SAND92-2289
Unlimited Release
Printed December, 1992

Distribution
Category UC-705

GROPE:
A GENESIS/EXODUS
Database Examination Program

Gregory D. Sjaardema
Solid and Structural Mechanics Department
Sandia National Laboratories
Albuquerque, New Mexico 87185

Abstract

GROPE is a program that examines the input to a finite element analysis (which is in the GENESIS database format) or the output from an analysis (in the EXODUS database format). GROPE allows the user to examine any value in the database. The display can be directed to the user's terminal or to a print file.

Acknowledgement

The original version of GROPE was written by Amy P. Gilkey and documented in an internal memorandum. The documentation is being rereleased as a SAND Document with minor changes.

Contents

1 Introduction **7**

2 Command Input **9**
2.1 Database Selection and Display Commands 10
2.2 Miscellaneous Commands 14

3 Informational and Error Messages **15**

4 Executing GROPE **17**
4.1 Execution Files 17
4.2 Special Software 17

5 References **19**

A The EXODUS Database Format **21**

1 Introduction

GROPE is a program that examines the input to a finite element analysis (which is in the GENESIS[1] database format) or the output from an analysis (in the EXODUS[2]database format). GROPE allows the user to examine any value in the database. The display can be directed to the user's terminal or to a print file.

This manual assumes that the reader is familiar with the GENESIS and EXODUS database formats. The formats are described briefly in appendix A on page 21 . The GENESIS[1] and EXODUS[2] manuals describe the formats in detail.

GROPE is very useful for locating database errors that have been flagged by another program. GROPE attempts to give as much information about an erroneous database as possible. If a read error occurs, an error message is displayed showing the record that was being read and the type of error. If a very minor read error occurs (such as a short title record), GROPE continues reading the database. Most errors cause GROPE to stop reading the database and prompt for user input.

When the program moves to a new time step, only the destination time step variables are read; the intermediate time steps are scanned over. The CHECK command verifies the correctness of all time step records.

All record items are set to a default value before each database record is read. If a short record is read, the default values remain in the unread portion of the record and are displayed when the user requests the contents of the record.

2 Command Input

The commands are in free-format and must adhere to the following syntax rules.

- Valid delimiters are a comma or one or more blanks.
- Either lowercase or uppercase letters are acceptable, but lowercase letters are converted to uppercase.
- A “$” character in any command line starts a comment. The “$” and any characters following it on the line are ignored.
- A command may be continued over several lines with an “>” character. The “>” and any characters following it on the current line are ignored and the next line is appended to the current line.

Each command has an action keyword or “verb” followed by a variable number of parameters. The command verb is a character string. It may be abbreviated, as long as enough characters are given to distinguish it from other commands.

The meaning and type of the parameters is dependent on the command verb. Most command parameters are optional. If an optional parameter field is blank, a command-dependent default value is supplied. Below is a description of the valid entries for parameters.

- A numeric parameter may be a real number or an integer. A real number may be in any legal FORTRAN numeric format (e.g., 1, 0.2, -1E-2). An integer parameter may be in any legal integer format.
- A string parameter is a literal character string. Most string parameters may be abbreviated.
- Variable names must be fully specified. The blank delimiter creates a problem with database variable names with embedded blanks. The program handles this by deleting all embedded blanks from the input database names. For example, the variable name “SIG R” must be entered as “SIGR”. The blank must be deleted in any references to the variable. All database names appear exactly as input in all displays.
- Several parameters allow a range of values. A range is in one of the following forms:
- “n_1” selects value n_1,
- “n_1 TO n_2” selects all values from n_1 to n_2,
- “n_1 TO n_2 BY n_3” selects all values from n_1 to n_2 stepping by n_3, where n_3 may be positive or negative.

If the upper limit of the range is greater than the maximum allowable value, the upper limit is changed to the maximum without a message.

The notation conventions used in the command descriptions are:

- The command verb is in **bold** type.
- A literal string is in all uppercase SANSERIF type and should be entered as shown (or abbreviated).
- The value of a parameter is represented by the parameter name in *italics*.
- A literal string in square brackets ("[ ]") represents a parameter option which is omitted entirely (including any following comma) if not appropriate. These parameters are distinct from most parameters in that they do not require a comma as a place holder to request the default value.
- The default value of a parameter is in angle brackets ("< >"). The initial value of a parameter set by a command is usually the default parameter value. If not, the initial setting is given with the default or in the command description.

2.1 Database Selection and Display Commands

SELECT *option* <No Default>

SELECT selects the database information for the LIST and PRINT commands. Items are displayed in the order listed in the SELECT command. If the ADD parameter is present, the listed values are added to the current selection.

SELECT NODES [ADD,] $node_range_1$, $node_range_2$, ... <all nodes>

selects the nodes. Each *node_range* must be in the range form described under Command Input.

SELECT ELEMENTS [ADD,] $element_range_1$, $element_range_2$, ... <all elements>

selects the elements. Each *element_range* must be in the range form described under Command Input. The selected element blocks are changed to select only those blocks that include a selected element. The selected elements are reordered so that they are grouped by element block.

SELECT BLOCKS or MATERIAL [ADD,] $block_id_1$, $block_id_2$, ... <all element blocks>

selects the element blocks. The *block_id* is the element block identifier displayed by the LIST BLOCKS command. The selected elements are changed to select all the elements in the selected blocks and no elements in non-selected blocks.

SELECT NSETS [ADD,] set_id_1, set_id_2, ... <all node sets>

selects the node sets. The *set_id* is the node set identifier displayed by the LIST NSETS command.

SELECT SSETS [ADD,] set_id_1, set_id_2, ... <all side sets>

selects the side sets. The *set_id* is the side set identifier displayed by the LIST SSETS command.

SELECT HVARS [ADD,] *history_variable*$_1$, *history_variable*$_2$, ... <all history variables>

selects the history variables.

SELECT GVARS [ADD,] *global_variable*$_1$, *global_variable*$_2$, ... <all global variables>

selects the global variables.

SELECT NVARS [ADD,] *nodal_variable*$_1$, *nodal_variable*$_2$, ... <all nodal variables>

selects the nodal variables.

SELECT EVARS [ADD,] *element_variable*$_1$, *element_variable*$_2$, ... <all element variables>

selects the element variables.

SELECT READ *nstep* <1>

changes the selected time step to *nstep* steps forward (if *nstep* is positive) or backward (if *nstep* is negative) of the current selected step.

SELECT STEP *nstep* <current selected step>

changes the selected time step to step *nstep*.

SELECT TIME *time* <current selected step time>

changes the selected time step to the step that has the time nearest *time*.

LIST *option* <No Default>

LIST displays the database information specified by *option* on the user's terminal. The "selected" items are specified with the SELECT command.

LIST VARS

displays a summary of the database. The summary includes the database title; the number of nodes, elements, and element blocks; the number of node sets and side sets; and the number of each type of variable.

LIST COORDINA

displays the coordinate values for the selected nodes.

LIST MAP

displays the element order map for all elements.

LIST BLOCKS or MATERIAL

displays a summary of the selected element blocks. The summary includes the block identifier, the number of elements in the block, the number of nodes per element, and the number of attributes per element.

LIST LINK or CONNECTI

displays the nodal connectivity of the selected elements.

LIST ATTRIBUT

displays the attribute values associated with the selected elements.

LIST NSETS

displays a summary of the selected node sets. The summary includes the set identifier and the number of nodes in the set.

LIST NNODES

displays the numbers of the nodes in the selected node sets.

LIST NFACTORS

displays the distribution factors of the nodes in the selected node sets.

LIST SSETS

displays a summary of the selected side sets. The summary includes the set identifier, the number of elements in the set, and the number of nodes in the set.

LIST SELEMS

displays the numbers of the elements in the selected side sets.

LIST SNODES

displays the numbers of the nodes in the selected side sets.

LIST SFACTORS

displays the distribution factors of the nodes in the selected side sets.

LIST QA

displays the QA records and the information records.

LIST NAMES

displays the names of the history, global, nodal, and element variables.

LIST HISTORY or HVARS

displays the values of the selected history variables for the selected time step.

LIST GLOBALS or GVARS

displays the values of the selected global variables for the selected time step.

LIST NODALS or NVARS

displays the values of the selected nodal variables for the selected nodes for the selected time step.

LIST ELEMENTS or EVARS

displays the values of the selected element variables for the selected elements for the selected time step.

LIST STEPS

displays the number of time steps and the minimum and maximum time step times.

LIST TIMES

displays the step numbers and times for all time steps on the database.

PRINT *option* <no parameter>

PRINT writes the database information specified by *option* to the print file. The valid *options* are described under the LIST command. If the file cannot be opened, a message is printed and the command is ignored.

2.2 Miscellaneous Commands

CHECK

CHECK checks the database for violations of the GENESIS or EXODUS standards [bib:genesis,bib:exodus] and prints a message if any violations are found. The following checks are made:

- The element order map contains each element exactly once.
- The element block identifiers are unique.
- The sum of the number of elements in all element blocks equals the total number of elements.
- The connectivity for each element contains valid node numbers.
- The node set identifiers are unique.
- The node set maximum node list index equals the length of the node set node list.

- The node set node list contains valid node numbers.
- The side set identifiers are unique.
- The side set maximum element list index equals the length of the side set element list.
- The side set element list contains valid element numbers.
- The side set maximum node list index equals the length of the side set node list.
- The side set node list contains valid node numbers.

CHECK also reads all the time step records and displays a message if an error is found in any record format.

MINMAX *variable* <No Default> , ALL or THIS <THIS>

MINMAX prints the minimum and maximum values of the given variable and at which step and node or element (if any) the first instance of the values was found. If the second parameter is ALL, all time steps are searched for the minimums and maximums; otherwise only the current time step is searched.

HELP *command* <no parameter>

HELP displays information about the program command given as the parameter. If no parameter is given, all the command verbs are displayed. HELP is system-dependent and may not be available on all systems.

EXIT

EXIT exits from the program.

3 Informational and Error Messages

GROPE first reads the database and scans the time step information. If a database format error is discovered before the time steps are read, the program prints an error of the following format:

DATABASE ERROR - Reading *database* item

and the type of error that was detected. If a format error is found while scanning the time steps, the following error message is printed:

WARNING - End-of-file during time steps.

The program stops reading the database when an error is found, but all information read before the error (including information about the time step that contains the error) is available for display.

After the database is read, command input is requested from the user. Most commands request a display of information. The response from GROPE is either an error message or the information requested. The print file is opened when the first PRINT command is issued. If the print file cannot be opened, the following message appears:

WARNING - Print file cannot be opened

and the PRINT command is ignored. If the information is directed to the print file, no text appears on the user's terminal except error messages. An appropriate header identifies the information on the print file.

The program allocates memory dynamically as it is needed. If the system runs out of memory, the following message is printed:

FATAL ERROR - Too much dynamic memory requested

and the program aborts. The user should try to obtain more memory on the system.

GROPE has certain programmer-defined limitations. The limits are not specified in this manual since they may change. In most cases the limits are chosen to be more than adequate. If the user exceeds a limit, a message is printed. If the user feels the limit is too restrictive, the code sponsor should be notified so the limit may be raised in future releases of GROPE .

4 Executing GROPE

The details of executing GROPE are dependent on the system being used. The system manager of any system that runs GROPE should provide a supplement to this manual that explains how to run the program on that particular system. Site supplements for all currently supported systems are in Appendix [Ref: appx:site] .

4.1 Execution Files

The table below summarizes GROPE 's file usage.

Description	**Unit Number**	**Type**	**File Format**
User input	standard input	input	Section 2
User output	standard output	output	ASCII
EXODUS database	11	input	Appendix A
Print file	20	optional output	ASCII

All files must be connected to the appropriate unit before GROPE is run. Each file (except standard input and output) is opened with the name retrieved by the EXNAME routine of the SUPES library[4].

4.2 Special Software

GROPE is written in ANSI FORTRAN-77[3] with the exception of the following system-dependent features:

- the VAX VMS help facility and
- the OPEN options for the files.

GROPE uses the following software packages:

- the SUPES package [4] which includes dynamic memory allocation, a free-field reader, and FORTRAN extensions.

5 References

[1]Lee M. Taylor, Dennis P. Flanagan, and William C. Curran, “The GENESIS Finite Element Mesh File Format,” SAND86-0910, Sandia National Laboratories, Albuquerque, New Mexico, May 1986.

[2]William C. Mills-Curran, Amy P. Gilkey, Dennis P. Flanagan, “EXODUS: A Finite Element File Format for Pre- and Post-Processing,” SAND87-2997, Sandia National Laboratories, Albuquerque, New Mexico, September 1988.

[3]*American National Standard Programming Language FORTRAN*, American National Standards Institute, ANSI X3.9-1978, New York, 1978.

[4]John R. Red-Horse, William C. Curran, and Dennis P. Flanagan, “SUPES Version 2.1 – A Software Utility Package for the Engineering Sciences,” SAND90–0247, Sandia National Laboratories, Albuquerque, New Mexico, May 1990.

A The EXODUS Database Format

The following code segment reads an EXODUS database. The first segment is the GENESIS database format.

```
C   --Open the EXODUS database file
      NDB = 9
      OPEN (UNIT=NDB, ..., STATUS='OLD', FORM='UNFORMATTED')
C   --Read the title
      READ (NDB) TITLE
C      --TITLE - the title of the database (CHARACTER*80)
C   --Read the database sizing parameters
      READ (NDB) NUMNP, NDIM, NUMEL, NELBLK,
     &   NUMNPS, LNPSNL, NUMESS, LESSEL, LESSNL, NVERSN
C      --NUMNP - the number of nodes
C      --NDIM - the number of coordinates per node
C      --NUMEL - the number of elements
C      --NELBLK - the number of element blocks
C      --NUMNPS - the number of node sets
C      --LNPSNL - the length of the node sets node list
C      --NUMESS - the number of side sets
C      --LESSEL - the length of the side sets element list
C      --LESSNL - the length of the side sets node list
C      --NVERSN - the file format version number
C   --Read the nodal coordinates
      READ (NDB) ((CORD(INP,I), INP=1,NUMNP), I=1,NDIM)
C   --Read the element order map (each element must be listed once)
      READ (NDB) (MAPEL(IEL), IEL=1,NUMEL)
C   --Read the element blocks
      DO 100 IEB = 1, NELBLK
C      --Read the sizing parameters for this element block
         READ (NDB) IDELB, NUMELB, NUMLNK, NATRIB
C         --IDELB - the element block identification (must be unique)
C         --NUMELB - the number of elements in this block
C         --   (the sum of NUMELB for all blocks must equal NUMEL)
C         --NUMLNK - the number of nodes defining the connectivity
C         --   for an element in this block
C         --NATRIB - the number of element attributes for an element
C         --   in this block
C      --Read the connectivity for all elements in this block
         READ (NDB) ((LINK(J,I), J=1,NUMLNK, I=1,NUMELB)
C      --Read the attributes for all elements in this block
         READ (NDB) ((ATRIB(J,I), J=1,NATRIB, I=1,NUMELB)
  100 CONTINUE
C   --Read the node sets
      READ (NDB) (IDNPS(I), I=1,NUMNPS)
C      --IDNPS - the ID of each node set
      READ (NDB) (NNNPS(I), I=1,NUMNPS)
C      --NNNPS - the number of nodes in each node set
      READ (NDB) (IXNNPS(I), I=1,NUMNPS)
C      --IXNNPS - the index of the first node in each node set
C      --   (in LTNNPS and FACNPS)
      READ (NDB) (LTNNPS(I), I=1,LNPSNL)
```

```
C      --LTNNPS - the nodes in all the node sets
      READ (NDB) (FACNPS(I), I=1,LNPSNL)
C      --FACNPS - the factor for each node in LTNNPS
C   --Read the side sets
      READ (NDB) (IDESS(I), I=1,NUMESS)
C      --IDESS - the ID of each side set
      READ (NDB) (NEESS(I), I=1,NUMESS)
C      --NEESS - the number of elements in each side set
      READ (NDB) (NNESS(I), I=1,NUMESS)
C      --NNESS - the number of nodes in each side set
      READ (NDB) (IXEESS(I), I=1,NUMESS)
C      --IXEESS - the index of the first element in each side set
C      --   (in LTEESS)
      READ (NDB) (IXNESS(I), I=1,NUMESS)
C      --IXNESS - the index of the first node in each side set
C      --   (in LTNESS and FACESS)
      READ (NDB) (LTEESS(I), I=1,LESSEL)
C      --LTEESS - the elements in all the side sets
      READ (NDB) (LTNESS(I), I=1,LESSNL)
C      --LTNESS - the nodes in all the side sets
      READ (NDB) (FACESS(I), I=1,LESSNL)
C      --FACESS - the factor for each node in LTNESS
```

A valid GENESIS database may end at this point or at any point until the number of variables is read.

```
C   --Read the QA header information
      READ (NDB, END=900) NQAREC
C      --NQAREC - the number of QA records (must be at least 1)
      DO 110 IQA = 1, MAX(1,NQAREC)
         READ (NDB) (QATITL(I,IQA), I=1,4)
C         --QATITL - the QA title records; each record contains:
C         --   1) analysis code name (CHARACTER*8)
C         --   2) analysis code qa descriptor (CHARACTER*8)
C         --   3) analysis date (CHARACTER*8)
C         --   4) analysis time (CHARACTER*8)
  110 CONTINUE
C   --Read the optional header text
      READ (NDB, END=900) NINFO
C      --NINFO - the number of information records
      DO 120 I = 1, NINFO
         READ (NDB) INFO(I)
C         --INFO - extra information records (optional) that contain
C         --   any supportive documentation that the analysis code
C         --   developer wishes (CHARACTER*80)
  120 CONTINUE
C   --Read the coordinate names
      READ (NDB, END=900) (NAMECO(I), I=1,NDIM)
C      --NAMECO - the coordinate names (CHARACTER*8)
C   --Read the element type names
      READ (NDB, END=900) (NAMELB(I), I=1,NELBLK)
C      --NAMELB - the element type names (CHARACTER*8)
```

The GENESIS section of the database ends at this point.

```
C   --Read the history, global, nodal, and element variable information
      READ (NDB, END=900) NVARHI, NVARGL, NVARNP, NVAREL
C      --NVARHI - the number of history variables
C      --NVARGL - the number of global variables
C      --NVARNP - the number of nodal variables
C      --NVAREL - the number of element variables
      READ (NDB)
     &   (NAMEHV(I), I=1,NVARHI),
     &   (NAMEGV(I), I=1,NVARGL),
     &   (NAMENV(I), I=1,NVARNP),
     &   (NAMEEV(I), I=1,NVAREL)
C      --NAMEHI - the history variable names (CHARACTER*8)
C      --NAMEGV - the global variable names (CHARACTER*8)
C      --NAMENV - the nodal variable names (CHARACTER*8)
C      --NAMEEV - the element variable names (CHARACTER*8)
      READ (NDB) ((ISEVOK(I,J), I=1,NVAREL), J=1,NELBLK)
C      --ISEVOK - the name truth table for the element blocks;
C      --   ISEVOK(i,j) refers to variable i of element block j;
C      --   the value is 0 if and only if data will NOT be output for
C      --   variable i for element block j (otherwise the value is 1)
C   --Read the time steps
  130 CONTINUE
         READ (NDB, END=900) TIME, HISTFL
C         --TIME - the time step value
C         --HISTFL - the time step type flag:
C         --   0.0 for all variables output ("whole" time step) else
C         --   only history variables output ("history-only" time step)
C
         READ (NDB) (VALHV(IVAR), IVAR=1,NVARGL)
C         --VALHV - the history values for the current time step
         IF (HISTFL .EQ. 0.0) THEN
            READ (NDB) (VALGV(IVAR), IVAR=1,NVARGL)
C            --VALGV - the global values for the current time step
            DO 140 IVAR = 1, NVARNP
               READ (NDB) (VALNV(INP,IVAR), INP=1,NUMNP)
C               --VALNV - the nodal variables at each node
C               --   for the current time step
  140       CONTINUE
            DO 160 IBLK = 1, NELBLK
               DO 150 IVAR = 1, NVAREL
                  IF (ISEVOK(IVAR,IBLK) .NE. 0) THEN
                     READ (NDB) (VALEV(IEL,IVAR,IBLK),
     &                  IEL=1,NUMELB(IBLK))
C                     --VALEV - the element variables at each element
C                     --   for the current time step
                  END IF
  150          CONTINUE
  160       CONTINUE
         END IF
C      --Handle time step data
         ...
      GOTO 130
  900 CONTINUE
```

```
C   --Handle end of file on database ...
```

Distribution

1	1400	E. L. Barsis
1	1401	J. R. Asay
1	1402	S. S. Dosanjh
1	1403	G. S. Davidson
1	1404	J. A. Ang
13	1425	J. H. Biffle & staff
50	1425	M. K. Smith
1	1431	J. M. McGlaun
1	1431	K. G. Budge
1	1431	J. S. Peery
1	1432	W. T. Brown
1	1433	J. W. Swegle
15	1434	D. R. Martinez & staff
1	1500	D. J. McCloskey
1	1501	C. W. Peterson
1	1502	P. J. Hommert
1	1503	L. W. Davison
1	1504	D. J. McCloskey, actg
1	1511	J. S. Rottler
1	1511	D. K. Gartling
1	1511	M. W. Glass
1	1511	P. L. Hopkins
1	1511	M. J. Martinez
1	1511	P. A. Sackinger
1	1511	P. R. Schunk
1	1511	J. D. Zepper
1	1512	A. C. Ratzel
1	1513	R. D. Skocypec
1	1513	R. G. Baca
1	1513	B. L. Bainbridge
1	1513	R. E. Hogan, Jr.
1	1513	J. L. Moya
1	1551	W. P. Wolfe
1	1552	C. E. Hailey
1	1553	W. L. Hermina
1	1554	W. H. Rutledge
15	1561	H. S. Morgan & staff
13	1562	R. K. Thomas & staff
10	1562	G. D. Sjaardema
1	1832	J. M. Ramage
1	2565	S. T. Montgomery
1	6313	J. Jung
1	6411	A. S. Benjamin
1	6423	J. F. Dempsey
1	6513	D. S. Oscar
1	6522	J. D. Miller
5	7141	Technical Library
1	7151	Technical Publications
10	7613-2	Document Processing for DOE/OSTI
1	8523-2	Central Technical Files
6	8741	G. A. Benedetti & staff
1	8742	M. R. Birnbaum
1	8742	J. J. Dike
1	8742	L. I. Weingarten
5	8743	M. L. Callabresi & staff