATIC/NONLIN  INTERVAL=100.,101.
```

since /NONLINEAR is the default qualifier for the STATIC command.

(R8) Spaces must be used to separate any command components which do not have an alternate separator (such as / or =). Additional spaces may be used freely.

(R9) Multiple command expressions may be entered on a single line by separating with a semi-colon (;).

For example, the following expression represents 3 commands:

```
CLEAR ANALYSIS;  REVIEW JOB;  HELP ANALYSIS
```

(R10) Comments may be appended to command expressions by inserting an isolated period (.) between the command and the comment. Everything to the right of the period will be ignored by the command interpreter.

For example:

```
DYNAMIC INTERVAL=0.,1.          . Compute transient response
DYNAMIC STEPSIZE=.001           . Dt governed by load history
COMPUTE/BATCH  QUEUE=HUGE       . This may take some time ...
```

represents a sequence of commented GIST commands.

1.3 THE 'HELP' COMMAND

One of the best ways to learn, or freshen-up on, the GIST Command Language is to use the HELP command, frequently. The basic form of the command is:

G I S T > **HELP**

which provides the following general information:

```
Help on a particular GIST command or topic may be obtained via:

                    HELP  Name  Subname

where 'Name' is either a valid command or topic and 'Subname' is
typically an associated command qualifier, keyword or subtopic.

Available commands:

BUCKLING     BULKLB     BULKM     BULKS     CLEAR      COMPUTE
DYNAMIC      EDITLB     EDITM     HELP      JOBNAME    MANAGE
QUIT         RESULT     REVIEW    SETUP     STATIC     VIBRATION

Available topics:
                          OVERVIEW

LANGUAGE   PREPROCESSING   ANALYSIS   POSTPROCESSING   EXAMPLES

                          DATABASE
```

As you can see, the entire command menu is displayed, plus a set of key topics. Now, by using the more specific form of the HELP command:

G I S T > **HELP** Name

you may obtain information on any of the individual commands or topics displayed above.

For example:

G I S T > **HELP STATIC**

produces a brief (screen-size) description of the STATIC command and lists command qualifiers and keywords under which additional information is available.

Similarly:

G I S T > HELP EXAMPLE

displays the introduction to a GIST example problem and lists subtopics under which the details of the analysis, i.e., the command input stream, may be presented.

To obtain information on a command qualifier, command keyword, or subtopic, use the most explicit form of the HELP command:

G I S T > HELP Name Subname

For example:

G I S T > HELP STATIC/NONLINEAR

describes the /NONLINEAR form of the STATIC command, while:

G I S T > HELP STATIC INTERVAL

explains the significance of the keyword phrase, INTERVAL=pamin,pamax.

Similarly:

G I S T > HELP EXAMPLE PART1

continues the presentation of the GIST example introduced under HELP EXAMPLE.

The "nested" structure of the HELP command facilitates a concise user/system dialogue, which becomes more and more desirable as the user gains experience. Note, however, that by systematically requesting HELP on each of the existing commands, topics, etc., a condensed version of the GIST Manual is available on-line.

Remark

The GIFTS pre- and post-processors also feature a HELP command, but it has a slightly different format than the GIST command described above. To obtain help from "within" a GIFTS processor, use the form:

HELP /Topic/

where 'Topic' may be the name of any GIFTS processor or GIFTS command.

1.4 COMMAND SUMMARY

The following table includes every command in the GIST language. Next to the brief functional description is a 3-character label that indicates with which phase of structural analysis the command is most closely associated. A complete description of each command may be found in Section 1.5. For guidance and examples on the integrated application of these commands to structural analysis consult Part 2 of this tutorial.

Command	Function	Phase
BUCKLING	Prepares strategy for a STAGS buckling analysis.	ANA
BULKLB	Invokes the GIFTS bulk load and b.c. generator.	PRE
BULKM	Invokes the GIFTS bulk model generator.	PRE
CLEAR	Erases all or part of the current Job Database.	AUX
COMPUTE	Invokes the STAGS "structural analyzer".	ANA
DYNAMIC	Prepares strategy for a STAGS dynamic analysis.	ANA
EDITLB	Invokes the GIFTS load and b.c. display editor.	PRE
EDITM	Invokes the GIFTS model display editor.	PRE
HELP	Displays information on GIST commands/topics.	AUX
INPUT	GIST "on-line batch" command switch	AUX
MANAGE	Invokes one of the system database "editors"	AUX
QUIT	Terminates the current GIST session.	AUX
RESULT	Displays all solution quantities, spatially.	POS
REVIEW	Monitors all aspects of the current case.	AUX
SETUP	Invokes the STAGS "pre-analysis" procedure.	ANA
STATIC	Prepares strategy for a STAGS static analysis.	ANA
VIBRATION	Prepares strategy for a STAGS vibration analysis	ANA

where:

PRE => Pre-processing
ANA => Analysis
POS => Post-processing
AUX => Auxiliary

1.5 COMMAND DESCRIPTIONS

This subsection contains detailed descriptions of each GIST command. The commands are listed in alphabetical order, with the command name appearing at the top of the first and every page of the individual command description.

Before proceeding with the command descriptions, we introduce the following set of notational conventions which will be used throughout. The notation also appears in the displays produced by the HELP command, and in Control Module prompting messages:

Notation for Command Descriptions

- Upper-case words are to be taken literally, i.e., entered as part of the command. Lower-case words are not to be taken literally. They are to be replaced with an appropriate name, number or array.
- To distinguish between "numeric" replacement and "alphanumeric" replacement, check the first letter of the word. If the first letter is lower-case, a numeric value is expected. If the first letter is upper-case and the rest of the word is lower case, a name is expected.
- Square brackets [ ] indicate that the enclosed items are optional. The brackets are not to be entered as part of the command.
- Curly brackets { } indicate that at least one of the enclosed items should be selected. The brackets are not to be entered as part of the command.
- A vertical bar | is used to separate items which are mutually exclusive. The bar should not be entered as part of the command.

B U C K L I N G

Function

Defines strategy for a "bifurcation buckling" analysis to be performed about either a linear or nonlinear prestress state. This is an eigenvalue analysis in which the results represent "critical load" factors and corresponding displacement modes.

Strategy parameters are saved in the global database for subsequent review and may be selectively revised without changing all previous settings. Once the strategy is complete, computation is initiated via the COMPUTE command which invokes the STAGS structural analyzer.

Formats

```
BUCKLING [/LINEAR]    {   PRELOAD = pa,[pb]
                          MODES   = maxnum
                        [ RANGE   = eigmin, eigmax ]
                        [ SHIFT   = eigshift       ]
                        [ MAXERR  = tolerance      ]
                        [ LIST [/Items] = switch   ]
                        [ SAVE [/Items] = switch   ]  }

BUCKLING /NONLINEAR   {   PRESTEP = stepnum
                          MODES   = maxnum
                         [ ... same options as /LINEAR ]  }
```

Qualifiers

/LINEAR (default)

Indicates that the buckling analysis is to be performed with respect to a linear pre-stress state.

/NONLINEAR

Indicates that the buckling analysis is to be performed with respect to a "previously computed" nonlinear prestress state.

Keywords

BUCKLING
PRELOAD

The keyword phrase:

PRELOAD = pa [,pb]

specifies the linear pre-buckling load factors corresponding to load systems A and B (or 1 and 2) respectively.

The resulting eigenvalues are to be interpreted as critical, (i.e., buckling) load factors according to the following expression:

{P}crit = eigenvalue * pa * {P}a + pb * {P}b

where {P}crit represents the critical load vector.

BUCKLING
PRESTEP

The keyword phrase:

PRESTEP = step

specifies the number of the load (or time) step corresponding to the nonlinear pre-buckling configuration. It is assumed that the displacement vector at this particular step has already been computed and saved.

BUCKLING
MODES

The keyword phrase:

MODES = maximum

specifies the maximum number of eigenvalues/eigenvectors (i.e., critical load-factors/modes) to be computed. The eigenvalues are extracted in order of increasing absolute value starting with the smallest, unless otherwise specified via the SHIFT or RANGE keywords.

BUCKLING
RANGE

The [optional] keyword phrase:

RANGE = eigmin,eigmax

specifies the lower and upper bounds of the eigenvalue interval. The eigenvalues are then extracted in order of increasing distance from the center of the interval. The first eigenvalue will therefore be the one closest to 'eigcen' where:

eigcen = (eigmin + eigmax) / 2

The computation automatically terminates when all eigenvalues within the specified interval have been determined, up to the 'maximum' set by MODES.

Default: RANGE = 0.,0.

BUCKLING
SHIFT

The [optional] keyword phrase:

SHIFT = target

specifies an initial eigenvalue shift, i.e., a target value. The eigenvalues (and corresponding modes) are then extracted in order of increasing distance from the shift value, up to the maximum number specified by MODES.

Default: SHIFT = 0.

Note: This option is superceded by the RANGE option.

BUCKLING
MAXERR

The [optional] keyword phrase:

MAXERR = tolerance

specifies the convergence criterion to be used for eigenvalue extraction. The value 'tolerance' represents the maximum acceptable relative error in each requested eigenvalue (as measured over two successive iterations).

Default: MAXERR = 1.E-5

BUCKLING
LIST

The [optional] keyword phrase:

LIST[/Items] = switch

indicates what is to be listed during the subsequent solution interval. Valid 'Item's are:

I (Intermediate iteration data)
D (Displacement modes)
Otherwise, all of the above are assumed.

The value 'switch' must be either: 0 => No list
or: 1 => List

Defaults: LIST/I = 1 LIST/D = 0

The default settings are recommended for interactive use, since the bulk of the solution may be reviewed during postprocessing.

BUCKLING
SAVE

The [optional] keyword phrase:

SAVE [/Item] = switch

indicates which results are to be saved (i.e, archived) during the subsequent solution interval. Valid 'Item's are:

D (Pre-buckling displacement vector)
M (Buckling mode vectors)
Otherwise, all of the above are assumed.

The value 'switch' must be either: 0 => no archival
or: 1 => archive

It is recommended to save solution vectors for potential post-processing and/or analysis continuation. [See HELP DATABASE]

Default: SAVE = 1

B U L K L B

Function

Invokes the GIFTS "Bulk Load and Boundary Condition Generator" of the same name.

BULKLB is a bulk load and boundary condition generator designed to apply loads and constraints to those portions of a model generated with BULKM, e.g., at key points, distributed along key lines, or distributed over surface grids. It may be used to apply distributed line, surface and inertial loads, prescribed displacements along lines and surfaces, simple Lagrangian constraints along lines and surfaces, as well as initial conditions (displacements and velocities) for an ensuing dynamic analysis.

Format

BULKLB

Qualifiers

None.

Keywords

None.

Remarks

GIFTS processors are driven by the GIFTS Command Language. Help is available from within the processor by entering the GIFTS command,

HELP /topic/

where 'topic' is the name of either a GIFTS processor or of a GIFTS command. To exit from the processor, use the GIFTS command: QUIT.

For comprehensive information, including examples, on any of the GIFTS processors, consult the GIFTS references listed at the end of this manual. For a sample of GIFTS utilization in a GIST analysis, refer to Section 5 herein.

B U L K M

Function

Invokes the GIFTS "Bulk Model Generator" of the same name.

BULKM is an automatic model generation processor with corresponding graphics capabilities. It is suitable for large, three dimensional beam, plate and shell type structures that can be easily modeled by repetitious generation of points and elements. Plotting is limited to the overall geometric features of the model, including key points, lines and grid boundaries (the mesh is suppressed). Once the "bulk" of the model has been defined with BULKM, a complementary processor: EDITM, is recommended for detailed, element-oriented verification, revision, and extension.

Format

BULKM

Qualifiers

None.

Keywords

None.

Remarks

GIFTS processors are driven by the GIFTS Command Language. Help is available from within the processor by entering the GIFTS command,

HELP /topic/

where 'topic' is the name of either a GIFTS processor or of a GIFTS command. To exit from the processor, use the GIFTS command: QUIT.

For comprehensive information, including examples, on any of the GIFTS processors, consult the GIFTS references listed at the end of this manual. For a sample of GIFTS utilization in a GIST analysis, refer to Section 5 herein.

C L E A R

Function

Clears all or part of the current GIST Job.

The CLEAR command may be used to clean-up after completing or falsely starting an analysis; to erase either the model or the solution without forfeiting the other; or to simply reset various global parameters.

Format

CLEAR { JOB | PREP | ANALYSIS | POST | BATCH }

Qualifiers

None.

Keywords

CLEAR JOB

Erases ALL of the current GIST Job. All files associated with the global database are automatically deleted. To begin work on a fresh Job, the analyst must QUIT and "log-on" again.

CLEAR PREP

Deletes the "pre-processing database" associated with the current Job. Note: this eliminates the GIFTS model definition and thus precludes any further GIFTS postprocessing of solution data. If no solution data exists, this form of the command has the same effect as: CLEAR JOB.

CLEAR ANALYSIS

Deletes the "analysis database" associated with the current Job, but leaves the "pre-/post-processing database" unscathed.

CLEAR STRATEGY

Erases the current solution strategy and resets all optional parameters to their default values.

CLEAR BATCH

Unlocks the "analysis database" for interactive access. Whenever a batch run is launched (via the COMPUTE or SETUP) commands, the analysis database is automatically locked for batch utilization. The analyst must therefore monitor all pending batch runs associated with the current Job manually, i.e., through the computer operating system, and CLEAR BATCH only when all such runs have reached a conclusion.

C O M P U T E

Function

Invokes the STAGS "structural analyzer" (STAGS2) to compute a solution interval based upon the currently defined solution strategy.

This is the "go-button" for analysis computation. It is unlocked only after the strategy for the next solution interval has been completely defined via the STATIC, DYNAMIC, BUCKLING or VIBRATION commands. The computation may be performed either on- or off-line by selecting the appropriate qualifier.

Formats

COMPUTE [/INTERACTIVE] [OUTPUT=Filename]

COMPUTE /BATCH QUEUE=Qname [MAXCOR=size] [OUTPUT=Filename]

Qualifiers

/INTERACTIVE (default)

Indicates that the computation is to be performed "on-line"; i.e., monitored from the analyst's interactive terminal. This mode of operation is recommended for "test" cases and preliminary studies.

/BATCH

Indicates that the computation is to be performed "off-line". The structural analyzer is submitted to a batch queue where it will await execution. The analyst may continue to interact with GIST although restricted from the "analysis database" until the batch process has reached its conclusion. This mode of operation is recommended for more realistic problems which are either too large or too long to be run at high priority from a dedicated computer terminal.

Keywords

COMPUTE
QUEUE

The keyword phrase:

QUEUE = Qname

specifies the name of the "batch queue" to which the processor is being sent as a "batch run".

Choices currently available are:

{ FAST | NORMAL | LONG | HUGE }

COMPUTE
MAXCORE

The [optional] keyword phrase:

MAXCORE = pages

specifies the maximum number of "pages" (128 blocks on VAX/VMS) to be used for central processing.

Default: MAXCORE = 300

COMPUTE
OUTPUT

The [optional] keyword phrase:

OUTPUT = Filename

specifies the name of a permanent disk file to which the output of the following interactive/batch process is to be disposed.

Filename conventions are those of the host operating system

Default: For interactive runs, the terminal;
For batch runs, the "LOG" file.

D Y N A M I C

Function

Defines strategy for a "dynamic response" analysis involving either linear or nonlinear geometric and/or material behavior.

Strategy parameters are saved in the global database for subsequent REVIEW and may be selectively revised without changing all previous settings. Once the strategy is complete, computation is initiated via the COMPUTE command which invokes the STAGS structural analyzer.

Formats

```
DYNAMIC [/NONLINEAR]     {  INTERVAL = tmin,tmax
                            STEPSIZE = dt
                            METHOD   = Name
                            NEWTON   = Type [,freq]
                            HISTORY  = Shape,values
                          [ START    = Type [,step]     ]
                          [ DAMPING  = alpha,beta,gamma ]
                          [ MINDTS   = mindts           ]
                          [ MAXDTS   = maxdts           ]
                          [ MAXERR   = maxerr           ]
                          [ LIST [/Item] = switch       ]
                          [ SAVE [/Item] = switch       ]  }

DYNAMIC /LINEAR          ( same as nonlinear excluding NEWTON )
```

Qualifiers

/NONLINEAR (default)

Indicates that the dynamic analysis is to account for nonlinear effects such as "moderately" large rotations and elastic-plastic material behavior.

/LINEAR

Indicates that the dynamic analysis is to be strictly linear. This limits its validity to problems in which displacements are "infinitesimal" and deformations are "elastic".

Keywords

DYNAMIC
INTERVAL

The keyword phrase:

INTERVAL = tmin, tmax

specifies the solution interval in terms of the independent variable, time ('t') A dynamic analysis usually begins with 'tmin' set to 0. and 'tmax' set to some value of 't' at which the analysis is to be interrupted for evaluation and/or change of strategy. For subsequent solution intervals, i.e. restarts, 'tmin' should correspond to a pre-computed (and archived) time step. [See HELP DYNAMIC STEPSIZE]

DYNAMIC
STEPSIZE

The keyword phrase:

STEPSIZE = dt

specifies the time increment to be used in the traversal of the forthcoming solution interval (see HELP DYNAMIC INTERVAL). Thus, assuming 'dt' remains fixed within the interval, the number of "solution steps" (computed configurations) should be:

nsteps = (tmax-tmin)/dt

[Solution steps are automatically assigned numbers 0,1,2,... starting with the initial conditions and proceeding "through" all subsequent solution intervals. These numbers provide a convenient indexing system for postprocessing reference.]

DYNAMIC
METHOD

The keyword phrase:

METHOD = { EXPLICIT | TRAPEZOIDAL | GEAR | PARK }

specifies the time integration method to be used on the spatially discretized equations of motion. The choices correspond to:

EXPLICIT ..... Explicit Central Difference Method

TRAPEZOIDAL .... Trapezoidal Rule

GEAR ..... Gear's 2nd-order Method

PARK ..... Park's 2nd-order Method

DYNAMIC
NEWTON

The keyword phrase:

NEWTON = { TRUE | MODIFIED,freq | SELECTIVE,max | FALSE }

specifies the "linearization" algorithm to be used for nonlinear analysis. All of the above choices are variations on the generalized Newton-Raphson procedure with the following distinctions:

TRUE ..... The "tangent stiffness" matrix is formed and re - factored upon every iteration of every time step.

MOD,freq ... The matrix is formed and refactored only at the beginning of every 'freq' th time step.

SEL,max .... The matrix is formed and refactored selectively (by the analyzer) up to a maximum of 'max' times.

FALSE ..... The matrix is formed and factored only at the beginning of the solution interval.

DYNAMIC
HISTORY

The keyword phrase:

HISTORY [/A/B] = { LIN | EXPO | TRIG }, values

specifies the "load amplitude history" for subsequent dynamic solution intervals. This may be changed as often as necessary. The qualifier /A or /B corresponds to the load system, and the array 'values' specifies the load factor as a function of time according to the load shape {...}.

LIN ..... Piecewise linear load-time function

EXPO .... Exponentially decaying load-time function

TRIG .... Trigonometric load-time function.

* See Subsection 2.2.2 for the precise definition of 'values'.

DYNAMIC
START

The [optional] keyword phrase:

START = { NEW | NEXT | FROM step }

specifies the starting conditions for the solution interval:

NEW ..... Start from initial conditions.

NEXT ..... Simple analysis continuation. Use the current, i.e., last-obtained, solution step as a starting approximation for the next logical solution step.

FROM step ... Recompute an earlier solution interval using step number 'old' as a starting approximation.

Default: START = NEXT

DYNAMIC
DAMPING

The [optional] keyword phrase:

DAMPING = alpha,beta,gamma

specifies the amount of "structural damping" to be included in the dynamic analysis. Each of the above coefficients correspond to a different source of velocity-dependent forces:

{f}v-d = alpha*[M]{v} + beta*[K]{v} + gamma*[D]{v}

where: [M] = mass matrix
[K] = stiffness matrix or operator
[D] = diagonal damping matrix (defined with loads)
{v} = structural velocity vector
{f} = internal force vector

Default: DAMPING = 0.,0.,0.

DYNAMIC
MINDTS

The [optional] keyword phrase:

MINDTS = mindts

specifies the limit for automatic stepsize decreases in the subsequent solution interval. The minimum possible value of 'dt' will then be:

dtmin = mindts * dt

where 'dt' is specified with the STEPSIZE keyword. If 'mindts' is set equal to 1.0, no decreases in stepsize will be engendered.

Default: MINDTS = 1.0

DYNAMIC
MAXDTS

The [optional] keyword phrase:

MAXDTS = maxdts

specifies the limit for automatic stepsize increases in the subsequent solution interval. The maximum possible value of 'dt' will then be:

dtmax = maxdts * dt

where 'dt' is specified with the STEPSIZE keyword. If 'maxdts' is set equal to 1.0, no increases in stepsize will be engendered.

[NOTE: Stepsize increases may inadvertently destroy the analyst's representation of the applied load history.]

Default: MAXDTS = 1.0

DYNAMIC
MAXERR

The [optional] keyword phrase:

MAXERR = tolerance

specifies the convergence criterion to be used for the nonlinear iteration. The value 'tolerance' represents the maximum acceptable relative error in either displacement or force (residual)

vector norms. Up to 12 iterations may be performed to meet this criterion, but less will be used if convergence appears unlikely.

Default: MAXERR = .001

DYNAMIC
LIST

The [optional] keyword phrase:

LIST [/Item] = switch

indicates what is to be listed during the subsequent solution interval. 'Item' may be any of the following:

D (Displacements and Velocities)
R (Stress resultants)
S (Stresses)
E (Strains)
F (Residual forces: {Fext} - [M]{a})
Otherwise, all of the above are assumed.

The value 'switch' may be: 0 => No list
or: >0 => List every 'switch' load steps

Default: LIST = 0 (No results listed until postprocessing)

DYNAMIC
SAVE

The [optional] keyword phrase:

SAVE [/Item] = switch

indicates which results are to be saved (i.e, archived) during the subsequent solution interval. Valid 'Item's are:

D (Displacements,velocities,accelerations,etc.)
S (Stress sets (stresses,strains,resultants))
F (Residual forces)
K (Assembled stiffness matrix)
Otherwise, all of the above are assumed.

The value 'switch' may be: 0 => no archival
or: >0 => archive every 'switch' steps

It is recommended to save solution vectors for potential post-processing and/or analysis continuation. [See HELP DATABASE]

Default: SAVE/D = 1 SAVE/S = 1

E D I T L B

Function

Invokes the GIFTS "Load and Boundary Condition Editor" of the same name.

EDITLB is a display and edit processor intended to provide local modification capability for loads, boundary conditions, etc. applied with BULKLB. It may also be used to apply simple, e.g., non-distributed loads and boundary conditions on models generated with BULKM and/or EDITM. It may be used to graphically verify all aspects of loading and initial conditions.

Format

EDITLB

Qualifiers

None.

Keywords

None.

Remarks

GIFTS processors are driven by the GIFTS Command Language. Help is available from within the processor by entering the GIFTS command,

HELP /topic/

where 'topic' is the name of either a GIFTS processor or of a GIFTS command. To exit from the processor, use the GIFTS command: QUIT.

For comprehensive information, including examples, on any of the GIFTS processors, consult the GIFTS references listed at the end of this manual. For a sample of GIFTS utilization in a GIST analysis, refer to Section 2.5 herein.

E D I T M

Function

Invokes the GIFTS "Model Editor" of the same name.

EDITM is a model editor designed primarily to correct, update and verify models generated with BULKM. However, it may also be used to define entire models, or subregions, which are not suitable for automatic generation. EDITM has extensive plotting capabilities which include node, element, material and thickness labeling options, as well as detailed illustrations of both the mesh and of local beam cross-sectional properties. Numerical data on all aspects of the model is also available in tabular form upon command.

Format

EDITM

Qualifiers

None.

Keywords

None.

Remarks

GIFTS processors are driven by the GIFTS Command Language. Help is available from within the processor by entering the GIFTS command,

HELP /topic/

where 'topic' is the name of either a GIFTS processor or of a GIFTS command. To exit from the processor, use the GIFTS command: QUIT.

For comprehensive information, including examples, on any of the GIFTS processors, consult the GIFTS references listed at the end of this manual. For a sample of GIFTS utilization in a GIST analysis, refer to Section 2.5 herein.

H E L P

Function

Provides information on all available GIST commands and general system utilization.

Format

HELP [Name] [Subname]

Qualifiers

None.

Keywords

'Name' may be any valid GIST command or topic name for which specific information is desired.

'Subname' may be any command component, i.e., qualifier or keyword, (or subtopic) associated with the command (or topic) specified by 'Name.

Notice that both 'Name' and 'Subname' are optional. If neither is specified, a general listing of all available commands and topics is presented.

Examples

1. G I S T > HELP DYNAMIC

   The HELP command displays a general description of the DYNAMIC command including its function, format, and qualifiers and keywords for which there is more information available.

2. G I S T > HELP DYNAMIC/NONLINEAR

   The HELP command displays specific information on the /NONLINEAR form of the DYNAMIC command.

3. G I S T > HELP DYNAMIC METHOD

   The HELP command displays specific information on the DYNAMIC keyword: METHOD.

M A N A G E

Function

Invokes any of the GIST system's interactive database editors.

Presently, GIST employs two special database managing processors; one for the (GIFTS) pre- and post-processing databases and one for the (STAGS) analysis database. These are intended for advanced users only. They provide a utility for performing detailed database maintenance and editing operations, which are usually only required when the database becomes either excessively large or of questionable integrity. Some information on the usage of the GIST database editors is provided in Section 2.4, but the primary documentation is to be found in references [G3] and [N4].

Format:

MANAGE { GIFTS | STAGS }

Qualifiers

None.

Keywords

MANAGE GIFTS

Invokes the GIFTS database "editor", called DUMP. This processor is not an editor, per se, but rather a comprehensive data-listing utility for those files comprising the pre- and post-processing databases (see Section 2.4 and [G3]).

MANAGE STAGS

Invokes the STAGS database editor, called CLAUDE. This processor does not belong to the STAGS family. It is a member of the NICE [N1] architecture library. Thus, it may be used to interactively manipulate any files which are of the so-called "DAL" or "GAL" variety. The STAGS 'STG' file, which is the principal member of the analysis database, fits into the former category (see Section 2.4, Appendix B, [N2] and [N4]).

Q U I T

Function

Terminates the current GIST session.

QUIT is a harmless command used to "log-off" of the GIST network. It neither creates nor destroys anything of importance in the database.

Format

QUIT

Qualifiers

None.

Keywords

None.

Remarks

GIFTS processors, e.g., BULKM, EDITM, etc. also feature a QUIT command. Do not confuse the GIST command with the GIFTS command. Unlike the GIST' QUIT command, the GIFTS' QUIT command does update the global database, and then subsequently transfers control back to the GIST Control Module. Thus, in order to "log-off" of the GIST network from within say BULKM, one would enter the command QUIT twice: once (as a GIFTS command) to exit from BULKM, and then again (as a GIST command) to leave the network.

R E S U L T

Function

Invokes the GIFTS "result-display" processor of the same name, and automatically precedes it with the STAGS->GIFTS (solution-> postprocessing) "adaptor".

RESULT may be used to display or tabulate the "physical" solution response at specific load- or time-steps, i.e., spatially. Display capabilities include deformed geometry, stress contours, element- coded stress levels, and principal stress directions. Numerical values may be selectively listed by node or element. Any sequence of pre-computed STAGS solutions may be processed in a single RESULT session by using the appropriate form of the command.

Formats

RESULT [/NEW] SOLUTION = step1 [,step2,increment]

RESULT /OLD

Qualifiers

/NEW (default)

Indicates that a new selection of solution steps is to be extracted from the analysis database for postprocessing. All available STAGS displacements, velocities, modes, stresses, strains, etc. corresponding to the selected steps will be transferred to the GIFTS postprocessing database. NOTE: This operation is not cumulative, i.e., the postprocessing database is always "erased" before anything from the analysis database is transferred. Thus, it may be necessary to specify the same step numbers on more than one occasion.

/OLD

Indicates that no new solution steps are to be extracted from the STAGS database for GIFTS postprocessing. In this case, the STAGS-> GIFTS "adaptor" is bypassed and the analyst may directly resume postprocessing the "old", i.e., left-over, solutions.

Keywords

RESULT
SOLUTIONS

The keyword phrase:

SOLUTIONS = step1 [,step2,incr]

specifies a sequence of solution steps to be extracted from the analysis database and prepared for postprocessing. The first step to be extracted (assuming it is present) will be 'step1', the last step 'step2', and the step increment will be 'incr'. If not specified, 'step2 = step2' and 'incr = 1'. Inside RESULT, the solution steps will be referred to as "load-cases" and numbered consecutively as 1,2,...n, where 'n' is the total number of solution steps actually transfered by the current RESULT [/NEW] command.

Remarks

GIFTS processors are driven by the GIFTS Command Language. Help is available from within the processor by entering the GIFTS command,

HELP /topic/

where 'topic' is the name of either a GIFTS processor or of a GIFTS command. To exit from the processor, use the GIFTS command: QUIT.

For comprehensive information, including examples, on any of the GIFTS processors, consult the GIFTS references listed at the end of this manual. For a sample of GIFTS utilization in a GIST analysis, refer to Section 2.5 herein.

REVIEW

Function

Provides feedback on all aspects of a GIST analysis.

The REVIEW command may be used to obtain current information on general Job status, model statistics, solution statistics, solution strategy and the contents of the database.

Formats

REVIEW { JOB | PREP | ANALYSIS | STRATEGY | SOLUTION }

REVIEW /TOC { JOB | PREP | ANALYSIS | POST }

Qualifiers

/TOC

Provides a database "table-of-contents".

Keywords

REVIEW [/Qualifier] JOB

Reviews the general status of the current GIST "Job", or for /TOC, summarizes the files comprising the Job database.

REVIEW [/Qualifier] PREP

Reviews the status of the current GIST model, or, for /TOC, summarizes the files comprising the "pre-processing database".

REVIEW [/Qualifier] ANALYSIS

Reviews the status of the current GIST analysis, or, for /TOC, summarizes the "data-sets" comprising the "analysis database".

REVIEW [/Qualifier] SOLUTION [=step1,step2,inc]

Reviews the status / computational-statistics of solution steps 'step1' thru 'step2'. If no steps are specified, the current, i.e., last computed, step is reviewed.

REVIEW STRATEGY

Reviews the current solution strategy showing the values of all parameters and indicating which parameters are undefined. This form of the command may be used in alternating fashion with the strategy definition commands (e.g., STATIC, DYNAMIC) to verify each parameter as it is specified.

S E T U P

Function

Invokes the STAGS "pre-analyzer" (STAGS1) and automatically precedes it with the GIFTS->STAGS (pre-processing -> analysis) "adaptor".

SETUP is required at the transition between the pre-processing phase and the analysis phase of a Job. It should be employed only after the model, loads, bcs, etc. have been completely defined, and need not be repeated unless pre-processing definitions are subsequently updated. The STAGS pre-analyzer and its role in the overall solution process is discussed in Section 2.2. The important thing to remember is that it sets the stage for all forms of solution strategy and computation involving the STAGS analyzer.

Formats

SETUP [/INTERACTIVE] [OUTPUT=Filename]

SETUP /BATCH QUEUE=Qname [MAXCOR=size] [OUTPUT=Filename]

Qualifiers

/INTERACTIVE (default)

Indicates that the SETUP procedure is to performed "on-line", i.e., monitored from the analyst's interactive terminal.

/BATCH

Indicates that the SETUP procedure is to be performed "off-line". The adaptor / pre-analyzer runstream is submitted to a batch queue where it will await execution. The analyst may continue to interact with GIST although restricted from the "analysis database" until the batch process has reached its conclusion.

Keywords

SETUP
QUEUE

The keyword phrase:

QUEUE = Qname

specifies the name of the "batch queue" to which the processor is being sent as a "batch run".

Choices currently available are:

{ FAST | NORMAL | LONG | HUGE }

SETUP
MAXCORE

The [optional] keyword phrase:

MAXCORE = pages

specifies the maximum number of "pages" (128 blocks on VAX/VMS) to be used for central processing.

Default: MAXCORE = 300

SETUP
OUTPUT

The [optional] keyword phrase:

OUTPUT = Filename

specifies the name of a permanent disk file to which the output of the following interactive/batch process is to be disposed.

Filename conventions are those of the host operating system

Default: For interactive runs, the terminal;
For batch runs, the "LOG" file.

S T A T I C

Function

Defines strategy for a "static response" analysis involving either linear or nonlinear geometric and/or material behavior.

Strategy parameters are saved in the global database for subsequent review and may be selectively revised without changing all previous settings. Once the strategy is complete, computation is initiated via the COMPUTE command which invokes the STAGS structural analyzer.

Formats

```
STATIC [/NONLINEAR]   INTERVAL= pamin, pamax [,pbmin,pbmax]
                      STEPSIZE= dpa [,dpb]
                      NEWTON  = Type [,freq]
                    [ START   = Type [,step] ]
                    [ MAXCUT  = maxno        ]
                    [ MAXERR  = maxerr       ]
                    [ EXTRAP  = { ON | OFF } ]
                    [ LIST [/Item] = switch  ]
                    [ SAVE [/Item] = switch  ]

STATIC /LINEAR   LOADS=pa,pb  [ LIST=switch ]  [ SAVE=switch ]
```

Qualifiers

/NONLINEAR (default)

Indicates that the static analysis is to account for nonlinear effects such as "moderately" large rotations and elastic-plastic material behavior.

/LINEAR

Indicates that the static analysis is to be strictly linear. This limits its validity to problems in which displacements are "infinitesimal" and deformations are "elastic".

Keywords

STATIC
INTERVAL

The Keyword phrase:

INTERVAL = pamin,pamax [,pbmin,pbmax]

specifies the solution interval in terms of "load factors". Thus, the starting and anticipated final load vectors will be:

{P}start = pamin * {P}a + pbmin * {P}b
{P}stop = pamax * {P}b + pbmax * {P}b

where {P}a and {P}b are the reference load vectors defined by the preprocessor corresponding to load systems A and B (i.e., load cases 1 and 2) respectively.

NOTE: For the initial interval, 'pamin' or 'pbmin' should be greater than zero to obtain a nontrivial solution. For subsequent intervals they should refer to a previous solution step.

STATIC
STEPSIZE

The Keyword phrase:

STEPSIZE = incpa [,incpb]

specifies the "load increment" to be used in the traversal of the solution interval. Thus, the increment in load vector will be:

{P}inc = incpa * {P}a + incpb * {P}b

and will be added to the total load vector at the beginning of each "solution step" beginning with {P}start and ending with {P}stop (as defined under STATIC INTERVAL), assuming convergence can be maintained to that point.

[Solution steps are numbered consecutively starting from 0 (the linear solution) and proceeding through all solution intervals.]

STATIC
NEWTON

The keyword phrase:

NEWTON = { TRUE | MODIFIED,freq | SELECTIVE,max | FALSE }

specifies the "linearization" algorithm to be used for nonlinear analysis. All of the above choices are variations on the generalized Newton-Raphson procedure with the following distinctions:

TRUE ..... The "tangent stiffness" matrix is formed and re - factored upon every iteration of every load step.

MOD,freq ... The matrix is formed and refactored only at the beginning of every 'freq' th load step.

SEL,max .... The matrix is formed and refactored selectively (by the analyzer) up to a maximum of 'max' times.

FALSE ..... The matrix is formed and factored only at the beginning of the solution interval.

STATIC
START

The [optional] keyword phrase:

START = { NEW | NEXT | FROM step }

specifies the starting conditions for the solution interval:

NEW ..... Start from scratch, i.e., with the linear solution.
NEXT ..... Simple analysis continuation. Use the current, i.e., last-obtained, solution step as a starting approximation for the next logical solution step.
FROM step ... Recompute an earlier solution interval using step number 'old' as a starting approximation.

Starting load factors should correspond to the solution step indicated by the above selection.

Default: START = NEXT

STATIC
MAXCUT

The [optional] keyword phrase:

MAXCUT = maxcut

specifies the maximum number of load-step cuts permitted during the solution interval. The decision to "cut", i.e., halve, the step is made by the analyzer based upon local convergence behavior and the allowable frequency of stiffness updates.

Default: MAXCUT = 0

STATIC
MAXERR

The [optional] keyword phrase:

MAXERR = tolerance

specifies the convergence criterion to be used for the nonlinear iteration. The value 'tolerance' represents the maximum acceptable relative error in either displacement or force (residual) vector norms. Up to 12 iterations may be performed to meet this criterion, but less will be used if convergence appears unlikely.

Default: MAXERR = .001 (.1%)

STATIC
EXTRAPOLATION

The [optional] keyword phrase:

EXTRAPOLATION = { ON | OFF }

specifies whether or not solution extrapolation is to be used during the interval. With extrapolation ON, as many as three previous displacement vectors may be used to form the initial approximation at each load level (i.e., a quadratic fit). If extrapolation is turned OFF, only the preceeding displacement vector (unscaled) is employed.

Default: EXTRAP = ON

STATIC
LOAD

The keyword phrase:

LOAD = pa [,pb]

specifies the load factors to be applied to load systems A and B, respectively, for a LINEAR analysis. In this case, an independent solution is obtained for each load system.

STATIC
LIST

The [optional] keyword phrase:

LIST [/Item] = switch

indicates what is to be listed during the subsequent solution interval. 'Item' may be any of the following:

D (Displacements)
R (Stress resultants)
S (Stresses)
E (Strains)
F (Residual forces (includes reactions))
Otherwise, all of the above are assumed.

The value 'switch' may be: 0 => No list
or: >0 => List every 'switch' load steps

Default: LIST = 0 (No results listed until postprocessing)

STATIC
SAVE

The [optional] keyword phrase:

SAVE [/Item] = switch

indicates which results are to be saved (i.e, archived) during the subsequent solution interval. Valid 'Item's are:

D (Displacements)
S (Stress sets (stresses,strains,resultants))
F (Residual forces)
K (Assembled stiffness matrix)
Otherwise, all of the above are assumed.

The value 'switch' may be: 0 => no archival
or: >0 => archive every 'switch' steps

It is recommended to save solution vectors for potential post-processing and/or analysis continuation. [See HELP DATABASE]

Default: SAVE/D = 1 SAVE/S = 1

V I B R A T I O N

Function

Defines strategy for a "small vibration" analysis to be performed about either an unstressed or a nonlinearly pre-stressed state. This is an eigenvalue analysis wherein the results represent "natural frequencies" and corresponding displacement modes.

Strategy parameters are saved in the global database for subsequent review and may be selectively revised without changing all previous settings. Once the strategy is complete, computation is initiated via the COMPUTE command which invokes the STAGS structural analyzer.

Formats

```
VIBRATION [/LINEAR]   {   MODES    = maxnum
                        [ RANGE    = eigmin, eigmax ]
                        [ SHIFT    = eigshift       ]
                        [ MAXERR   = tolerance      ]
                        [ LIST [/Items] = switch    ]
                        [ SAVE [/Items] = switch    ]  }

VIBRATION /NONLINEAR  {   PRESTEP = stepnum
                          MODES   = maxnum
                        [ ... same options as /LINEAR ]  }
```

Qualifiers

/LINEAR (default)

Indicates that the vibration analysis is to be performed with respect to an unstressed or (equivalently) a linearly prestressed state.

/NONLINEAR

Indicates that the vibration analysis is to be performed with respect to a "previously computed" nonlinear prestress state. The resulting vibration modes are to be viewed as "small-amplitude" oscillations about the nonlinearly deformed configuration.

Keywords

VIBRATION
PRESTEP

The keyword phrase:

PRESTEP = step

specifies the number of the load- (or time-) step corresponding to the nonlinear pre-stress configuration. It is assumed that the displacement vector at this particular step has already been computed and archived.

VIBRATION
MODES

The keyword phrase:

MODES = maxno

specifies the maximum number of eigenvalues/eigenvectors (i.e., natural frequencies/modes) to be computed The eigenvalues are extracted in order of increasing absolute value starting with the smallest, unless otherwise specified via the SHIFT or RANGE keywords. The interpretation of an eigenvalue is as a "natural frequency" (in cps).

VIBRATION
RANGE

The [optional] keyword phrase:

RANGE = eigmin,eigmax

specifies the lower and upper bounds of the eigenvalue interval. The eigenvalues are then extracted in order of increasing distance from the "center" of the interval. The first eigenvalue will therefore be the one closest to 'eigcen' where:

eigcen**2 = (eigmin**2 + eigmax**2) / 2

The computation automatically terminates when all eigenvalues within the specified interval have been determined, up to the maximum set by 'MODES'.

Default: RANGE = 0.,0.

VIBRATION
SHIFT

The [optional] keyword phrase:

SHIFT = target

specifies an initial eigenvalue shift, i.e., a target value. The eigenvalues (and corresponding modes) are then extracted in order of increasing distance from the shift value, up to the maximum number specified by MODES.

Default: SHIFT = 0.

Note: This option is superceded by the RANGE option.

VIBRATION
MAXERR

The [optional] keyword phrase:

MAXERR = tolerance

specifies the convergence criterion to be used for eigenvalue extraction. The value 'tolerance' represents the maximum acceptable relative error in each requested eigenvalue (as measured over two successive iterations).

Default: MAXERR = 1.E-5

VIBRATION
LIST

The [optional] keyword phrase:

LIST[/Items] = switch

indicates what is to be listed during the subsequent solution interval. Valid 'Item's are:

I (Intermediate iteration data)
D (Displacement modes)
Otherwise, all of the above are assumed.

The value 'switch' must be either: 0 => No list
or: 1 => List

Defaults: LIST/I = 1 LIST/D = 0

The default settings are recommended for interactive use, since the bulk of the solution may be reviewed during postprocessing.

VIBRATION
SAVE

The [optional] keyword phrase:

SAVE [/Item] = switch

indicates which results are to be saved (i.e, archived) during the subsequent solution interval. Valid 'Item's are:

D (Displacement mode vectors)
Otherwise, all of the above are assumed.

The value 'switch' must be either: 0 => no archival
or: 1 => archive

It is recommended to save solution vectors for potential post-processing and/or analysis continuation. [See HELP DATABASE]

Default: SAVE = 1

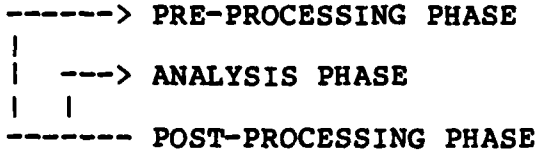
T H E G I S T T U T O R I A L

PART 2:

APPLICATION; GIST STRUCTURAL ANALYSIS

1.0 INTRODUCTION

Problem-solving with GIST, i.e., structural analysis, typically proceeds in the following cyclic manner:



where "preprocessing" represents the idealization and definition of the finite-element model, loads, etc.; "analysis" represents the numerical formulation, strategy and solution of the corresponding mathematical model; and "postprocessing" represents the evaluation of the solution from both physical and computational points of view.

While each of these phases is encountered in practically all classes of structural analysis, it is in nonlinear static and dynamic analysis that the "inner loop" indicated above becomes especially prominent.

For example, during a nonlinear static analysis, the solution is usually advanced in a step-wise fashion, with the analyst postprocessing the intermediate results after every so many steps, and then jumping back into the analysis phase to revise the computational strategy and continue along the solution path. A similar scenario exists for dynamic analysis.

It was to facilitate this essential need for intermediate feedback, and expedite the potentially frequent traversals from phase to phase, that the GIST system was developed in its present form.

As described in Part 1, the entire process of structural analysis is interactively "controlled" and "monitored" from the GIST Control Module employing the GIST Command Language. (This does not imply that all computations must be "performed" interactively. For example, a long solution interval may be interactively "submitted" to a batch queue and then interactively monitored and evaluated when it finishes.)

In accordance with the phases depicted above, the GIST Command Language may be logically partitioned into three groups:

(1) Pre-processing commands
(2) Analysis commands
(3) Post-processing commands

plus a number of auxiliary commands which are not tied to any particular phase (.e.g., data-management commands).

In Part 2, each phase of structural analysis will be discussed, in turn, and related to the appropriate set of GIST commands. The presentation is saturated with short examples of basic operations, building to the final section (2.5), which features comprehensive examples of interactive GIST applications.

For maximum benefit, we suggest referring to the command descriptions in Section 1.5 on a regular basis.

2.1 THE PRE-PROCESSING PHASE

The following GIST commands are associated with the "pre-processing phase" of structural analysis:

BULKM	.....	Invokes the GIFTS bulk model generator
EDITM	.....	Invokes the GIFTS model editor
BULKLB	....	Invokes the GIFTS bulk load/bc generator
EDITLB	....	Invokes the GIFTS load/bc editor
REVIEW	....	Monitors Job status (e.g., REVIEW PREPROCESSING)

The finite element model, i.e., geometry, material properties, fabrication and discretization, is typically defined via the BULKM processor and verified, extended and/or modified (in detail) via the EDITM processor. A similar relationship exists between BULKLB and EDITLB for definition of loads, boundary conditions, initial conditions and generalized constraints. Each of these GIFTS processors is interactive-graphics oriented and driven by the GIFTS command language.

The REVIEW command is actually associated with every phase of analysis. It is a Control Module utility for monitoring the status of a case throughout its evolution. Certain forms of the command, however, will be shown to be specifically relevant to pre-processing.

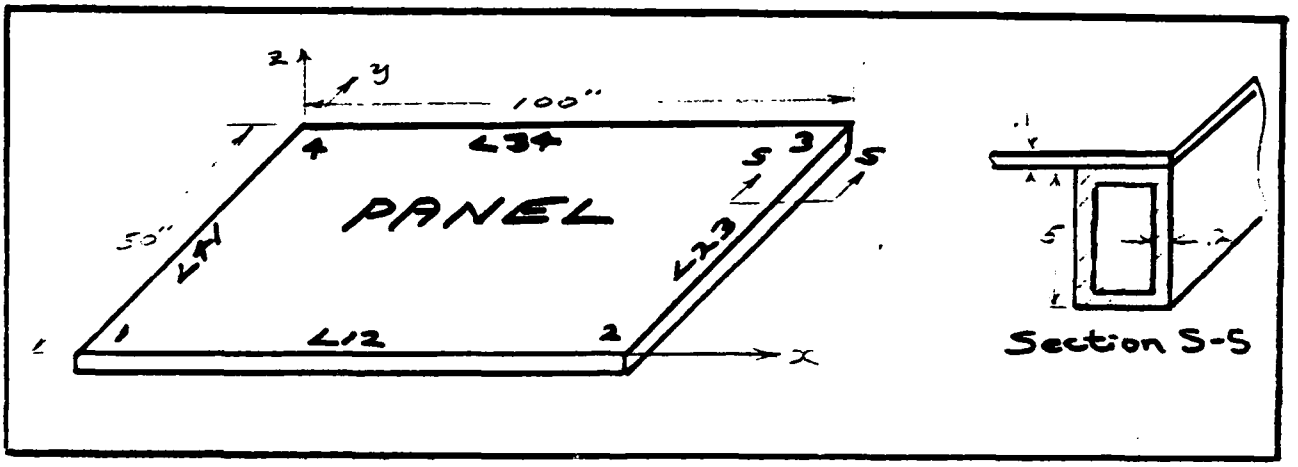
Once the model, loads, boundary conditions, etc. have been completely defined and verified, the user is then ready to enter the "analysis phase" which is described in Section 2.2. It is possible to return to the pre-processing phase as the structural analysis progresses to recheck, or in some cases even modify, the finite element idealization. However, post-facto changes to the model should be made with caution, as solution continuation will not always be guaranteed (see Subsection 2.2.1).

In the following subsections, the function of each of the above GIST commands is expanded upon and illustrated by example. Note that the GIFTS pre-processing command language will only be highlighted in this section (As usual the reader is referred to the GIFTS User's Manual for the details and generalization). Special consideration is given to those GIFTS commands whose interpretation is of special consequence to the STAGS analysis process. For a summary of GIFTS-STAGS "interface conventions" consult Appendix A.

2.1.1 MODEL GENERATION

Model generation typically begins by invoking the GIFTS bulk model generator, BULKM, and defining the basic geometry, fabrication and discretization for one or more subregions of the structure. EDITM is then used to edit the model, with the Control Module intermediating.

As a simple illustration, suppose we wish to model a stiffened rectangular plate with lateral imperfections. We begin by making a sketch of the following form:



The corner numbers: 1,2,3,4, the boundary names: B1,B2,B3,B4, and the surface name: PANEL, are all arbitrary, but will be useful for automatic mesh generation as we will see.

Now, having made such a sketch, we are ready to get on the computer. First, we will summon the Control Module (see Section 1.1) and establish a Jobname, e.g.,

```
==========================================================

<<>>  G I S T   [ GRAPHICS-INTERACTIVE STRUCTURAL ANALYSIS ]

==========================================================

Jobname:   PANEL1

<> New Job

Command | HELP | QUIT
G I S T >                                ( GIST command prompt )
```

To invoke BULKM, we call it by name, i.e.,

```
G I S T >   BULKM
```

and wait for the GIFTS processor to respond with:

```
-------------------------------------------------------------------
<<>>   B U L K M   [ Bulk Model Generator ]   GIFTS/Version xxx
-------------------------------------------------------------------
JOB:  PANEL1

    *                                        ( GIFTS command prompt )
```

Using the sketch for guidance we proceed with the logical sequence of generation: (i) key points, (ii) boundary lines, and (iii) surfaces; declaring material and fabrication properties as necessary:

```
    *   ELMAT,3  /1/  10000., 1.37, .3 /
    *   ETH,1    /1/ .1 /
    *   RECTH    /2/ 5., 2. / .2, .2 /

    *   KPOINT
    >    1/0.
    >    2/100.
    >    3/100.,50.
    >    4/0.,50.
    >    5/50.,25.

    *   LETY/BEAM2,210/1,2

    *   SLINE,10
    >    S1 /1,2,21  /5
    >    S2 /2,3,11  /5
    >    S3 /3,4,21  /5
    >    S4 /4,1,11  /5

    *   GETY /QB4,,410/1,1

    *   GRID4
    >    PANEL /S1,S2,S3,S4

    *   KN/LN/GN/   PLOT
```

In the above command stream, the first command, ELMAT, tabulates the elastic material properties for future reference as "material group" number 1. Similarly, the next two commands, ETH and RECTH, tabulate the section properties for the plate and hollow rectangular stiffener as "thickness group" number 1 and 2, respectively.

The KPOINT command defines the coordinates of the plate's four corner points, plus an additional reference point, 5, which will be used for stiffener orientation. Note that these "Keypoints" are not to be thought of as "nodes", which generally possess degrees of freedom, but rather as labelled generating coordinates.

The subsequent commands utilize the above information to generate the boundaries of our model, which in this case also happen to have stiffeners. The LETY command designates the line-element type to be used during generation. (In the absence of stiffeners, LETY would have been unnecessary.)

The parameters, 'BEAM2' and '210', represent the line-element's "physical" (GIFTS) and "computational" (STAGS) names, respectively, while the next two parameters, '1,2', make the appropriate material and thickness group assignments. The SLINE command then generates the four straight lines labelled S1 through S4, incorporating the beam elements just specified. Notice that each line is described by its label (e.g. S1), key-point connectively (e.g. 1,2) and node density (e.g., 21). For instance, "stiffener" S1 is composed of 20 beam elements.

The last stage in the progression is surface (i.e., "grid") generation which is accomplished with the next two commands: The GETY command is the 2-dimensional equivalent of the LETY command. It designates a grid-element type for subsequent generation. In this case, 'QB4', (4-node quadrilateral bending element) and '410' are the physical and computational surface element names (see Appendix A), and the associated material and thickness group numbers are both 1.

The GRID4 command then generates the 4-sided surface bounded by lines S1, S2, S3, S4 by interpolation and fills in the mesh with the quadrilateral elements just prescribed. The mesh density is determined from the corresponding boundary node densities. Hence, the grid will possess a total of 21 x 11 (=231) nodes and 20 x 10 (=200) quadrilateral elements.

The final command sequence, 'KN/LN/GN/PLOT', displays the model geometry (excluding elements), labelling all key points, boundary lines and grids. The picture should look something like the original sketch.

Oh yes, we have inadvertently generated a "perfectly" flat plate (a physical impossibility). Now to introduce some transverse waviness, we may, for example, enter the following command:

```
*   SIMP
>    PANEL /3,1 /.03,.01
```

which specifies 3 simusoidal half-waves parallel to the x direction and 1 parallel to the y direction, with respective amplitudes of .03 and .01 inches. Now when we say:

```
*   PLOT
```

the grid will be regenerated with corresponding perturbations in the nodal coordinates (which can be scaled for display).

To verify the model in greater detail, or make "local" changes to our "bulk" definition, we must employ the GIFTS model editor, EDITM. First, to exit from BULKM we enter:

```
*   QUIT
```

and wait for the familiar greeting:

```
Command | HELP | QUIT
G I S T >
```

which means we have been returned to the Control Module and may enter a GIST command. We proceed with:

```
G I S T  >   EDITM
```

which invokes the EDITM processor and catapults us right back to the GIFTS domain.

```
-------------------------------------------------------
<<>>   E D I T M   [ Model Editor ]   GIFTS / Version xxx
-------------------------------------------------------
JOB:  PANEL1

    *
```

We are now in a position to display, tabulate or update the model by command.

For instance, to display the element discretization, labelling all node and element numbers, we may enter:

```
*   ELEMENTS/PN/EN/ PLOT
```

To replace the element number labels with material group numbers and re-display, we enter:

```
*    MN/ PLOT
```

In each case the screen is immediately cleared before displaying the new plot. Since we are looking at a planform view, we may want to rotate the model to give it some perspective and also verify the imperfections, ala:

```
*   ROTV/-60, -30/ PLOT
```

and so on.

To list the nodal coordinates, we use the command:

```
*   INFP /1,100
```

and to list the material properties (for group 1):

```
*   INFM/1
```

Similar INFormation commands are available for practically all model and solution parameters.

Now, suppose we wish to change our definition of the material from elastic-isotropic to plastic-isotropic. We simply redefine material number 1 as follows:

```
*   NLMAT,3 /1,4/ 10000., 1.E7, .3
>   .001, 10000.
>   .002, 15000.
>   .003, 20000.
>   .004, 22000.

*   INFM /1/
```

The NLMAT command defines a "nonlinear material" by setting the stress-free material properties as in the elastic case, and additionally specifying 4-points on the stress-strain curve. The integer 1 on the first line is the material group number. Consequently, the previous definition of material group 1 is completely overwritten. The INFM command is then used to verify the new definition.

To verify our stiffener cross-sectional geometry, we select an especially suited graphics option:

```
*   PLOTBC /2/
```

The above command will display the channel section which we defined as 'thickness group 2' labelling all dimensions and indicating the locations of the attachment point, centroid and shear center, as well as the orientation of the principal axes with respect to the nodal reference plane.

Eventually, after satisfying ourselves that the model is absolutely correct (though not necessarily valid) we are ready to prescribe its environment, i.e., loads and boundary conditions.

Recall that we are currently (in our illustrative example) communicating with EDITM. To return to the Control Module, we again use the command:

```
*   QUIT
     .
     .
     .
G I S T >
```

and ponder our next move.

Before proceeding with load and boundary condition generation, we might wish to confirm our status, via:

```
G I S T >   REVIEW JOB
```

which will show that model generation has been initiated.

For a more descriptive report, we could enter:

```
G I S T >  REVIEW PREP
```

The latter command produces a summary of model statistics, e.g., the number of nodes, elements, grids, materials, etc., currently defined.

Finally, if we happen to be database conscious:

```
G I S T >  REVIEW/TOC PREP
```

will summarize the current preprocessing contribution to the Job database.

Now, suppose we have forgotten which set of loads and boundary conditions are of highest priority to our client, boss, or whomever, and need to make a long-distance call before proceeding. We could then enter:

```
G I S T >  QUIT
```

and feel free to resume later ...

==========

(See Subsection 2.1.2 for the sequel)

Remarks

1. The above example, although simple, illustrates the basic procedure for generating even the most complex models. By using generalizations of the GIFTS commands shown, we can generate "sequences" of arbitrary three dimensional curves and surfaces featuring a variety of materials, fabrications and element types.

2. GIFTS commands do not have to be entered interactively, i.e., one at a time. Instead, entire command streams may be prepared apriori (e.g., with a text editor) and entered via a single command: OLB. For example, if the BULKM command stream were prepared on a file called PANELM.SRC we could have entered:

```
* OLB/PANELM
```

   at the beginning of our interaction with BULKM, and viewed the model generation in "spectator mode".

3. EDITM may be used not only to modify material properties, nodal coordinates, element connectivity, etc., but also to generate entire subregions of the model which do not lend themselves to automatic (i.e., bulk) generation. Conversely, BULKM may be used to make large scale modifications to the model involving redefinitions of keypoints, lines or grids. For example, grid refinement can be accomplished by re-entering the corresponding boundary line generation commands (e.g., SLINE) with larger node-density parameters. The grid will then automatically be re-generated with the appropriate density upon exiting from BULKM. Another popular option is to update the original command file (see previous remark) with a text editor and simply regenerate the entire model.

4. It is often convenient to save frequently used commands or groups of commands in separate text files. These can then be accessed (via the OLB command) as "micro-commands". For example, the command string:

```
*  LPROJ
>   LINE1/SURF1A,SURF1B
>   LINE2/SURF2A,SURF2B
>   LINE3/SURF3A,SURF3B
```

   forces 3 separate lines to be on 3 separate surface intersections. By saving this expression in a text file called PROJ.SRC, it can then be invoked from BULKM at will, via OLB/PROJ, and used to reinforce the intersections whenever necessary, e.g., following a change in one of the surface geometrics.

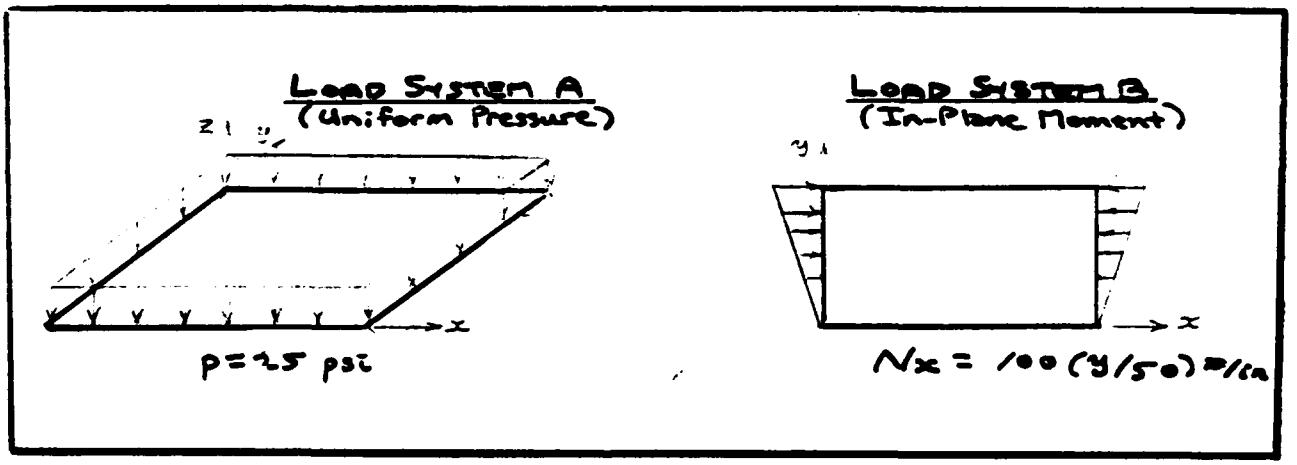
5. The reader is urged to consult the GIFTS Primer [G2] for a broader look at model generation procedures.

2.1.2 LOADS, BOUNDARY CONDITIONS, ETC.

The specification of all external loads, boundary conditions, initial conditions and any generalized constraints, is performed by another set of GIFTS processors: BULKLB and EDITLB.

BULKLB is primarily used to apply distributed loads, b.c.'s, etc. along the "key lines" and "grids" generated with BULKM. EDITLB is then used for graphical verification and local modification of these external conditions. (It can therefore be viewed as the load and b.c. counterpart of EDITM.) Both are invoked from the GIST Control Module by name.

For clarity and continuity, we shall proceed with the example introduced in Subsection 2.1.1 as a means of illustrating the essential features of load and b.c. generation. If you recall, we had generated an "imperfect" rectangular plate, bounded by four eccentric stiffeners. Now suppose we wish to investigate its structural response under the following two external load systems:



which may act simultaneously and in various combinations. For the time being, all boundaries are assumed to be simply-supported. (We will later modify this to exploit the stiffeners.)

Again, armed with the above sketches, we return to the computer and re-awaken the Control Module:

```
==========================================================

<<>>  G I S T   [ Graphics-Interactive Structural Analysis ]

==========================================================
Jobname:  PANEL1

<> Existing Job

G I S T >
```

After confirming that things are as we left them, via:

```
G I S T >   REVIEW PREP
```

we proceed directly to the bulk load and b.c. generator, i.e.,

```
G I S T >   BULKLB

-------------------------------------------------------------
<<>>  B U L K L B    [ BULK LOAD/BC GENERATOR ]   GIFTSxxx
-------------------------------------------------------------
Jobname:  PANEL1

Loading Case 1

   *
```

The GIFTS message "Loading Case 1" implies that all subsequently generated loads will be attributed to the first loading case. (This will become clear in a moment.)

The situation shown in the above sketch may be idealized by employing the following command stream, which will serve as a focal point for our general discussion.:

```
*   SUPL,3 /S1/S2/S3/S4/
*   SUPL,1 /S1/S3/
*   SUPL,2 /S2/S4/
*   SUPL,4 /S2/S4/
*   SUPL,5 /S1/S3/

*   LIVE
*   LOADG
>    PANEL /-25.,-25.,--25.,-25./
*   DEAD

*   LDCASE/2

*   LOADL
>    S2 /0.,-100.
>    S4 /100.,0.

*   QUIT
```

The first block of commands (the SUPL's) introduce the desired boundary conditions. For instance, the SUPL,3 command suppresses the third nodal degree-of-freedom, w, (i.e., the displacement component in the GLOBAL z-direction) on boundaries S1 through S4. The remaining SUPL commands complete the simple-support representation.

The LOADG ("load grid") command then applies a uniform <u>pressure</u> of 25 psi to the top surface of the panel. Note that the pressure value is specified at each of the grid's 4 corner points which

allows for a bilinear variation over the interior. The LIVE command preceding LOADG turns on the "live load" mode switch while the DEAD command turns it off. In the default, or "dead", mode, the pressure would have been constrained to remain parallel to the global z-axis throughout the plate's deformation; while in the "live" mode, the pressure is allowed to rotate and thereby maintain its normality to the surface. If we assume our pressure is being induced by a fluid interface, e.g., hydrostatic conditions, the latter mode is obviously more realistic.

Next, the LDCASE command is used to establish "loading case" number 2. This will segregate the previously defined lateral loads from the yet to be defined in-plane loads. All subsequent loads (until the appearance of another LDCASE command) will be associated with the second loading case, and hence accumulated in a separate array.

Finally, the LOADL ("load-line") command is used to apply the linearly varying Nx (see sketch) on boundaries S2 and S4. Notice that the force per unit length (directed in the global x-direction) is specified at the (two) key points defining each boundary.

Now, to verify and/or modify what has just been (hypothetically) generated with BULKLB, we will transfer to EDITLB, thusly:

```
G I S T >   EDITLB

-------------------------------------------------------------
<<>>  E D I T L B   [ Load/BC Editor ]   GIFTS /Version xxx
-------------------------------------------------------------
JOB:  PANEL1

Loading Case 1

    * ELEMENTS/ELDON/ PLOT
```

In the above scenario, we invoke EDITLB and immediately direct it to plot. The picture will feature a "vectorial" characterization of the pressure field (corresponding to loading case 1) superimposed on the discrete model. The EDITLB model display options are identical to that which may be obtained via EDITM. (Note that the orientation of the model is always remembered from session to session, while such options as ELEMENTS and ELDON are not.)

To display the second loading case, we may enter:

```
    *   LDCASE/ 2
    *   ELDOFF/ PLOT
```

which would display the distributed line loads (on boundaries S2 and S4) as equivalent nodal force vectors. The ELDON/ELDOFF commands switch the element-load display mode on and off.

We may also print the actual force components, via:

```
*   INFLD /1,100/
```

To check the boundary conditions,

```
*   TRANFR/PLOT
```

will display all active translational freedoms as un-headed "arrows", while:

```
*   ROTFR/PLOT
```

will do the same for rotational degrees of freedom. The missing "arrows" will correspond to those freedoms which have been supressed.

We may also display all active freedoms corresponding to a particular component, via 'TRANFR,n', where 'n' is the nodal dof number (1 thru 6). Alternatively, the INFP command may be employed to list the full freedom pattern at any sequence of nodal points.

Now suppose we have changed our minds on the simple-support conditions, and would like to release the boundary except for a discrete "hinge" at each of the four corners. This would allow the stiffeners to play a more substantial role (see sketch). We could always go back to BULKLB and revise our definition of the boundary conditions; but for demonstration purposes we will employ EDITLB to directly release the "internal" boundary nodes. This can be done as follows:

```
*   PN/PLOT
      .
      .
      .

*   RELP
>    5,23
>    24,42
>    43,61
>    62,80
>
*
```

where the PLOT command has been used to graphically recall the node numbers, and the RELP ("release-point") command to release all freedoms at the appropriate nodes. We can now verify the new boundary conditions by redisplaying the active freedoms, i.e., TRANFR/PLOT ... /ROTFR/PLOT.

Another interesting application of EDITLB is for the introduction of

concentrated forces at interior nodes. For example, to "trigger" a particular buckling mode in a nonlinear collapse analysis, we might introduce a small transverse point force (or couple) in the vicinity of the anticipated wave-crest. Point forces (or moments) may be applied to any node in the model with the following command:

```
*   LOADP,k /n/v
```

where 'k' indicates the direction, 'n' is the node number, and 'v' is the magnitude of the force (or moment).

Once the correctness of the model, loads, boundary conditions, etc., have all been established, it is time to get on with the analysis. The user should then return to the Control Module and prepare accordingly, e.g.,

```
*   QUIT
     .
     .
     .

Command | HELP | QUIT
G I S T > HELP ANALYSIS
     .
     .
     .
```

==========

(The "analysis-phase" is taken up in the following section.)

Remarks

1. The orientation of nodal degrees of freedom will depend on the coordinate system which was active when the node was generated. If a "local" coordinate system was not employed, all degrees of freedom will be alligned with the "global" x, y, z axes. If a "local" coordinate system was used, e.g., cylindrical or spherical, the degrees of freedom will be alligned with the local basis vectors, which may vary from point to point. For example, a surface grid generated in a cylindrical coordinate system with the radial coordinate, r, fixed, will have all of its degrees of freedom either normal or tangential to the underlying cylindrical surface. See the GIFTS User's Manual [G1] for the various conventions and restrictions surrounding "local degrees of freedom".

   NOTE: Temporarily, all loads, (except for pressures) must be defined in the global coordinate system; even if the displacement dof's are oriented in some other system. This inconvenient restriction will be removed as soon as possible.

2. Loading cases may be used to define both "load systems" and "initial conditions". By default, each loading case corresponds to a "load system", which is simply a collection of prescribed forces, displacements, temperatures, etc. which are intended to act in unison, i.e., in fixed relative proportion to one another. During the analysis, individual load systems may then be scaled and combined as a matter of solution strategy (see Section 2.2). The current (i.e., STAGS) limitation on the nubmer of active load systems is 2. These will often be referred to as load systems "A" and "B", respectively.

   To designate a loading case as an initial condition vector, use the following form of the LDCASE command:

   ```
   LDCASE,i/n
   ```

   where 'n' is the number of the loading case and 'i' may be either 1, for initial displacements, or 2, for initial velocities. Nonzero initial displacement or velocity components are then specified via the GIFTS 'DISP' commands.

3. Boundary conditions involving displacement degrees of freedom are handled differently depending on whether they are prescribed to be zero (i.e., homogeneous) or nonzero (i.e., nonhomogeneous). The zero displacement conditions are normally imposed via SUP-type commands ("suppress"-freedom) which may be entered at any time, while the nonzero conditions are "prescribed" via DIS-type commands and must be associated with a particular loading case. Note that, initially, all nodal degrees of freedom (i.e., u,v,w,ru,rv,rw) are assumed present at all nodes. The user may then begin "suppressing" or "prescribing" degrees of freedom as necessary.

4. "Live" loads are currently restricted to be normal surface tractions, i.e., pressures. Hence, LOADG is the only GIFTS load- generating command which may legitimately follow the LIVE mode switch. For information on the use and range of validity of the live pressure option in nonlinear analysis, consult the STAGS Theoretical Manual (or call B.O. Almroth at xxx-xxxx).

2.2 THE ANALYSIS PHASE

The following GIST commands are associated with the "analysis phase", in which the user employs the system to compute the structural response, stability or vibrational characteristics of the preprocessed finite element model:

SETUP	Invokes (1) the GIFTS->STAGS Adaptor (G2S) and (2) the STAGS Pre-Analyzer (STAGS2)
STATIC	Prepares strategy for a STAGS static analysis
DYNAMIC	Prepares strategy for a STAGS dynamic analysis
BUCKLING	Prepares strategy for a STAGS buckling analysis
VIBRATION ...	Prepares strategy for a STAGS vibration analysis
COMPUTE	Invokes the STAGS Analyzer (STAGS2) for solution
REVIEW	Monitors Job status (e.g., REVIEW ANALYSIS)

Once the model (i.e., geometry, fabrication, discretization, loads, boundary conditions, etc.) have been completely defined, the SETUP command is required to make the transition from preprocessing to analysis. The command accomplishes two primary functions: (i) the adaptation of the GIFTS-generated model into a suitable STAGS format by the G2S "Adaptor" and (ii) conversion of the physical representation of the model into a computationally convenient representation by the STAGS "Pre-Analyzer" (STAGS1). The combined procedure thus provides the "setup" for all possible forms of STAGS analysis and need not be repeated unless the model is subsequently updated BY THE USER.

The strategy commands: STATIC, DYNAMIC, BUCKLING and VIBRATION, may then be used to specify algorithmic parameters for the forthcoming solution interval. For example, a dynamic solution interval would require specification of the time interval involved, the stepsize to be used to traverse the interval, the desired time integration method, the load (amplitude) history, and various optional parameters (e.g., print and save switches). The solution strategy is automatically saved in the database so that it can be reviewed and selectively updated as the analysis progresses.

When the analyst has finished defining or revising the strategy, solution computation may be initiated via the COMPUTE command. This command invokes the STAGS "Analyzer" (STAGS2) to act on the current strategy, and to compute those solution vectors (i.e., displacements, velocities, stresses, etc.) within the specified interval. If the analyst has opted to save the necessary results, the solution may be advanced, i.e., "restarted" from a previous solution interval. This simply involves updating the strategy and re-issuing the COMPUTE command.

Note that solution computation may be performed in either interactive or batch modes, according to the qualifier selected, i.e., /INTERACTIVE or /BATCH.

The REVIEW command, in the context of the analysis phase, provides a convenient means for checking the overall status of the anaysis and for verifying the current solution strategy. For example, the REVIEW STRATEGY option may be used in conjunction with the strategy definition commands, STATIC, DYNAMIC, etc., to provide a "strategy editing" capability. Other useful REVIEW options will be demonstrated throughout this section.

In the following discussion, each of the above commands will be considered in greater depth, primarily in regard to their respective functions and limitations. (For the details of the command formats, refer to Section 1.5: GIST Command Descriptions, where the commands are presented in alphabetical order.] For more comprehensive information on STAGS analysis capabilities, including theoretical formulations/assumptions, numerical implementation and advice on solution-strategy selection, consult the STAGS Theoretical Manual [S2] and Section 6 of the STAGS User's Manual [S1].

2.2.1 ANALYSIS "SETUP"

The function of the SETUP command is to effect the necessary transition from the preprocessing phase to the analysis phase. This is a two- stage procedure which is automatically administered by the Control Module.

The first stage is the adaptation of the model from GIFTS to STAGS formats, which is performed by the GIFTS->STAGS Adaptor (G2S). The G2S processor operates on the GIFTS database representation of the model and produces a corresponding "input file" for the STAGS Pre-Analyzer (STAGS1).

The Pre-Analyzer is then invoked to produce a computationally-ready version of the model for utilization by the STAGS Analyzer (STAGS2). The resulting data structures, e.g., element kinematic and constitutive arrays, are added to the database and may be subsequently employed for any form of STAGS analysis (irrespective of solution strategy).

Hence, the SETUP procedure is typically required only once during the course of an analysis, i.e., following the initial definition of the complete model. However, should the user subsequently update the model (or somehow destroy the analysis database), the SETUP command will then have to be repeated.

Adaptation from GIFTS to STAGS (G2S)

The G2S "Adaptor" module is formally presented in Part 3 of this text. Essentially, this processor, which is automatically engaged by the SETUP command, finds the appropriate correspondence between GIFTS and STAGS model descriptions, and produces a card-image input file for the STAGS Pre-Analyzer (see below). When running in an interactive mode, the user will see only the initiation and termination of this processor, unless something has gone wrong. For example, if an unrecognized (or unaccomodated) element type is detected in the GIFTS database, G2S will print a corresponding message and terminate. The user should then employ one of the GIFTS processors to correct the definition and repeat the SETUP command. (The summary of GIFTS-STAGS conventions in Appendix A may be helpful.)

Upon successful completion of G2S, the total number of STAGS input records (i.e., card-images) produced is listed. The disk file containing these records is then automatically accessed by the STAGS Pre-Analyzer, which follows immediately. Note that the name of the file created by G2S is of the form:

'Job'.G2S

where 'Job' represents the current 'Jobname' (see Section 1.1). Since it is a standard "formatted" file, it may be easily

transported to another computer for "remote" STAGS executions. Otherwise, the file should only be of interest to the user in the event of an unreconcilable G2S problem; in which case it may be printed and used as a debugging aid.

Pre-Analysis

The STAGS Pre-Analyzer (STAGS1) is normally invoked by the SETUP command as soon as the G2S Adaptor has successfully prepared its input. The function of the Pre-analyzer is to facilitate subsequent element stiffness and internal force calculations, which may occur quite frequently in nonlinear analysis. To this end, the following tasks are performed:

(1) Re-tabulation of element, fabrication, and nodal data in conveniently accessible file structures.

(2) Evaluation and storage of element strain-displacement interpolation arrays and constitutive matrices at Gaussian integration points.

(3) Assignment of computational equation numbers employing a bandwidth optimization scheme.

(4) Assembly of mass matrix and displacement-independent external load vectors in computationally ordered arrays (i.e., according to (3)).

(5) Generation of stiffness profile (or "skyline") vector to facilitate ("compacted-column") stiffness factorization.

(6) Estimation of global stiffness formation, assembly and factorization times (and operation counts).

(7) Archival of results in "analysis database" for utilization during both analysis and postprocessing.

These operations are performed apriori to avoid excessive re-computation by the STAGS Analyzer, but, of course, this is at the expense of increased data storage.

The printed output from the Pre-Analyzer is usually just a summary of the above tasks. However, in case of an inexplicable error, the user may obtain a highly verbose listing from the Pre-Analyzer by using the following option:

```
G I S T >   SETUP   LIST=FULL

     or:

G I S T >   SETUP   LIST=FULL  OUTPUT=Somefile
```

Note also that the SETUP procedure (G2S/STAGS1) may be exercised in a batch mode via:

G I S T > SETUP/BATCH [options]

where the precise command description is presented in Section 1.5.

The eventual result of using the SETUP command is to create an embryonic "analysis database" (see Section 2.4) from which all kinds of (STAGS) structural analyses may evolve.

2.2.2 STRATEGY PREPARATION

Once the structural analysis has been "setup", as described in the previous subsection, the user must specify certain strategy parameters before the STAGS Analyzer may be invoked to perform a solution. These parameters are specified via one of the "strategy commands". The nature of the parameters will depend on the "problem class" selected, and this is explicitly indicated by the command name (and qualifier).

The following commands are presently available for specifying solution strategy:

(1) STATIC
(2) DYNAMIC
(3) BUCKLING
(4) VIBRATION

Each command corresponds to a separate problem class and requires the specification of an associated set of strategy parameters. The parameters are defined in "keyword phrases" (cf. Section 1.2) and may be specified either all at once or gradually.

For example, strategy for a linear static analysis might be specified as:

```
G I S T >   STATIC/LIN   LOAD = 1.   LIST = 0   SAVE = 1

G I S T >   REVIEW STRATEGY
      .
      .   (strategy listed)
```

or, alternatively, as:

```
G I S T >   STATIC/LIN   LOAD=1.

G I S T >   REVIEW STRAT
      .
      .   (strategy listed)

G I S T >   STATIC/LIN     LIST=0   SAVE=1

G I S T >   REV STRAT
      .
      .   (updated strategy listed)
```

where the REVIEW command is used extensively for verification.

A more informative prompt may be obtained by entering just the strategy command 'name' and 'qualifier' on the first line, e.g.,

```
Command|HELP|QUIT
G I S T >   STATIC/LINEAR
            .
            .
```

```
Keywords: LOAD  [LIST]  [SAVE]
G I S T  STATIC >   LOAD = 1.  LIST=0  SAVE=1

Command|HELP|QUIT
G I S T >
```

Notice that the keywords associated with STATIC/LIN are splashed on the screen. Then, after the user has responded, the prompt reverts back to its more basic form. Note also, that the keywords enclosed in brackets, i.e., LIST and LOAD, are not mandatory -- default settings exist for their associated parameters.

For more "on-line" information, the user will have to employ the HELP command. For example:

```
G I S T >   HELP  STATIC/LIN
```

or:

```
G I S T >   HELP  STATIC LOAD
```

or:

```
G I S T >   HELP STATIC LIST
```

and so on.

Another important feature of the strategy commands is that all parameter (i.e., keyword) definitions are saved in the "analysis database" (see Section 2.4). This means that the user has a permanent record of all previous strategies and, furthermore, may update current strategy parameters selectively.

For instance, suppose the user has already performed the linear static analysis "strategized" above, and would like to repeat it with an extended list option. The user could return, say, on another day, "log-on" under the original Jobname, REVIEW the old strategy, and then modify the appropriate keyword, i.e.,

```
G I S T >   STATIC/LIN   LIST=1
```

The new strategy would thus be completely defined and ready for analysis.

In the following paragraphs, each of the basic problem classes and corresponding strategy commands will be discussed in turn. The emphasis will be on providing some insight into the relative significance of the various strategy parameters.

STATIC/NONLINEAR Strategy

In a (STAGS) nonlinear static analysis, the time-independent, i.e., equilibrium, response of the loaded structure is computed with the possibility of accounting for "moderately" large rotations and plastic deformations. For example, such an analysis is often used to predict the collapse load of a general shell structure or to assess its postbuckling strength. There is much controversy as to when it becomes necessary to perform a dynamic analysis, when a linear analysis is sufficient, when a linearized bifurcation buckling analysis is sufficient or, generally, just what is the range of validity of a nonlinear static analysis. However, a good deal of insight regarding these questions may be grasped from Section 6 of the STAGS User's Manual ("Modeling and Strategy").

The solution to a nonlinear static problem is usually obtained in a stepwise fashion. The total load is thus applied incrementally, with stiffness updates and equilibrium iterations performed according to both the user's specifications and various built-in program criteria.

At each step, the equilibrium equations are reformulated with respect to the original (reference) configuration, and a complete solution is sought. This approach is sometimes referred to in the literature as a "Total Lagrangian" implementation.

Each converged solution on the load-displacement path will be referred to as a "load step", or more generally, as a "solution step". The "stepsize" is then defined to be the load increment between two successive solution steps. (This same terminology may be applied to dynamic analysis by replacing the word "load" with "time".)

In static analysis, there are potentially 2 stepsizes; one for "load system A" and one for "load system B" (corresponding to the first 2 GIFTS standard load cases; cf. Section 2.1.2). The specification of each stepsize is made in terms of an amplitude or "load factor". The following two equations should clarify these notions:

$$\{F\} = \{Fa\}ref * pa + \{Fb\}ref * pb \qquad <1>$$

$$\{F\}inc = \{Fa\}ref * incpa + \{Fb\}ref * incpb \qquad <2>$$

where:

{F} ..... Combined structural load vector at a given point on the load path.

{Fa}ref ... Reference load vector defined as load system A (usually equivalent to "loading case" 1).

{Fb}ref ... Reference load vector defined as load system B (usually equivalent to "loading case" 2).

pa/b ..... Load factors for load systems A and B at given point on the load path.

incpa/b ... Specified increments in load factors for load systems A and B. These are the "stepsizes".

{F}inc .... Combined "incremental" structural load vector

Since a single set of stepsizes ('incpa' and 'incpb') and a single stiffness-updating/iteration strategy is not usually sufficient to obtain an accurate solution for the total load, a series of such strategies may be employed. Thus, the total load path may be computed in a series of "solution intervals", each of which corresponds to a separate solution strategy, and represents a separate execution by the STAGS Analyzer. The number of expected solution steps in a particular solution interval is governed by the strategy specification for that interval.

A continuation from one interval to the next (or "restart", to use the historical term) usually involves a certain amount of solution evaluation, or postprocessing, followed by a suitable change in strategy. The "smoothness" of the continuation will depend on the number of previous solution steps which have been saved and whether or not "extrapolation" has been selected. The GIST command language and database system make this process relatively automatic.

The strategy command which is used to plan a nonlinear static solution interval is:

STATIC [/NONLINEAR]

where the brackets indicate that the qualifier /NONLINEAR is optional, i.e., nonlinearity is the default mode. The following keywords represent the various parameters which may be set with this command:

INTERVAL STEPSIZE NEWTON

[START MAXCUT MAXERR EXTRAP LIST SAVE]

where the keywords in brackets are optional in the sense that default settings are provided for the corresponding parameters. The first three keywords, INTERVAL, STEPSIZE and NEWTON are essential and must be set by the user "at some time".

We will now discuss these keywords, leaving most of the details to Section 1.5.

INTERVAL is used to define the forthcoming solution interval in terms of load factors, i.e.,

INTERVAL = pamin,pamax [pbmin,pbmax]

specifies the starting and stopping load factors for systems A and B, respectively. At the start of the first solution interval either 'pamin' or 'pbmin' should be greater than zero (load factors must always be positive). Ideally, these initial load factors are selected so that the very first solution step is practically linear in behavior; but since this is not always easy to assess, it may be necessary to repeat the first interval with progressively smaller initial load factors.

The parameters 'pamax' and 'pbmax' represent the load levels sought at the end of the solution interval. Keep in mind that these are just target values, and will be attained only if there are no convergence (or execution) problems.

For all subsequent intervals beyond the first, the starting load factors, 'pamin' and 'pbmin' should correspond to a previously computed (and saved) solution step. For example, the following two consecutive strategy specifications:

```
G I S T >   STATIC INTERVAL = .1, .5
G I S T >   COMPUTE                    . (invokes Analyzer)
G I S T >   STATIC INTERVAL = .5, .7
```

represent an ideally smooth continuation from one solution interval to the next.

STEPSIZE is used to specify the "load steps", i.e., load factor increments, to be used while traversing the forthcoming solution interval. The specification is of the form:

```
STEPSIZE = incpa [,incpb]
```

where 'incpa' and 'incpb' are defined in equation <2>. Again, these are target values and may be adjusted by the program if convergence is either too slow or too fast.

(NOTE: The program will not reduce the stepsize unless permitted via the MAXCUT keyword.)

If the stepsizes happen to remain constant throughout the solution, then the following relationship applies:

$$\{F\}_i = \{Fa\}_{ref} * (pamin + (i-1)*incpa) + \{Fb\}_{ref} * (pbmin + (i-1)*incpb)$$

where "i" is the solution "step number", an index which starts at 1, and runs consecutively through all solution intervals. (The linear solution, which is computed as an initial guess to solution step 1, is referred to as solution step 0.)

The reason for allowing a different stepsize for each load system is to enable nonproportional loading. For example, a cylindrical

structure submerged under hydrostatic pressure (e.g., load system A) might be subjected to an independently applied axial force (e.g., load system B). In such a case, load system 'A' would probably be applied first, incrementally, until the total pressure was achieved; then load system 'B' would be applied, also incrementally, with system 'A' held fixed, i.e., with 'incpa' = 0.

While stepsizes can often be rather large initially, they will usually require cutting as the load approaches a "critical" value, e.g., a bifurcation point. The optimal stepsize will not only depend on the physics, but also on various other strategy parameters such as the frequency of stiffness updates (NEWTON keyword) and the allowable convergence error (MAXERR keyword).

NEWTON is used to specify the frequency of stiffness updates in the forthcoming solution interval. The name derives from the fact that the user is actually specifying the form of "Newton linearization algorithm" to be used. For example:

```
NEWTON = TRUE
```

.end lilteral
refers to the familiar "true Newton" algorithm in which the tangent stiffness matrix is formed and refactored upon every iteration of every solution step. This is a very expensive way to travel and only
recommended for extremely unwieldly problems. A more practical method, the "modified Newton" algorithm, may be selected via:
.literal

```
NEWTON = MOD, period
```

where 'period' is the number of solution steps between successive refactorings. For example, the so-called "one-step Newton" is obtained via:

```
NEWTON = MOD,1
```

In this case the stiffness is updated (i.e., formed and factored) at the beginning of every solution step. Note that no additional stiffness updates are performed during equiliibrium iterations; only the right-hand-side (i.e., the internal force vector) is updated each iteration.

A slight variation on the modified Newton method is what we shall call the "selective Newton" and is specified by:

```
NEWTON = SEL,max
```

where 'max' is the maximum number of stiffness updates permitted during the solution interval. In this case, the program decides when to refactor, (based on the observed convergence behavior and other available options) and does so until it reaches the specified

limit. Again, refactorings will be made only at the beginning of a prospective solution step.

At the lower extreme, the phrase:

NEWTON = FALSE

may be used to suppress all stiffness updating except upon restart. (Presently, one stiffness update is performed at the beginning of every solution interval, regardless of the NEWTON setting.)

The optimal frequency of stiffness updates will be influenced by the stepsize, the convergence error tolerance and just how rapidly the load-deflection "slope" is actually changing. Another aspect which must be "factored" in to the user's strategy is the computational expense of stiffness formation + assembly + factorization, i.e., the "size" of the problem.

The remaining STATIC/NONLINEAR keywords are optional [as indicated]. They can, however, be extremely useful at times and hence warrant some discussion.

START is used to tell the Analyzer how to begin, or "restart" the next solution interval. The default setting is:

START = NEXT

which implies that the solution is to continue from where it left off, employing the last computed solution step as the starting approximation (possibly with extrapolation) for the first solution step of the new interval.

To make this clearer; consider the following scenario: Suppose you have successfully completed a solution interval, and the last solution step saved in the database is step number 4. Now, you issue the GIST command:

G I S T > REVIEW ANALYSIS

and display the corresponding load factors, i.e.,

```
    .
    .
Current Analysis:  Static/Nonlinear
Current Step    :      4
Current Loads   :  PA = 1.5,  PB = 0
    .
    .
```

Now, to advance the solution, the next specified interval must begin with these load factors. Hence the new strategy might be:

```
G I S T >  STATIC  INTERVAL=1.5, 2.0  STEP=.25
```

Then, by default, solution step number 4 would become the starting step for the new interval. If, on the other hand, the solution at step 4 did not instill confidence (say, you had inadvertently passed a bifurcation point) and you would like to repeat it with a smaller stepsize, you might employ a strategy such as:

```
G I S T >  STATIC  INTERVAL=1.,2.  STEP=.1  START=FROM 3
```

where, in this case, the program is forced to restart from solution step 3, which is presumabley is in the database and corresponds to load factor PA = 1.

Hence, by using the starting option:

```
START = FROM step
```

the user may redo any solution interval for which the necessary starting solution steps are available.

The keyword, MAXCUT, is used to allow the program to cut, i.e., halve, the stepsize(s) when convergence difficulties arise. It specifies the maximum number of allowable cuts. The default option is:

```
MAXCUT = 0
```

which does not allow any cuts to be made during the interval.

NOTE: The program may increase the stepsize(s) if convergence occurs too easily. (For instance, if only 1 iteration is required for convergence at two consecutive solution steps.)

MAXERR is used to change the maximum allowable value for the equilibrium-iteration convergence-error. The default setting is:

```
MAXERR=.001
```

which means that the program will continue to iterate (generally, up to 12 times) until the relative error in both displacement and force vector norms is less than one-tenth of a percent.

NOTE: By setting MAXERR to some abnormally high value, e.g., 1., the program will be inhibited from performing iterations. If, in addition, NEWTON = MOD,1, the resulting algorithm is then equivalent to the classical "incremental" approach to nonlinear equation solving.

EXTRAP is used to turn the solution extrapolation switch on or off. By default, up to quadratic extrapolation is used at the beginning of each solution step. That is, the previous 3 solution steps are employed to obtain a first approximation of the displacement vector for the new step. However this is not always desirable (or possible, and so to inhibit extrapolation, the user may employ the phrase:

EXTRAP=OFF

which causes the previous solution step to be directly used as the initial iterate for the new solution step.

Finally, we arrive at the LIST and SAVE keywords, which are common to all of the strategy commands. They are used to control the flow of output from the STAGS Analyzer. LIST controls the amount of data to be printed, while SAVE controls the amount of data to be archived in the analysis database (see Section 2.4). The general keyword-phrases are of the form:

LIST [/Item] = period

SAVE [/Item] = period

where 'period' represents the number of solution steps between printing or saving, respectively (the value 0 turns the LIST or SAVE option off completely).

For example, the default settings are:

LIST = 0 SAVE/D = 1 SAVE/S = 1

which suppresses all solution data printout, but saves displacements and stresses (plus stress resultants and strains) at every solution step. This is probably the most convenient mode of operation (disk-space permitting) since: (1) all results can then be displayed at the analyst's leisure via GIFTS interactive postprocessing, and (2) by saving all solutions, the analyst achieves maximum "restart" flexibility.

NOTE: Excess solution data can later be discarded from the database as described in Section 2.4.

With LIST=0, the printout emanating from the STAGS Analyzer will be limited to the computational statistics of the solution process, e.g., the number of iterations/refactorings actually performed at each solution step. This is just as well, since the STAGS printout of displacements, stresses, etc. will not in general follow the same node and element ordering as the GIFTS pre- and post-processors, and may therefore lead to confusion. (If you do use a nonzero LIST option, employ Appendix A as an interpreter.)

STATIC/LINEAR Strategy

There is really little strategy to speak of when it comes to linear static analysis. The only pertinent command keywords are:

LOAD [LIST] [SAVE]

where LIST and SAVE are practically as defined for nonlinear static analysis, and LOAD simply defines the load factors which are to be applied to load systems A and B, i.e.,

LOAD = pa [,pb]

Instead of combining the 2 load systems as in nonlinear static analysis, they are kept separate, and 2 independent solutions are obtained, the first corresponding to the applied load vector:

{Fa} = pa * {Fa}ref

and the second to:

{Fb} = pb * {Fb}ref

where {Fa}ref and {Fb}ref represent the load vectors corresponding to load systems A and B, as defined during pre-processing (e.g., GIFTS load cases 1 and 2).

NOTE:

The user is reminded that, strictly speaking, LINEAR static response analysis is valid only for problems characterized by "small", elastic deformations.

DYNAMIC/NONLINEAR Strategy

A dynamic response analysis is usually performed to account for the effects of rapidly varying loads and/or initial conditions. The resulting equations of motion now include structural inertia and damping oporators and hence require a temporal discretization algorithm in addition to the finite element spatial discretization. The STAGS Analyzer employs a variety of "direct" numerical integration schemes to discretize the time domain, and thereby "march" the solution through a sequence of "time steps" (same as "solution steps").

If the problem contains nonlinearity, either geometric or material, additional stiffness updates (i.e., refactorings) and/or iteration may be required at each step. If the problem is purely linear, the solution is then achieved with neither of the above. Due to the potentially high cost of a nonlinear dynamic response analysis, it should be approached with both caution and understanding (cf. [S1]).

Basically, the process of conducting a nonlinear dynamic analysis is similar to that outlined for nonlinear static analysis, except that instead of incrementing load factors, we increment the time, with load factors following according to some prescribed historical function. In addition, dynamic solution strategy requires the selection of a "time integrator" and involves a few extra options which have no counterpart in static analysis.

Like nonlinear static analysis, dynamic analysis is usually performed as a series of "solution intervals" (i.e., STAGS executions), each of which may contain many solution steps.

Nonlinear dynamic solution intervals are prescribed with the following strategy command.

DYNAMIC [/NONLINEAR]

which features the following set of "keywords" for parameter definition:

INTERVAL STEPSIZE METHOD HISTORY NEWTON

[START DAMPING MINDTS MAXDTS MAXERR LIST SAVE]

As usual, the unbracketed keywords require user specification while the bracketed ones have default settings and are therefore optional. Since many of the basic notions associated with these keywords have already been discussed under "STATIC/NONLINEAR Strategy", we will restrict our comments here to those features which are unique to dynamic analysis.

INTERVAL and STEPSIZE are used in the same way as in nonlinear static analysis strategy except that the parameters refer to time

interval (tmin, tmax) and time step (dt), respectively, rather than to load interval and load step. The first solution interval is usually kicked-off at 'tmin=0' and may employ initial displacements and/or velocities (defined during pre-processing). The selection of the stepsize, dt, is especially crucial in dynamic analysis with respect to stability and accuracy considerations. Also, as suggested in the STAGS Manual [S1], a preliminary vibration analysis may be required just to establish a "ballpark" time step.

A "key" keyword for dynamic strategy is METHOD, which declares the time integration algorithm. Both the time stepsize and the nonlinear solution strategy (see NEWTON) below) will depend strongly on the choice of time integrator. The methods currently available are:

METHOD=EXPLICIT

which corresponds to the explicit (2nd order) control difference algorithm,

METHOD=TRAPEZOIDAL

which corresponds to the implicit (2nd order) trapezoidal rule

METHOD=GEAR

which corresponds to the implicit (2nd order accurate) Gear algorithm and:

METHOD=PARK

which corresponds to the implicit (2nd order accurate) Park algorithm.

The explicit method, by definition, requires no matrix factorizations but may require prohibitively small stepsizes to maintain stability. On the other hand, the implicit methods require at least one matrix factorization, but permit exceptionally large stepsizes.

As can be established from the literature, each of the above methods may be considered superior for a particular class of problems. For example, in "structural dynamics" (i.e., low-frequency response) situations, the PARK method is often recommended for nonlinear systems, while the TRAPezoidal rule rule is hard to beat for linear systems. Consule [S1] and [S2] for additional guidance and references.

The NEWTON keyword, which controls the frequency of tangent "stiffness" updating has the same options and implications for nonlinear dynamic analysis as it does for nonlinear static analysis;

with the following qualifications. First, if the time integration algorithm (selected via METHOD) is explicit, then NEWTON becomes irrelevant, since there is no tangent "stiffness" matrix in the problem to update and factorize. Second, with regard to implicit methods, since there are typically many more solution steps in dynamic analysis than in static analysis, frequent matrix updating may be cost-prohibitive, and, for a sufficiently small time step, even unnecessary.

As mentioned earlier, the load "history" is part of the dynamic strategy prescription. The reference load systems (A and B) have already been defined during pre-processing. What needs to be specified here is the variation of their respective amplitudes, i.e., "load factors", with time.

The keyword provided for this purpose is called HISTORY and is specified as follows:

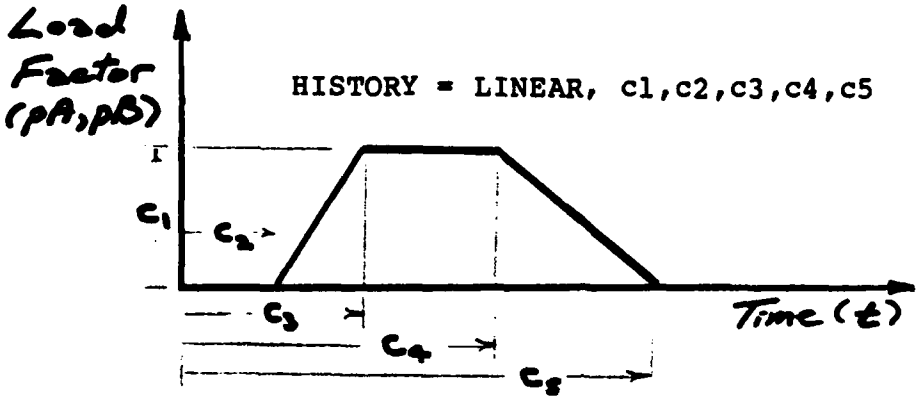
HISTORY [/System] = Shape, coefficents

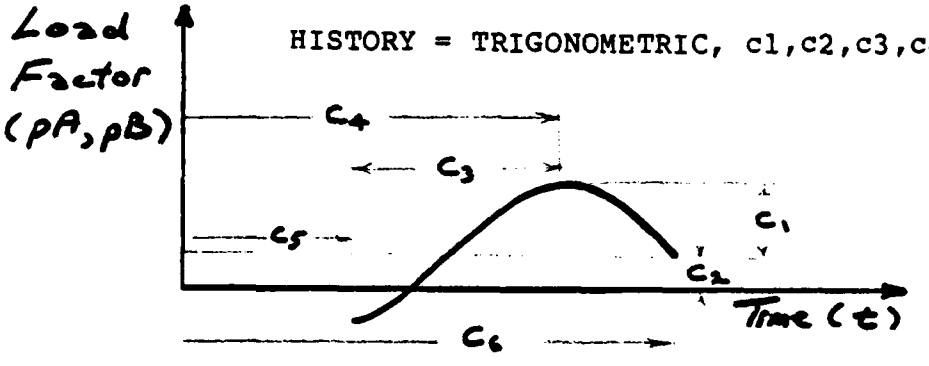
where 'Load-System' may be either 'A' or 'B', 'Shape' may be 'LIN' (linear), 'EXPO' (exponential) or 'TRIG' (trigonometric), and 'coefficients' represents a set of up to 6 constants which correspond to 'Shape' and together with it fully characterize the load factor vs. time function for the indicated load system.

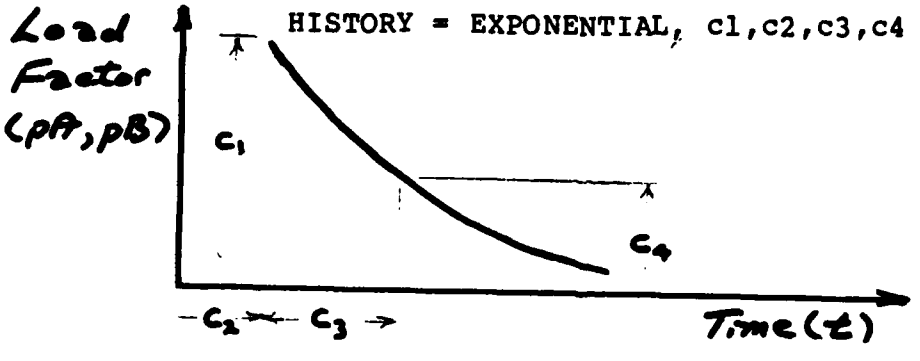
where 'System' refers to the load system and may be either A or B; 'Shape' refers to the type of load-time function and may be LINEAR, EXPONENTIAL, or TRIGONOMETRIC; and 'coefficients' is a set of values which, together with 'Shape', fully characterize the load history.

The illustrations on the following page define the set of 'coefficients' required for each of the allowable load history 'Shape's:

Load HISTORY Parameters







Remark

Admittedly, the current mechanism for defining the load history is neither convenient, nor sufficiently general. A better procedure is not hard to imagine. For example, a graphically defined load-time function capability could be introduced in one of the GIFTS preprocessors. We hope to incorporate such improvements in future versions of the system.

The next DYNAMIC keyword is START. The description is identical to the STATIC/NONLINEAR case except that a 'solution step' now refers to a 'time step', rather than a 'load step'. Note that a re-START may be accomplished from any previous solution step, providing it has been saved in the database (see SAVE). However, if an 'm-step' METHOD is being employed for time integration, a "smooth" continuation will then be possible only if 'm' consecutive previous solution steps are available.

For example, to pick up smoothly from step number 10 using PARK's (3-step) method would require that solution steps 8, 9 and 10 all be present in the database. Under normal circumstances this will be achieved automatically via the SAVE keyword (see below).

The next three (optional) keywords, DAMPING, MAXDTS and MINDTS, are unique to DYNAMIC strategy. They are pretty well documented in Section 1.5; however, a few additional comments are in order:

First, the parameter 'gamma' defined with the DAMPING keyword is presently inactive. It is supposed to multiply a user-defined diagonal damping matrix. However, there is no current mechanism for the user to define such a matrix.

Regarding MINDTS and MAXDTS: The former is analogous to the STATIC MAXCUT keyword except that instead of specifying the maximum number of stepsize reductions, MINDTS specifies the minimum allowable time step scale-factor (between 0 and 1). The latter keyword, MAXDTS, specifies the maximum stepsize scale-factor (greater than or equal to 1). Its presence is basically to enable the user to inhibit automatic time-step increases, which might otherwise destroy the required resolution of the response (or load history). Note that by setting both MINDTS and MAXDTS to 1, a constant time-step is assured.

Finally, we arrive again at the LIST and SAVE keywords (see STATIC /NONLINEAR Strategy). The only additional comment required here is that the 'D' qualifier, as in:

```
SAVE/D=period
```

refers not only to displacement vectors, but also includes velocity vectors, acceleration vectors and <u>whatever else is necessary for subsequent restarts</u>.

As a closing remark on the use of the DYNAMIC [/NONLINEAR] strategy command, we note that a dynamic solution interval may be initiated from a previously computed static solution. This will occur automatically if the step number referenced with the START keyword happens to correspond to a static solution step. The static solution will then represent the "initial conditions" for the dynamic analysis.

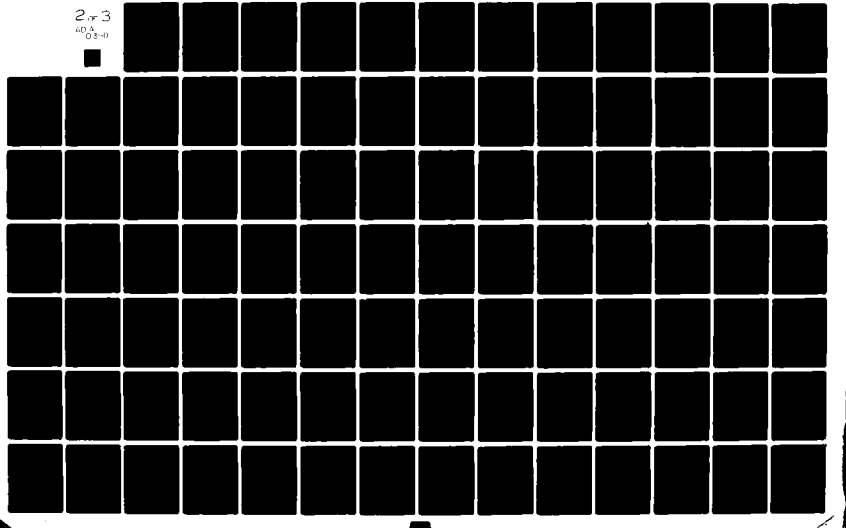
DYNAMIC/LINEAR Strategy

A linear dynamic response analysis is performed in the same manner as a nonlinear dynamic response analysis, except that the linearity is exploited by precluding any stiffness updates or iterations between time steps.

Hence, the keywords used to set the /LINEAR Strategy are identical to the /NONLINEAR keywords with the exception of NEWTON and MAXERR, which have no relevance to linear dynamics.

Another distinguishing feature, of course, is that the selection of a time integration METHOD may be quite different for linear and nonlinear problems. Once again the reader is advised to consult [S1], [S2], and references contained therein.

AD-A103 801 LOCKHEED MISSILES AND SPACE CO INC PALO ALTO CA PALO --ETC F/G 13/13
INTERACTIVE NONLINEAR STRUCTURAL ANALYSIS: ENHANCEMENT.(U)
JUL 81 G M STANLEY N00014-80-C-0831
UNCLASSIFIED LMSC-D811535 NL



BUCKLING/LINEAR Strategy

In this text, we use the term "buckling analysis" loosely to refer to what is commonly called a "bifurcation buckling analysis", or alternatively a "stability analysis". In any case, what we are talking about is the generalized eigenvalue problem:

$$([K]ref + eig * [K]geom) * \{d\} = 0$$

where '[K]ref' is (in the /LINEAR case) the linear stiffness matrix; '[K]geom' is the so-called "geometric" stiffness matrix, which is a function of the pre-buckling (or "pre-stress") state; and 'eig' and '{d}' represent, respectively, one of the corresponding eigenvalues and eigenvectors of the problem. The proper interpretation of 'eig' is as a "critical" load scale-factor which, when multiplied by the load vector producing the (equilibrium) pre-buckling state yields an estimated "buckling load". Formally, the definition is:

$$\{F\}crit = eig * pa * \{Fa\}ref$$

where '{F}crit' is the critical, or buckling, load vector, '{Fa}ref' is the reference load vector defined during pre-processing as "load system A", and 'pa' is a corresponding load-factor which is defined (arbitrarily) by the user as part of the buckling strategy.

The qualifier, /LINEAR, implies that the pre-buckling configuration is to be one which is the result of a linear static analysis. The pre- buckling solution, however, is performed as an integral part of the buckling analysis, rather than a priori (e.g., in a separate STATIC/LINEAR analysis).

STAGS actually allows for an even more general form of the eigenvalue problem (again for the /LINEAR case), i.e.,

$$([K]ref + [Kb]geom + eig * [Ka]geom) * \{d\} = 0$$

where '[Ka]geom' is the geometric stiffness due to a user-scaled load- system A and '[Kb]geom' is the geometric stiffness due to a user-scaled load-system B.

The result is that the critical load vector becomes:

$$\{F\}crit = (eig * pa * \{Fa\}ref) + (pb * \{Fb\}ref)$$

which answers the question: "how much can load-system A be scaled, given a fixed load system B, before the total load becomes critical?" Such an analysis would be useful, for instance, in determining the axial buckling load of a cylindrical shell subjected to a fixed internal pressure.

The "multi-load-system" form of the problem is activated automatically by setting 'pb' greater than zero (see PRELOAD below).

With the above discussion as background, we will examine the various parameters used to define the buckling strategy. The strategy command:

BUCKLING [/LINEAR]

features the following keywords:

PRELOAD MODES [RANGE SHIFT MAXERR LIST SAVE]

with default settings provided for those in brackets.

PRELOAD specifies the linear pre-buckling load factors for load systems A and B via:

PRELOAD = pa [,pb]

The linear solutions for each of the two thusly-scaled load systems will be computed independently by the analyzer and used in the formation of the matrix eigenvalue problem presented above.

MODES specifies the maximum number of eigenvalues (and corresponding) eigenvectors desired from the analysis. Typically, only the first few ,i.e., the lowest, critical loads are of interest, but there are cases when it will be safer to request more than are actually desired -- for example, to sort out some unexpected "spurious nodes" (see [S2]). Note that the eigenvalues ('eig') are extracted in order of increasing absolute value, or distance from some specified shift value.

The next two keywords, RANGE and SHIFT, enable the analyst to focus in on a particular neighborhood where the eigenvalues are expected to lie, or cluster. By specifying:

RANGE = eigmin, eigmax

the eigenspectrum origin will be shifted from 0 to the point:

eigcen = (eigmin + eigmax)/2

and eigenvalues will then be extracted in order of increasing distance from 'eigcen'. Furthermore, 'eigmin' and 'eigmax' are treated as an upper and lower bound, so that the eigenvalue extraction process will automatically cease when all eigenvalues in the specified range have been determined. Alternatively,

SHIFT = target

also causes a direct eigenspectrum shift, i.e., to the value

'target', but sets no upper or lower bound on the eigenvalues thus obtained. Hence, the maximum number of modes extracted is then equal to the number indicated by MODES. (Note that the RANGE option supercedes the SHIFT option.)

The relative accuracy of eigenvalue computation is governed by the keyword, MAXERR, which specifies the maximum acceptable convergence error. The default setting is:

MAXERR = 1.E-5

Note that when the lowest eigenvalues have converged according to the MAXERR criterion some of the higher ones may be "practically" converged and will be available from the output listing (see the LIST keyword).

LIST and SAVE control the output emanating from the STAGS analyzer. As in other types of analysis strategy, LIST controls the direct printout from the analyzer and SAVE controls the amount of solution data to be archived in the database. If nothing is specified, the default settings are:

LIST/I = 1 LIST/D = 0

and:

SAVE/D = 1

which causes only the eigenvalues and intermediate iteration data to be printed directly, while both the eigenvalues and eigenvectors are saved in the database. The rationale is that the evaluation of displacement modes is much more effectively performed via interactive post-processing (see Section 2.3). Note also that a full STAGS displacement vector printout may require some unravelling (as described in Appendix A) and is therefore not recommended except perhaps as a diagnostic tool (or as a "security blanket").

BUCKLING/NONLINEAR Strategy

This analysis option, which is supposed to assess stability with respect to a nonlinearly stressed configuration is "presently inoperative".

VIBRATION / LINEAR Strategy

The discussion here is meant to complement the discussion of BUCKLING /LINEAR strategy. These are both structural eigenvalue analyses, but with different interpretations and restrictions. A STAGS "vibration analysis" determines the natural frequencies and corresponding displacement modes corresponding to "small" vibrations about some equilibrium reference configuration. The qualifier /LINEAR further indicates that the equilibrium configuration to be employed shall be the "unstressed" state.

Formally, the eigenvalue problem may be expressed as:

$$([K]ref - eig * [M]) * \{d\} = 0$$

where [K]ref is the linear stiffness matrix; [M] is the mass matrix; {d} represents a displacement mode vector, and 'eig', a corresponding eigenvalue. The eigenvalue is "physically" defined by:

$$eig = (2*pi*f) **2$$

where 'f' is the natural frequency in cycles per second (cps). Note that when preparing vibration strategy, the analyst should refer to 'f' as if it were the eigenvalue.

The vibration strategy command:

VIBRATION [/LINEAR]

features the same keywords as BUCKLING/LINEAR except that the BUCKLING keyword, PRELOAD, is absent. The list is therefore:

MODES [RANGE SHIFT MAXERR LIST SAVE]

The absence of PRELOAD simply reflects the fact that no pre-stress state is required for a "linear" vibration analysis. In fact, any load systems defined during pre-processing will just be ignored.

For a discussion of the remaining keywords, refer to the paragraphs under BUCKLING/LINEAR Strategy with the following qualifications:

First, as mentioned above, the eigenvalue parameter, 'eig', in the buckling discussion translates as 'f' (natural frequency in cps) when discussing vibration analysis. Note that the natural frequencies are output directly from the analyzer and thus require no further interpretation.

Second, the eigenvalue shift induced by the RANGE keyword is no longer to the center of the interval [eigmin,eigmax], but instead to the point:

$$fshift = SQRT ((fmin**2 + fmax**2)/2)$$

where 'fmin' and 'fmax' are specified bounds on the natural

frequency (in cps).

VIBRATION/NONLINEAR Strategy

This analysis option, which is supposed to determine natural frequencies and corresponding "small" vibration modes with respect to a nonlinearly stressed configuration is "presently inoperative".

2.2.3 ANALYSIS COMPUTATION

As soon as the strategy for the forthcoming solution interval has been completely prescribed and verified (as discussed in the previous subsection), the analyst may invoke the STAGS analyzer to act on that strategy via:

G I S T > COMPUTE

Interactive Computation

The above GIST command, by default, will lead directly to on-line computation by the analyzer. This is the most "interactive" mode of operation, since the solution interval is then monitored directly from the screen and may be advanced or revised without delay.

Batch Computation For large-scale problems, on-line computation may be undesirable, due to either the time or cost involved. In such cases, the solution interval may be processed in a "batch" mode, via:

G I S T > COMPUTE/BATCH Parameters...

With this form of the command, an operating system procedure to perform the desired computation is automatically prepared and submitted, as a "batch run", to a "batch queue" where it will await execution. The analyst may then continue interacting (on a restricted basis) with GIST while the solution interval is being processed independently.

In the present implementation of the network, the "analysis" database automatically "locks" to the interactive user when a GIST batch run is in progress. This means that the user will be restricted from checking analysis status, defining or reviewing strategy, or processing any new solutions until the batch run has terminated. However, the user may continue to postprocess "old" results even during a batch solution interval (see Section 2.3).

Another condition for batch processing is that the USER MUST MONITOR THE STATUS OF A GIST BATCH-RUN MANUALLY, i.e., through the computer operating system. When the COMPUTE/BATCH command is issued, a "batch run identification number" (or "run ID") should appear on the screen. To find out where the corresponding batch run is, (e.g., waiting to execute, executing, waiting to print or finished) the user must exit from GIST and make the inquiry directly to the computer operating system, employing the "run ID" for proper identification. (The form of this request will vary from computer to computer.) Finally, when the user is sure that the batch job has finished, the GIST command:

G I S T > CLEAR BATCH

may be issued to "unlock" the analysis database for interactive processing. (The whole procedure isn't as cumbersome as it sounds. The excuse for not synchronizing batch and interactive operations automatically is the tremendous degree of machine-dependence currently associated with such a task.)

Computer Output

Printed output from the STAGS analyzer is controlled by setting the LIST keyword under any of the strategy commands discussed in Section 2.2.2. By default, all physical results, e.g., displacements and stresses, are suppressed from direct printout, and only a summary of computational statistics is displayed as the solution progresses.

By saving the physical results in the database at sufficiently frequent intervals (via the strategy keyword: SAVE), they may then be viewed at the analyst's leisure, i.e., in a relaxed, interactive-graphics postprocessing mode (see Section 2.3). Hence, there is usually no need for extensive analyzer-printout. However, if such a need arises, the output may be disposed to a permanent file (rather than say directly to the screen or the printer) by employing the OUTPUT keyword, e.g.,

```
G I S T >   COMPUTE  OUTPUT = Filename
```

or:

```
G I S T >   COMPUTE/BATCH  OUTPUT = Filename...
```

Computational "Review"

While between solution intervals and before actually getting into a formal postprocessing phase, the user may monitor the "status" of the solution by employing the GIST 'REVIEW' command. For example,

```
G I S T >   REVIEW ANALYSIS
```

will display the current analysis type, solution step number, and corresponding time/load-factors. For an informative, chronological look at what has been accumulated in the database, simply add the /TOC qualifier, e.g.,

```
G I S T >   REVIEW/TOC  ANALYSIS
```

which is elucidated upon in Section 2.4. Finally, computational statistics may be reviewed via:

```
G I S T >   REVIEW SOLUTION = steps
```

where 'Job' is symbolic for the actual Jobname.

Note that the above files must be maintained on disk throughout the analysis as they are required for both interactive pre- and post-processing.

The data within these files is stored and manipulated automatically by the GIFTS processors, i.e., BULKM, EDITM, BULKLB and EDITLB. However, for a detailed description of the data structures and access method employed in GIFTS data management, the interested reader is referred to the GIFTS Systems Manual (Section II: The Unified Database). Such knowledge is usually only required when developing interface software or for debugging system errors.

The Analysis Database

The analysis database consists of 1 general purpose file which contains a number of named "data-sets". (This type of file is also referred to as a "data library", or a DAL file [N3].) The file name is of the form:

Job .STG

where the extension, STG, is an abbreviation for STAGS. The following data-sets are (potentially) stored in the STG file:

File: 'Jobname'.STG

Data-set Name	Contents
STAG.STAT. 0. 0	STAGS problem status parameters
STAG.MATL. 0. 0	Material properties
STAG.FABR.type. 0	Fabrication properties (1D/2D)
STAG.CONF.unit. 0	Structural configuration (reference)
STAG.LOAD.syst. 0	Load system 'syst' vectors
STAG.DOFS.vers. 0	Degree of freedom map
STAG.ICON.type. 0	Initial conditions ('type'=0: displ; 'type'=1: veloc)
STAG.STRA.vers. 0	Solution strategy parameters
STAG.SOLN. 0.step	Solution statistics for step 'step'
STAG.DISP. 0.step	Displacement vector at step 'step'
STAG.VELO. 0.step	Velocity vector " " "
STAG.STRE. 0.step	Stresses, strains, etc. " " "
STAG.PLAS. 0.step	Plasticity data " " "
STAG.AUXI. 0.step	Dynamic auxiliary vectors " " "
STAG.MODE.mode.step	Mode vector no. 'mode' " " "
STAG.FORC. 0.step	Internal force vector " " "

The STG file is created by the STAGS Preanalyzer (STAGS1) and extended during the analysis by the STAGS Analyzer (STAGS2). The data-sets created by the Pre-analyzer (i.e., STAT through ICON) are stored automatically. The solution data-sets generated by the Analyzer, however, e.g., DISP, VELO, STRE, and so forth, may be selectively archived as explained under Strategy in Section 2.2 (cf. the SAVE keyword).

The user is reminded that a reasonable frequency of solution data-archival is required for satisfactory postprocessing and restart capability. Since this particular file has a tendency to grow without bound (especially in a nonlinear or dynamic response analysis) it may be necessary to compress it every so often, as described in Subsection 2.4.4.

A detailed description of the STG file (for software afficianados) may be found in Appendix B of this tutorial.

In addition to the STG file, the analysis database may (i.e., for most implementations of the system) contain a special file which is used as an intermediary between the Pre-Analyzer and the Analyzer (alias STAGS1 and STAGS2). This file will appear with the name:

Job .SIN

Although it may initially seem quite substantial in size with respect to the STG file, the SIN file, once created by the Pre-Analyzer, will never expand. It contains a computationally primed version of the element kinematic and constitutive data (as discussed in Subsection 2.2.1) which is utilized by the Analyzer at the start of each solution interval. Furthermore, if the file is destroyed (or misplaced) it can always by regenerated simply by rerunning the Pre-Analyzer via

The SETUP command. (Note: The SIN file will not appear in implementations of the system which "hard-wire" the Pre-Analyzer and the Analyzer together as one COMPUTE operation.)

The Post-processing Database

The post-processing database consists of a set of transient files which are used exclusively by the GIFTS result display module (RESULT). The potential members of this database are:

File Name	Contents
Job .DNS	Displacement-type vectors corresponding to a sequence of solution steps.
Job .STR	Stress-type data corresponding to the same sequence of solution steps as in .DNS.

The DNS and STR files are created at the time RESULT is invoked from the GIST Control Module. As explained in Section 2.3, the STAGS->GIFTS Adaptor (S2G) is at that time employed (by default) to transfer a prescribed set of solutions from the analysis database to the post- processing database in an appropriate GIFTS format. The post-processing files are re-created upon each new RESULT session (unless the /OLD qualifier is used) and the analysis database is left unperturbed. Hence, the post-processing database may be viewed simply as a movable window to the analysis database.

Note that the DNS and STR files are not in themselves sufficient for GIFTS postprocessing; the pre-processing database must be present as well to maintain the definition of the basic model.

The formal description of the internal data structures for DNS and STR, though not usually required by the user, per se, is available in the GIFTS System Manual (under "The Unified Database").

Remark

The actual form of the file names appearing in the Job database will depend, of course, on operating system conventions. For example, if the Jobname is 'TEST', the Job database would consist of files such as TEST.PAR, TEST.PTS, TEST.STG, etc. on a VAX/VMS installation; whereas the same files would be named TESTPAR, TESTPTS, TESTSTG, etc. on a CDC/NOS installation.

2.4.2 MONITORING THE DATABASE; THE 'REVIEW/TOC' COMMAND

By employing the GIST REVIEW command with the /TOC qualifier, the user may list, in various ways, the current database "table-of-contents".

For example, by entering:

```
G I S T >   REVIEW/TOC   JOB
```

the Job database is summarized by listing all files currently assigned to the Job.

Similarly:

```
G I S T >   REVIEW/TOC   PREP
```

lists those files corresponding to the pre-processing database, and:

```
G I S T >   REVIEW/TOC   POST
```

lists those files corresponding to the postprocessing database.

A somewhat more detailed listing is available for the analysis database, via:

```
G I S T >   REVIEW/TOC   ANALYSIS
```

which produces a tabulated listing of all data-sets currently residing in the 'Job'.STG file. The analysis database "table-of-contents" provides, to some extent, an historical progress record. All successfully archived solution data-sets are listed in chronological order, and feature such statistics as size, date and time of creation. Thus, the user may verify (at a glance) what is available for postprocessing and/or solution continuation, and what is not. More specific information at a particular solution step can then be obtained via the REVIEW SOLUTION form of the command, as described in Subsection 2.3.2.

Remark

The REVIEW command provides a convenient mechanism for monitoring the accumulation of data during an analysis. To examine the database in any greater detail, or to "control" the propagation of data, the user will have to use one of the interactive database managers described in Subsection 2.4.3.

2.4.3 CLEANING-UP AFTER YOURSELF; THE 'CLEAR' COMMAND

Since the database can require (at times) a substantial disk allocation, the user might be interested in a "magic button" which will make it all disappear, e.g., at the end of an analysis, or perhaps after a false start. Such a button is to be found in the GIST' CLEAR command, when used as follows:

CLEAR JOB will eliminate the entire Job database, i.e., all files generated under the current Jobname. The list of potential files was given in Subsection 2.4.1.

CLEAR PREPROCESSING will eliminate only those files associated with the pre-processing database. This is not a recommended option, as the analysis and postprocessing databases (if they exist) will be rendered useless for subsequent interactive pre-/postprocessing.

CLEAR ANALYSIS will elimnate the analysis database associated with the current Job, i.e., the STG (and SIN) file(s). This option might be used, for instance, to re-initiate the entire solution process without forfeiting the model definition.

CLEAR POSTPROCESSING will eliminate the postprocessing database associated with the current Job. This is a very safe option (although rarely necessary) since the postprocessing database contains only "reformatted copies" of various solutions from the analysis database. Hence, as long as the comprehensive analysis database is available, the post-processing database can always be regenerated. (In fact, the post-processing database is intended to have a rapid turnover; it is automatically destroyed and recreated each time the GIST command: RESULT[/NEW] is succesfully employed.)

Remark

Any of the database files may also be deleted (or itemized) externally, i.e., by employing the computer operating system, which usually has a set of commands for this purpose. However, the GIST 'CLEAR' command is usually more convenient and less hazardous, since the user is spared from having to refer to each of the individual file names.

2.4.4 "EDITING" THE DATABASE; THE 'MANAGE' COMMAND

The GIST network provides a couple of interactive database managers which may be employed by the user to effectively "edit" the database (if the need arises). By "edit", we refer to such operations as disabling and enabling individual data-sets, data compression (e.g., removing unwanted data-sets to drastically reduce file size), copying data-sets from one file to another, and obtaining detailed data-listing (or "dump") capability.

These database "editors" are invoked by using the GIST command:

MANAGE { GIFTS | STAGS }

where the options, GIFTS and STAGS, each lead to an independent processor as will now be explained.

MANAGE GIFTS invokes an interactive processor called 'DUMP' which may be employed to list in detail, or "dump", any file in the pre- and postprocessing databases. The DUMP processor may be considered a member of the GIFTS family (as are the pre- and post- processing files) and consequently must be driven via the GIFTS command language. However, the relevant GIFTS commands are easy to remember. For example: 'DMPPAR' will dump the contents of the PAR file, 'DMPPTS' will dump the contents of the PTS file, etc. Furthermore, the GIFTS 'LPON' command may be used to divert the data listings to the line-printer.

For more information on this processor, the user is referred to [G1,G4]. Note that DUMP is primarily intended for software development and debugging purposes, but the user/analyst should at least know of its existence.

MANAGE STAGS will invoke an interactive processor called 'CLAUDE' which may be employed to do just about anything with the analysis database. CLAUDE is truly a database editor, in the sense described at the beginning of this subsection. It is designed to operate on DAL-type files [N1], of which the STAGS-generated STG file is one. Besides providing extremely selective data-listing options, its most important capability (in the context of GIST) is data-set manipulation. For example, the user may employ CLAUDE to create a new, reduced analysis database (i.e., STG file) from one which has become unnecessarily bloated with solution steps. This is performed by selectively copying data-sets from the old file to the new file. Hence, it should only be performed by users who are well aware of the significance of the various STG data-sets (see Appendix B). In other words, CLAUDE is not for the casual user.

Regarding the operation of CLAUDE, i.e., learning and understanding its command language, a comprehensive manual should soon be

available. In the meantime, there is considerable on-line HELP documentation available. Just follow the sequence:

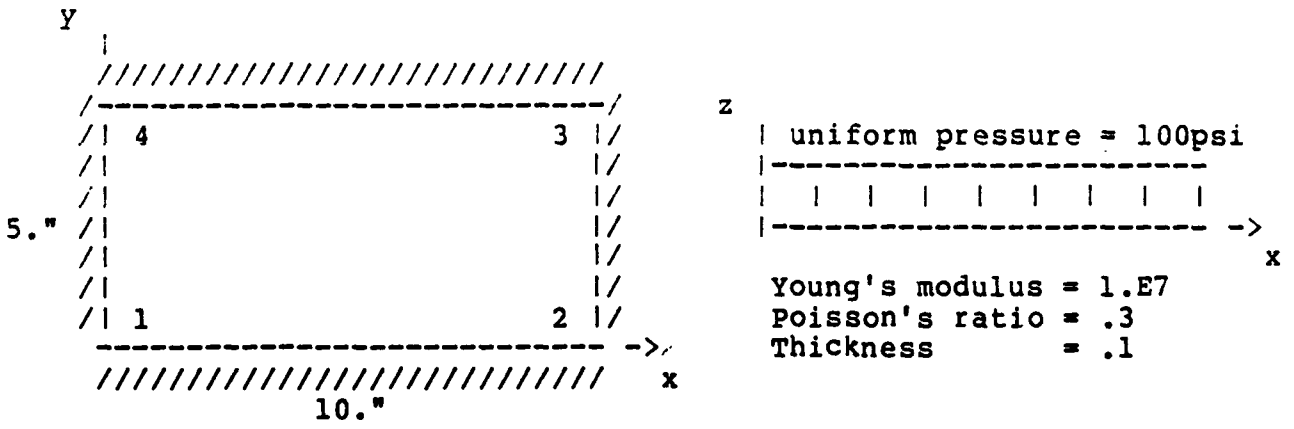
```
G I S T >   MANAGE STAGS
          .
          .
          .
CLAUDE >    HELP
          .
          .
          .
```

to its logical conclusion.

2.5 INTERACTIVE SAMPLE CASES

In this section, we present the complete interactive dialogue for a set of very simple structural analyses performed with the GIST system. These "sample cases" are intended for two purposes: (1) to give the prospective user a glimpse of the actual system in operation, while filling in on many of the details suggested in previous sections, and (2) to serve as a "check-out" procedure for new installations of the system, i.e., by having the user enter all of the indicated commands and verify all of the corresponding program responses, before going on to perform more meaningful analyses.

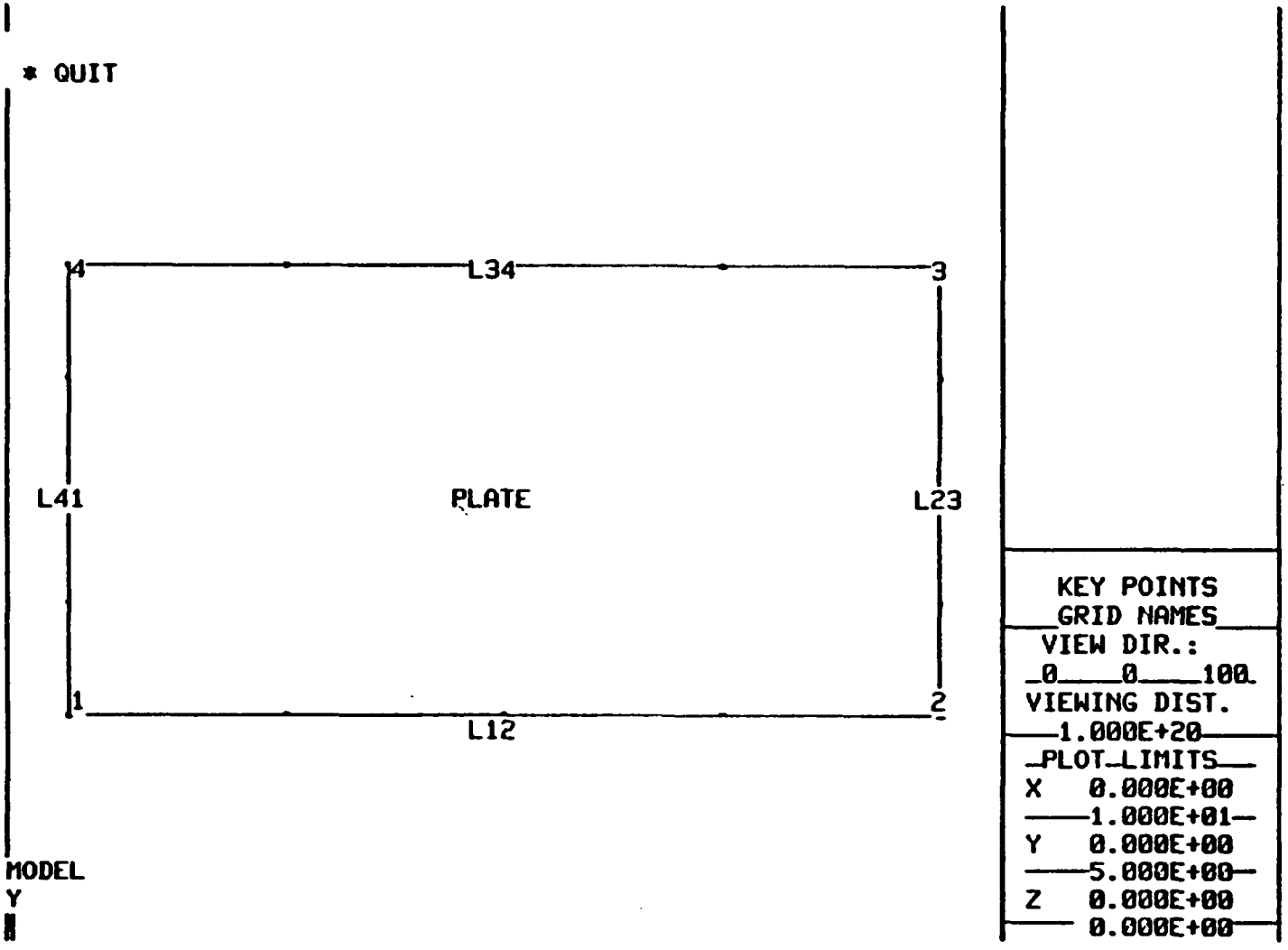
2.5.1 SAMPLE CASE 1: NONLINEAR STATIC ANALYSIS OF A CLAMPED PLATE



The setup is as shown above. We are going to generate a flat rectangular plate, clamp it on all four sides, and apply a uniform pressure in a series of statically incremented steps. The user is urged to follow the various transitions from pre-processing to analysis to postprocessing phases, as presented on the following pages, and to reproduce the results at the first possible opportunity.

```
<<>>  G I S T  [ GRAPHICS-INTERACTIVE STRUCTURAL ANALYSIS ]   Version 1

                    "Welcome to the GIFTS/STAGS Network"

Jobname: PLATE

<> NEW JOB

COMMAND : HELP : QUIT
G I S T > REVIEW JOB

<>  J O B   S T A T U S     [ PLATE ]

    Preprocessing Tasks       Status
    MODEL GENERATION ........ REQUIRED
    LOAD  GENERATION ........ REQUIRED
    BOUNDARY CONDITIONS ..... REQUIRED

    Analysis Tasks            Status
    ANALYSIS "SETUP"  ....... REQUIRED
    SOLUTION STRATEGY  ...... REQUIRED
    SOLUTION COMPUTATION .... REQUIRED

    Postprocessing Tasks      Status
    SPATIAL DISPLAY ......... REQUIRED
    TEMPORAL DISPLAY ........ N/A

COMMAND : HELP : QUIT
G I S T > BULKM
-
```

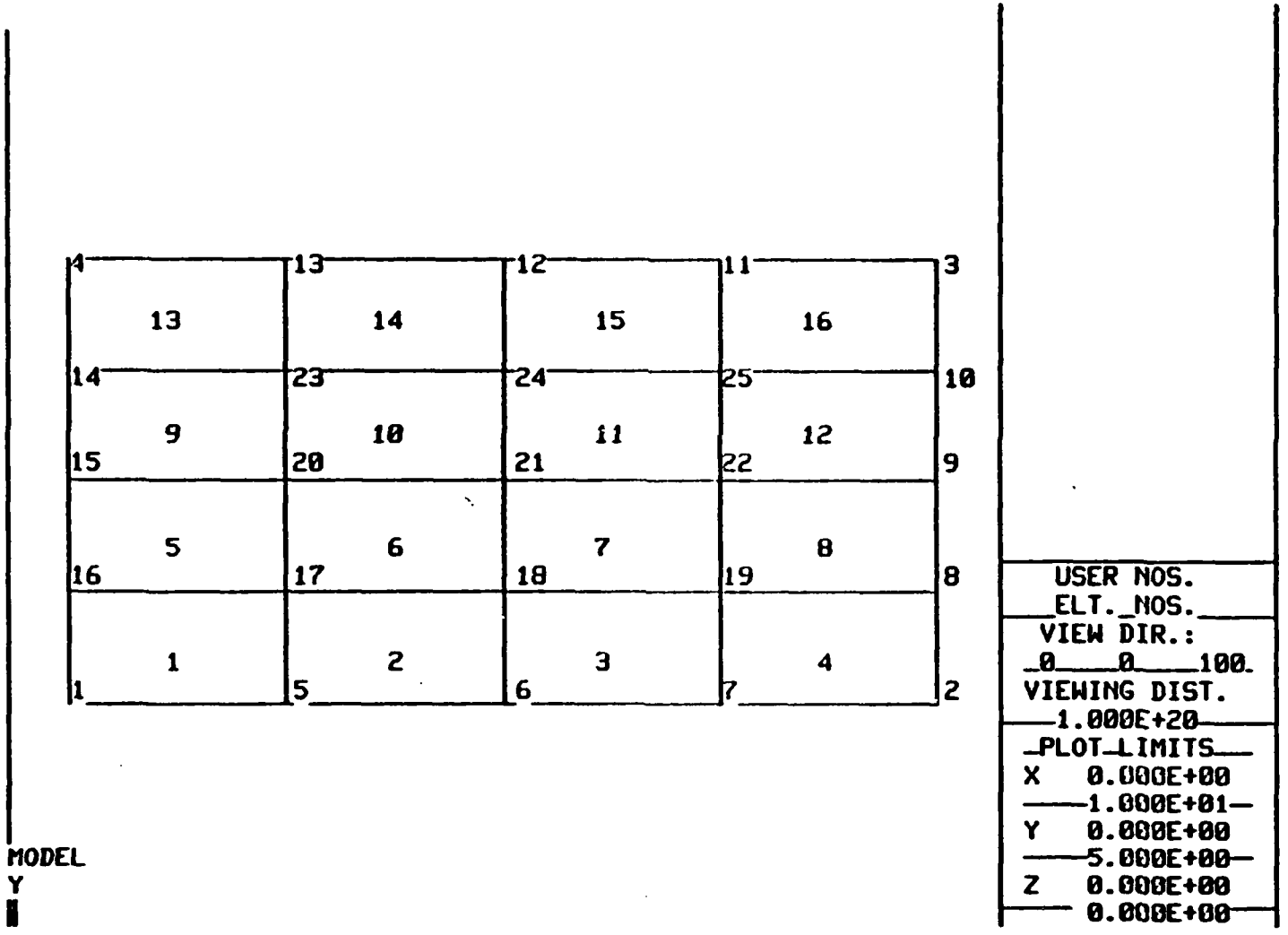
```
<<>>  B U L K M          [ BULK MODEL GENERATOR ]          GIFTS/5.03
JOB:  PLATE

JOB:  PLATE    BEING CREATED
 * KPOINT
 > 1/0.
 > 2/10.
 > 3/10.,5.
 > 4/0.,5.
 >
 * SLINE
 > L12/1,2 5
 > L23/2,3 5
 > L34/3,4 5
 > L41/4,1 5
 >
 * ELMAT,3 /1/ 10000., 1.E7, .3
 >
 * ETH,1 /1/ .1
 >
 * GETY/QB4/1,1
 *
 * GRID4
 > PLATE/L12,L23,L34,L41
 >
 * KN/LN/GN/ PLOT
-
```


* QUIT
4
L34
3
L41
PLATE
L23
1
2
L12
MODEL
Y
KEY POINTS
GRID NAMES
VIEW DIR.:
0 0 100
VIEWING DIST.
1.000E+20
PLOT LIMITS
X 0.000E+00
1.000E+01
Y 0.000E+00
5.000E+00
Z 0.000E+00
0.000E+00

```
COMMAND : HELP : QUIT
G I S T > REVIEW PREP

<> M O D E L  S T A T U S  [ PLATE ]

    SPATIAL DIMENSIONS .......    3
    SURFACE GRIDS ............    1
    SOLID GRIDS ..............    0
    KEY LINES ................    4
    KEY POINTS ...............    4
    NODES ....................   25
    ELEMENTS .................   16
    MATERIALS ................    1
    THICKNESSES ..............    1
    LOADING CASES ............    0

COMMAND : HELP : QUIT
G I S T > EDITM_
```

```
<<>> E D I T M          [ MODEL EDITOR ]          GIFTS/5.03
```

```
JOB:  PLATE

 * ELEMENTS
 * PN
 * EN
 * PLOT
-
```


4
13
12
11
3
13
14
15
16
14
23
24
25
10
9
10
11
12
15
20
21
22
9
5
6
7
8
16
17
18
19
8
1
2
3
4
1
5
6
7
2
USER NOS.
ELT. NOS.
VIEW DIR.:
0 0 100.
VIEWING DIST.
1.000E+20
PLOT LIMITS
X 0.000E+00
1.000E+01
Y 0.000E+00
5.000E+00
Z 0.000E+00
0.000E+00
MODEL
Y

```
 * PAGEOFF
 * INFP
 > 1,100

JOB:PLATE                5-JUN-81              22:39:35              PAGE  1
  NP          X             Y             Z          DEP     MFP      LPD
    1   0.0000E+00    0.0000E+00    0.0000E+00        0    111111   000000
    2   1.0000E+01    0.0000E+00    0.0000E+00        0    111111   000000-------
    3   1.0000E+01    5.0000E+00    0.0000E+00        0    111111   000000TS/5.03
    4   0.0000E+00    5.0000E+00    0.0000E+00        0    111111   000000-------
    5   2.5000E+00    0.0000E+00    0.0000E+00        0    111111   000000
    6   5.0000E+00    0.0000E+00    0.0000E+00        0    111111   000000
    7   7.5000E+00    0.0000E+00    0.0000E+00        0    111111   000000
    8   1.0000E+01    1.2500E+00    0.0000E+00        0    111111   000000
    9   1.0000E+01    2.5000E+00    0.0000E+00        0    111111   000000
   10   1.0000E+01    3.7500E+00    0.0000E+00        0    111111   000000
   11   7.5000E+00    5.0000E+00    0.0000E+00        0    111111   000000
   12   5.0000E+00    5.0000E+00    0.0000E+00        0    111111   000000
   13   2.5000E+00    5.0000E+00    0.0000E+00        0    111111   000000
   14   0.0000E+00    3.7500E+00    0.0000E+00        0    111111   000000
   15   0.0000E+00    2.5000E+00    0.0000E+00        0    111111   000000
   16   0.0000E+00    1.2500E+00    0.0000E+00        0    111111   000000
   17   2.5000E+00    1.2500E+00    0.0000E+00        0    111111   000000
   18   5.0000E+00    1.2500E+00    0.0000E+00        0    111111   000000
   19   7.5000E+00    1.2500E+00    0.0000E+00        0    111111   000000
   20   2.5000E+00    2.5000E+00    0.0000E+00        0    111111   000000
   21   5.0000E+00    2.5000E+00    0.0000E+00        0    111111   000000
   22   7.5000E+00    2.5000E+00    0.0000E+00        0    111111   000000
   23   2.5000E+00    3.7500E+00    0.0000E+00        0    111111   000000
   24   5.0000E+00    3.7500E+00    0.0000E+00        0    111111   000000
   25   7.5000E+00    3.7500E+00    0.0000E+00        0    111111   000000
 *
 *
-
```

```
 * INFE
 > 1,1000
JOB:PLATE                5-JUN-81              22:44:17              PAGE  1
 NEL    TYPE      NCP  NGP  NTT  NTH          CORNERS
    1 F-QB4        4    4    1    1      1     5    16    17
    2 F-QB4        4    4    1    1      5     6    17    18
    3 F-QB4        4    4    1    1      6     7    18    19
    4 F-QB4        4    4    1    1      7     2    19     8
    5 F-QB4        4    4    1    1     16    17    15    20
    6 F-QB4        4    4    1    1     17    18    20    21
    7 F-QB4        4    4    1    1     18    19    21    22
    8 F-QB4        4    4    1    1     19     8    22     9
    9 F-QB4        4    4    1    1     15    20    14    23
   10 F-QB4        4    4    1    1     20    21    23    24
   11 F-QB4        4    4    1    1     21    22    24    25
   12 F-QB4        4    4    1    1     22     9    25    10
   13 F-QB4        4    4    1    1     14    23     4    13
   14 F-QB4        4    4    1    1     23    24    13    12
   15 F-QB4        4    4    1    1     24    25    12    11
   16 F-QB4        4    4    1    1     25    10    11     3
 *
 * INFM
 > 1,1000
JOB:PLATE                5-JUN-81              22:44:28              PAGE  1
NM TYP     E             V             G             SY            P            A
 1  1  1.000E+07  3.000E-01  3.846E+06  1.000E+04  7.339E-04  6.500E-06
 *
 * INFT
 > 1,1000
JOB:PLATE                5-JUN-81              22:44:38              PAGE  1
NTH                          THICKNESS VALUES
  1    1.000E-01
 * QUIT
-
```

```
COMMAND : HELP : QUIT
G I S T > REVIEW PREP

<> M O D E L  S T A T U S  [ PLATE ]

    SPATIAL DIMENSIONS .......    3
    SURFACE GRIDS ............    1
    SOLID GRIDS ..............    0
    KEY LINES ................    4
    KEY POINTS ...............    4
    NODES ....................   25
    ELEMENTS .................   16
    MATERIALS ................    1
    THICKNESSES ..............    1
    LOADING CASES ............    0

COMMAND : HELP : QUIT
G I S T > BULKLB_
```

<<>> B U L K L B [BULK LOAD/BC GENERATOR] GIFTS/5.03

```
JOB:  PLATE
LOADING CASE  1
 * SUPL
 > L12
 > L23
 > L34
 > L41
 >
 * LOADG
 > PLATE
 ? -100.,-100.,-100.,-100.
 >
 * QUIT
-
```

```
COMMAND : HELP : QUIT
G I S T > EDITLB_
```

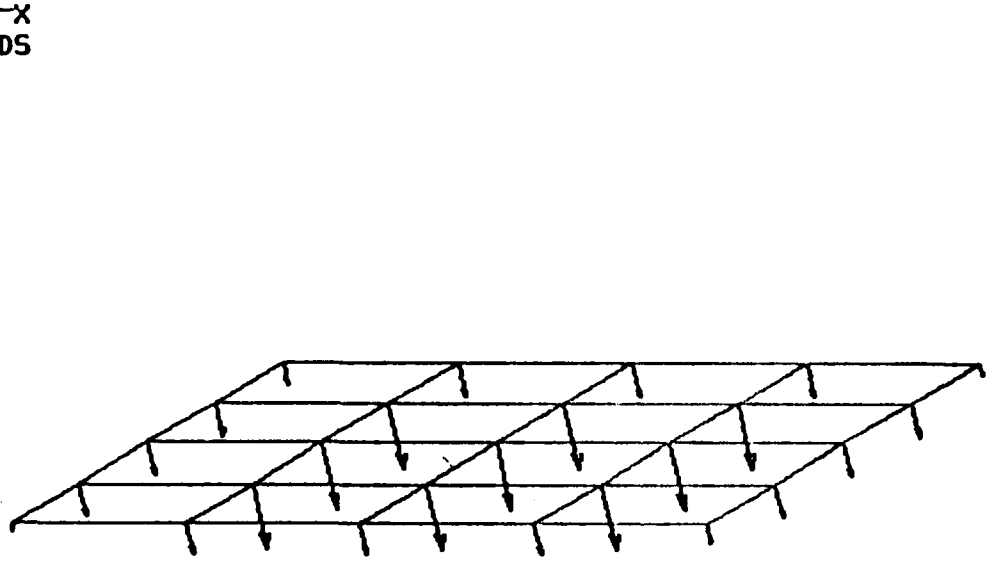
```
<<>>  E D I T L B          [ LOAD/BC EDITOR ]                GIFTS/5.03
```

```
JOB:  PLATE

LOADING CASE  1
 * ELEMENTS
 * ROTV
 > -70,-40
 * PLOT
-
```

000E+02 LOADING CASE 1

*

* SCALELD/0/TRANFR/PLOT



LOAD PLOT

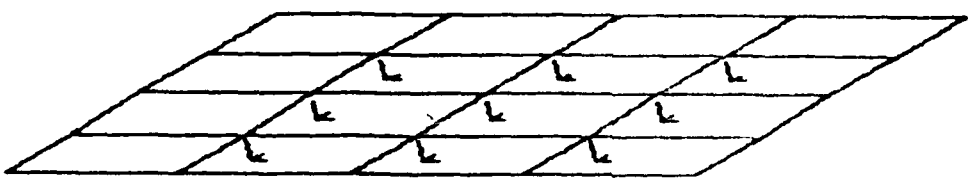
PT. LOADS

VIEW DIR.:
64 -72 26
VIEWING DIST.
1.000E+20

PLOT LIMITS
X 0.000E+00
1.000E+01
Y 0.000E+00
5.000E+00
Z 0.000E+00
0.000E+00

LOADING CASE 1

* ROTFR/PLOT



MODEL

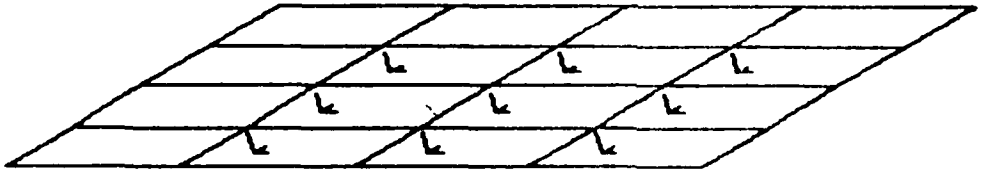
FREEDOM PLOT

TRANSLATIONAL

VIEW DIR.:
64 -72 26
VIEWING DIST.
1.000E+20

PLOT LIMITS
X 0.000E+00
1.000E+01
Y 0.000E+00
5.000E+00
Z 0.000E+00
0.000E+00

LOADING CASE 1



MODEL

FREEDOM PLOT

ROTATIONAL

VIEW DIR.:
64 -72 26
VIEWING DIST.
1.000E+20
PLOT LIMITS
X 0.000E+00
1.000E+01
Y 0.000E+00
5.000E+00
Z 0.000E+00
0.000E+00

```
*
*
* PAGEOFF
* INFP/1,1000

JOB:PLATE                4-JUN-81                23:21:39                PAGE  1
```

NP	X	Y	Z	DEP	INFP	LPD
1	0.0000E+00	0.0000E+00	0.0000E+00	0	000000	000000
2	1.0000E+01	0.0000E+00	0.0000E+00	0	000000	000000
3	1.0000E+01	5.0000E+00	0.0000E+00	0	000000	000000
4	0.0000E+00	5.0000E+00	0.0000E+00	0	000000	000000
5	2.5000E+00	0.0000E+00	0.0000E+00	0	000000	000000
6	5.0000E+00	0.0000E+00	0.0000E+00	0	000000	000000
7	7.5000E+00	0.0000E+00	0.0000E+00	0	000000	000000
8	1.0000E+01	1.2500E+00	0.0000E+00	0	000000	000000
9	1.0000E+01	2.5000E+00	0.0000E+00	0	000000	000000
10	1.0000E+01	3.7500E+00	0.0000E+00	0	000000	000000
11	7.5000E+00	5.0000E+00	0.0000E+00	0	000000	000000
12	5.0000E+00	5.0000E+00	0.0000E+00	0	000000	000000
13	2.5000E+00	5.0000E+00	0.0000E+00	0	000000	000000
14	0.0000E+00	3.7500E+00	0.0000E+00	0	000000	000000
15	0.0000E+00	2.5000E+00	0.0000E+00	0	000000	000000
16	0.0000E+00	1.2500E+00	0.0000E+00	0	000000	000000
17	2.5000E+00	1.2500E+00	0.0000E+00	0	111111	000000
18	5.0000E+00	1.2500E+00	0.0000E+00	0	111111	000000
19	7.5000E+00	1.2500E+00	0.0000E+00	0	111111	000000
20	2.5000E+00	2.5000E+00	0.0000E+00	0	111111	000000
21	5.0000E+00	2.5000E+00	0.0000E+00	0	111111	000000
22	7.5000E+00	2.5000E+00	0.0000E+00	0	111111	000000
23	2.5000E+00	3.7500E+00	0.0000E+00	0	111111	000000
24	5.0000E+00	3.7500E+00	0.0000E+00	0	111111	000000
25	7.5000E+00	3.7500E+00	0.0000E+00	0	111111	000000

```
*
```

```
*
* INFLD/1,1000

JOB:PLATE          4-JUN-81      LOADING CASE  1      23:22:19      PAGE  1
                                    LOADS
 NP       VX           VY           VZ           MX           MY           MZ
  1   0.           0.          -5.200E+01   0.           0.           0.
  2   0.           0.          -1.042E+02   0.           0.           0.
  3   0.           0.          -5.208E+01   0.           0.           0.
  4   0.           0.          -1.042E+02   0.           0.           0.
  5   0.           0.          -1.563E+02   0.           0.           0.
  6   0.           0.          -1.563E+02   0.           0.           0.
  7   0.           0.          -1.563E+02   0.           0.           0.
  8   0.           0.          -1.563E+02   0.           0.           0.
  9   0.           0.          -1.563E+02   0.           0.           0.
 10   0.           0.          -1.563E+02   0.           0.           0.
 11   0.           0.          -1.563E+02   0.           0.           0.
 12   0.           0.          -1.563E+02   0.           0.           0.
 13   0.           0.          -1.563E+02   0.           0.           0.
 14   0.           0.          -1.563E+02   0.           0.           0.
 15   0.           0.          -1.563E+02   0.           0.           0.
 16   0.           0.          -1.563E+02   0.           0.           0.
 17   0.           0.          -3.125E+02   0.           0.           0.
 18   0.           0.          -3.125E+02   0.           0.           0.
 19   0.           0.          -3.125E+02   0.           0.           0.
 20   0.           0.          -3.125E+02   0.           0.           0.
 21   0.           0.          -3.125E+02   0.           0.           0.
 22   0.           0.          -3.125E+02   0.           0.           0.
 23   0.           0.          -3.125E+02   0.           0.           0.
 24   0.           0.          -3.125E+02   0.           0.           0.
 25   0.           0.          -3.125E+02   0.           0.           0.
*
* QUIT
```

```
COMMAND : HELP : QUIT
G I S T > REVIEW JOB

<> J O B   S T A T U S     [ PLATE ]

    Preprocessing Tasks        Status
    MODEL GENERATION ........ UPDATED
    LOAD  GENERATION ........ UPDATED
    BOUNDARY CONDITIONS ..... UPDATED

    Analysis Tasks             Status
    ANALYSIS "SETUP"  ....... REQUIRED
    SOLUTION STRATEGY  ...... REQUIRED
    SOLUTION COMPUTATION .... REQUIRED

    Postprocessing Tasks       Status
    SPATIAL DISPLAY ......... REQUIRED
    TEMPORAL DISPLAY ........ N/A

COMMAND : HELP : QUIT
G I S T > REVIEW ANALYSIS

>>> Sorry >>>   ANALYSIS NOT INITIATED

COMMAND : HELP : QUIT
G I S T > REVIEW STRATEGY

>>> Sorry >>>   NO STRATEGY TO REVIEW

COMMAND : HELP : QUIT
G I S T > _
```

```
COMMAND : HELP : QUIT
G I S T > HELP ANALYSIS
<GIST>
    ANALYSIS

    The following commands are associated with analysis proper:

    SETUP  .........  Invokes the STAGS Structural Pre-Analyzer
    STATIC  ........  Prepares strategy for STAGS static analysis
    DYNAMIC  .......  Prepares strategy for STAGS dynamic analysis
    BUCKLING  ......  Prepares strategy for STAGS buckling analysis
    VIBRATION  .....  Prepares strategy for STAGS vibration analysis
    COMPUTE  .......  Invokes the STAGS Structural Analyzer for solution

    SETUP is required at the transition from the pre-processing phase
    to the analysis phase.  The strategy commands may then be used to
    prescribe a forthcoming solution interval in terms of an analysis
    type and corresponding algorithmic parameters (+ data-processing
    switches).  The strategy definition is immediately archived in the
    database and hence is available for subsequent review and selective
    modification.  Finally, the COMPUTE command is used to initiate
    each solution interval, employing the analyst's current strategy.

    For additional information on STAGS analysis capabilities, consult
    (1) HELP STATIC,DYNAMIC etc., or (2) the STAGS User's Manual (S6).

COMMAND : HELP : QUIT
G I S T > SETUP_
```

```
------------------------------------------------------------------------
<<>>  S T A G S 1     [ STRUCTURAL PRE-ANALYZER ]              STAGS/C1.0
------------------------------------------------------------------------
JOB:  PLATE    /NEW


<>  M O D E L   S U M M A R Y

    STRUCTURAL UNITS ..............     1
    LOAD SYSTEMS ..................     1
    MATERIALS .....................     1
    1-D (BEAM) FABRICATIONS .......     0
    2-D (SHELL) FABRICATIONS ......     1


<>  P R E - A N A L Y S I S   S U M M A R Y :  STRUCTURAL UNIT 1

    o  GENERAL CONFIGURATION

       NODES ..............   25
       ELEMENTS ...........   16
       KINEMATICS ......... NONLINEAR
       CONSTITUTION ........ ELASTIC
       NET WEIGHT .......... 0.3669E-02
       GROSS WEIGHT ........ 0.3669E-02

 -> o  COMPUTATIONAL STATISTICS

       EQUATIONS ..................    54
       AVG. HALF-BANDWIDTH .......    19
       NONZERO STIFFNESSES .......  1053
       EST. FLOATING OPERATONS ... 12600
       EST. FACTORIZATION TIME ...    0.016560 SECS
-
```

```
      EST. FACTORIZATION TIME ...    0.016560 SECS

<> READY FOR ANALYSIS.
-------------------------------------------------------------------------

COMMAND : HELP : QUIT
G I S T > REVIEW JOB

<>  J O B   S T A T U S     [ PLATE ]

    Preprocessing Tasks       Status
    MODEL GENERATION ........ READY
    LOAD  GENERATION ........ READY
    BOUNDARY CONDITIONS ..... READY

    Analysis Tasks            Status
    ANALYSIS "SETUP"  ....... READY
    SOLUTION STRATEGY  ...... REQUIRED
    SOLUTION COMPUTATION .... REQUIRED

    Postprocessing Tasks      Status
    SPATIAL DISPLAY ......... REQUIRED
    TEMPORAL DISPLAY ........ N/A

COMMAND : HELP : QUIT
G I S T > STATIC

Keywords:   INTERVAL   STEPSIZE   NEWTON   [ Options ]
G I S T  STATIC > INTERVAL=.1,.5  STEPSIZE=.2

COMMAND : HELP : QUIT
G I S T > REVIEW STRATEGY_
```

```
<>  S T R A T E G Y :   STATIC / NONLINEAR

    INTERVAL (pamin,pamax) .......  0.1000E+00, 0.5000E+00
    STEPSIZE (incpa) .............  0.2000E+00
    NEWTON ....................... ** UNDEFINED **
    EXTRAP ....................... "ON"
    MAXCUT .......................   0
    MAXERR .......................  0.1000E-02
    LIST (D,S,R,E,F)  ............   0,  0,  0,  0,  0
    SAVE (D,S,F,K,KI) ............   1,  1,  0,  0,  0
    START ........................ "NEXT"

COMMAND : HELP : QUIT
G I S T > STATIC NEWTON=MOD,1

COMMAND : HELP : QUIT
G I S T > REV STRAT

<>  S T R A T E G Y :   STATIC / NONLINEAR

    INTERVAL (pamin,pamax) .......  0.1000E+00, 0.5000E+00
    STEPSIZE (incpa) .............  0.2000E+00
    NEWTON ....................... "MODIFIED" /  1
    EXTRAP ....................... "ON"
    MAXCUT .......................   0
    MAXERR .......................  0.1000E-02
    LIST (D,S,R,E,F)  ............   0,  0,  0,  0,  0
    SAVE (D,S,F,K,KI) ............   1,  1,  0,  0,  0
    START ........................ "NEXT"

COMMAND : HELP : QUIT
G I S T > COMPUTE_
```

```
<<>> S T A G S 2      [ STRUCTURAL ANALYZER ]                    STAGS/C1.0

JOB:  PLATE

<> I N P U T   P H A S E

    o  ANALYSIS:  NONLINEAR STATIC RESPONSE

    o  INTERVAL (PA):  0.1000E+00, 0.5000E+00;  0.2000E+00

    o  START FROM STEP:    0

<> S O L U T I O N   P H A S E

ELEMENT STIFFNESS MATRICES COMPUTED FOR UNIT     1 (ELEMENT)

 ASSEMBLY OF TOTAL STIFFNESS MATRIX COMPLETED.

DETERMINANT OF STIFFNESS MATRIX=   0.3857078E+01*10.**   270.   NUMBER OF NEGATI

VE ROOTS =   0
       54 EQUATIONS.  AVERAGE BAND WIDTH =          18
 MATRIX DECOMPOSITION COMPLETED.

    o  ARCHIVING STEP   0   [ PA =0.1000E+00,   PB =0.0000E+00 ]

    _
```

[illegible] OF INITIAL STIFFNESS MATRICES COMPLETED.

[illegible] OF STIFFNESS MATRIX= [illegible] ZRO. NUMBER OF NEGATIVE ROOTS = 0
54 EQUATIONS. MAXIMUM BAND WIDTH = 19
MATRIX RECONFIGURATION COMPLETED.

0 ARCLENGTHING STEP 0 [PA =0.1000E+00, PB =0.0000E+00]

BEGIN NON-LINEAR ITERATIONS FOR LOAD STEP 1, PA= 0.1000000E+00, PB= 0.0000000E+00, INCREMENT LOAD= 0.0000000E+00

ITERATION	RESIDUAL ERROR	DISPLACEMENT ERROR	RELAXATION FACTOR	MINIMUM RESIDUAL	VARIABLE NUMBER
1	[illegible]	[illegible]	[illegible]	[illegible]	44
2	[illegible]	[illegible]	[illegible]	[illegible]	44
3	[illegible]	[illegible]	[illegible]	[illegible]	44
4	[illegible]	[illegible]	[illegible]	[illegible]	44
5	[illegible]	[illegible]	[illegible]	[illegible]	0

CONVERGENCE HAS BEEN OBTAINED FOR LOAD STEP 1
ERROR 0 ESTIMATE ALLOCATION WILL BE USED FOR LOAD STEP 2

0 ARCLENGTHING STEP 1 [PA =0.1000E+00, PB =0.0000E+00]

TOTAL STIFFNESS MATRIX, LOAD STEP 2, PN= 0.[illegible]E+00, PB= 0.[illegible]E+00

ELEMENT STIFFNESS MATRICES OBTAINED FOR UNIT 1 (ELEMENT)

ASSEMBLY OF TOTAL STIFFNESS MATRIX COMPLETED.

DETERMINANT OF STIFFNESS MATRIX= 0.[illegible]E+00 10.## 27. NUMBER OF NEGATIVE ROOTS = 0
54 EQUATIONS. AVERAGE BAND WIDTH = 19
MATRIX DECOMPOSITION COMPLETED.

BEGIN NON-LINEAR ITERATIONS FOR LOAD STEP 2, PA= 0.30000E+00, PB= 0.00000E+00, THERMAL LOAD= 0.00000E+00

ITERATION	RESIDUAL ERROR	DISPLACEMENT ERROR	RELAXATION FACTOR	MAXIMUM RESIDUAL	VARIABLE NUMBER
1	0.35858E+01	0.14822E+00	0.10000E+01	-0.27837E+03	44
2	0.79216E-01	0.94282E-02	0.10000E+01	-0.40558E+01	31
3	0.11540E-01	0.11755E-02	0.10000E+01	0.87451E+00	8
4	0.14222E-02	0.15221E-03	0.10000E+01	-0.98490E-01	44
5	0.18781E-03	0.19833E-04	0.10000E+01	-0.13100E-01	44

CONVERGENCE HAS BEEN OBTAINED FOR LOAD STEP 2
ORDER 1 EXTRAPOLATION WILL BE USED FOR LOAD STEP 3

o ARCHIVING STEP 2 [PA =0.300E+00, PB =0.000E+00]

TOTAL STIFFNESS MATRIX, LOAD STEP 3, PA= 0.50000E+00, PB= 0.00000E+00

TOTAL STIFFNESS MATRIX, LOAD STEP 3, PR= 0.000000E+00, PI= 0.000000E+00

ELEMENT STIFFNESS MATRICES OBTAINED FOR UNIT 1 (ELEMENT)

ASSEMBLY OF TOTAL STIFFNESS MATRIX COMPLETED.

DETERMINANT OF STIFFNESS MATRIX= 0.XXXXXXE+XX*10.** 20. NUMBER OF NEGATIVE ROOTS = 0
54 EQUATIONS. AVERAGE BAND WIDTH = 19
MATRIX DECOMPOSITION COMPLETED.

BEGIN NON-LINEAR ITERATIONS FOR LOAD STEP 4, TF= 0.[illegible], TF= 0.[illegible], INCREMENT LOAD= 0.[illegible]

ITERATION	RESIDUAL FORCE	DISPLACEMENT ERROR	RELAXATION FACTOR	MAXIMUM RESIDUAL	VARIABLE NUMBER
1	0.[illegible]E+01	0.[illegible]E+00	0.[illegible]E+01	-0.[illegible]E+03	44
2	0.[illegible]E-01	0.[illegible]E-02	0.[illegible]E+01	0.[illegible]E+01	19
3	0.[illegible]E-01	0.[illegible]E-02	0.[illegible]E+01	-0.[illegible]E+01	44
4	0.[illegible]E-02	0.[illegible]E-03	0.[illegible]E+01	0.[illegible]E+00	8
5	0.[illegible]E-03	0.[illegible]E-04	0.[illegible]E+01	-0.[illegible]E-01	44

CONVERGENCE HAS BEEN OBTAINED FOR LOAD STEP 3

MAXIMUM LOAD VALUE ATTAINED

NON-LINEAR COLLAPSE ANALYSIS DISCONTINUED

0 ARCHIVING STEP 3 [PA =0.[illegible]E+00, PR =0.[illegible]E+00]

0 SOLUTION READY.

[illegible] ; HELP ; QUIT
(LIST)

```
COMMAND : HELP : QUIT
G I S T >   REVIEW/TOC AN

<> A N A L Y S I S   D A T A B A S E

++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
+    TABLE OF CONTENTS, Library 16,  File:  PLATE.STG                        +
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
 Seq DMGASP                E                                  Data set name
   # Addres  Date    Time   r   Words  Recs  Size Typ Nam1 Nam2 Nam3 Nam4
   1       8 060581 232652 0     540     9    60   0 STAG STAT    0    0
   2       9 060581 232705 0      20     2    10   0 STAG MATL    0    0
   3      11 060581 232705 0      20     2    10   0 STAG FABR    2    0
   4      13 060581 232705 0     894     3    20   0 STAG CONF    1    0
   5      28 060581 232705 0     118     2    10   0 STAG LOAD    1    0
   6      31 060581 232705 0     215     2    40   0 STAG DOFS    1    0
   7      35 060581 233005 0      90     1    90   0 STAG STRA    0    0
   8      37 060581 233029 0     128     2    20   0 STAG DISP    0    0
   9      40 060581 233030 0     549     3    20   0 STAG STRE    0    0
  10      50 060581 233113 0     128     2    20   0 STAG DISP    0    1
  11      53 060581 233114 0     549     3    20   0 STAG STRE    0    1
  12      63 060581 233137 0     128     2    20   0 STAG DISP    0    2
  13      66 060581 233140 0     549     3    20   0 STAG STRE    0    2
  14      76 060581 233202 0     128     2    20   0 STAG DISP    0    3
  15      79 060581 233204 0     549     3    20   0 STAG STRE    0    3
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
+      Library has    15 data sets and occupies      89 sectors              +
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

    File 2:  PLATE.SIN     (STAGS INternal File)

COMMAND : HELP : QUIT
G I S T > RESULT SOLUTION=0,3_
```

```
COMMAND : HELP : QUIT
G I S T > RESULT SOLUTION=0,3


<<>>  S 2 G            < STAGS->GIFTS ADAPTOR >                GIST/1.0/
JOB:
JOB:
<CL> U00,L0002,C00001>PLATE
S 2 G >
<CL> U00,L0003,C00002> ADAPT STEPS =    0,    3,    0

<> GIFTS SOLUTION NUMBER   1  <==>  STAGSDISP   0   0
<> GIFTS SOLUTION NUMBER   1  <==>  STAGSSTRE   0   0
<> GIFTS SOLUTION NUMBER   2  <==>  STAGSDISP   0   1
<> GIFTS SOLUTION NUMBER   2  <==>  STAGSSTRE   0   1
<> GIFTS SOLUTION NUMBER   3  <==>  STAGSDISP   0   2
<> GIFTS SOLUTION NUMBER   3  <==>  STAGSSTRE   0   2
<> GIFTS SOLUTION NUMBER   4  <==>  STAGSDISP   0   3
<> GIFTS SOLUTION NUMBER   4  <==>  STAGSSTRE   0   3
+++  CLOSE, 12

<> READY FOR DISPLAY


-------------------------------------------------------------------------
<<>>  R E S U L T        [ SOLUTION DISPLAY ]                GIFTS/5.03
-------------------------------------------------------------------------
JOB:  PLATE

LOADING CASE  1
 * ELEMENT/ PLOT
_
```

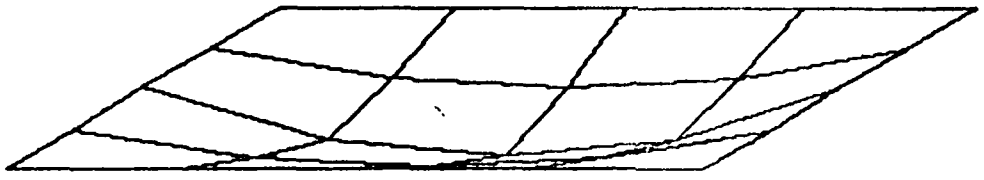
LOADING CASE 1

```
 * PAGEOFF
 * INFDN/21/
```

JOB:PLATE 5-JUN-81 LOADING CASE 1 23:44:10 PAGE 1

NP	U	V	W	THX	THY	THZ
21	0.000E+00	0.000E+00	-1.891E-02	-4.317E-19	-4.291E-19	0.000E+00

*



VIEW DIR.:
64 -72 26
VIEWING DIST.
1.000E+20

PLOT LIMITS
X 0.000E+00
1.000E+01
Y 0.000E+00
5.000E+00
Z 0.000E+00
0.000E+00

MODEL

DEFLS

```
 *
 *
 * LDCASE/2
LOADING CASE  2
 *
 * INFDN/21/

JOB:PLATE          5-JUN-81      LOADING CASE  2      23:46:06       PAGE  1
 NP        U             V             W            THX           THY           THZ
  21  2.938E-22 -2.024E-22 -1.830E-02  1.356E-19 -1.248E-19  3.031E-21
 *
 *
 * LDCASE/3
LOADING CASE  3
 *
 * INFDN/21/

JOB:PLATE          5-JUN-81      LOADING CASE  3      23:46:17       PAGE  1
 NP        U             V             W            THX           THY           THZ
  21  3.481E-21 -6.960E-21 -4.646E-02  1.927E-20 -7.034E-20 -1.107E-20
 *
 *
 * LDCASE/4
LOADING CASE  4
 *
 * INFDN/21/

JOB:PLATE          5-JUN-81      LOADING CASE  4      23:46:34       PAGE  1
 NP        U             V             W            THX           THY           THZ
  21  1.529E-21  2.390E-21 -6.541E-02 -4.292E-19 -2.764E-19  1.368E-20
 *
 *
 * QUIT
```

```
G I S T > REVIEW ANALYSIS

<> A N A L Y S I S   S T A T U S   [ PLATE ]

    PROBLEM ..... STATIC / NONLINEAR
    SOLUTION STEP .....     3
    LOAD FACTORS (PA,PB): 0.5000E+00, 0.0000E+00
    NUMBER OF EQNS .....    54

COMMAND : HELP : QUIT
G I S T > REVIEW STRAT

<> S T R A T E G Y :   STATIC / NONLINEAR

    INTERVAL (pamin,pamax) .....  0.1000E+00, 0.5000E+00
    STEPSIZE (incpa) ..........  0.2000E+00
    NEWTON ..................... "MODIFIED" /  1
    EXTRAP ..................... "ON"
    MAXCUT .....................   0
    MAXERR .....................  0.1000E-02
    LIST (D,S,R,E,F) ..........    0,  0,  0,  0,  0
    SAVE (D,S,F,K,KI) .........    1,  1,  0,  0,  0
    START ...................... "NEXT"

COMMAND : HELP : QUIT
G I S T > STATIC

Keywords:   INTERVAL   STEPSIZE   NEWTON   [ Options ]
G I S T  STATIC > INTERVAL=.5,.1 STEPSIZE=.5

>>> Invalid Data >>>   INTERVAL = what?

COMMAND : HELP : QUIT
G I S T > STATIC  INTERVAL=.5,1. STEPSIZE=.5_
```

```
COMMAND : HELP : QUIT
G I S T > REVIEW STRATEGY

<> S T R A T E G Y :   STATIC / NONLINEAR

    INTERVAL (pamin,pamax) .......  0.5000E+00, 0.1000E+01
    STEPSIZE (incpa) .............  0.5000E+00
    NEWTON ....................... "MODIFIED" /  1
    EXTRAP ....................... "ON"
    MAXCUT .......................   0
    MAXERR .......................  0.1000E-02
    LIST (D,S,R,E,F)  ............   0,  0,  0,  0,  0
    SAVE (D,S,F,K,KI) ............   1,  1,  0,  0,  0
    START ........................ "NEXT"

COMMAND : HELP : QUIT
G I S T > COMPUTE

-------------------------------------------------------------------------
<<>> S T A G S 2       [ STRUCTURAL ANALYZER ]                STAGS/C1.0
-------------------------------------------------------------------------
JOB:  PLATE


<> I N P U T   P H A S E


    o  ANALYSIS:  NONLINEAR STATIC RESPONSE

    o  INTERVAL (PA):  0.5000E+00, 0.1000E+01;  0.5000E+00

_   o  START FROM STEP:    3
```

◇ INPUT PHASE

- ANALYSIS: NONLINEAR STATIC RESPONSE
- INTERVAL (PA): 0.5000E+00, 0.1000E+01; 0.5000E+00
- START FROM STEP: 3

◇ SOLUTION PHASE

- RESTORING STEP 3 [PA=0.5000E+00, PB=0.0000E+00]
- RESTORING STEP 2 [PA=0.3000E+00, PB=0.0000E+00]
- RESTORING STEP 1 [PA=0.1000E+00, PB=0.0000E+00]

ORDER 2 EXTRAPOLATION WILL BE USED FOR LOAD STEP 4

TOTAL STIFFNESS MATRIX, LOAD STEP 4, PA= 0.1000E+01, PB= 0.0000E+00

TOTAL STIFFNESS MATRIX, LOAD STEP 4, FR= 0.[illegible], FI= 0.[illegible]

ELEMENT STIFFNESS MATRICES COMPUTED FOR UNIT 1 (ELEMENT)

ASSEMBLY OF TOTAL STIFFNESS MATRIX COMPLETED.

DECOMPOSITION OF STIFFNESS MATRIX= 0.[illegible] 24. NUMBER OF NEGATIVE POINTS = 0
24 EQUATIONS. AVERAGE BAND WIDTH = 19
MATRIX DECOMPOSITION COMPLETED.

ITERATION	RESIDUAL ERROR	DISPLACEMENT ERROR	RELAXATION FACTOR	MAXIMUM RESIDUAL	VARIABLE NUMBER
1	0.191514E+01	0.236195E+00	0.10000E+01	-0.84700E+03	8
2	0.173598E+00	0.623787E-01	0.10000E+01	0.61048E+02	8
3	0.859828E-01	0.222707E-01	0.10000E+01	-0.37108E+02	8
4	0.363028E-01	0.117228E-01	0.10000E+01	0.13490E+02	8
5	0.158028E-01	0.509314E-02	0.10000E+01	0.61118E+01	44
6	0.677172E-02	0.217898E-02	0.10000E+01	-0.25078E+01	44
7	0.291621E-02	0.937407E-03	0.10000E+01	-0.11008E+01	8
8	0.125088E-02	0.402267E-03	0.10000E+01	0.46824E+00	8
9	0.537088E-03	0.172888E-03	0.10000E+01	-0.20174E+00	8

CONVERGENCE HAS BEEN OBTAINED FOR LOAD STEP 4

MAXIMUM LOAD VALUE ATTAINED

NONLINEAR COLLAPSE ANALYSIS DISCONTINUED

o ARCHIVING STEP 4 [PA =0.1000E+01, PB =0.000E+00]

◊ SOLUTION READY.

```
COMMAND : HELP : QUIT
G I S T > RESULT SOLUTION=4


<<>>  S 2 G            < STAGS->GIFTS ADAPTOR >                GIST/1.0/
JOB:
JOB:
<CL> U00,L0002,C00001>PLATE
S 2 G >
<CL> U00,L0003,C00002> ADAPT STEPS =    4,    0,    0

<> GIFTS SOLUTION NUMBER   1  <==>  STAGSDISP   0   4
<> GIFTS SOLUTION NUMBER   1  <==>  STAGSSTRE   0   4
+++  CLOSE, 12

<> READY FOR DISPLAY_
```

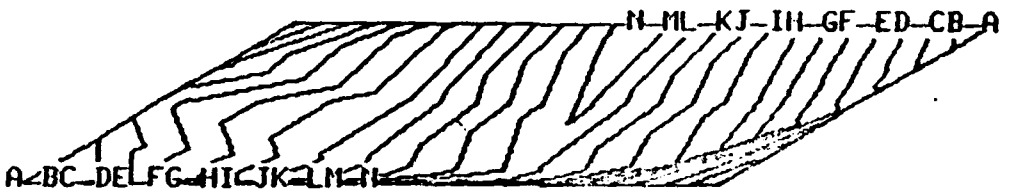
```
--------------------------------------------------------------------------------
<<>>  R E S U L T          [ SOLUTION DISPLAY ]                    GIFTS/5.03
--------------------------------------------------------------------------------
JOB:  PLATE

LOADING CASE  1
 *
 * INFDN/21/
JOB:PLATE          5-JUN-81      LOADING CASE  1      23:59:01       PAGE  1
 NP        U            V            W           THX          THY          THZ
  21  2.822E-20  3.901E-20 -9.584E-02  1.330E-18 -3.111E-20  2.314E-20
 *
 * INFST/5,12/
JOB:PLATE          6-JUN-81      LOADING CASE  1      00:00:10       PAGE  1
 NE                             STRESSES -- MIDDLE
    5  4.815E+03  3.540E+03  2.610E+02
    6  4.123E+03  9.360E+03 -6.463E+02
    7  4.123E+03  9.360E+03  6.463E+02
    8  4.815E+03  3.540E+03 -2.610E+02
    9  4.815E+03  3.540E+03 -2.610E+02
   10  4.123E+03  9.360E+03  6.463E+02
   11  4.123E+03  9.360E+03 -6.463E+02
   12  4.815E+03  3.540E+03  2.610E+02
 *
 * CONTOUR/ PLOT
-
```

RESET

* CONTOUR

* PLOT

LOADING CASE 1



MODEL

DEFLS

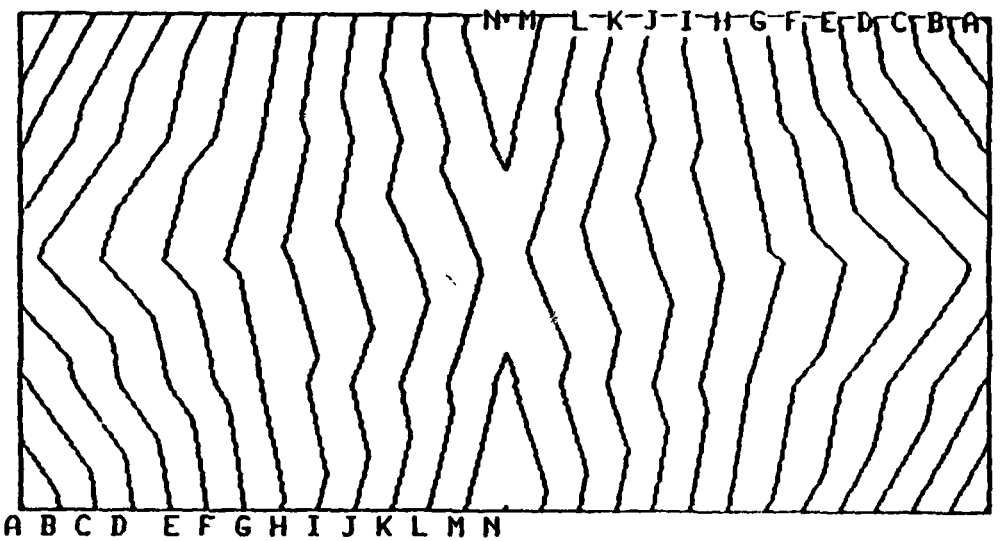
STRESS CONTOURS	
A	2.500E+01
C	3.500E+01
D	4.000E+01
E	4.500E+01
F	5.000E+01
G	5.500E+01
H	6.000E+01
I	6.500E+01
J	7.000E+01
L	8.000E+01
M	8.500E+01
N	9.000E+01

VIEW DIR.:
64 -72 26
VIEWING DIST.
1.000E+20

PLOT LIMITS

X	0.000E+00
	1.000E+01
Y	0.000E+00
	5.000E+00
Z	0.000E+00
	0.000E+00

LOADING CASE 1



STRESS CONTOURS	
A	2.500E+01
C	3.500E+01
D	4.000E+01
E	4.500E+01
F	5.000E+01
G	5.500E+01
H	6.000E+01
I	6.500E+01
J	7.000E+01
L	8.000E+01
M	8.500E+01
N	9.000E+01

VIEW DIR.:
0 0 100
VIEWING DIST.
1.000E+20

PLOT LIMITS

X	0.000E+00
	1.000E+01
Y	0.000E+00
	5.000E+00
Z	0.000E+00
	0.000E+00

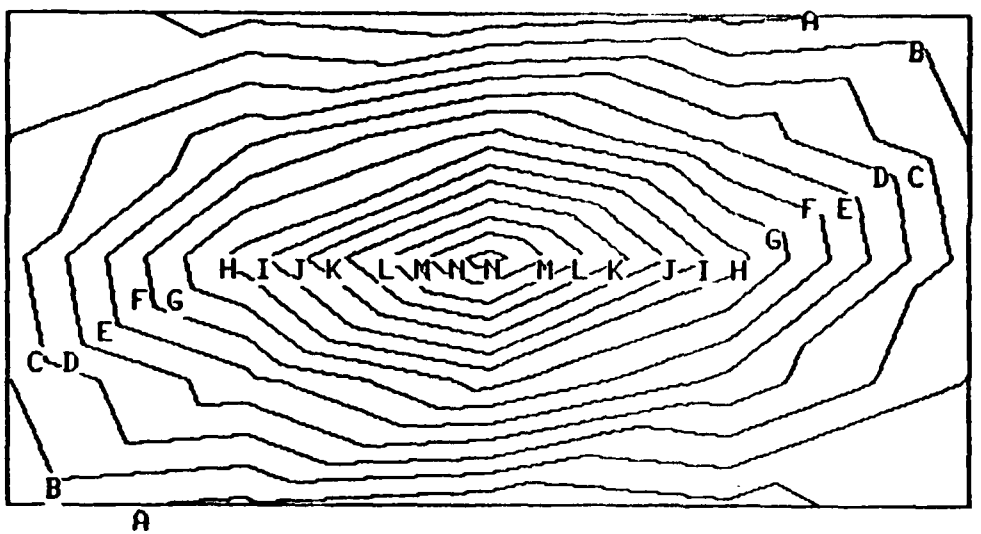
MODEL
Y

DEFLS
Y

1-05

LOADING CASE 1

* BOTTOM/PLOT



MODEL
Y

DEFLS
Y

STRESS CONTOURS

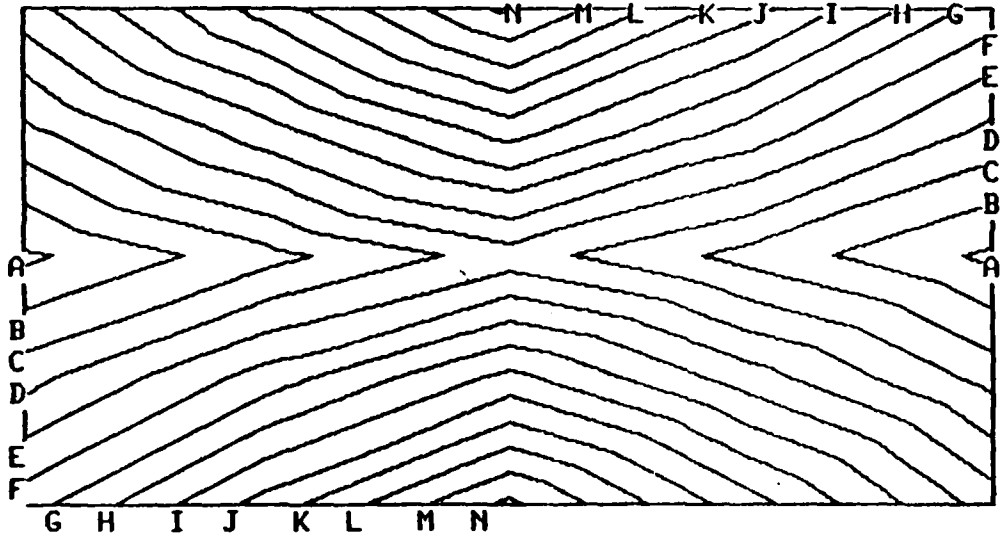
A	1.400E+04
C	1.800E+04
D	2.000E+04
E	2.200E+04
F	2.400E+04
G	2.600E+04
H	2.800E+04
I	3.000E+04
J	3.200E+04
L	3.600E+04
M	3.800E+04
N	4.000E+04

VIEW DIR.:
0 0 100
VIEWING DIST.
1.000E+20
PLOT LIMITS

X	0.000E+00
	1.000E+01
Y	0.000E+00
	5.000E+00
Z	0.000E+00
	0.000E+00

LOADING CASE 1

* INFST/1,1000/



STRESS CONTOURS

A	1.000E+04
C	1.400E+04
D	1.600E+04
E	1.800E+04
F	2.000E+04
G	2.200E+04
H	2.400E+04
I	2.600E+04
J	2.800E+04
L	3.200E+04
M	3.400E+04
N	3.600E+04

VIEW DIR.:
0 0 100
VIEWING DIST.
1.000E+20
PLOT LIMITS

X	0.000E+00
	1.000E+01
Y	0.000E+00
	5.000E+00
Z	0.000E+00
	0.000E+00

MODEL
Y

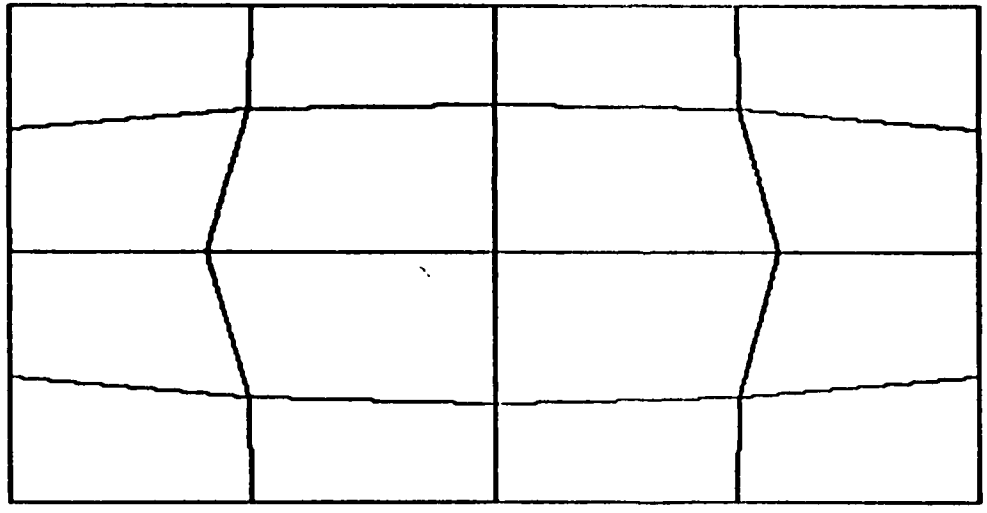
DEFLS
Y

```
 * INFST/1,1000/
JOB:PLATE          6-JUN-81      LOADING CASE  1      00:10:40      PAGE  1
 NE                             STRESSES -- BOTTOM
   1  5.583E+03  1.316E+04  7.465E+03
   2  8.438E+03  3.154E+04  8.494E+02
   3  8.438E+03  3.154E+04 -8.494E+02
   4  5.583E+03  1.316E+04 -7.465E+03
   5  5.057E+03 -5.692E+03  4.540E+03
   6 -5.712E+03 -1.338E+04  4.559E+01
   7 -5.712E+03 -1.338E+04 -4.559E+01
   8  5.057E+03 -5.692E+03 -4.540E+03
   9  5.057E+03 -5.692E+03 -4.540E+03
  10 -5.712E+03 -1.338E+04 -4.559E+01
  11 -5.712E+03 -1.338E+04  4.559E+01
  12  5.057E+03 -5.692E+03  4.540E+03
  13  5.583E+03  1.316E+04 -7.465E+03
  14  8.438E+03  3.154E+04 -8.494E+02
  15  8.438E+03  3.154E+04  8.494E+02
  16  5.583E+03  1.316E+04  7.465E+03
 *
 * TOP/ INFST/6/
JOB:PLATE          6-JUN-81      LOADING CASE  1      00:10:55      PAGE  1
 NE                             STRESSES -- TOP
   6  1.396E+04  3.210E+04 -1.338E+03
 *
 *
 *
 *
 * INFST/7/
JOB:PLATE          6-JUN-81      LOADING CASE  1      00:11:18      PAGE  1
 NE                             STRESSES -- TOP
   7  1.396E+04  3.210E+04  1.338E+03
 *
-
```

```
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 * INFST/1,1000/
JOB:PLATE          6-JUN-81      LOADING CASE  1      00:13:24       PAGE  1
 NE                             STRESSES -- TOP
    1 -3.137E+03 -8.061E+03 -6.236E+03
    2 -2.028E+03 -1.119E+04 -6.585E+02
    3 -2.028E+03 -1.119E+04  6.585E+02
    4 -3.137E+03 -8.061E+03  6.236E+03
    5  4.573E+03  1.277E+04 -4.017E+03
    6  1.396E+04  3.210E+04 -1.338E+03
    7  1.396E+04  3.210E+04  1.338E+03
    8  4.573E+03  1.277E+04  4.017E+03
    9  4.573E+03  1.277E+04  4.017E+03
   10  1.396E+04  3.210E+04  1.338E+03
   11  1.396E+04  3.210E+04 -1.338E+03
   12  4.573E+03  1.277E+04 -4.017E+03
   13 -3.137E+03 -8.061E+03  6.236E+03
   14 -2.028E+03 -1.119E+04  6.585E+02
   15 -2.028E+03 -1.119E+04 -6.585E+02
   16 -3.137E+03 -8.061E+03 -6.236E+03
 *
 * RESET/ELEMENTS/SCALEDN/100/ PLOT
```

LOADING CASE 1

* RESET
* ELEMENTS
* ROTV/-90/PLOT



MODEL
Y

DEFLS
Y

VIEW DIR.:
0 0 100
VIEWING DIST.
1.000E+20
PLOT LIMITS
X 0.000E+00
1.000E+01
Y 0.000E+00
5.000E+00
Z 0.000E+00
0.000E+00

LOADING CASE 1

* QUIT

MODEL
Z

DEFLS
Z

VIEW DIR.:
0 -100 0
VIEWING DIST.
1.000E+20
PLOT LIMITS
X 0.000E+00
1.000E+01
Y 0.000E+00
5.000E+00
Z 0.000E+00
0.000E+00

```
COMMAND : HELP : QUIT
G I S T > REVIEW/TOC JOB

<> J O B   D A T A B A S E
```

File	Name	Type	Usage
1	PLATE.PAR	GIFTS	PRE-PROCESSING
2	PLATE.FIL	GIFTS	PRE-PROCESSING
3	PLATE.GRD	GIFTS	PRE-PROCESSING
4	PLATE.LIN	GIFTS	PRE-PROCESSING
5	PLATE.MAT	GIFTS	PRE-PROCESSING
6	PLATE.THS	GIFTS	PRE-PROCESSING
7	PLATE.PTS	GIFTS	PRE-PROCESSING
8	PLATE.ELT	GIFTS	PRE-PROCESSING
9	PLATE.ELD	GIFTS	PRE-PROCESSING
10	PLATE.LDS	GIFTS	PRE-PROCESSING
11	PLATE.STG	STAGS	ANALYSIS
12	PLATE.SIN	STAGS	ANALYSIS
13	PLATE.DNS	GIFTS	POSTPROCESSING
14	PLATE.STR	GIFTS	POSTPROCESSING

```
COMMAND : HELP : QUIT
G I S T > CLEAR JOB_
```

```
COMMAND : HELP : QUIT
G I S T > REVIEW/TOC JOB

<> J O B   D A T A B A S E

          -- EMPTY --

COMMAND : HELP : QUIT
G I S T > QUIT

<> Job PLATE Suspended
------------------------------ G I S T ------------------------------

$ _
```

T H E G I S T T U T O R I A L

PART 3:

ARCHITECTURE; GIST SOFTWARE

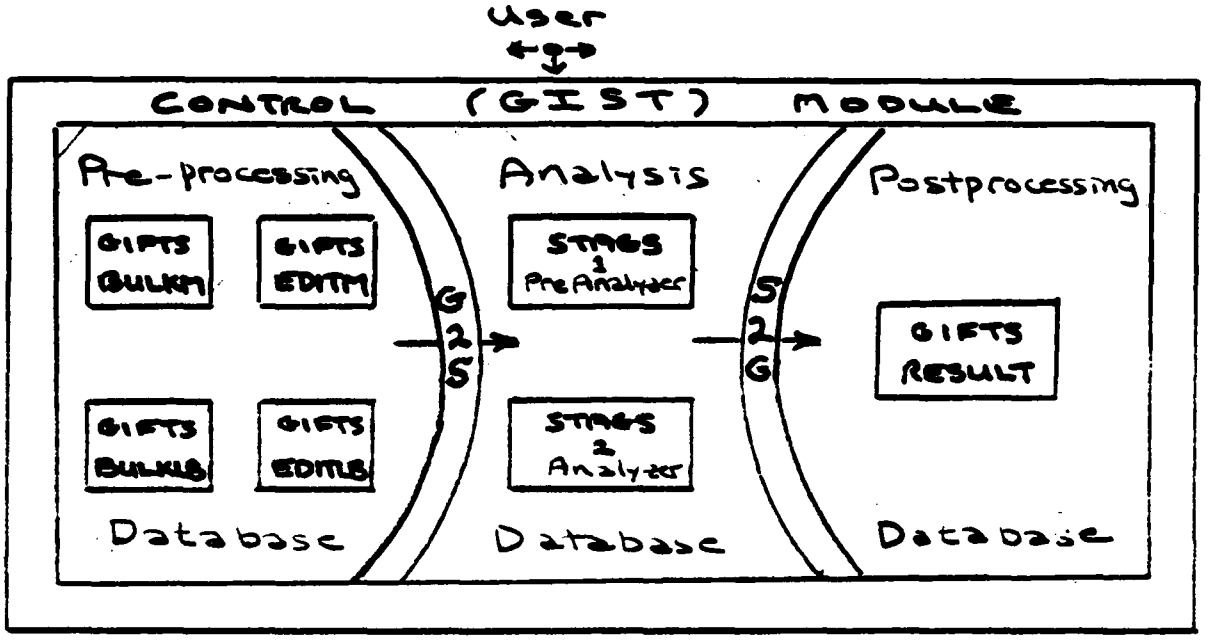
3.0 INTRODUCTION

In this Part of the tutorial, we focus on the actual software comprising the GIST system, and in particular on the architectural components which cement the basic building blocks into an integrated whole. The enclosed material is therefore not intended for the "pure" user, i.e., the analyst, but rather for the developer. By developer, we refer to anyone interested in either continuing or maintaining the development of the GIST system, or even in building other systems which have similar architectural requirements.

A schematic of the software system is illustrated on the following page. Three basic software "families" can be identified: GIFTS (pre/post-processing software), STAGS (structural analysis software), and GIST (architectural software). Each family consists of a number of "processors", i.e., independently executable program modules. Furthermore, the processors within each software family communicate with each other by means of a "database", i.e., a formal collection of named data-sets which are saved, or archived, on permanent disk storage. Notice that the GIST architectural components have access to both the GIFTS and the STAGS databases. It is the database system that makes the concept of "a network of integrated processors" work in practice. The architectural components simply exploit this convenient mechanism, guiding the flow of data both from database to database and between the database and the user, as necessary for compatible and coordinated operations.

In the following sections, each software family will be examined with respect to both processor function and database interaction. Since documentation on GIFTS and STAGS is available elsewhere (e.g., [G3] and [S4]), the most substantial sections will be on the GIST architectural components: the GIFTS->STAGS Adaptor (Section 3.3), the STAGS->GIFTS Adaptor (Section 3.4) and the GIST Control Module (Section 3.5). Included in these latter sections are full source code listings (in FORTRAN 77), which, with a few isolated exceptions are essentially machine independent. These listings may serve as a sample for constructing arbitrary interfaces to GIFTS (as a pre-/post- processing system), and also for developing command-driven processors based on the NICE architectural utility library [N1].

ARCHITECTURAL VIEW OF THE GIST SYSTEM



NOTES:

- Shaded regions represent databases (///=>GIFTS; \\\=>STAGS).
- Non-shaded regions represent software processors.
- Processors have access only to the database (or databases) with which they come in contact in the drawing.

3.1 GIFTS SOFTWARE ARCHITECTURE

This section will, for the time being, constitute nothing more than a pointer to the literature. The reader interested in the underlying organization of the GIFTS software family is referred to the GIFTS Systems Manual [G3]. Also included therein is a detailed discription of the GIFTS database (or Unified DataBase) and the corresponding software utilities (i.e., subroutines) available for accessing it.

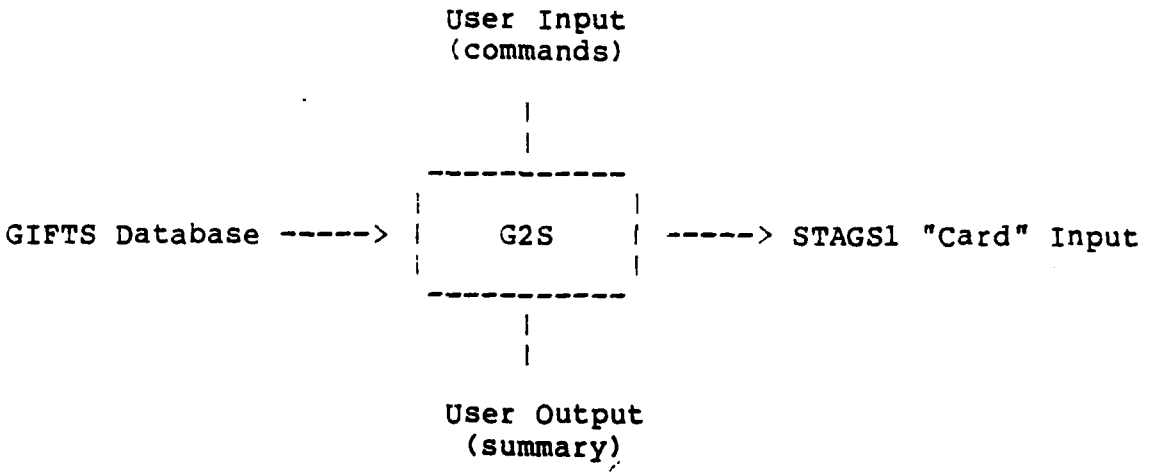
3.2 STAGS SOFTWARE ARCHITECTURE

Due to present plans to perform a massive re-structuring of STAGS software into a system more aligned with the NICE concept of engineering analysis softare-architecture [N1], it is not considered worthwhile, at this time, to go into any detail on the present "state of the architecture". However, at the soonest possible opportunity, a detailed description of the newly implemented DAL database [N2], which was introduced in Part 2 (Section 2.4), will appear in this very section ...

3.3 THE GIFTS->STAGS (PRE-PROCESSING) ADAPTOR: G2S

The G2S processor is the GIST architectural component which constitutes the "pre-processing interface" between GIFTS and STAGS. The term "Adaptor" implies that it renders something suitable for a use other than was originally intended. In fact, G2S "adapts" the GIFTS database definition of a finite element model for utilization in STAGS analysis. The result of this adaptation is a STAGS1 (pre-analyzer) "card-image" input file, precisely in the format specified in the STAGS User's Manual [S1], with the exception of some modifications noted in Appendix B of this tutorial.

The function of G2S may therefore be concisely represented by the following diagram:



The internal structure, or architecture, of the G2S module, i.e., the way it breaks down its primary function into tasks, is summarized in the following list of basic subroutines:

G2SX ..... Main Program ("Executive")
G2SI ..... Initializes program operations
G2SU ..... User (command) interface routine
G2SMS ..... Creates STAGS1 Model Summary data
G2SMR ..... Creates STAGS1 Model Resources (matls/fabrications)
G2SMC ..... Creates STAGS1 Model Configuration (discrete geom.)
G2SML ..... Creates STAGS1 Model Load and B.C. data
G2SESP ..... Generates GIFTS "element stress pointers"
G2SF ..... Finalizes program operations

A listing of the FORTRAN 77 source code for all of the above routines is presented on the following pages (in alphabetical order). Note that a number of GIFTS and STAGS utility subroutines (and COMMON blocks) are employed throughout the processor. Documentation on these routines may be found in [G3] and [N5], respectively.

```
C=DECK G2SESP
      SUBROUTINE G2SESP (IEGIFT,NSTRP,IESTAG)
C
C=PURPOSE   OUTPUT GIFTS ELEMENT STRESS POINTERS
C=AUTHOR    G.M.STANLEY
C=VERSION   NOV24/1980
C=EQUIPMENT INDEPENDENT
C
C                    D E C L A R A T I O N S
C
      COMMON /ELT/ NEU,NES,ISTRKE,IELPLT,IT,IORD,IST,IAPL,NLDRC2,
     1             NCP,NGP,NLFLG,NSTPT,NLAYR,ISTPTR,NMAT,NTHS,
     2             FACT(5),ECCEN,ELTPAD(3),ALPHID,LCP(27)
C
C                          L O G I C
C
      NSTPT = 1
      NLAYR = 1
      IF (IT .GT. 2)   NLAYR = 2
      ISTPTR = NSTRP
      NSTRP  = NSTRP + NSTPT*(NLAYR+1)
      ISTRKE = IESTAG
      CALL  OUT ELT (IEGIFT)
C
      RETURN
      END
```

```
C=DECK G2SF
      SUBROUTINE G2SF
C
C=PURPOSE   FINALIZE OPERATIONS FOR G2S MODULE
C=AUTHOR    G.M.STANLEY
C=VERSION   DEC05/1980
C=EQUIPMENT CDC          VAX
C=KEYWORDS  FINALIZE     G2S
C
C                    C O M M O N
C
      COMMON /STATUS/ ISTAT,ISLG2S,NG2S
C
C.... GIFTS
      COMMON /CMDSLT/ ISLIN,ISLOUT,ISLLST,ISLTTI,ISLTTO,ISLLPT
      COMMON /JOB/ XJOB(2),LJOB(5)
      COMMON /PAR/ LHV(13),LGL(5),LP(25),LS(16),LM(11)
C
C                    E Q U I V A L E N C E
C
       EQUIVALENCE (LS(1), ISTM), (LS(6), ISTLD), (LS(7), ISTBC)
C
C                    D A T A
C
      DATA XG2S /4H G2S/
C
C                    L O G I C
C
      IF (ISTAT .LT. 0)   GO TO 800
C
C.... RESET GIFTS MODEL STATUS PARAMETERS
      ISTM  = 1
      ISTLD = 1
      ISTBC = 1
      CALL  OUT PAR
C
C.... CLOSE 'G2S' FILE
      CALL CLOSEF (ISLG2S)
C
C.... PRINT CLOSING MESSAGE
      IF (NG2S .LE. 0)   WRITE (ISLOUT,2000)
      IF (NG2S .GT. 0)   WRITE (ISLOUT,2001) NG2S
C
      RETURN
C
C.... ERROR EXIT: DELETE 'G2S' FILE
  800 IF (ISLG2S .GT. 0)   CALL DELETE (G2S)
      STOP
C
C                    F O R M A T S
C
 2000 FORMAT (/ 3H <> /)
 2001 FORMAT (/32H <> READY FOR STAGS PRE-ANALYSIS
     1          3H  [,I5,16H INPUT RECORDS ]/)
C
      END
```

```
C=DECK G2SI
      SUBROUTINE G2SI
C
C=PUROSE    INITIALIZE OPERATIONS FOR G2S MODULE
C=AUTHOR    G.M.STANLEY  [ JAN 1980 ]
C=VERSION   1.0
C=EQUIPMENT CDC         VAX
C
C=UPDATED   JAN30/1981
C
C                     C O M M O N
C
      COMMON /JOB/ XJOB(2),LJOB(5)
      COMMON /G2SVER/ G2SVER(2)
      COMMON /STATUS/ ISTAT,ISLG2S,NG2S
      COMMON /OPTION/ LISTOP
C
C.... GIFTS
      COMMON /CMDSLT/ ISLIN,ISLOUT,ISLLST,ISLTTI,ISLTTO,ISLLPT
      COMMON /DISKIO/ INUSER(10),INUSES(3),IFS,IFN
      COMMON /PAR/ LHV(13),LGL(5),LP(25),LS(16),LM(11)
      COMMON /PTS/ NU,NS,ISTRKT,ISTIPLT,VCSMC(9),WRKPAD(8),
     1             LMN,INTP,NLDREC,NBL,NFR,MFP
      COMMON /PTSBUF/ ISLPTS,LOCPTS,NBPTS,LPTSSZ(5),
     1                IBPTS(102),FBPTS(170)
      COMMON /ELTBUF/ ISLELT,LOCELT,NBELT,LELTSZ(5),
     1                IBELT(252),FBELT(100)
      COMMON /MATBUF/ ISLMAT,LOCMAT,NBMAT,LMATSZ(5),
     1                IBMAT(27),FBMAT(55)
      COMMON /THSBUF/ ISLTHS,LOCTHS,NBTHS,LTHSSZ(5),
     1                IBTHS(32),FBTHS(75)
      COMMON /LDSBUF/ ISLLDS,LOCLDS,NBLDS,LLDSSZ(5),
     1                IBLDS(2),FBLDS(80)
      COMMON /ELDBUF/ ISLELD,LOCELD,NBELD,LELDSZ(5),
     1                IBELD(32),FBELD(160)
C
C.... STAGS
      COMMON /STAGS/ NSTGM,NSTGF1,NSTGF2,NSTGN,NSTGE(4),NSTGL
C
C                D I M E N S I O N  /  T Y P E
C
      CHARACTER*24   CASENAM, CCLVAL
      INTEGER        ICLIP(4)
      LOGICAL        PRESNT
C
C                     E Q U I V A L E N C E
C
      EQUIVALENCE (LS(1), ISTM ), (LS(6), ISTLD)
      EQUIVALENCE (LS(7), ISTBC), (LS(2), ISTPM)
C
C                          D A T A
C
      DATA ICLIP /4*0/
C
      DATA   ISTAT,   XG2S,   XPAR
     1     /     0, 4H G2S, 4H PAR /
      DATA   XLDS / 4H LDS /
C
```

```
C
C                          L O G I C
C
C.... INITIALIZE I/O DEVICE NUMBERS
      CALL  INITIO
      ISLIN  = 5
      ISLOUT = 6
      ISLTTI = 5
      ISLTTO = 6
      ISLNET = 25
C
C.... PRINT OPENING MESSAGE
      WRITE (ISLOUT,2000) G2SVER(1)
C
C.... READ CASE NAME
C
C *** CHECK FOR NETWORK CONTROL
      CALL  CLMODE (ICLIP,1,0)
C *** READ (ISLNET,1000,END=10,ERR=10) CASENAM
C *** WRITE (ISLOUT,2001) CASENAM
C *** GO TO 20
C
   10 CALL CLNEXT (' JOB: ', '1 ', ITEMS)
      IF (ITEMS .EQ. 0)   GO TO 800
      CASENAM = CCLVAL (1)
C
   20 CALL CC2H (CASENAM(1:1),XJOB(1),4)
      CALL CC2H (CASENAM(5:5),XJOB(2),4)
C
C.... CHECK PRESENCE OF GIFTS DATA-BASE
      IF (.NOT.PRESNT(XPAR))              GO TO 801
      CALL  IN PAR
C
C.... CHECK MODEL DEFINITION STATUS
      IF (ISTM.LT.1)                      GO TO 810
C
C *** TEMPORARY:  SET LOAD STATUS SWITCH IF 'LDS' FILE EXISTS
      IF (PRESNT(XLDS))   ISTLD = 1
      IF (ISTLD.LT.1)                     GO TO 820
C
      IF (ISTBC.LT.1)                     GO TO 830
      IF (ISTPM.GT.0) WRITE (ISLOUT,8400)
C
C.... INITIALIZE GIFTS I/O BUFFERS
      LOCPTS = 0
      NBPTS  = 1
      LOCELT = 0
      NBELT  = 1
      LOCMAT = 0
      NBMAT  = 1
      LOCTHS = 0
      NBTHS  = 1
      LOCLDS = 0
      NBLDS  = 1
      LOCELD = 0
      NBELD  = 1
C
C.... CREATE STAGS PREPROCESSING INPUT FILE: 'CASE'.G2S
```

```
      NG2S = 0
C
      RETURN
C
C                    E R R O R   E X I T S
C
  800 ISTAT = -1
      CALL  G2S F
  801 WRITE (ISLOUT,8010)
 8010 FORMAT (/41H >> ERROR >>   GIFTS DATA-BASE NOT FOUND.
     1 5X,6H[G2SI]/)
      ISTAT = -1
      CALL  G2S F
  810 WRITE (ISLOUT,8100)
 8100 FORMAT (/49H >> ERROR >>   GIFTS MODEL NOT GENERATED.  [G2SI])
      ISTAT = -1
      CALL  G2S F
  820 WRITE (ISLOUT,8200)
 8200 FORMAT (/49H >> ERROR >>   GIFTS LOADS NOT GENERATED.  [G2SI])
      ISTAT = -1
      CALL  G2S F
  830 WRITE (ISLOUT,8300)
 8300 FORMAT (/44H >> ERROR >>   GIFTS BOUNDARY CONDITIONS NOT
     1         20H INTRODUCED.  [G2SI])
      ISTAT = -1
      CALL  G2S F
 8400 FORMAT (/52H >> WARNING >>   GIFTS POINT MASSES WILL BE IGNORED.
     1          8H  [G2SI])
C
C                       F O R M A T S
C
 1000 FORMAT (A9)
 2000 FORMAT (//12H <<>>  G 2 S  ,11X,25H [ GIFTS->STAGS ADAPTOR ]
     1,                           14X, 6H GIST/A4 )
 2001 FORMAT (7H JOB:  A9 /)
C
      END
```

```
C=DECK G2SMC
      SUBROUTINE G2SMC
C
C=PURPOSE   TRANSFORM GIFTS MODEL CONFIGURATION DATA TO STAGS INPUT
C=AUTHOR    G.M.STANLEY
C=VERSION   APR 20 1981
C=EQUIPMENT CDC          VAX
C=KEYWORDS  G2S          MODEL       CONFIGURATION
C=KEYWORDS  GIFTS        STAGS       ADAPTOR
C
C=BLOCK     ABSTRACT
C
C        G2SMC TRANSFORMS GIFTS NODE AND ELEMENT DEFINITION DATA
C        INTO THE CORRESPONDING STAGS (GENERAL UNIT) INPUT DATA.
C        THE INPUT RECORDS ARE GENERATED ON FILE 'CASE'.G2S.
C
C=END       ABSTRACT
C
C                    C O M M O N
C
      COMMON /STATUS/ ISTAT,ISLG2S,NG2S
      COMMON /FAC/ FAT(20,20)
      INTEGER FAT
      COMMON /THSX/ A,AQ,AP,IPP,IQQ,JJ,ZG1,YG1,ZG2,YG2,
     1                                 ZO1,YO1,ZO2,YO2,ALFA
      REAL IPP,IQQ,JJ
C
C.... GIFTS
      COMMON /CMDSLT/ ISLIN,ISLOUT,ISLLST,ISLTTI,ISLTTO,ISLLPT
      COMMON /PAR/ LHV(13),LGL(5),LP(25),LS(16),LM(11)
      COMMON /PTS/ NU,NS,ISTRKP,IPTPLT,VCSMC(9),PTSPAD(8),
     1             LMN,INTP,NLDREC,NBL,NFR,MFP
      COMMON /ELT/ NEU,NES,ISTRKE,IELPLT,IT,IORD,IST,IAPL,NLDRC2,
     1             NCP,NGP,NLFLG,NSTPT,NLAYR,ISTPTR,NMAT,NTHS,FACT(5),
     2             ECCEN,ELTPAD(3),ALPHID,LCP(27)
      COMMON /MAT/ MATPTR,IMT,LCOLRM(3),PMAT(11)
      COMMON /THS/ ITHPTR,ITT,IPTRCS,LCOLRT(3),TH(15)
C
C.... STAGS
      COMMON /STAGS/ NSTGM,NSTGF1,NSTGF2,NSTGN,NSTGE(4),NSTGL
      COMMON /STGE/ LCPS(10)
      COMMON /STGN/ XS(3),DS(6)
C
C                    D I M E N S I O N
C
      DIMENSION THX(15),LF(12)
C
C                    E Q U I V A L E N C E
C
      EQUIVALENCE (THX(1), A    ), (LP(1), NGPT ), (LP(2), NGPA)
      EQUIVALENCE (LP(3), NELTT ), (LP(8), NMATT), (LP(9), NTHST)
      EQUIVALENCE (LP(11), NBCPT), (TH(15),THETA), (LP(18), NSTRP)
C
C                    D A T A
C
      DATA   DEG2RAD/.017453293/, RAD2DEG/57.29577951/
      DATA   BLANK,  E210,  E300,  E320,  E410
     1    / 4H    , 3H210, 3H300, 3H320, 3H410 /
C
```

```
C                    L O G I C
C
C.... OPEN GIFTS DATA-BASE
      CALL  OPN PTS
      CALL  OPN ELT
      CALL  OPN THS
      CALL  OPN MAT
C
C     ----------------------
C     GENERATE NODAL RECORDS
C     ----------------------
C
      DO 100 IN=1,NGPT
C
C     .... INPUT GIFTS NODAL RECORD
           IR = IN
           CALL  IN PTS (IR)
C
C     .... INITIALIZE STAGS NODAL RECORD
           DO 30 I=1,3
   30      XS(I) = VCSMC(I)
           DO 40 I=1,6
   40      LF(I) = 0
           IAUX  = 0
C
C     .... IGNORE DOF FOR NON-ACTIVE NODES
           IF (NU.LE.0)                  GO TO 70
           IF (NU.EQ.32767)              GO TO 70
C
C     .... TRANSFER DOF PATTERN
           CALL  UPF (MFP,LF)
           IF (LMN.GT.0)   IAUX = 1
C
C     .... WRITE 'PRIMARY NODAL' RECORD (S-1)
   70      WRITE (ISLG2S,2001)  IN,XS,(LF(I),I=1,6),IAUX
           NG2S = NG2S + 1
C
C     .... GET AUXILIARY COORDINATE TRANSFORMATION
C     .... (COLUMNS OF MATRIX = ROTATED UNIT VECTORS)
           IF (IAUX.LE.0)                GO TO 100
           CALL  IN PTS (LMN)
           DO 80 I=1,6
           DS(I) = VCSMC(I)
C *
C **
C *** COMPENSATE FOR GIFTS "ERROR" -- LEFT-HANDED TRANSFR. MATRIX
C **
C *
           IF (I.GT.3)   DS(I) = -DS(I)
C
   80 CONTINUE
C
C     .... WRITE 'AUXILLIARY NODAL' RECORD (S-2)
           WRITE (ISLG2S,2002)   DS
           NG2S = NG2S + 1

C
  100 CONTINUE
```

```
C     ------------------------
C     GENERATE ELEMENT RECORDS
C     ------------------------
C
      NT0 = 0
      NM0 = 0
      IESTAG = 0
      NSTRP = 1
C
C     ---------------
C     LUMPED ELEMENTS (INOPERATIONAL)
C     ---------------
      IF (NSTGE(1).EQ.0)                    GO TO 200
      GO TO 200
C
C     ------------
C     1-D ELEMENTS
C     ------------
  200 IF (NSTGE(2).EQ.0)                    GO TO 300
      NE2 = 0
      DO 290 IE=1,NELTT
           IEGIFT = IE
C
C     .... INPUT GIFTS ELEMENT RECORD
           CALL  IN ELT (IE)
           IF (NEU.LE.0)                    GO TO 290
           IF (IAPL.NE.0)                   GO TO 290
           IF (IT.NE.2)                     GO TO 290
C
C     .... STRAIGHT BEAMS
C
           BM = E210
C ***      IF (ALPHID.NE.BLANK)    BM = E210
           DO 207 I=1,3
  207      LCPS(I) = LCP(I)
           ICROSS = FAT(NTHS,NMAT)
C
C     .... DEFINE CROSS-SECTION ORIENTATION
           IF (NTHS.EQ.NT0)                 GO TO 220
           CALL  IN THS (NTHS)
           IF (ITHPTR.NE.NTHS)   CALL  IN THS (ITHPTR)
           DO 210 I=1,15
                THX(I) = TH(I)
                TH(I)  = 0.
  210      CONTINUE
           IF (IPTRCS.GT.0)   CALL  IN THS (IPTRCS)
           XSI = - (ALFA*RAD2DEG + 90.)

           S = SIN (THETA*DEG2RAD)
           C = COS (THETA*DEG2RAD)
           ECY =  ZG1*C - YG1*S
           ECZ = -ZG1*S - YG1*C
           NT0 = NTHS
C
C     .... SET NONLINEARITY SWITCHES
C
C *** TEMPORARY CORRECTION TO BEAM DEFAULT SWITCH SETTINGS
  220      IF (NLFLG.EQ.3)   NLFLG = 0
           ILIN = NLFLG/2
```

```
            IPLAS = NLFLG - 2*ILIN
            IPLAS = 1 - IPLAS
            CALL  IN MAT (NMAT)
            IF (MATPTR.NE.NMAT)   CALL  IN MAT (MATPTR)
            IF (IMT .NE. 5)   IPLAS = 0
C
C     .... WRITE 'BEAM ELEMENT' RECORD (T-2)
            WRITE (ISLG2S,2003)  (LCPS(I),I=1,3),BM,ICROSS,
     1                            XSI,ECY,ECZ,ILIN,IPLAS
            NG2S = NG2S + 1
            IESTAG  = IESTAG + 1
            NE2  = NE2 + 1
C
C..... OUTPUT GIFTS ELEMENT STRESS POINTER
            CALL  G2S ESP (IEGIFT,NSTRP,IESTAG)
            IF (NE2.GE.NSTGE(2))                    GO TO 300
C
  290 CONTINUE
C
C     --------------------------
C     2-D ELEMENTS -- TRIANGULAR
C     --------------------------
  300 IF (NSTGE(3).EQ.0)                            GO TO 400
      NE3 = 0
      DO 390 IE=1,NELTT
            IEGIFT = IE
C
C     .... INPUT GIFTS ELEMENT RECORD
            CALL  IN ELT (IE)
            IF (NEU.LE.0)                           GO TO 390
            IF (IAPL.NE.0)                          GO TO 390
            IF (IT.NE.3.AND.IT.NE.4)                GO TO 390
C
C     .... STRAIGHT TRIANGLES
C
            IF (IT.EQ.3)   TRI = E300
            IF (IT.EQ.4)   TRI = E320
C ***       IF (ALPHID.NE.BLANK)   TRI = ALPHID
            DO 307 I=1,3
  307       LCPS(I) = LCP(I)
            IWALL = FAT(NTHS,NMAT)
            ZETA  =  FACT(5)
            ECZ   = -ECCEN
C
C     .... SET NONLINEARITY SWITCHES
            ILIN  = NLFLG/2
            IPLAS = NLFLG - 2*ILIN
            IPLAS = 1 - IPLAS
            CALL  IN MAT (NMAT)
            IF (MATPTR.NE.NMAT)   CALL  IN MAT (MATPTR)
            IF (IMT .NE. 5)   IPLAS = 0
C
C     .... WRITE 'TRIANGULAR ELEMENT' RECORD (T-3)
            WRITE (ISLG2S,2004)  (LCPS(I),I=1,3),TRI,IWALL,
     1                            ZETA,ECZ,ILIN,IPLAS
            NG2S = NG2S + 1
            IESTAG  = IESTAG  + 1
            NE3  = NE3  + 1
```

```
C......... OUTPUT GIFTS ELEMENT STRESS POINTER
           CALL  G2S ESP (IEGIFT,NSTRP,IESTAG)
           IF (NE3.GE.NSTGE(3))                    GO TO 400
C
  390 CONTINUE
C
C     ----------------------------
C     2-D ELEMENTS -- QUADRILATERAL
C     ----------------------------
  400 IF (NSTGE(4).EQ.0)                           GO TO 500
      NE4 = 0
      DO 490 IE=1,NELTT
           IEGIFT = IE
           CALL  IN ELT (IE)
           IF (NEU.LE.0)                           GO TO 490
           IF (IAPL.NE.0)                          GO TO 490
           IF (IT.NE.5.AND.IT.NE.6)                GO TO 490
C
C     .... STRAIGHT QUADRILATERALS
C
           QUAD = E410
C ***      IF (ALPHID.NE.BLANK)   QUAD = ALPHID
           LCPS(1) = LCP(1)
           LCPS(2) = LCP(2)
           LCPS(3) = LCP(4)
           LCPS(4) = LCP(3)
           IWALL = FAT(NTHS,NMAT)
           ZETA  = FACT(5)
           ECZ   = -ECCEN
C
C     .... SET NONLINEARITY SWITCHES
           ILIN  = NLFLG/2
           IPLAS = NLFLG - 2*ILIN
           IPLAS = 1 - IPLAS
           CALL  IN MAT (NMAT)
           IF (MATPTR.NE.NMAT)   CALL  IN MAT (MATPTR)
           IF (IMT .NE. 5)   IPLAS = 0
C
C     .... WRITE 'QUADRILATERAL ELEMENT' RECORD (T-4)
           WRITE (ISLG2S,2005)  (LCPS(I),I=1,4),QUAD,IWALL,
     1                           ZETA,ECZ,ILIN,IPLAS
           NG2S = NG2S + 1
           IESTAG  = IESTAG  + 1
           NE4  = NE4  + 1
C
C......... OUTPUT GIFTS ELEMENT STRESS POINTER
           CALL  G2SESP (IEGIFT,NSTRP,IESTAG)
           IF (NE4.GE.NSTGE(4))                    GO TO 500
C
  490 CONTINUE
C
  500 CONTINUE
C
C .... CLOSE GIFTS DATA-BASE
       CALL  CLS PTS
       CALL  CLS ELT
       CALL  CLS THS
       CALL  CLS MAT
C
```

```
      RETURN
C
C                    F O R M A T S
C
 2001 FORMAT (1X,I5,4H,,,, 3(5X,E10.4),2(1X,3I1),1X,I1,1X, 5H$ S-1)
 2002 FORMAT (6(2X,E10.4))
 2003 FORMAT (1X,3I5,1X,A3,1X,I3,3(2X,E10.4),2(1X,I1),2X, 5H$ T-2)
 2004 FORMAT (1X,3I5,1X,A3,1X,I3,2(2X,E10.4),12X,2(1X,I1),2X, 5H$ T-3)
 2005 FORMAT (1X,4I5,1X,A3,1X,I3,1H, 36X,5H$ T-4,
     1        /25X,2(2X,E10.4),2(1X,I1),1X, 1H$)
C
      END
```

```
C=DECK G2SML
      SUBROUTINE G2SML
C
C=PURPOSE   TRANSFORM GIFTS LOAD DATA TO STAGS INPUT
C=AUTHOR    G.M.STANLEY
C=VERSION   DEC  5 1980
C=UPDATED   MAY 29 1980 [G.M.STANLEY]
C=EQUIPMENT CDC          VAX
C=KEYWORDS  G2S          LOADS
C=KEYWORDS  GIFTS        STAGS        ADAPTOR
C
C=BLOCK     ABSTRACT
C
C        G2SML TRANSFORMS GIFTS SPECIFIED FORCE AND DISPLACEMENT
C        DATA INTO THE CORRESPONDING STAGS (LOAD SYSTEM) INPUT DATA.
C        THE INPUT RECORDS ARE GENERATED ON FILE 'JOB'.G2S.
C
C=END       ABSTRACT
C
C                    C O M M O N
C
      COMMON /STATUS/ ISTAT,ISLG2S,NG2S
C
C.... GIFTS
      COMMON /CMDSLT/ ISLIN,ISLOUT,ISLLST,ISLTTI,ISLTTO,ISLLPT
      COMMON /PAR/ LHV(13),LGL(5),LP(25),LS(16),LM(11)
      COMMON /PTS/ NU,NS,ISTRKT,IPTPLT,VCSMC(9),PTSPAD(8),
     1             LMN,INTP,NLDREC,NBL,NFR,MFP
      COMMON /LDS/ PV(8)
      COMMON /ELT/ NEU,NES,ISTRKE,IELPLT,IT,IORD,IST,IAPL,NLDRC2,
     1             NCP,NGP,NLFLG,NSTPT,NLAYR,ISTPTR,NMAT,NTHS,FACT(5),
     2             ECCEN,ELTPAD(3),ALPHID,LCP(27)
      COMMON /ELD/ NELDE,NELDC,NELDT,NELDP,NELDNP,NELDNC,ELDV(3,8),
     1             ELDVR(3),ELDVRM(3),ELDRES,ELDREM
C
C.... STAGS
      COMMON /STAGS/ NSTGM,NSTGF1,NSTGF2,NSTGN,NSTGE(4),NSTGL
C
C                    D I M E N S I O N
C
      DIMENSION NRECL(2),NRECI(2),LF(12),PL(6),TL2G(3,3)
      LOGICAL   PRESNT
C
C                    E Q U I V A L E N C E
C
      EQUIVALENCE (PV(1), LDTYPE), (PV(2) , VAL ), (PV(4) , LIVE)
      EQUIVALENCE (ISTRKE,IESTAG)
      EQUIVALENCE (LP(1), NGPT  ), (LP(14), NLCA), (LP(3), NELTG)
C
C                    D A T A
C
      DATA   XELD/4H ELD/
C
C                    L O G I C
C
C.... OPEN GIFTS DATA-BASE
      CALL  OPN LDS
      CALL  OPN PTS
      IELD = 0
```

```
      IF (.NOT.PRESNT(XELD))   GO TO 10
      CALL  OPN ELD
      CALL  OPN ELT
      IELD = 1
C
C     ------------------------------------------------------------
C     P R E L I M I N A R Y   P A S S  --  C O U N T  R E C O R D S
C     ------------------------------------------------------------
C
   10 NSTGL = 0
      NSTGI = 0
      DO 20 I=1,2
      NRECL(I) = 0
   20 NRECI(I) = 0
      NLIVE = 0
C
C.... EXAMINE GIFTS "LOAD-CASES"
C
      DO 200 LCASE = 1,NLCA
      CALL  IN LDS (0,LCASE)
      IF (LDTYPE.EQ.0)   GO TO 220
      IF (LDTYPE.EQ.1 .AND. VAL.LT.0)   GO TO 270
      GO TO 810
C
C
C.... "LOAD SYSTEMS"
C
C
  220 IF (NSTGL .GE. 2)   GO TO 291
      NSTGL = NSTGL + 1
      ILIVE = LIVE
      IF (ILIVE .GT. 0)    NLIVE = NLIVE + 1
      IF (NLIVE .GT. 1)   GO TO 820
C
C.... COUNT NODAL LOAD RECORDS
C
      DO 100 IN = 1, NGPT
      CALL  IN PTS (IN)
C
C.... SKIP DELETED AND "BOUNDARY CONDITON" PTS
      IF (NU .LE. 0)       GO TO 100
      IF (NU .EQ. 32767)   GO TO 100
C
      CALL  IN LDS (IN,LCASE)
C
C.... UNPACK GIFTS NODAL DOF PATTERN
      CALL  UPF (MFP,LF)
C
      IF (LMN .LE. 0)   GO TO 50
C
C.... Transform "Forces" from Global to Local
C
      IAUX = LMN
      CALL  IN PTS (IAUX)
      K = 0
      DO 302 J=1,3
      DO 302 I=1,3
      K = K + 1
      TL2G(I,J) = VCSMC(K)
```

```
C
C.... COMPENSATE FOR GIFTS TRANSFORMATION ERROR
      IF (J.EQ.2)   TL2G(I,J) = -TL2G(I,J)
C
  302 CONTINUE
C
      DO 310 I=1,3
      PL(I) = 0.
      PL(I+3) = 0.
      DO 310 J=1,3
      K = J + 6
      IF (LF(K)  .EQ.0)   PL(I)   = PL(I)   + TL2G(J,I) * PV(J)
      IF (LF(K+3).EQ.0)   PL(I+3) = PL(I+3) + TL2G(J,I) * PV(J+3)
  310 CONTINUE
C
      DO 320 I=1,6
  320 IF (LF(I+6).EQ.0)   PV(I) = PL(I)
C
C.... OUTPUT (OVERWRITE) "TRANSFORMED" GIFTS NODAL LOAD RECORD
      CALL  OUT LDS (IAUX,LCASE)
C
C.... COUNT ACTIVE LOAD COMPONENTS
C
   50 DO 60 ID = 1, 6
      IF (LF(ID+6))   30, 30, 40
C
C.... Specified Force
   30 IF (PV(ID).NE.0.)   NRECL(NSTGL) = NRECL(NSTGL) + 1
      GO TO 60
C
C.... Specified Displacement
   40 NRECL(NSTGL) = NRECL(NSTGL) + 1
   60 CONTINUE
  100 CONTINUE
C
C..... COUNT ELEMENT ("LIVE") LOAD RECORDS
C
      IF (ILIVE.LE.0 .OR. IELD.LE.0)   GO TO 200
      DO 250 IE = 1, NELTG
C
C.... Input GIFTS Element Directory Record
      CALL  IN ELT (IE)
C
      IF (NEU .LE. 0)       GO TO 250
      IF (NLDRC2 .EQ. 0)    GO TO 250
C
C.... Input GIFTS Element Load Record
      CALL  IN ELD (NLDRC2)
C
  255 IF (NELDC .EQ. LCASE)   GO TO 260
      IF (NELDNP .EQ. 0  .OR.  NELDNC .EQ. 0)   GO TO 250
      CALL  IN ELD (NELDNP)
      GO TO 255
  260 IF (NELDT .NE. 3)   GO TO 250
C
      W = 0.
      DO 263 I=1,NCP
      IF (ELDV(3,I) .EQ. 0.)   GO TO 263
      W = W + ELDV(3,I)
```

```
  263 CONTINUE
C
      IF (W .NE. 0.)    NRECL(NSTGL) = NRECL(NSTGL) + 1
C
  250 CONTINUE
      GO TO 200
C
C
C.... INITIAL CONDITIONS
C
C
  270 IF (NSTGI .GE. 2)    GO TO 292
      NSTGI = NSTGI + 1
      DO 275 IN = 1, NGPT
      CALL  IN PTS (IN)
      IF (NU .EQ. 32767)    GO TO 275
      CALL  IN LDS (IN,LCASE)
      DO 280 ID = 1, 6
  280 IF (PV(ID) .NE. 0.)    NRECI(NSTGI) = NRECI(NSTGI) + 1
  275 CONTINUE
      GO TO 200
C
  291 WRITE (ISLOUT,9001)     LCASE
      GO TO 200
  292 WRITE (ISLOUT,9002)     LCASE
C
  200 CONTINUE
C
C.... Write STAGS "Load Summary" Record
      WRITE (ISLG2S,2001)    NSTGL, NSTGI
      NG2S = NG2S + 1
C
C      ------------------------------------------------------
C      F I N A L   P A S S   --   G E N E R A T E   R E C O R D S
C      ------------------------------------------------------
C
      ISTGL = 0
      ISTGI = 0
      DO 500 LCASE = 1,NLCA
      CALL  IN LDS (0,LCASE)
      IF (LDTYPE .EQ. 0)    GO TO 410
      IF (LDTYPE.EQ.1  .AND.  VAL.LT.0.)    GO TO 510
      GO TO 810
C
C
C.... "LOAD SYSTEM" (GENERATION)
C
C
  410 IF (ISTGL .GE. 2)    GO TO 500
      ISTGL = ISTGL + 1
      ILIVE = LIVE
      IFLG  = 0
      IF (ILIVE .GT. 0)    IFLG = 2
C
C.... Write Load-System Summary Record
      WRITE (ISLG2S,2002)    ISTGL,NRECL(ISTGL),IFLG
      NG2S = NG2S + 1
C
C     ------------------------------------------------------
```

```
C
      DO 300 IN=1,NGPT
C
      CALL  IN PTS (IN)
C
C.... SKIP DELETED AND "BOUNDARY CONDITION" PTS
      IF (NU .LE. 0)        GO TO 300
      IF (NU .EQ. 32767)    GO TO 300
C
      CALL  IN LDS (IN,LCASE)
C
C.... UNPACK GIFTS NODAL DOF PATTERN
      CALL  UPF (MFP,LF)
C
C.... INPUT "TRANSFORMED" GIFTS NODAL LOAD RECORD
      IF (LMN .GT. 0)    CALL  IN LDS (LMN,LCASE)
C
      DO 360 ID=1,6
      LFI = 1 - LF(ID+6)
      LT  = 2*LFI - 1
      IF (LT.EQ.1.AND.PV(ID).EQ.0.)     GO TO 360
C
C.... Write STAGS Nodal Load Record
      IE = 0
      WRITE (ISLG2S,2003)  PV(ID), LT, ID, IN, IE
      NG2S = NG2S + 1
  360 CONTINUE
  300 CONTINUE
C
      IF (ILIVE.LE.0 .OR. IELD.LE.0)    GO TO 500
C
C.... GENERATE ELEMENT ("LIVE") LOAD RECORDS
C
      DO 450 IE = 1, NELTG
C
      CALL  IN ELT (IE)
      IF (NEU .LE. 0)       GO TO 450
      IF (NLDRC2 .EQ. 0)    GO TO 450
C
      CALL  IN ELD (NLDRC2)
  420 IF (NELDC .EQ. LCASE)     GO TO 430
      IF (NELDNP .EQ. 0  .OR.  NELDNC .EQ. 0)    GO TO 450
      CALL  IN ELD (NELDNP)
      GO TO 420
  430 IF (NELDT .NE. 3)         GO TO 450
C
      W = 0.
      DO 433 I=1,NCP
      IF (ELDV(3,I) .EQ. 0.)    GO TO 433
      W = W + ELDV(3,I)
  433 CONTINUE
      IF (W .EQ. 0.)    GO TO 450

      LT = 3
      ID = 3
      IN = 0
      PV = W/NCP

      C.... STAGS Element Load Record
```

```
      WRITE (ISLG2S,2003)   PP,LT,ID,IN,IESTAG
      NG2S = NG2S + 1
C
  450 CONTINUE
      GO TO 500
C
C
C.... INITIAL CONDITIONS (GENERATION)
C
C
  510 IF (ISTGI .GE. 2)   GO TO 500
      ISTGI = ISTGI + 1
      ICFLG = VAL + 1
C
C.... Write Initial-Condition Summary Record
      IFLG = 0
      WRITE (ISLG2S,2002)   ICFLG, NRECI(ISTGI), IFLG
      NG2S = NG2S + 1
C
C,,,, GENERATE NODAL DISPLACEMENT/VELOCITY RECORDS
C
      DO 520 IN = 1, NGPT
C
      CALL  IN PTS (IN)
      IF (NU .LE. 0  .OR.  NU .EQ. 32767)   GO TO 520
      CALL  IN LDS (IN,LCASE)
C
      LT = -1
      DO 530 ID = 1, 6
      IF (PV(ID) .EQ. 0.)   GO TO 530
C
C.... Write Nodal Displ./Veloc. Record
      IE = 0
      WRITE (ISLG2S,2003)   PV(ID), LT, ID, IN, IE
C
  530 CONTINUE
  520 CONTINUE
C
  500 CONTINUE
C
C
C
C.... WRITE "TEMPORARY" 'OUTPUT CONTROL' RECORD (V-1)
      WRITE (ISLG2S,2005)
      NG2S = NG2S + 1
C
C.... CLOSE GIFTS DATA-BASE
      CALL  CLS LDS
      CALL  CLS PTS
      IF (IELD .EQ. 0)   RETURN
      CALL  CLS ELD
      CALL  CLS ELT
C
      RETURN
C
C             E R R O R   E X I T S
C
  810 WRITE (ISLOUT,8100)  LCASE,LDTYPE
 8100 FORMAT (/52H >> ERROR >>   UNADAPTABLE LOAD-CASE TYPE.  [G2SML!
```

```
     1        /15X, 9HLOAD CASE, 3X, 9HLOAD TYPE/19X,I5, 2X,I5/
     2        /20X, 7H VAL = ,E12.4,5X,7H LIVE =,I4/)
      ISTAT = -1
      CALL  G2S F
C
  820 WRITE (ISLOUT,8200)  LCASE
 8200 FORMAT (/49H >> ERROR >>   "LIVE" LOADS DETECTED IN MORE THAN
     1         23H ONE ACTIVE LOAD-SYSTEM//20X,12H LOAD-CASE = I5/
     2         12H << G2SML <</)
      ISTAT = -1
      CALL  G2S F
C
C                    F O R M A T
C
 2001 FORMAT (2(1X,I5),54X, 5H$ U-1)
 2002 FORMAT (3(1X,I5),48X, 5H$ U-2)
 2003 FORMAT (1X,5X,E10.4,4(1X,I5),26X, 5H$ U-3)
 2005 FORMAT (1X,6(4X,1H0),35X,5H$ EOF)
C
 9001 FORMAT (/51H >> WARNING >>   ONLY TWO "LOAD-SYSTEMS" PERMITTED;
     1         /16X,              20H GIFTS LOAD-CASE NO. I5,9H IGNORED./)
 9002 FORMAT (/48H >> WARNING >>   ONLY TWO "I.C. SETS" PERMITTED;
     1         /16X,              20H GIFTS LOAD-CASE NO. I5,9H IGNORED./)
C
      END
```

```
C=DECK G2SMR
      SUBROUTINE G2SMR
C
C=PURPOSE   TRANSFORM GIFTS MODEL RESOURCE DATA TO STAGS INPUT
C=AUTHOR    G.M.STANLEY
C=VERSION   JAN18/1981
C=EQUIPMENT CDC          VAX
C=KEYWORDS  G2S          MODEL       RESOURCES
C
C=BLOCK     ABSTRACT
C
C        G2SMR TRANSFORMS GIFTS MATERIAL AND THICKNESS GROUP DATA
C        INTO STAGS MATERIAL AND FABRICATION DATA, GENERATING
C        THE APPROPRIATE INPUT RECORDS ON FILE 'CASE'.G2S.
C
C=END       ABSTRACT
C
C                  C O M M O N
C
      COMMON /STATUS/ ISTAT,ISLG2S,NG2S
      COMMON /FAC/ FAT(20,20)
      INTEGER FAT
      COMMON /THSX/ A,AQ,AP,IPP,IQQ,JJ,ZG,YG,ZG2,YG2,ZO,YO,ZO2,YO2,ALFA
      REAL IPP,IQQ,JJ
C
C.... GIFTS
      COMMON /CMDSLT/ ISLIN,ISLOUT,ISLLST,ISLTTI,ISLTTO,ISLLPT
      COMMON /PAR/ LHV(13),LGL(5),LP(25),LS(16),LM(11)
      COMMON /MAT/ MATPTR,IMT,LCOLRM(3),PMAT(11)
      COMMON /THS/ ITHPTR,ITYPE,IPTRCS,LCOLRT(3),TH(15)
C
C.... STAGS
      COMMON /STAGS/ NSTGM,NSTGF1,NSTGF2,NSTGN,NSTGE(4),NSTGL
      COMMON /STGF1/ SA,SY,SZ,SIY,SIZ,SIYZ,ISO
      COMMON /STGF2/ MATL,TL,ZETL,LSO
C
C                  D I M E N S I O N
C
      DIMENSION THX(15)
C
C                  E Q U I V A L E N C E
C
      EQUIVALENCE (A, THX(1))
      EQUIVALENCE (LP(8),NMATT  ), (LP(9 ),NTHST )
      EQUIVALENCE (PMAT(1), NSEG), (PMAT(2), NREC)
      EQUIVALENCE (PMAT(1), E   ), (PMAT(2), VNU ), (PMAT(3), G)
      EQUIVALENCE (PMAT(5), RHO ), (PMAT(7), TEX ), (TH(15) , THETA)
      EQUIVALENCE (PMAT(1), SIG ), (PMAT(4), EPS )
C
C                  D A T A
C
      DATA D2R /.017453293/
C
C
C                  L O G I C
C
C.... OPEN GIFTS DATA-BASE
      CALL  OPN MAT
      CALL  OPN THS
```

```
C
C     ---------------------------------
C     GENERATE MATERIAL PROPERTY RECORDS
C     ---------------------------------
C
      DO 100 IM=1,NMATT
           ITAM = IM
           IR   = IM
           CALL  IN MAT (IR)
           IF (MATPTR.EQ.IR)              GO TO 5
           IF (MATPTR.LE.0)               GO TO 10
           IR = MATPTR
           CALL  IN MAT (IR)
    5      GO TO (10,801,801,40,50,801,801), IMT
C
C          ----------------------------
C          ISOTROPIC MATERIAL -- ELASTIC
C          ----------------------------
C
C     .... WRITE 'MATERIAL HEADER' RECORD (I-1)
   10      WRITE (ISLG2S,2001)  ITAM
           NG2S = NG2S + 1
C
C     .... WRITE 'ELASTIC PROPERTY' RECORD (I-2)
           G = 0.
           WRITE (ISLG2S,2002)  E,VNU,G,RHO,TEX
           NG2S = NG2S + 1
           GO TO 100
C
C          ---------------------------------
C          2-D ORTHOTROPIC MATERIAL -- ELASTIC
C          ---------------------------------
C
C     .... WRITE 'MATERIAL HEADER' RECORD (I-1)
   40      WRITE (ISLG2S,2001)  ITAM
           NG2S = NG2S + 1
C
C     .... WRITE 'ELASTIC PROPERTY' RECORD (I-2)/PART 1
           WRITE (ISLG2S,2003)  E,VNU,G,RHO,TEX
           NG2S = NG2S + 1
C
C     .... ADD PROPERTIES FOR SECOND DIRECTION
           CALL  IN MAT (IR+1)
           WRITE (ISLG2S,2004)  E,TEX
           NG2S = NG2S + 1
           GO TO 100
C
C          ----------------------------
C          NONLINEAR MATERIAL -- PLASTIC
C          ----------------------------
C
   50      NESP = NSEG-1
C
C     .... WRITE 'MATERIAL HEADER' RECORD (I-1)
           WRITE (ISLG2S,2005)  ITAM,NESP
           NG2S = NG2S + 1
C
C     .... WRITE 'ELASTIC PROPERTY' RECORD (I-2)
           IR = IR+1
```

```
            CALL  IN MAT (IR)
C       .... SET STAGS ISOTROPY FLAG: G=0
            G = 0.
            WRITE (ISLG2S,2002)  E,VNU,G,RHO,TEX
            NG2S = NG2S + 1
C
C       .... WRITE 'PLASTICITY CURVE' RECORDS (I-3)
            DO 55 I=1,NESP
                 IR = IR+1
                 CALL  IN MAT (IR)
                 IF (I.LT.NESP)  WRITE (ISLG2S,2006) EPS,SIG
                 IF (I.EQ.NESP)  WRITE (ISLG2S,2007) EPS,SIG
                 NG2S = NG2S + 1
   55       CONTINUE
C
  100 CONTINUE
C
C       ---------------------------------------
C       GENERATE 1-D (BEAM) FABRICATION RECORDS
C       ---------------------------------------
C
        IF (NSTGF1.EQ.0)                    GO TO 250
        ITAB = 0
        DO 210 NT=1,NTHST
        DO 200 NM=1,NMATT
            IF (FAT(NT,NM).NE.-1)           GO TO 200
            ITAB = ITAB+1
            FAT(NT,NM) = ITAB
C
C       .... INPUT GIFTS THICKNESS RECORD
            IR = NT
            CALL  IN THS (IR)
            IF (ITHPTR.EQ.NT)               GO TO 110
            IF (ITHPTR.LE.0)                GO TO 810
            IR = ITHPTR
            CALL  IN THS (IR)
  110       IF (ITYPE.NE.0)                 GO TO 812
C
C       .... CHECK EXISTENCE OF MATERIAL NM
            JR = NM
            CALL  IN MAT (JR)
            IF (MATPTR.LE.0)                GO TO 814
C
C       .... SAVE COMPUTED PROPERTIES, GET USER PROPERTIES
            DO 112 I=1,15
                 THX(I) = TH(I)
                 TH(I)  = 0.
  112       CONTINUE
            IF (IPTRCS.GT.0)   CALL  IN THS (IPTRCS)
C
C       .... WRITE 'CROSS-SECTION HEADER' RECORD (J-1)
            KCROSS = 2
            NSUB   = 1
            MATB   = NM
            TORJ   = JJ
            BETA   = ALFA - THETA*D2R
            SCY    = (ZO-ZG) * COS(BETA)  +  (YO-YG) * SIN(BETA)
            SCZ    = (ZO-ZG) * SIN(BETA)  -  (YO-YG) * COS(BETA)
            WRITE (ISLG2S,2008)  ITAB,KCROSS,MATB,NSUB,SCY,
```

```
            NG2S = NG2S + 1
C
C     .... WRITE 'SUBELEMENT PROPERTY' RECORD (J-3A)
            SA   = A
            SY   = 0.
            SZ   = 0.
            SIY  = IQQ
            SIZ  = IPP
            SIYZ = 0.
            ISO  = 1
            WRITE (ISLG2S,2010)  SA,SY,SZ,SIY,SIZ,SIYZ,ISO
            NG2S = NG2S + 1
C
  200 CONTINUE
  210 CONTINUE
C
C     ----------------------------------------
C     GENERATE 2-D (SHELL) FABRICATION RECORDS
C     ----------------------------------------
C
  250 IF (NSTGF2.EQ.0)                    GO TO 400
      ITAW = 0
      DO 310 NT=1,NTHST
      DO 300 NM=1,NMATT
            IF (FAT(NT,NM).NE.-2)         GO TO 300
            ITAW = ITAW+1
            FAT(NT,NM) = ITAW
C
C     .... INPUT GIFTS THICKNESS RECORD
            IR = NT
            CALL  IN THS (IR)
            IF (ITHPTR.EQ.NT)             GO TO 220
            IF (ITHPTR.LE.0)              GO TO 810
            IR = ITHPTR
            CALL  IN THS (IR)
  220       IF (ITYPE.NE.0)               GO TO 812
C
C     .... CHECK EXISTENCE OF MATERIAL NM
            JR = NM
            CALL  IN MAT (JR)
            IF (MATPTR.EQ.NM)             GO TO 225
            IF (MATPTR.LE.0)              GO TO 814
            JR = MATPTR
            CALL  IN MAT (JR)
C
C     .... CHECK FOR COMPOSITE MATERIAL
  225       IF (IMT.EQ.7)                 GO TO 801
C
C     .... WRITE 'WALL-SECTION HEADER' RECORD (K-1)
            KWALL = 1
            NLAY  = 1
            NLIP  = 5
            NSMRS = 0
            WRITE (ISLG2S,2012)  ITAW,KWALL,NLAY,NLIP,NSMRS
            NG2S = NG2S + 1
C
C     .... WRITE 'GENERAL LAYERED SHELL PROPERTY' RECORD (K-2)
            MATL = NM
            TL   = TH(1)
```

```
            ZETL = 0.
            LSO  = 1
            WRITE (ISLG2S,2014)  MATL,TL,ZETL,LSO
            NG2S = NG2S + 1
C
  300 CONTINUE
  310 CONTINUE
C
  400 CONTINUE
C
C.... CLOSE GIFTS DATA-BASE
      CALL  CLS THS
      CALL  CLS MAT
C
      RETURN
C
C                    E R R O R   E X I T S
C
  801 WRITE (ISLOUT,8010)  IM,IMT
 8010 FORMAT (/50H >> ERROR >>   UNADAPTABLE MATERIAL TYPE.  [G2SMR]
     1//15X,12HMATERIAL NO.,3X,4HTYPE/21X,I5,3X,I5/)
      ISTAT = -1
      CALL  G2S F
  810 WRITE (ISLOUT,8100)  NT
 8100 FORMAT (/30H >> ERROR >>   THICKNESS GROUP I5
     1,37H IS REFERENCED BUT UNDEFINED. [G2SMR]/)
      ISTAT = -1
      CALL  G2S F
  812 WRITE (ISLOUT,8120)  NT,ITYPE
 8120 FORMAT (/51H >> ERROR >>   UNADAPTABLE THICKNESS TYPE.  [G2SMR]
     1//15X,13HTHICKNESS NO.,3X,4HTYPE/22X,I5,3X,I5/)
      ISTAT = -1
      CALL  G2S F
  814 WRITE (ISLOUT,8140)  NM
 8140 FORMAT (/29H >> ERROR >>   MATERIAL GROUP I5
     1,37H IS REFERENCED BUT UNDEFINED. [G2SMR]/)
      ISTAT = -1
      CALL  G2S F
C
C                    F O R M A T S
C
 2001 FORMAT (1X,I5,60X,5H$ I-1)
 2002 FORMAT (1X,5(2X,E10.4),5X,5H$ I-2)
 2003 FORMAT (1X,5(2X,E10.4),1H,4X,5H$ I-2)
 2004 FORMAT (1X,2X,E10.4,36X,2X,E10.4,5X,1H$)
 2005 FORMAT (1X,I5,5X,I5,50X,5H$ I-1)
 2006 FORMAT (1X,2(5X,E10.4),1H,34X,5H$ I-3)
 2007 FORMAT (1X,2(5X,E10.4),35X,5H$ I-3)
 2008 FORMAT (1X,4I5,3(2X,E10.4),9X,5H$ J-1)
 2010 FORMAT (3(5X,E10.4),1H,20X,5H$ J-3/3(5X,E10.4),1X,I5,15X,1H$)
 2012 FORMAT (1X,5I5,40X,5H$ K-1)
 2014 FORMAT (1X,I5,2(5X,E10.4),1X,I5,24X,5H$ K-2)
C
      END
```

```
C=DECK G2SMS
      SUBROUTINE G2SMS
C
C=PURPOSE   TRANSFORM MODEL SUMMARY DATA GIFTS-TO-STAGS
C=AUTHOR    G.M.STANLEY
C=VERSION   1.0
C=EQUIPMENT CDC          VAX
C=KEYWORDS  G2S          MODEL       SUMMARY
C
C=UPDATED   MAR 03 1981
C
C=BLOCK     ABSTRACT
C
C         G2SMS COLLECTS MODEL SUMMARY PARAMETERS FROM THE GIFTS
C         DATA-BASE AND GENERATES THE CORRESPONDING STAGS INPUT
C         RECORDS ON FILE 'CASE'.G2S.
C
C=END       ABSTRACT
C
C                        C O M M O N
C
      COMMON /STATUS/ ISTAT,ISLG2S,NG2S
      COMMON /JOB/ XJOB(2),LJ(5)
      COMMON /OPTION/ LISTOP
      COMMON /FAC/ FAT(20,20)
      INTEGER FAT
C
C.... GIFTS
      COMMON /CMDSLT/ ISLIN,ISLOUT,ISLLST,ISLTTI,ISLTTO,ISLLPT
      COMMON /PAR/ LHV(13),LGL(5),LP(25),LS(16),LM(11)
      COMMON /ELT/ NEU,NES,ISTRKT,IELPLT,IT,IORD,IST,IAPL,NLDRC2,
     1             NCP,NGP,NLFLG,NSTPT,NLAYR,ISTPTR,NMAT,NTHS,
     2             FAC(5),ECCEN,WRKPAD(3),ALPHID,LCP(27)
      COMMON /THS/ ITHPTR,ITYPE,IPTRCS,LCOLRT(3),TH(15)
C
C.... STAGS
      COMMON /STAGS/ NSTGM,NSTGF1,NSTGF2,NSTGN,NSTGE(4),NSTGL
C
C
C                      E Q U I V A L E N C E
C
      EQUIVALENCE (LP(1),NGPT),  (LP(2),NGPA),  (LP(3),NELTT)
      EQUIVALENCE (LP(8),NMATT), (LP(9),NTHST), (LP(13),NLCT),
     1                                          (LP(14),NLCA)
      EQUIVALENCE (LS(1),ISTM),  (LS(6),ISTLD), (LS(7),ISTBC)
C
C                            D A T A
C
      DATA   MAXMAT,  MAXTHS,  MAXEGO
     1     /     20,       20,       1 /
      DATA   NSTGM,NSTGF1,NSTGF2,NSTGN,NSTGE,NSTGL /9*0/
C
C                         L  O  G  I  C
C
C.... OPEN GIFTS DATA-BASE
      CALL  OPN ELT
      CALL  OPN THS
      CALL  OPN MAT
C
```

```
C     ----------------------
C     GENERAL SUMMARY RECORDS
C     ----------------------
C
C.... WRITE 'CASE TITLE' RECORD (A-1)
      WRITE (ISLG2S,2000)  XJOB
      NG2S = NG2S + 1
C
C.... WRITE 'CONTROL' RECORD (B-1)
      ICASE = 1
      ILIST = LISTOP
      ISAVE = 1
      IMASS = 1
      ITEMP = 0
      INERT = 0
      GRAVC = 0.
      WRITE (ISLG2S,2001) ICASE,ILIST,ISAVE,IMASS,ITEMP,INERT,GRAVC
      NG2S = NG2S + 1
C
C.... WRITE 'MODEL SUMMARY' RECORD (B-2)
      NUNITS = 0
      NUNITE = 1
      NSTFS  = 0
      NINTS  = 0
      NPATS  = 0
      NCONS  = 0
      NLOADS = MIN (NLCA,2)
      WRITE (ISLG2S,2002) NUNITS,NUNITE,NSTFS,NINTS,NPATS,NCONS,NLOADS
      NG2S = NG2S + 1
C
C     ----------------
C     MATERIAL SUMMARY
C     ----------------
C
      NSTGM = NMATT
C
C     ---------------------------------------
C     FABRICATION, ELEMENT AND NODE SUMMARIES
C     ---------------------------------------
C
C.... NOTE: A STAGS FABRICATION CORRESPONDS TO A GIFTS MAT-THS PAIR.
C
      IF (NTHST.GT.MAXTHS.OR.NMATT.GT.MAXMAT)     GO TO 801
      DO 100 I=1,MAXTHS
      DO 100 J=1,MAXMAT
           FAT(I,J) = 0
  100 CONTINUE
C
C.... LOOP ON GIFTS ELEMENTS / COUNT STAGS ELEMENTS, FABRICATIONS
C
      DO 200 IELT=1,NELTT
           IE = IELT
           CALL  IN ELT (IE)
           IF (NEU.LE.0)              GO TO 200
           IF (IAPL.NE.0)             GO TO 200
           IF (IORD.GT.MAXEGO)                          GO TO 802
           IF (IT.GT.6)                                 GO TO 803
           GO TO (803, 210, 220, 220, 220, 220), IT
C
```

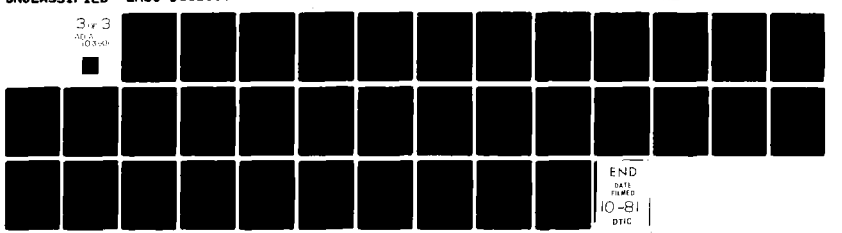
```
C     .... 1-DIMENSIONAL ELEMENTS / FABRICATIONS
  210       NSTGE(2) = NSTGE(2) + 1
            IF (FAT(NTHS,NMAT).LT.0)              GO TO 200
            NSTGF1 = NSTGF1 + 1
            FAT(NTHS,NMAT) = -1
            GO TO 200
C
C     .... 2-DIMENSIONAL ELEMENTS / FABRICATIONS
  220       I = 3
            IF (IT.GT.4)   I = 4
            NSTGE(I) = NSTGE(I) + 1
            IF (FAT(NTHS,NMAT).LT.0)                GO TO 200
            NSTGF2 = NSTGF2 + 1
            FAT(NTHS,NMAT) = -2
C
  200 CONTINUE
C
C.... WRITE 'RESOURCE SUMMARY' RECORD (B-3)
      WRITE (ISLG2S,2003)  NSTGM,NSTGF1,NSTGF2
      NG2S = NG2S + 1
C
C.... WRITE 'DISCRETIZATION SUMMARY' RECORD (H-1)
      WRITE (ISLG2S,2005)  NGPT,NSTGE
      NG2S = NG2S + 1
C
C.... CLOSE GIFTS DATA-BASE
      CALL  CLS ELT
      CALL  CLS THS
      CALL  CLS MAT
C
      RETURN
C
C                E R R O R   E X I T S
C
  801 WRITE (ISLOUT,8010)  MAXMAT
 8010 FORMAT (/67H >> ERROR >>   NO. OF MATERIAL/THICKNESS GROUPS TOO LA
     1RGE.  [G2SMS]//29X,9H/ LIMIT =,I3,2H / /)
      ISTAT = -1
      CALL  G2S F
  802 WRITE (ISLOUT,8010)  NEU,IT,IORD,MAXEGO
 8020 FORMAT (/57H >> ERROR >>   ELEMENT GEOMETRIC ORDER TOO HIGH.  [G2S
     1MS]//15X,11HELEMENT NO.,3X,4HTYPE,3X,5HORDER / 20X,I5,2(3X,I5),
     2 10H -- LIMIT=,I1/)
      ISTAT = -1
      CALL  G2S F
  803 WRITE (ISLOUT,8030)  NEU, IT
 8030 FORMAT (/49H >> ERROR >>   UNADAPTABLE ELEMENT TYPE.  [G2SMS]
     1//15X,11HELEMENT NO.,3X,4HTYPE / 20X,I5,3X,I5 /)
      ISTAT = -1
      CALL  G2S F
C
C                    F O R M A T S
C
 2000 FORMAT (2A4,57X, 6H $ A-1)
 2001 FORMAT (6I5,1H,E10.4,25X, 5H$ B-1)
 2002 FORMAT (7I5,31X, 5H$ B-2)
 2003 FORMAT (1X,3I5,50X, 5H$ B-3)
 2005 FORMAT (1X,5I5,40X, 5H$ H-1)
```

```
      END
```

```
C=DECK G2SU
      SUBROUTINE G2SU (ICOMM)
C
C=PURPOSE   USER COMMAND INTERFACE WITH 'G2S' ADAPTOR
C=AUTHOR    G.M.STANLEY
C=VERSION   1.0
C=EQUIPMENT INDEPENDENT
C
C=UPDATED   JAN30/1981
C
C                  D E C L A R A T I O N S
C
      COMMON /CMDSLT/ ISLIN,ISLOUT,ISLLST,ISLTTI,ISLTTO,ISLLPT
      COMMON /STATUS/ ISTAT,ISLG2S,NG2S
      COMMON /OPTION/ LISTOP
C
      CHARACTER*7   KEY, CCLVAL
C
C                        L  O  G  I  C
C
C.... INITIALIZE
      ICOMM = 0
C
C.... PROMPT USER FOR COMMAND
   10 CALL CLNEXT (' G 2 S > ', '1COMMANDS:   ADAPT | QUIT', ITEMS)
      IF (ITEMS .LE. 0)   GO TO 10
C
      KEY = CCLVAL(1)
      IF (KEY(1:3) .EQ. 'ADA')   GO TO 100
      IF (KEY(1:3) .EQ. 'QUI')   GO TO 200
      WRITE (ISLOUT,2010) KEY
      GO TO 800
C
C.... 'ADAPT' COMMAND
C
  100 CALL CLSTAT (NEXT,ITYPE,NV)
      IF (NEXT .GT. 0)   GO TO 110
      LISTOP = 0
      ICOMM  = 1
      GO TO 900
C
  110 KEY = CCLVAL (NEXT)
      IF (KEY(1:3) .EQ. 'LIS')   GO TO 120
      WRITE (ISLOUT,2110) KEY
      GO TO 800
C
  120 KEY = CCLVAL (NEXT+1)
      LISTOP = 0
      IF (KEY(1:3) .EQ. 'BRI')   LISTOP = 1
      IF (KEY(1:3) .EQ. 'FUL')   LISTOP = 2
      IF (LISTOP .GT. 0)   GO TO 130
      WRITE (ISLOUT,2120)
      GO TO 800
  130 ICOMM = 1
      GO TO 900
C
C.... 'QUIT' COMMAND
C
  200 ICOMM = 2
```

```
      GO TO 900
C
C.... ERROR TRAP
  800 ISTAT = -1
      GO TO 10
C
C.... COMMAND INTERPRETED.
  900 RETURN
C
C                    F O R M A T S
C
 2010 FORMAT (/27H >>> INVALID COMMAND >>>   A7/)
 2110 FORMAT (/27H >>> INVALID KEYWORD >>>   A7/)
 2120 FORMAT (/47H >>> INVALID DATA >>>   LIST = { BRIEF | FULL })
C
      END
```

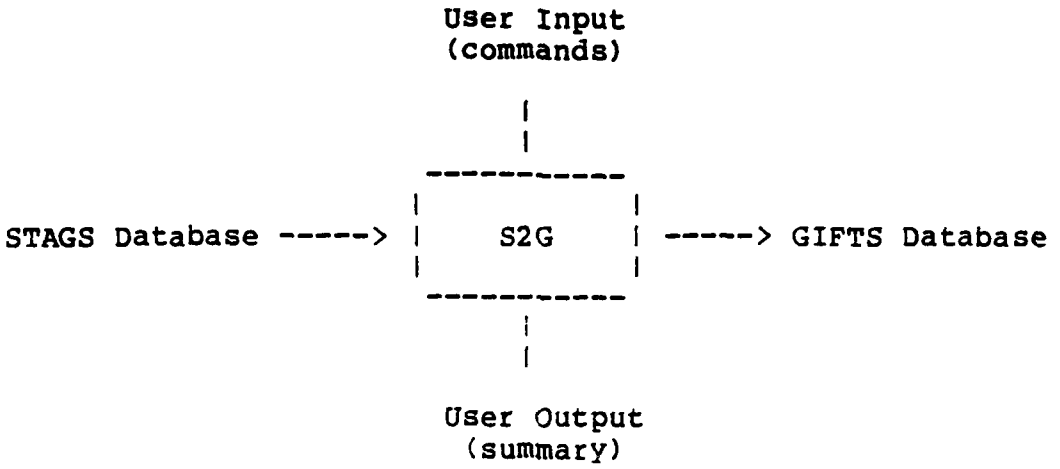
AD-A103 801 LOCKHEED MISSILES AND SPACE CO INC PALO ALTO CA PALO --ETC F/6 13/13
INTERACTIVE NONLINEAR STRUCTURAL ANALYSIS: ENHANCEMENT.(U)
JUL 81 G M STANLEY N00014-80-C-0831
UNCLASSIFIED LMSC-D811535 NL



```
C=DECK G2SX
      PROGRAM G2SX
C
C=CDC1(INPUT=512,OUTPUT=512,TAPE5=INPUT,TAPE6=OUTPUT)
C
C=PURPOSE   EXECUTIVE FOR GIFTS->STAGS ADAPTOR
C=AUTHOR    G.M.STANLEY
C=VERSION   JAN18/1981
C=EQUIPMENT CDC          VAX
C=KEYWORDS  EXECUTIVE    GIFTS       STAGS       ADAPTOR
C
C=BLOCK     ABSTRACT
C
C       G2SX IS THE EXECUTIVE FOR THE GIFTS-TO-STAGS PREPROCESSING
C       ADAPTOR.  IT SUPERVISES THE CREATION OF A FORMATTED
C       (CARD-IMAGE) INPUT FILE, 'CASE'.G2S, FOR THE STAGS1 MODULE.
C
C=END       ABSTRACT
C
C                          L O G I C
C
C.... INITIALIZE OPERATIONS
      CALL  G2S I
C
C.... READ COMMAND
      CALL  G2S U (ICOMM)
      IF (ICOMM .EQ. 2)   GO TO 900
C
C.... TRANSFORM MODEL SUMMARY DATA
      CALL  G2S MS
C
C.... TRANSFORM MODEL RESOURCE DATA
      CALL  G2S MR
C
C.... TRANSFORM MODEL CONFIGURATION DATA
      CALL  G2S MC
C
C.... TRANSFORM MODEL LOAD DATA
      CALL  G2S ML
C
C.... FINALIZE OPERATIONS
  900 CALL  G2S F
C
      END
```

3.4 THE STAGS->GIFTS (POSTPROCESSING) ADAPTOR: S2G

The S2G processor, or Adaptor, is the complement of the G2S processor (see Section 3.3). It is the GIST architectural component which constitutes the "postprocessing interface" between STAGS and GIFTS. S2G "adapts" the results of a STAGS analysis for GIFTS postprocessing, i.e., interactive-graphics evaluation. This is a database to database transfer function as illustrated in the following diagram:



The internal structure of the S2G Adaptor reflects the requirements of its output destination, the GIFTS (postprocessing) database. The processor is composed of the following basic routines:

S2GX ..... Main Program ("Executive")
S2GI ..... Initializes program operations
S2GU ..... User (command) interface routine
S2GSPA .... Oversees adaptation of spatial solutions (disp/stress)
S2GDNS .... Creates DNS file containing "displacement" solutions
S2GSTR .... Creates STR file containing "stress" solutions
S2GF ..... Finalizes program operations

A listing of the FORTRAN 77 source code for all of the above routines is presented on the following pages (in alphabetical order). Note that a number of GIFTS and STAGS utility subroutines (and COMMON blocks) are employed throughout the processor. Documentation on these utilities may be found in [G3] and [S4], respectively.

```
C=DECK S2GDNS
      SUBROUTINE S2GDNS (ISTEP,IMODE,NSOLN,SOLN,ISTAT)
C
C=PURPOSE   ADAPT SOLUTION VECTOR FROM STAGS->GIFTS
C=AUTHOR    G.M.STANLEY
C=VERSION   JAN25/1981
C=EQUIPMENT INDEPENDENT
C
C                        D E C L A R A T I O N S
C
      COMMON /USERIO/ IDI,IDO
      COMMON / PREC / IPREC
C
      COMMON /STAT/ LHARD(10),LSOFT(20),LCASE(30)
      COMMON /DOFS/ NRDOFS,IDOFA,NDOFA,NDOFP(31),IFDOFS(6)
      COMMON /STEP/ LSTEP(10),FSTEP(10)
      COMMON /SOLN/ NWSOLN,IWSOLN,ISOLN(8),FSOLN(10)
C
      COMMON /UNIFI/ UNICF,UNISF,UNIDF,UNIVF,IUNIT,KUNIT
      INTEGER        UNICF,UNISF,UNIDF,UNIVF
      COMMON /UNISTA/ NHED(4),UNICA(31),UNISA(31),UNIDA(31),UNIVA(31)
      INTEGER                 UNICA    ,UNISA    ,UNIDA    ,UNIVA
      COMMON /NDTAB/ IUPT,ISYS,NUJ,JP,XS,YS,XG,YG,ZG,TG(3,3),IACT
     1,              IBC,NOFF,NMS,LMS
C
      COMMON /JOB/ XJOB,XJOB2,YJOB(5)
      COMMON /PAR/ LHV(13),LGL(5),LP(25),LS(16),LM(11)
      COMMON /DNS/ DN(8)
      COMMON /PTS/ NU,NS,ISTRKT,ISTIPLT,VCSMC(9),WRKPAD(8),
     1             LMN,INTP,NLDREC,NBL,NFR,MFP
C
      DOUBLE PRECISION   VN(6)
C=CDC DIMENSION   VN(6)
C
      EQUIVALENCE   (LP(2),NGPA), (LP(15),NDST), (LS(9),ISTDN)
      EQUIVALENCE   (LCASE(12),NDOFS)
      EQUIVALENCE   (ISOLN(1),ISTYPE), (ISOLN(5),ISDOFS)
      EQUIVALENCE   (FSOLN(1),FSPALD), (FSOLN(2),FSPBLD)
      EQUIVALENCE   (FSOLN(3),FSTIME), (FSOLN(4),FSMODE)
      EQUIVALENCE   (DN(1),IDNTYP), (DN(4),IDNSTP), (DN(5),IDNMOD)
C
      DATA   ICONF/1/
C
C                           L O G I C
C
C     ------------------------------------------------------------
C               A D A P T   S E L E C T E D   V E C T O R
C     ------------------------------------------------------------
C
      ISTAT = 0
           CALL  IN SOLN (SOLN,IMODE,ISTEP,0,0,0,IS)
           IF (IS .LE. 0)   GO TO 900
C
           IF (ISDOFS .EQ. NDOFS)   GO TO 200
C
C..... BOUNDARY CONDITION TRANSFORMATION REQ'D (UN-IMPLEMENTED)
           WRITE (IDO,2445)  ISTEP
           GO TO 900
C
```

```
  200       WRITE (IDO,2200)  NSOLN,SOLN,IMODE,ISTEP
C
            CALL  IN SOLN (SOLN,IMODE,ISTEP,1,UNIDF,1,IS)
C
C......... OUTPUT GIFTS HEADER RECORD
C
  450       IDNTYP = 0
            IF (ISTYPE .EQ. 6)    IDNTYP=1
            IF (IMODE  .GT. 0)    IDNTYP=2
            IDNSTP = ISTEP
            IDNMOD = IMODE
            DN(2)  = FSTIME
            DN(6)  = FSMODE
            DN(7)  = FSPALD
            DN(8)  = FSPBLD
C
            CALL  OUT DNS (0,NSOLN)
C
C......... GENERATE NODAL VECTORS
C
            DO 500 NODE=1,NGPA
                 CALL MOVER (0,0,DN,1,8)
                 CALL VECIO (ICONF,NODE,6,IPREC,VN,1)
                 DO 600 I=1,6
  600            DN(I) = VN(I)
                 DN(7) = SQRT (DN(1)**2 + DN(2)**2 + DN(3)**2)
                 DN(8) = SQRT (DN(4)**2 + DN(5)**2 + DN(6)**2)
C
C *** CHECK FOR COORDINATE TRANSFORMATIONS
                 N = NODE
                 CALL  IN PTS (N)
                 IPAUX = NODE
                 IF (LMN .GT. 0)              IPAUX = LMN
C
C *** OUTPUT DISPLACEMENTS, AS COMPUTED
                 CALL  OUT DNS (IPAUX,NSOLN)
                 IF (LMN .LE. 0)   GO TO 500
C
C *** TRANSFORMATIONS EXIST; GET THEM
                 CALL  IN PTS (IPAUX)
                 K = 0
                 DO 601 J=1,3
                 DO 601 I=1,3
                 K = K + 1
                 TG(I,J) = VCSMC(K)
C
C *** COMPENSATE FOR GIFTS TRANSFORMATION ERROR
                 IF (J.EQ.2)   TG(I,J) = -TG(I,J)
  601            CONTINUE
C
C *** TRANSFORM "LOCAL" DISPLACEMENTS TO "GLOBAL"
                 DO 610 I=1,3
                 DN(I) = 0
                 DN(I+3) = 0
                 DO 610 J=1,3
                 DN(I) = DN(I) + TG(I,J) * VN(J)
                 DN(I+3) = DN(I+3) + TG(I,J) * VN(J+3)
  610            CONTINUE
C
```

```
C *** OUTPUT "GLOBAL" DISPLACEMENTS (IFF "LOCAL" SYSTEM EXISTS)
                CALL  OUT DNS (NODE,NSOLN)
C
  500       CONTINUE
C
  700 ISTAT = 1
      ISTDN = 1
C
  900 RETURN
C
C                        F O R M A T S
C
 2200 FORMAT (25H <> GIFTS SOLUTION NUMBER I4,13H  <==>  STAGS A4,2I4 )
 2440 FORMAT (//51H >>> E R R O R >>>   CURRENT STAGS DOF MAP MISSING.
     1,          10H  [S2GDNS]/)
 2445 FORMAT (/38H >>> WARNING >>>   BOUNDARY CONDITIONS
     1,          19H UPDATED SINCE STEP I4,10H. [S2GDNS]/)
C
       END
```

```
C=DECK S2GF
      SUBROUTINE S2GF (ISTAT)
C
C=PURPOSE   FINALIZE OPERATIONS FOR 'S2G' MODULE
C=AUTHOR    G.M.STANLEY [ January, 1981 ]
C=VERSION   1.0
C=EQUIPMENT INDEPENDENT
C=UPDATED   March 7 1981
C
C                   D E C L A R A T I O N S
C
      COMMON /USERIO/ IDI,IDO
      COMMON /STG/ CASE(2),IGOPEN,IGUNIT,IGBUFF
      COMMON /PAR/ LHV(13),LGL(5),LP(25),LS(16),LM(11)
      LOGICAL   PRESNT
C
      EQUIVALENCE   (LP(15),NDST), (LS(9),ISTDN), (LS(13),ISTES)
C
      DATA XPAR /4H PAR/
C
C                       L O G I C
C
C.... Update GIFTS Database
      IF (ISTAT.GT.0)   CALL  OUT PAR
C
C.... Release STAGS Database
      CALL  CLS STG (0,0,0,0,IS)
C
      IF (ISTAT .LE. 0)   GO TO 800
      IF (NDST  .GT. 0)   WRITE (IDO,2001)
      RETURN
C
C.... Error Termination
  800 STOP
C
C                         F O R M A T S
C
 2001 FORMAT (/21H <> READY FOR DISPLAY /)
C
      END
```

```
C=DECK S2GI
      SUBROUTINE S2GI
C
C=PURPOSE   INITIALIZE OPERATIONS FOR S2G MODULE
C=AUTHOR    G.M.STANLEY
C=VERSION   1.0
C=EQUIPMENT CDC   UNIVAC   VAX
C
C=UPDATED   MAY 27 1981
C
C                      D E C L A R A T I O N S
C
      COMMON A(1)
      COMMON /VMBUFT/ V(1)
      COMMON /S2GVER/ S2GVER(2)
      COMMON /PREC/ IPREC
      COMMON /USERIO/ IDI,IDO, IDNET
      COMMON /NITNOT/ NIT,NOT
C
      COMMON /STG/ CASE(2),IGOPEN,IGUNIT,IGBUFF
      COMMON /STAT/ LHARD(10),LSOFT(20),LCASE(30)
      COMMON /SOLN/ NWSOLN,IWSOLN,ISOLN(8),FSOLN(10)
      COMMON /CONF/ NWCONF,IWCONF,KCONF,JNCONF(3),JECONF(3),JMCONF(3)
     1,             IFCONF(8)
C
      COMMON /UNIFI/ UNICF,UNISF,UNIDF,UNIVF,IUNIT,KUNIT
      COMMON /UNISTA/ NHED,NUNIT,NUNITS,NUNITE
     1,               UNICA(31),UNISA(31),UNIDA(31),UNIVA(31)
      COMMON /UNICI/ LENC,LNOD,LELT,LMID,NNOD,NELT,NWN,NFN,NFM,NDOFL,IUC
C
      COMMON /JOB/ XJOB(2),YJOB(5)
      COMMON /PAR/ LHV(13),LGL(5),LP(25),LS(16),LM(11)
      COMMON /PTSBUF/ ISLPTS,LOCPTS,NBPTS,LPTSSZ(5),
     1                IBPTS(102),FBPTS(170)
      COMMON /ELTBUF/ISLELT,LOCELT,NBELT,LELTSZ(5),IBELT(252),FBELT(100)
      COMMON /DNSBUF/ISLBUF,LOCDNS,NBDNS,LDNSSZ(5),IBDNS(2),FBDNS(80)
      COMMON /STRBUF/ISLSTR,LOCSTR,NBSTR,LSTRSZ(5),IBSTR(2),FBSTR(80)
C
      COMMON /STEPS/ NSTEPS,ISTEPS(3)
C
      INTEGER UNICF,UNISF,UNIDF,UNIVF,UNICA,UNIDA,UNISA,UNIVA
      INTEGER VMPARS(6), VMINFO(7)
      INTEGER ICLIP(4)
      LOGICAL PRESNT
C
      CHARACTER*24   CASENAM, CCLVAL
C
      EQUIVALENCE   (LP(15),NDST), (LS(9),ISTDN), (LS(13),ISTES)
C
      DATA   XPAR/4H PAR/, XDNS/4H DNS/, XSTR/4H STR/
      DATA   UNICA,UNISA,UNIDA,UNIVA / 31*1,31*65,31*1,31*1 /
      DATA   IUC/0/
      DATA   VMPARS /0,0,0,0,0,0/, NOT/26/
      DATA   ICLIP /4*0/
      DATA   NSTEPS,ISTEPS /4*0/
C
C                          L  O  G  I  C
C
C.... IDENTIFY PROCESSOR
```

```
      WRITE (IDO,2000) S2GVER(1)
C
C.... READ CASE NAME
C
C *** CHECK FOR NETWORK CONTROL
      CALL  CLMODE (ICLIP,1,0)
C *** READ (IDNET,1000,END=10,ERR=10) CASENAM
C *** WRITE (IDO,2001) CASENAM
C *** GO TO 20
C
   10 CALL CLNEXT (' JOB:  ', '1 ', ITEMS)
      IF (ITEMS .EQ. 0)   GO TO 800
      CASENAM = CCLVAL (1)
C
   20 CALL CC2H (CASENAM(1:1),CASE(1),4)
      CALL CC2H (CASENAM(5:5),CASE(2),4)
      CALL CC2H (CASENAM(1:1),XJOB(1),4)
      CALL CC2H (CASENAM(5:5),XJOB(2),4)
C
C.... ACCESS STAGS DATABASE
C
      IGBUFF = 1
      IOPEN  = 0
      ILIST  = 0
      CALL  OPN STG (IOPEN,ILIST,0,0,IS)
      IF (IS .LE. 0)   GO TO 801
C
C.... PREPARE FOR STAGS LOCAL DATA MANAGEMENT
C
      CALL VM INIT (0)
      CALL VM OPEN (LNAM,UNICF,VMPARS)
      CALL VM OPEN (LNAM,UNIDF,VMPARS)
      CALL VM OPEN (LNAM,UNISF,VMPARS)
      CALL VM OPEN (LNAM,UNIVF,VMPARS)
C
C.... RETRIEVE STAGS STATISTICS DATA-SET
C
      CALL  IN STAT (0,0,0,0,IS)
C
C.... RETRIEVE STAGS CONFIGURATION DATA-SET
C
      ICONF = 1
      CALL  IN CONF (ICONF,0,0,0,IS)
      IF (IS .LE. 0)   GO TO 803
      CALL  IN CONF (ICONF,1,UNICF,UNICA(ICONF),IS)
C
      UNICA(ICONF) = 1
      CALL UNITIO (ICONF,1)
C
C.... ACCESS GIFTS DATABASE
C
      CALL INITIO
      ISLIN  = IDI
      ISLOUT = IDO
      ISLTTI = IDI
      ISLTTO = IDO
      IF (.NOT.PRESNT(XPAR))   GO TO 802
C
```

```
C
C.... PREPARE FOR GIFTS LOCAL DATA MANAGEMENT
C
      LOCDNS = 0
      NBDNS  = 1
      LOCSTR = 0
      NBSTR  = 1
      LOCELT = 0
      NBELT  = 1
      LOCPTS = 0
      NBPTS  = 1
C
C.... DELETE EXISTING GIFTS POSTPROCESSING FILES
C
      CALL DELETE (XDNS)
      CALL DELETE (XSTR)
      NDST  = 0
      ISTDN = 0
      ISTES = 0
C
      RETURN
C
C                    E R R O R    E X I T S
C
  800 CALL S2GF (-1)
  801 WRITE (IDO,2010)
      CALL S2GF (-1)
  802 WRITE (IDO,2020)
      CALL S2GF (-1)
  803 WRITE (IDO,2030)
      CALL S2GF (-1)
C
C                         F O R M A T S
C
 1000 FORMAT (A9)
 2000 FORMAT (//12H <<>>  S 2 G  ,11X,25H < STAGS->GIFTS ADAPTOR >
     1,                           14X, 6H GIST/A4 )
 2001 FORMAT (7H CASE: A9 /)
 2010 FORMAT (/46H >>> E R R O R >>>   STAGS DATABASE NOT FOUND.
     1,         7H [S2GI] /)
 2020 FORMAT (/46H >>> E R R O R >>>   GIFTS DATABASE NOT FOUND.
     1,         7H [S2GI] /)
 2030 FORMAT (/52H >>> E R R O R >>>   STAGS MODEL DEFINITION MISSING.
     1,          7H [S2GI] /)
C
      END
```

```
C=DECK S2GSPA
      SUBROUTINE S2GSPA
C
C=PURPOSE   ADAPT SPATIAL SOLUTIONS FROM GIFTS -> STAGS
C=AUTHOR    G.M.STANLEY
C=VERSION   JAN27/1981
C=EQUIPMENT INDEPENDENT
C
C                D E C L A R A T I O N S
C
      COMMON /STEPS/ NSTEPS,ISTEPS(3)
C
      COMMON /STAT/  LHARD(10),LSOFT(20),LCASE(30)
      COMMON /DOFS/  NRDOFS,IDOFA,NDOFA,NDOFP(31),IFDOFS(6)
      COMMON /SOLN/  NWSOLN,IWSOLN,ISOLN(8),FSOLN(10)
C
      COMMON /PAR/   LHV(13),LGL(5),LP(25),LS(16),LM(11)
C
      COMMON /UNIFI/  UNICF,UNISF,UNIDF,UNIVF,IUNIT,KUNIT
      INTEGER         UNICF,UNISF,UNIDF,UNIVF
      COMMON /UNISTA/ NHED(4),UNICA(31),UNISA(31),UNIDA(31),UNIVA(31)
      INTEGER                 UNICA    ,UNISA    ,UNIDA    ,UNIVA
C
      COMMON /USERIO/ IDI,IDO
C
      EQUIVALENCE (ISTEPS(1),ISTEP1),(ISTEPS(2),ISTEP2),(ISTEPS(3),INC)
      EQUIVALENCE (LCASE(12),NDOFS)
      EQUIVALENCE (LP(15),NDST), (LS(9),ISTDN), (LS(13),ISTES)
C
C                    L  O  G  I  C
C
C.... SCAN STAGS DATABASE; COUNT AVAILABLE SOLUTIONS
C
      DO 100 ISTEPP = ISTEP1,ISTEP2,INC
           ISTEP = ISTEPP-1
C
           CALL  IN SOLN ('DISP',0,ISTEP,0,0,0,IS)
           IF (IS .GT. 0)   NDST = NDST + 1
C
           CALL  IN SOLN ('VELO',0,ISTEP,0,0,0,IS)
           IF (IS .GT. 0)   NDST = NDST + 1
C
           DO 200 IMODE = 1,10
                CALL  IN SOLN ('MODE',IMODE,ISTEP,0,0,0,IS)
                IF (IS .GT. 0)   NDST = NDST + 1
  200      CONTINUE
  100 CONTINUE
      IF (NDST .GT. 0)   GO TO 300
      WRITE (IDO,2100)
      GO TO 800
C
C
C.... INPUT CURRENT STAGS DOF MAP
C
  300 CALL  IN DOFS (NDOFS,0,0,0,IS)
      IF (IS .GT. 0)   GO TO 310
      WRITE (IDO,2300)
      GO TO 800
C
```

```
  310 ICONF = 1
      CALL  IN DOFS (NDOFS,1,UNIVF,UNIVA(ICONF),IS)
C
      WRITE (IDO,2000)
C
C.... ADAPT SELECTED SOLUTIONS
C
      CALL  OPN DNS
      CALL  OPN STR
      CALL  OPN ELT
      CALL  OPN PTS
C
      NSOLN = 1
C
      DO 400 ISTEPP = ISTEP1,ISTEP2,INC
           ISTEP = ISTEPP-1
C
C..... ADAPT DISPLACEMENT VECTOR
           CALL  S2GDNS (ISTEP,0,NSOLN,'DISP',ISTAT)
C
C..... ADAPT "STRESSES"
           CALL  S2GSTR (ISTEP,0,NSOLN,'STRE',JSTAT)
C
           IF (ISTAT.GT.0 .OR. JSTAT.GT.0)    NSOLN = NSOLN+1
C
C..... ADAPT VELOCITY VECTOR
           CALL  S2GDNS (ISTEP,0,NSOLN,'VELO',ISTAT)
           IF (ISTAT .GT. 0)    NSOLN = NSOLN + 1
C
C..... ADAPT EIGENVECTORS
           DO 450 IMODE=1,10
                CALL  S2GDNS (ISTEP,IMODE,NSOLN,'MODE',ISTAT)
                IF (ISTAT .LE. 0)    GO TO 400
                NSOLN = NSOLN + 1
  450      CONTINUE
  400 CONTINUE
C
      NSOLN = NSOLN - 1
      IF (NSOLN .GT. 0)    GO TO 500
      WRITE (IDO,2600)
      GO TO 800
C
  500 CONTINUE
C
      CALL  CLS PTS
      CALL  CLS ELT
      CALL  CLS STR
      CALL  CLS DNS
C
  700 RETURN
C
C.... ERROR TRAP
  800 CALL S2GF (-1)
C
C                        F O R M A T S
C
 2000 FORMAT ()
 2100 FORMAT (/47H >>> SORRY >>>   REQUESTED SOLUTIONS NOT FOUND. /)
```

```
     1          10H  [S2GSPA]/)
 2600 FORMAT (/38H >>> SORRY >>>   NO SOLUTIONS ADAPTED.  /)
C
      END
```

```
C=DECK S2GSTR
      SUBROUTINE S2GSTR (ISTEP,IMODE,NSOLN,SOLN,ISTAT)
C
C=PURPOSE   ADAPT STRESSES FROM GIFTS->STAGS DATABASES
C=AUTHOR    G.M.STANLEY
C=VERSION   JAN25/1981
C=EQUIPMENT INDEPENDENT
C
C                    D E C L A R A T I O N S
C
      COMMON /USERIO/ IDI,IDO
      COMMON /PRECIS/ IPREC
C
      COMMON /STAT/   LHARD(10),LSOFT(20),LCASE(30)
      COMMON /STEP/   LSTEP(10),FSTEP(10)
      COMMON /SOLN/   NWSOLN,IWSOLN,ISOLN(8),FSOLN(8)
C
      COMMON /UNIFI/  UNICF,UNISF,UNIDF,UNIVF,IUNITS,KUNITS
      INTEGER         UNICF,UNISF,UNIDF,UNIVF
      COMMON /UNISTA/ NHED(4),UNICA(31),UNISA(31),UNIDA(31),UNIVA(31)
      INTEGER               UNICA    ,UNISA    ,UNIDA    ,UNIVA
      COMMON /ELTAB/  LEVM,LEFM,NEFM,IET,AET,IOPE,JOPE,MFAB,
     1                ANGLE,ECY,ECZ,ILIN,IPLAS,NECP,NCP(10),LOCE
      COMMON /UNISTR/ LENS,LSTR(5)
C
      COMMON /PAR/    LHV(13),LGL(5),LP(25),LS(16),LM(11)
      COMMON /ELT/    NEU,NES,ISTRKT,IELPLT,IT,IORD,IST,IAPL,NLDRC2,
     1                NCPE,NGP,NLFLG,NSTPT,NLAYR,ISTPTR,NMAT,NTHS,
     2                FAC(5),ECCEN,WRKPAD(3),ALPHID,LCP(8)
      COMMON /STR/    ST(8)
C
      DIMENSION STRESS(6),RESULT(1),STRAIN(1)
C
      EQUIVALENCE   (STRESS,RESULT), (STRESS,STRAIN)
      EQUIVALENCE   (LSTR(2),IETS), (LSTR(3),IDVR),
     1              (LSTR(4),KFAB), (LSTR(5),NLAY)
      EQUIVALENCE   (LP(3),NELTG), (LP(15),NDST), (LS(13),ISTES)
      EQUIVALENCE   (ST(1),ISTTYP), (ST(4),ISTSTP)
      EQUIVALENCE   (ST(5),ISTMOD), (ST(7),FC1)
      EQUIVALENCE   (ISOLN(1),ISTYPE), (ISOLN(5),ISNDOF)
      EQUIVALENCE   (FSOLN(1),FSPALD), (FSOLN(2),FSPBLD)
      EQUIVALENCE   (FSOLN(3),FSTIME), (FSOLN(4),FSMODE)
      EQUIVALENCE   (ISTPTR,IPGIFT), (ISTRKT,IESTAG)
C
      DATA   ICONF /1/
      DATA   DEG2RAD /.017453293/
C
C                         L O G I C
C
C     ------------------------------------------------------------
C         A D A P T   S E L E C T E D   S T R E S S   S T E P
C     ------------------------------------------------------------
C
      ISTAT = 0
C
C..... SCAN STAGS DATA-BASE FOR STEP 'ISTEP' STRESSES
C
      CALL  IN SOLN (SOLN,0,ISTEP,0,0,0,IS)
```

```
C
C......... STRESSES EXIST, LOCALIZE
C
           WRITE (IDO,2210)   NSOLN,SOLN,IMODE,ISTEP
C
           CALL  IN SOLN (SOLN,0,ISTEP,1,UNISF,1,IS)
C
C......... OUTPUT GIFTS STRESS-GROUP DESCRIPTOR RECORD
C
           ISTTYP = 0
           IF (ISTYPE .EQ. 6)   ISTTYP = 1
           ISTSTP = ISTEP
           ST(2)  = FSTIME
           ST(7)  = FSPALD
           ST(8)  = FSPBLD
           CALL  OUT STR (0,NSOLN)
C
C......... GENERATE GIFTS ELEMENT STRESS RECORDS
C
           IEGIFT = 0
           IPGIFT = 0
           DO 300 IE=1,NELTG
C
C.............. GET ELEMENT DIRECTORY RECORDS
C
                IEGIFT = IEGIFT + 1
                CALL  IN ELT (IEGIFT)
                IF (NEU .LE. 0)   GO TO 300
                IF (IESTAG .LE. 0)   GO TO 300
                CALL  ELTIO (IESTAG,ICONF,0,1)
                CALL  ESTIO (IESTAG,ICONF,0,0,0,0,1)
C
                GO TO (310,320,330), IDVR
C
C.............. BEAM ELEMENTS [RESULTANTS: N,QY,QZ,MY,MZ,T]
C
  310           CALL  ESTIO (IESTAG,ICONF,RESULT,1,0,0,1)
                CALL  MOVER (0,0,ST,1,8)
                ST(1) = RESULT(1)
                ST(2) = RESULT(4)
                XSI   = ANGLE*DEG2RAD
                ST(5) = RESULT(2)*COS(XSI) - RESULT(3)*SIN(XSI)
                ST(6) =-RESULT(2)*SIN(XSI) - RESULT(3)*COS(XSI)
                CALL  OUT STR (IPGIFT,NSOLN)
                GO TO 300
C
C.............. SHELL ELEMENTS [STRESSES: SIGXX,SIGYY,SIGXY (BOT/TOP)]
C
  320           DO 325 IFIBER=1,2
                     CALL  ESTIO (IESTAG,ICONF,STRESS,3,1,IFIBER,1)
                     CALL  MOVER (0,0,ST,1,8)
                     CALL  MOVER (STRESS(4),1,ST,1,3)
                     FC1 = SQRT(ST(1)**2 + ST(1)*ST(2) + ST(2)**2
     1                                              + 3.*ST(3)**2)
                     IPGIFT = IPGIFT+1
                     CALL  OUT STR (IPGIFT,NSOLN)
  325           CONTINUE
                GO TO 300
C
```

```
C.............. SOLID ELEMENTS (UN-IMPLEMENTED)
C
  330           GO TO 300
C
  300      CONTINUE
C
  700 ISTAT = 1
      ISTES = 1
C
  900 RETURN
C
C                        F O R M A T S
C
 2210 FORMAT (25H <> GIFTS SOLUTION NUMBER I4,13H  <==>  STAGS A4,2I4 )
C
      END
```

```
C=DECK S2GU
      SUBROUTINE S2GU (ICOMM)
C
C=PURPOSE   USER INTERACTION ROUTINE FOR 'S2G' MODULE
C=AUTHOR    G.M.STANLEY
C=VERSION   1.0
C=EQUIPMENT CDC   UNIVAC   VAX
C
C=UPDATED   JAN 31 1981
C
C                        D E C L A R A T I O N S
C
      INTEGER          S2GCOM,S2GDAT
      COMMON /USERIO/  IDI,IDO
      COMMON /STEPS/   NSTEPS,ISTEPS(3)
C
      CHARACTER*7 KEY, CCLVAL
C
C                                L O G I C
C
C.... INITIALIZE
      ICOMM = 0
C
C.... PROMPT USER FOR COMMAND
C
   10 CALL CLNEXT (' S 2 G > ', '1COMMANDS:   ADAPT | QUIT', ITEMS)
      IF (ITEMS .EQ. 0)   GO TO 10
      KEY = CCLVAL (1)
      IF (KEY(1:3) .EQ. 'ADA')   GO TO 100
      IF (KEY(1:3) .EQ. 'QUI')   GO TO 300
      WRITE (IDO,2011) KEY
      GO TO 10
C
C.... 'ADAPT' COMMAND
C
  100 CALL CLSTAT (NEXT,TYPE,NV)
      IF (NEXT .GT. 0)   GO TO 110
      CALL CLNEXT (' G 2 S  ADAPT > ', '1KEYWORD:   STEPS', ITEMS)
      IF (ITEMS .EQ. 0)   GO TO 10
      NEXT = 1
C
  110 KEY = CCLVAL (NEXT)
      IF (KEY(1:4) .EQ. 'STEP')   GO TO 120
      WRITE (IDO,2111) KEY
      GO TO 10
C
C.... READ 'STEP' DATA
  120 NSTEPS = 0
      CALL CLOADI (NEXT+1,-3,0,ISTEPS,NV)
      IF (NV .GT. 0)   GO TO 140
  135 WRITE (IDO,2121) ISTEPS
      GO TO 10
  140 DO 160 I=1,2
      IF (ISTEPS(I) .LT. 0)   GO TO 135
  160 ISTEPS(I) = ISTEPS(I) + 1
      ISTEPS(2) = MAX (ISTEPS(2),ISTEPS(1))
      ISTEPS(3) = MAX (ISTEPS(3),1)
      NSTEPS = (ISTEPS(2)-ISTEPS(1))/ISTEPS(3) + 1
      ICOMM = 1
```

```
      RETURN
C
C.... 'QUIT' COMMAND
C
  300 ICOMM = 3
      RETURN
C
C                          F O R M A T S
C
 2011 FORMAT (/27H >>> INVALID COMMAND >>>   A7)
 2111 FORMAT (/27H >>> INVALID KEYWORD >>>   A7)
 2121 FORMAT (/47H >>> INVALID DATA >>>   STEPS = step1,step2,inc)
C
      END
```

```
C=DECK S2GX
      PROGRAM S2GX
C=CDC1(INPUT=512,OUTPUT=512,TAPE5=INPUT,TAPE6=OUTPUT)
C
C=PURPOSE   EXECUTIVE FOR STAGS->GIFTS ADAPTOR MODULE
C=AUTHOR    G.M.STANLEY
C=VERSION   JAN25/1981
C=EQUIPMENT CDC   VAX
C
C=BLOCK ABSTRACT
C
C       PROGRAM S2GX IS THE EXECUTIVE FOR THE STAGS->GIFTS SOLUTION
C       ADAPTOR.   IT SUPERVISES THE ADAPTATION OF SOLUTION VECTORS
C       AND ELEMENT STRESSES FROM THE STAGS DATABASE INTO THE GIFTS
C       DATABASE (I.E., CASE.STG -> CASE.DNS,CASE.STR).
C
C=END   ABSTRACT
C
C                       D E C L A R A T I O N S
C
      COMMON A(500)
      COMMON /VMBUFT/ V(20000)
      COMMON /S2GVER/ S2GVER(2)
      COMMON /STEPS/ NSTEPS,ISTEPS(3)
      COMMON /PREC/ IPREC
      COMMON /USERIO/ IDI,IDO, IDNET
C
      COMMON /STG/ CASE(2),IGOPEN,IGUNIT,IGBUFF
      COMMON /UNIFI/ UNICF,UNIDF,UNISF,UNIVF,IUNIT,KUNIT
      INTEGER        UNICF,UNIDF,UNISF,UNIVF
C
      COMMON /JOB/ XJOB,XJOB2,YJOB(5)
C
C                             D A T A
C
      DATA   S2GVER /4H1.0/,4HVAX /
C
      DATA   IDI,IDO, IDNET /5,6, 25/
      DATA   IGUNIT /10/
      DATA   UNICF,UNIDF,UNISF,UNIVF /15,16,17,18/
      DATA   IPREC/2/
C
C                             L O G I C
C
C.... INITIALIZE PROGRAM OPERATIONS
      CALL S2GI
C
C.... READ COMMAND
      CALL S2GU (ICOMM)
      IF (ICOMM .EQ. 1)   GO TO 100
      IF (ICOMM .EQ. 3)   GO TO 300
C
C.... ADAPT SOLUTION VECTORS/STRESSES AT SELECTED STEPS
  100 CALL  S2G SPA
C
C.... GENERATE SOLUTION HISTORIES  (UN-IMPLEMENTED)
  200 CONTINUE
C
C.... FINALISE PROGRAM OPERATION
```

```
  300 CALL S2GF (1)
C
      END
```

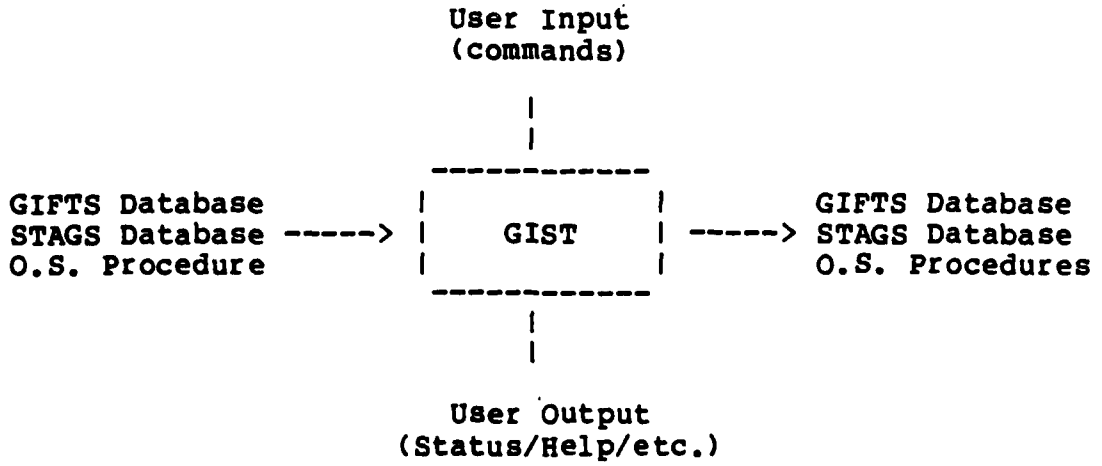
3.5 THE GIFTS/STAGS CONTROL MODULE: GIST

The GIST Control Module is the unifying agent of the GIST system. It is the command-driven processor which constitutes essentially all of the user/system interface outside of the GIFTS pre- and postprocessors (see Part 1: The GIST Command Language). It is also the means by which a "structural analysis operating system" effect is created, i.e., by invoking and monitoring the execution of the various other processors without noticeable intervention (with the user) from the actual operating system.

Conceptually, the Control Module may be thought of as a "capsule" which completely surrounds the GIST system and the computer operating system as well. In actuality, however, it is only another processor. What makes it special, is its responsibility for "hiding" the computer operating system from the user, and for coordinating, and providing a common means of access to, the rest of the GIST system.

By necessity, the Control Module has access to both the GIFTS and the STAGS database AND in addition must interact (in place of the user) with the computer operating system. To perform this last function, a certain amount of "machine-dependence" (or to be more precise, "operating system dependence") has been engendered; but this has been isolated (we hope) within a small number of subroutines.

The following diagram represents the flow of data through the Control Module:



The mechanism employed by the Control Module for communicating with the computer operating system is the "procedure file" (or O.S. Procedure). Practically all operating systems have this capability, which amounts to allowing the user or user-program to prepare a series of operating system commands in a so-called "procedure file" which may then be added to the "runstream" via another, special, operating system command. For example, on CDC/NOS, an operating system procedure file is invoked via: CALL Filename, while on the VAX/VMS system, an expression of the form: @Filename, is used.

With the above information as background, we will now look beyond the black walls surrounding the Control Module into the interior. What we will find is an internal structure which essentially parallels the GIST Command Language introduced in Part 1.

(NOTE: This parallelism is a common attribute of "command-driven" processors, as opposed to "data-driven" processors which often read everything at the outset and then proceed to process it in some arbitrary order which bears little resemblance to the initial input. As may be discerned from this description, the author is somewhat biased towards "command-driven" processors. This position has evolved from experience and from the inherent neatness and flexibility for both batch and interactive operation derived from such a design; see [N1] and [N5] for amplification.)

The following is a brief summary of the basic subroutines comprising the GIST Control Module:

```
    GISTX ..... Main Program ("Executive")
    GISTI ..... Initializes program operations
    GSGIFT .... Supervises action on all GIFTS-related commands
    GSSTAG .... Supervises action on all STAGS-related commands
    GSSETU .... Acts on the SETUP command
    GSSTAS .... Acts on the STATIC command
    GSDYNS .... Acts on the DYNAMIC command
    GSEIGS .... Acts on the BUCKLING and VIBRATION commands
    GSSTAC .... Acts on the COMPUTE command for static analysis
    GSDYNC .... Acts on the COMPUTE command for dynamic analysis
    GSEIGC .... Acts on the COMPUTE command for eigenvalue analysis
    GSREV ..... Supervises action on the REVIEW command
    GSREVJ .... Acts on the REVIEW JOB command
    GSREVM .... Acts on the REVIEW PREP command
    GSREVA .... Acts on the REVIEW ANALYSIS command
    GSREVY .... Acts on the REVIEW STRATEGY command
    GSREVS .... Acts on the REVIEW SOLUTION command
    GSREVR .... Acts on the REVIEW POST command
    GSCLR ..... Acts on the CLEAR command
    GSMAN ..... Acts on the MANAGE command
  * GSOS ...... Creates all operating system procedure files
  * GSBS ...... Prepares for /BATCH operations
    GSHELP .... Acts on the HELP command
    GSQUIT .... Acts on the QUIT command
    GISTF ..... Finalizes program operations
```

where the subroutines flagged with an asterisk are inherently operating system dependent (or OD). Note that the main program (GISTX) may also be a bit OD in the sense that adustable array dimensions and FORTRAN unit numbers are contained therein. Also, there are a number of GIFTS, STAGS and NICE utilities that are called from within the above routines. The documentation for these utilities may be found in: [G3], [S4], and [N5] respectively.

A source code listing of the Control Module will not be presented here, simply to avoid unnecessary bulk (approximately 3000 lines). However, it is provided as part of the standard GIST installation package (see Appendix C).

A P P E N D I C E S

("Currently in Preparation")

R E F E R E N C E S

[G1] H.A. Kamel, M.W. McCabe, et. al., "The GIFTS System; Version 5.0 User's Manual", University of Arizona, Interactive Graphics Engineering Laboratory (IGEL), May 1979.

[G2] H.A. Kamel, M.W. McCabe, et. al., "GIFTS Primer; A First Introduction to the GIFTS System", University of Arizona, Interactive Graphics Engineering Laboratory (IGEL), May 1979.

[G3] H.A. Kamel, M.W. McCabe, et. al., "GIFTS-5 Systems Manual", University of Arizona, Interactive Graphics Engineering Laboratory, May 1980.

[G4] H.A. Kamel, M.W. McCabe, et. al., "GIFTS-5 Pocket Guide", University of Arizona, Interactive Graphics Engineering Laboratory (IGEL), March 1980.

[N1] C.A. Felippa, "Architecture of a Distributed Analysis Network for Computational Mechanics", Proceedings of the Symposium on Computational Methods in Nonlinear Structural and Solid Mechanics (Washington, D.C., October 6-8, 1980), Pergammon Press, London, pp. 405-414.

[N2] C.A. Felippa, "The Input-Output Manager DMGASP and the Direct Access Library Manager EZ-DAL of the NOSTRA Data Management System, LMSC-D628839, July 1978.

[N3] C.A. Felippa, "The Global Database Manager EZGAL, LMSC-D766995, October 1980.

[N4] C.A. Felippa, "A Command Language Advanced Utility for Database Editing: CLAUDE" -- to be released.

[N5] C.A. Felippa, "A Command Language for Applied Mechanics Processors, LMSC-D63582, March 1980 (revised April 1981).

[S1] B.O. Almroth, F.A. Brogan, and G.M. Stanley, "Structural Analysis of General Shells; Volume II: User Instructions for STAGS-C1", LMSC-D633873, July 1979.

[S2] B.O. Almroth et. al., "Structural Analysis of General Shells; Volume I: Theory Manual for STAGS-C1", LMSC, March 1978.

[S3] B.O. Almroth et. al., "Structural Analysis of General Shells; Volume III: Example Manual for STAGS-C1" -- to be released.

2.3 THE POSTPROCESSING PHASE

The following GIST commands are associated with the "postprocessing" phase" of structural analysis, in which the solutions obtained by the analyzer are evaluated interactively:

RESULT .... Invokes the GIFTS "Result Display" processor

REVIEW .... Monitors Job status (e.g., REVIEW SOLUTION)

RESULT is used to evaluate the solution from a "physical" perspective. The analyst may employ it to view, either graphically or in tabular form, such things as displacements, stresses, strains, resultants, etc. at any number of discrete solution steps. The prerequisite is, of course, that some solutions have been computed AND SAVED by the STAGS analyzer (see Section 2.2). The adaptation of the results for GIFTS display is automatic, as will be explained later in this section.

REVIEW (esp. REVIEW SOLUTION) may be used to evaluate the solution from a "computational" perspective. By this, we refer to the re-examination of such things as convergence error, iteration count, stiffness update status and determinant, on a step by step basis. Such information is indispensable when formulating or revising strategy for subsequent solution intervals, or just for assessing the accuracy of the current solution.

In the following subsections, we will outline and demonstrate both forms of GIST postprocessing.

2.3.1 "PHYSICAL" EVALUATION

The evaluation of the structural response (or modal characteristics) thus far computed by the STAGS analyzer is performed exclusively with the interactive GIFTS processor: RESULT. RESULT may be used to selectively display such physical parameters as displacements, velocities, stresses, strains and stress resultants in a variety of both graphical and tabular formats. For example, graphical displays include: deformed geometry, stress contours, element-labelled stress levels (normalized w.r.t. a failure criterion), and principal stress directions.

as are many discrete solution steps (or modes) as are available may be observed in a single RESULT session. However, the view is presently restricted to be spatial (i.e., one "frame" at a time) rather than temporal. (This means that an external processor will be required if such things as response history "graphs" of individual displacement components are desired.)

To check on the availability of solutions in the database, which is largely determined by previous solution strategies, the user may employ the REVIEW command, in the form:

```
REVIEW/TOC ANALYSIS
```

The above command lists a "table of contents" of the "analysis database" (discussed further in Section 2.4) and will indicate which solution steps (and/or modes) are currently available for postprocessing.

To select the desired solution steps and initiate interactive post-processing with RESULT, the user enters the GIST command:

```
RESULT  SOLUTIONS = stepmin [,stepmax,stepinc]
```

where 'stepmin, stepmax' represent a range of solution step numbers and 'stepinc' is an optional increment. The requested sequence may include steps which are not available in the database; such "holes" will be politely skipped over.

Before turning the user over to the GIFTS' postprocessor, the GIST Control Module normally (i.e., by default) activates the so-called STAGS-GIFTS "Adaptor" (S2G), which extracts the requested solution data from the analysis database and recreates it in the "postprocessing database", in a form convenient for GIFTS utilization. In the process, a correspondence table between RESULT's "Load Case" numbers (not to be confused with the load case numbers employed during Pre-processing) and STAGS "solution step" numbers will be printed on the screen.

The above description of the RESULT procedure will now be made clear through an illustrative example followed by some remarks.

Illustrative Example

Suppose we have computed and saved the first 10 solution steps of a nonlinear static response analysis, and wish to sample the results at steps 0, 5 and 10. Assuming we are already communicating with the GIST Control Module, we thus enter the appropriate command:

```
G I S T >   RESULT  SOL = 0, 10, 5
```

Moments later, the STAGS->GIFTS Adaptor (S2G) goes to work and eventually prints the following table:

POSTPROCESSING CORRESPONDENCE TABLE

GIFTS RESULTS	STAGS SOLUTIONS
"LOAD-CASE" NO. 1	STEP 0 -- DISPLACEMENTS -- STRESSES
"LOAD-CASE" NO. 2	STEP 5 -- DISPLACEMENTS -- STRESSES
"LOAD-CASE" NO. 3	STEP 10 -- DISPLACEMENTS -- STRESSES

This is immediately followed by:

```
------------------------------------------------
<<>>  R E S U L T   [ Result Display ]   GIFTS/xxx
------------------------------------------------
JOB: xxxx

LOADING CASE 1.

    *
```

We are "now" communicating with the RESULT postprocessor. Since it is one of the GIFTS interactive processors, we will henceforth have to employ the GIFTS command language [G1], as explained in Section 1.1. The following is a hypothetical dialogue with RESULT which should serve to highlight some of its key features as well as establishing some basic conventions for result interpretation:

First, we reflect on the GIFTS message which was printed as we entered RESULT:

```
Loading Case 1
```

The so-called "loading case" number mentioned in this module is NOT the same as the loading case number referenced during pre-processing, e.g., in BULKLB or EDITLB. Instead, it is just a pointer to the set of solutions currently residing in the

postprocessing database, and should be interpreted according to the POSTPROCESSING CORRESPONDENCE TABLE printed above. Hence, for the given example, "Loading Case 1" actually represents Solution Step 0, "Loading Case 2" represents Solution Step 5, and so on.

So, since we are presently switched (by default) to "Loading Case" 1, we may begin our evaluation of Solution Step 0, (i.e., the linear version of Step 1).

For instance, if we enter:

```
* ELEM / PLOT
```

we would obtain a picture of the deformed structure, scaled in some reasonable manner, and labelled on the periphery with such information as the model orientation, geometric scale factors, "load case", etc.

To rotate the model, say 70 degrees about the screen's x-axis, followed by 20 degrees about the screen's y-axis, and then amplify the displacements by a factor of 2, we would enter:

```
* ROTV/70,20/ SCALEDN/2/ PLOT
```

which is the corresponding sequence of GIFTS commands. (Remember, GIFTS employs the slash (/) to allow multiple commands on a single line; while GIST employs it to designate qualifiers).

To zoom in on the area of apparent maximum deformation, and find out which nodes are involved, we might enter something like:

```
* BOX / 40,50 / 100,200 / 0,300 / PN / PLOT
```

which would display that portion of the deformed structure which is contained in the global Cartesian "box" bounded by the planes X=40, X=50; y=100, y=200; and z=0, z=100; and label all nodes therein.

Then, to list the actual displacement values at the nodes we have just displayed, we may use the information command:

```
* INFDN / n1, n2, ntot /
```

where 'n1, n2, ntot' represents a sequence of node numbers that presumably containes the nodes of interest.

For extremists, a full table of nodal displacements may be obtained and sent to the line-printer via:

```
* LPON / INFDN/1,nmax/ LPOFF
```

where 'nmax' is greater than or equal to the total number of nodes in the structure, and LPON and LPOFF respectively turn the line-printer mode-switch on and off. (Note that the printing does not actually take place until the user exits from RESULT via QUIT.)

Stresses may be viewed in a variety of ways. For example, we may list them, contour them, element-code them, or indicate their principal values and directions.

To display stress contours (involving shell or continuum-type elements only) we may simply enter:

```
* CONTOUR / PLOT
```

which, by default, will contour the mid-surface values of an "effective stress" parameter (currently, the Von Mises stress). The stress ranges corresponding to each contour line are computed automatically, but may be overridden by the user via the RANGE command (e.g., for higher resolution of a particular area or stress regime). Note that all stresses are normalized with respect to the "yield stress" initially defined for the material (see Section 2.1.1).

To shift attention from the middle surface to either the top or bottom surface, the user may enter:

```
* TOP
```

or:

```
* BOTTOM
```

respectively. These surface-switch commands: TOP, MIDDLE and BOTTOM, apply to all surface elements (e.g., membranes, plates and shells) and remain active for all stress-related displays until another such command is issued. The interpretation of "top" and "bottom" is related to the element coordinate systems: x', y', z'. The top surface is at z' = z'(max) and the bottom surface at z' = z'(min).

Another way of evaluating the "effective" stresses, but in a more discrete, element-oriented fashion, is to use the FC (failure criterion) option. For example, by entering:

```
* FC / PLOT
```

the stress levels of individual elements will be indicated by special symbols appearing at their centroids. Each symbol (usually an alphanumeric character) will represent a different range of "normalized" Von Mises stress, and a 'symbol-key' (showing the correspondences) will be displayed to the right of the plot frame. (The "yield stress", which was defined for each material way back in pre-processing, is used as the normalization factor.) This type of stress display can be useful for evaluating element "mesh patterns".

To display the principal stress directions and relative magnitudes, we could enter:

```
* PRINST / PLOT
```

The principal directions at the centroid of each surface element are displayed as a tiny pair of intersecting vectors, whose lengths are proportional to the absolute value of the corresponding stress "components. To display "compressive" principal stresses only, the command CPRINST may be used instead of PRINST. Such displays may, for instance, facilitate "load path" visualization for the design-oriented analyst.

Remember that the surface on which stresses are currently being evaluated always depends on the last TOP, BOTTOM or MIDDLE command issued; the displays will be labelled accordingly.

Finally, to list the actual values of individual stress components, an appropriate information command is available:

```
* INFST /nel,ne2,ntot/
```

where 'nel,ne2,ntot' represents a sequence of element numbers. As usual, to review the element numbers before issuing the information command, the user may BOX-in on the region of interest and plot them via EN/PLOT.

The INFST command will list the basic stress components for each element, which will of course depend on type. For example: surface elements will list SIGMAX, SIGMAY and TAUXY evaluated at a centroidal location on either top, middle or bottom surfaces; while beam elements will list the axial and torsional stresses, SIGMAX, TAUXY and TAUXZ at a number of points on the mid-span cross-section.

Similar listings can be obtained for stress-resultants via the command:

```
* INFSTR /nel,ne2,ntot/
```

Keep in mind that element stresses, stress-resultants and strains are always oriented with respect to the "element" coordinate systems. For the precise definitions of these coordinate systems and the various conventions associated with them, refer to [G1] in conjunction with Appendix A.

Having evaluated the important results for one solution step, we may turn our attention to another by issuing the LDCASE command. For instance:

```
* LDCASE/2
```

would switch the RESULT pointer to "Loading Case" number 2, which,

in our illustrative example, is equivalent to Solution Step 5. We could then proceed to display and list this solution just as we did the former one.

However, if we wish to view a solution step which was not included in our original RESULT request (i.e., at the time of invocation), we will have to return to the Control Module and ask for more.

For example, suppose we wanted to go back and fetch solution step number 3. We might proceed as follows:

```
  *   QUIT
       .
       .
       .

G I S T >   RESULT  SOL = 3, 10, 7
       .
       .
       .
```

POSTPROCESSING CORRESPONDENCE TABLE

GIFTS RESULTS	STAGS SOLUTIONS
"LOAD-CASE" NO. 1	STEP 3 -- DISPLACEMENTS -- STRESSES
"LOAD-CASE" NO. 2	STEP 10 -- DISPLACEMENTS -- STRESSES

```
       .
       .
       .

-------------------------------------------------
<<>>  R E S U L T   [ Result Display ]   GIFTS/xxx
-------------------------------------------------
JOB: xxxx

LOADING CASE 1.      (Now correspondes to Solution Step 3)

    *
```

Note that the original set of solution steps (0,5 and 10) has been cleared and replaced with the new set requested. (Since we had not yet looked at step number 10, we requested it again.) This just reflects the volatility of the postprocessing datatbase, which acts essentially as a "window" to the analysis database.

Remarks

1. The above illustration of a RESULT session represents only a sampling of GIFTS postprocessing capabilities. The reader is referred to the usual references [G1-G4] for a more complete description of the commands, options and conventions.

2. The term "loading case" used within RESULT is a hold-over from the GIFTS linear analysis package. The term should eventually be removed and replaced with something more neutral, like "result number". Also, an information command to identify the contents of a particular "result number" with such things as "solution step" and "solution type" would be preferrable to the present POSTPROCESSING CORRESPONDENCE TABLE.

3. Eigenvectors are automatically transferred to the postprocessing database and assigned a RESULT "Load Case" number when the corresponding solution step is requested.

   For example, if a linear buckling analysis has been performed the GIST command:

   ```
   RESULT SOLUTION = 0
   ```

   will prepare both the linear (pre-buckling) solution and up to the first three buckling mode vectors for RESULT postprocessing. The POSTPROCESSING CORRESPONDENCE TABLE might then look as follows:

GIFTS RESULTS	STAGS SOLUTIONS
"LOAD-CASE" NO. 1	STEP 0 -- DISPLACEMENTS -- STRESSES
"LOAD-CASE" NO. 2	STEP 0 -- MODE 1
"LOAD-CASE" NO. 3	STEP 0 -- MODE 2
"LOAD-CASE" NO. 4	STEP 0 -- MODE 3

   To transfer more than 3 modes the MODE keyword may be employed, e.g.,

   ```
   G I S T >   RESULT   SOLUTION = 0   MODES = 1,10
   ```

   which would pick up the first 10 modes at solution step 0.

4. Velocity vectors will also be transferred automatically upon RESULT invocation when a particular (dynamic) solution step is

requested. Velocity components may then be listed via the GIFTS 'INFDN' command, after switching to the appropriate "Load Case" via the GIFTS 'LDCASE' command.

5. Stress resultants and strains do not yet enjoy the standard stress display options described above. To list stress resultants, use the GIFTS 'INFSTR' command which is analogous to the 'INFST' (stress information) command.

   NOTE: For beam elements, stresses must also be examined via the information command ('INFST'). The beam stress display capability present in the "GIFTS Linear Analysis Package" relies on an internal stress computation algorithm which is not applicable to external analyzers such as STAGS. Hence, all beam stress-display options are temporarily "off-limits" to the GIST user.

6. The "effective stress" computed and employed for contour and failure criterion stress plots is presently hard-wired as the Von Mises yield criterion, defined by:

```
SVM = SQRT ( .5*( SS12 + SS23 + SS31) )

  where:
          SS12 = (S1-S2)**2
          SS23 = (S2-S3)**2
          SS31 = (S3-S1)**2
  and:
          S1, S2, S3 are the principal stress
values.
```

   The effective stress is normalized with respect to the "yield stress" which was defined with the material at pre-processing time. To change the definition of effective stress, a slight modification to the STAGS->GIFTS Adaptor (G2S) would be required. Clues on performing this simple operation are given in Section 3.3. It is conceivable that a number of different effective stress measures could be programmed into the Adaptor as options to be selected at the time of RESULT invocation.

   Eventually (after surface coordinates have become fully implemented in GIFTS it should be possible to obtain contour or element-coded (perhaps with color) displays of any arbitrary stress, strain or stress resultant component. In the meantime, this item sits very high on the "GIST WISH-LIST".

7. Rather than requesting a new set of solution vectors every time RESULT is called upon, it is also possible to postprocess a fixed set of solutions over many RESULT sessions, i.e., without re-accessing the analysis database. By entering the "qualified" GIST command:

```
RESULT/OLD
```

   the RESULT processor is invoked directly, i.e., without

intervention by the STAGS-GIFTS adaptor. The user may thus resume an evaluation of whatever was last transferred to the postprocessing database. This not only eliminates the invocation delay, but enables the user to postprocess previously computed solutions WHILE WAITING FOR A BATCH-RUN TO PRODUCE NEW ONES. If you recall, the analyzer ties up the "analysis database" only; the postprocessing database may therefore be pre-stocked with solutions and evaluated while a batch computation is actually in progress.

(Note: This remark may be unnecessarily abused by pushy project managers and should therefore be whited-out after a first reading by the analyst.)

2.3.2 "COMPUTATONAL" EVALUATION

It is especially important in nonlinear structural analysis to evaluate both the "physics" and the "numerics" of the solution before pushing on to higher and higher levels of loading (or time, in transient analysis). One reason for this is the ever-present possiblity of inadvertently obtaining either unconverged or spurious solutions. In the latter case, a physical display may reveal the "spuriousness", while in the former case, a careful look at the computational statistics (e.g., relative errors, no. of iterations, stiffness update frequency and the sign of the stiffness determinant) over a series of solution steps may be required. Hence, both forms of "postprocessing" are complementary, but neither is sufficient for assuring the accuracy of a particular analysis.

There are presently three mechanisms provided by GIST for computational postprocessing. The first is to save and scrutinize the solution commentary which is automatically produced by the STAGS analyzer each time the COMPUTE command is issued. This is easily accomplished by using the OUTPUT keyword, as described in Section 1.5 under the COMPUTE command.

The second mechanism is to employ the REVIEW SOLUTION command. This GIST Control Module feature provides essentially the same information as in the previous case, except only for those solution steps which have been successfully archived. (Again, see Section 1.5 for the proper usage of this command.)

The third mechanism for computational postprocessing is just barely developed, but potentially powerful. It is to obtain a spatial display of the "residual force" vector at specific solution steps. This gives the analyst an idea of how well "equilibrium" is being satisfied throughout the structure. At the moment the best that can be obtained is either a selective print-out or a "deformed geometry" - type plot of "internal forces" via the GIFTS RESULT postprocessor (see previous subsection). The difference between the "internal force" vector and the "residual force" vector is the "external force" vector. Hence, the internal forces will approximate (in magnitude) the external forces, and will yield the "reaction forces" at degrees of freedom where displacement boundary conditions have been prescribed. See the STAGS User's Manual (Section 6) for a more precise definition of the internal force vector (which is alternatively referred to as the "equilibrium force" vector therein).

It would be desirable in future development efforts to introduce full graphical-display capabilities for computational statistics. This would allow the analyst to obtain, for example, contour plots of residual forces, or "historical" plots of virtually any solution parameter as a function of load or time step.

2.4 DATABASE MANAGEMENT

The following GIST commands are associated with database management:

CLEAR	....	Erases bulk portions of the database
MANAGE	....	Invokes an interactive database "editor"
REVIEW	...	Lists current database table of contents

Since most data-management operations with GIST are automatic (going on quietly behind the scenes of structural analysis) only a few basic maintenance functions are left to the user. The purpose of this section is to describe the database, or disk archival, layout for a GIST analysis and to show the user how to (1) monitor it periodically, (2) discard it when finished, and (3) edit it when necessary. These functions are handled by the REVIEW, CLEAR, and MANAGE commands, respectively.

2.4.1 THE GIST DATABASE LAYOUT

All data generated by the GIST system that may be of lasting significance either during or after the course of structural analysis is stored in a "database". The GIST database actually represents a collection of (permanent) disk files on which data is neatly archived, for convenient access by both the user and the software network.

Every GIST analysis, or "Job", generates its own independent database and corresponding set of disk files. Files associated with a particular Job are referred to collectively as the "Job database", and are accordingly tagged by the "Jobname" (defined in Section 1.1). The first part of each file name is identical to the Jobname, and the last part (or "extension") reveals something about its contents.

The Job database is partitioned (both logically and physically) into:

(1) the Pre-processing Database
(2) the Analysis Database
(3) the Post-processing Databse

As expected, the pre- and post-processing databases are principally the domain of the GIFTS processors, while the analysis database is principally the domain of the STAGS processors. All three, however, are accessible to the user from the GIST Control Module.

The Pre-processing Database

The pre-processing database consists of a set of files which contain the current GIFTS definition of the model. They are summarized as follows:

File Name	Contents
Job .PAR	GIFTS problem status parameters
Job .FIL	File size and data management parameters
Job .LIN	Key line generation data
Job .GRD	Grid (surface) generation data
Job .PTS	Basic nodal point data
Job .ELT	Basic element data
Job .MAT	Material properties
Job .THS	Thickness (i.e., section) properties
Job .LDS	Nodal load vectors
Job .ELD	Element loads