NASA-CR-172244
19850026204

NASA Contractor Report 172244

SYSTEM MAINTENANCE MANUAL FOR MASTER: MODELING OF AERODYNAMIC SURFACES BY THREE DIMENSIONAL EXPLICIT REPRESENTATION

S. G. Gibson

BOEING COMMERCIAL AIRPLANE COMPANY
P.O. BOX 3707, SEATTLE, WASHINGTON 98124

CONTRACT NAS1-15325
APRIL 1983

National Aeronautics and
Space Administration

Langley Research Center
Hampton, Virginia 23665

NASA Contractor Report 172244

SYSTEM MAINTENANCE MANUAL FOR MASTER: MODELING OF AERODYNAMIC SURFACES BY THREE DIMENSIONAL EXPLICIT REPRESENTATION

S. G. Gibson

BOEING COMMERCIAL AIRPLANE COMPANY
P.O. BOX 3707, SEATTLE, WASHINGTON 98124

CONTRACT NAS1-15325
APRIL 1983

National Aeronautics and
Space Administration

Langley Research Center
Hampton, Virginia 23665

N85-34517#

FOREWORD

This report specifies the implementation of MASTER, a system of computer programs used to model three-dimensional geometry configurations. The work was conducted under Subtask 4.3.3 of NASA contract NAS1-15325 from February 1981 through February 1983. The contract was managed by the NASA Energy Efficient Transport Office (EETPO), headed by Mr. R.V. Hood, which is a part of the Aircraft Energy Efficiency (ACEE) program organization at the Langley Research Center. Mr. D.B. Middleton was the technical monitor for the contract. Mr. D.E. Reubush of the Propulsion Aerodynamics group at Langley assisted in the installation of MASTER. The work was performed in the Engineering organization of the Boeing Commercial Airplane Company and in the Energy and Technology Applications organization of the Boeing Computer Services Company. Key contractor personnel responsible for the material in this report were:

G.W. Hanks
Program Manager

B.W. Farquhar
Project Manager

J.L. Colehour
BCAC Propulsion Technology

D.R. Ferguson
ETA Computational Matematics

S.G. Gibson
BCAC Propulsion Technology

R.A. Mastro
ETA Computational Mathematics

B.S. Terrill
BCAC Propulsion Technology

D.A. Vanrossum
ETA Information Services

CONTENTS

1.0 INTRODUCTION

This document is the system maintenance manual for "Modeling of Aerodynamic Surfaces by Three-dimensional Explicit Representation" (MASTER), a system of programs with which engineers can loft the surface geometry of configurations and extract geometric information from the surface model. It explains the details of MASTER's implementation, and is written for maintenance programmers. The user's manual describes the function of the system and explains how to operate it. (See Reference 1.) The user's manual also defines data formats and gives a set of examples. (This manual refers to the user's manual, rather than duplicating its contents.)

2.0 ACRONYMS AND KEYWORDS

1. BEGIN
   Control statement calling a MASTER procedure (or any other CCL procedure)
2. CCL
   Cyber Control Language
3. CDC
   Control Data Corporation
4. CFD
   Computational Fluid Dynamics (also the data format for combined mesh and intersection-normal output)
5. CUR
   Curve data format (also file name for procedure MSHNRM intersection curves).
6. DDP
   The program that creates SDMS data-definition files from an input text file.
7. DRAWIT
   Procedure for graphic display of surface models and intersection normals
8. GENTRN
   Procedure to generate coordinate-transformation data
9. INPUT
   Default file name for job-input data
10. LSTOPT
    Keyword to change the level of listing from procedure MSHNRM (within mesh data)
11. MASTER
    Modeling of Aerodynamic Surfaces by Three-dimensional Explicit Representation
12. MSH
    Mesh data format (also file name for procedure MSHNRM or procedure NRMCFD input)
13. MSHNRM
    Procedure to compute mesh/surface intersection normals
14. NEWCFD
    File name for CFD-format output (from procedure NRMCFD)
15. NEWNRM
    File name for intersection- normal output (from procedure NRMREV)
16. NEWSIL
    File name for surface-description output (from procedure TRNSIL or from procedure REGSIL)
17. NOS
    Network Operating System (for CDC computers)
18. NRM
    Intersection-normal data format (also file name for procedure MSHNRM output or procedure NRMCFD input)
19. NRMCFD
    Procedure to format intersection normals for CFD input
20. NRMREV
    Procedure for intersection-normal reversal
21. OLDCFD
    File name for CFD-format input (for procedure NRMCFD)
22. OLDNRM
    File name for intersection-normal input (for procedure NRMREV)
23. OLDSIL
    File name for surface-description input (for procedure TRNSIL or for procedure REGSIL)

24. OPTION
    File name for program-control selection input for one of the additional procedures of MASTER
25. OUT
    Listing file from MASTER procedures in an interactive job
26. OUTPUT
    Default file name for job output data, which contains listing from MASTER procedures in a batch job
27. PC
    Parametric Cubic
28. PLINT
    The MASTER subroutine that computes plane/patch intersection curves
29. PLOT-10
    The subroutine library supporting Tektronix storage-tube graphics terminals
30. REGSIL
    Procedure to regulate point spacing along input surface-description curves
31. SDMS
    Scientific Data Management System. The database manager used within MASTER
32. SEC
    Section-curve data format (also file name for procedure SRFINT output)
33. SELECT
    A program associated with MASTER that replaces REGSIL
34. SIL
    Surface Input Language, the surface-description data format (also file name for procedure SILSRF output)
35. SILSRF
    Procedure to model surfaces
36. SRF
    Surface-model data format (also file name for procedure SILSRF output or procedure MSHNRM input)
37. SRFINT
    Procedure for surface/surface intersection
38. TRN
    Coordinate-transformation data format (also file name for procedure TRNSIL input)
39. TRNSIL
    Procedure for coordinate transformation within surface-description data
40. UPDATE
    The CDC utility program used to maintain MASTER source code

3.0 ENVIRONMENT

This section specifies the computing environment in which MASTER is currently implemented. (The environment of the implementation on the NASA Langley Research Center (LaRC) ACD computing system is documented.)

3.1 COMPUTING HARDWARE

MASTER is executed on Cyber series computers, manufactured by the Control Data Corporation.

3.2 OPERATING SYSTEM

MASTER is executed by Version 1.2 of the NOS operating system for CDC computers. (See Reference 2.) The LaRC version is at level 528.

3.3 COMPILER

MASTER is programmed in FORTRAN. Version 4.8 of the FTN4 compiler is used. (See Reference 3.) This is the CDC version of FORTRAN 66.

3.4 UTILITIES

A pair of other utility programs are required for the maintenance of MASTER. Version 1 of the CDC UPDATE utility is used to control source-code revisions. (See Reference 4. Version 1.4 is used at LARC.) An interactive text editor is used to prepare system-update job decks from the templates described in Section 4.3.3. Any editor which supports global string substitutions and accepts the EOR record marks of CDC computers is sufficient.

3.5 DATABASE MANAGER

The user's data is handled by MASTER as local files, but some programs use the SDMS database manager to handle structured temporary data. (See Reference 5.)

3.6 GRAPHICS HARDWARE

DRAWIT, the graphic-display program within MASTER, is designed to be executed from Textronix 4010-series interactive graphics terminals. These are stroke graphics terminals which use a storage tube, rather than a refreshed display. The DRAWIT graphic display functions use the standard grahics software supplied by Tektronix with these terminals, the PLOT-10 package. (See Reference 6.)

4.0 SYSTEM FILES

This section lists the system files found on the MASTER system account. It also shows the relationships between these files.

4.1 FILE NAME CONVENTIONS

Some general conventions are used to group related files together, and some parts of file names are identified as abstract variables, which are referred to by their symbolic names.

More than one version of MASTER can exist on the same system account. Each version has a separate copy of all the program information. This permits a programmer to test modifications to an experimental version without affecting the version that is accessed by general users. Files from each version have the same initial letter, which identifies the version. (The symbol "+" is used in this document to stand for this letter.)

The files for each version are mostly gathered into 5 parts, each of which has a 4-letter name:

1. Procedure files are found in the part named "PROC".
2. Sample data and miscellaneous files are found in the part named "DATA".
3. Main programs are found in the part named "MAIN".
4. Most of the subroutines are found in the part named "SUBS".
5. Selected subroutines are found in the part named "LIBS". (The selected subroutines are the ones which originate from NASA-funded work.)

An arbitrary part name is symbolized "XXXX" in this manual.

4.2 SOURCE FILES

Source data for each part is stored on an UPDATE program-library file, with the name "+PLXXXX". Each of these program libraries was initially formed from a source file with the name "+UPXXXX".

4.3 INTERMEDIATE BINARIES

The relocatable object code produced by compiling the FORTRAN source code is stored on 3 files: +BRMAIN, +BRSUBS, and +BRLIBS. File +MASBIN contains the combined contents of +BRSUBS and +BRLIBS, with the subroutines sorted alphabetically if possible.

File DDP is found on the MASTER system account. This file contains the program to convert SDMS data-definition text to binary form. (See Reference 5.)

4.4 INTERMEDIATE TEXT FILES

Files +PROC and +DATA contain listings of the corresponding source data, with UPDATE line numbers.

Files SILDBD and IBRDBD contain data-definition text for use with SDMS.

4.5 EXECUTION FILES

MASTER users access the system through a procedure file, whose name usually starts with the version letter "+", this file has the symbolic name "VERSION".

Each main program is executed from a file of relocatable object code. The symbolic name for an arbitrary main program is "PROGRAM", and the corresponding file is named "+PROGRAM". Most programs are stored without the associated subroutines, which are found in a library. This is called a user library in the operating-system manual, but this manual calls it the MASTER system library. The library is found on file +MASLIB.

The library SDMSLIB is found on the MASTER system account; the SDMS database manager is supported by this library. The binary data definitions used by SILSRF and SRFINT are found respectively on the direct-access files ZZZSIL and ZZZIBR; they are respectively formed with DDP from the SILDBD and IBRDBD definition text files.

File MESSAGE contains an interactive message-display program that is used by MASTER; its source code is on file FMESSAGE.

4.6 MAINTENANCE FILES

File +ERRLOG is an error log that contains listing information and data from aborted executions of MASTER procedures. (The first EOF-partition is a copy of file NEWPAGE that heads the list.) Any following EOF-partitions are error packages. The first record in an error package is listing information. The next record is a copy of the procedure record that was executed. Any following records contain data.

File J+INIT is the job deck to create a system version from the initial source data. File J+REGEN is a job deck which reads the source libraries and creates the other files for this version from the current source version.

System updates are made from job decks, which are stored as permanent files. These decks have file names like "+JAN01A" with the month, day, and final letter copied from the update identifier. Files JUPPROC, JUPDATA, JUPMAIN, JUPSUBS, and JUPLIBS are templates that can be edited into the job decks for system updates. File UPDATES is a log of system updates for all the versions at the installation.

4.7 ASSOCIATED PROGRAM FILES

There are some programs which are associated with MASTER but have not yet been included. Programs FIXNRM, MSH465, and SELECT are in this category. (They are mentioned in the demonstration of P465 input preparation, see Reference 7 for more details. They are not documented in this manual.) The general convention for these files is that the binary file has the same name as the program and that the source-code file has the letter "F" as a prefix.

4.8 DEMONSTRATION FILES

Some files on the system account demonstrate MASTER usage, to help users learn to operate the system. Job deck J+DEMO will demonstrate the examples from the user's manual. (See Section 6 of the MASTER user's manual.) Job decks JDMPREP, JDMSRF, JDMNRM, JDM465, and JDM465E form an application-oriented demonstration. The input files are these demonstration jobs are accessed from the system account, and copies of their correct outputs are there for comparison.

4.9 OTHER FILES

File NEWPAGE is simply a page header, which is used to start a new page when assembling error packages. File STDTRN contains a set of standard coordinate transformations, which are compatible with procedure TRNSIL.

5.0 SYSTEM MAINTENANCE

This section describes the major activities involved in maintaining MASTER according to current practices.

5.1 SYSTEM INSTALLATION

A new version of MASTER can be created by job J+INIT. This job reads an initial version of the source data for all 5 parts from files +UPPROC, +UPDATA, +UPMAIN, +UPSUBS, and +UPLIBS. It creates all the other system files. This is better suited than J+REGEN for creating the first version of MASTER at an installation, because a text form of the source data is used. (Text files are simpler to transmit between computers than binary files, such as UPDATE libraries.) After installation of the initial version, the existing system updates should be made to this version.

5.2 SYSTEM REGENERATION

Job J+REGEN also creates a new copy of the MASTER system files, but it starts with the UPDATE library files "+UPXXXX". This is convenient for recovering when files are lost or corrupted. It also is useful to create a listing of the current form of the programs, after the version has been updated many times.

5.3 SYSTEM UPDATES

After a version of MASTER is created, it is modified through system updates. This practice should be followed without exception, because it supports the reliable removal of the modifications. The desired changes to the UPDATE libraries are made, and then they are added to the intermediate and execution files.

The principal changes to the source code that can be made are the insertion of new lines and the deletion of existing ones. Deletions are made only by entering *DELETE directives to the UPDATE utility, while insertions can be made with the *INSERT and *DELETE directives. (See Reference 4.)

5.3.1 UPDATE IDENTIFIERS

Each update job adds a correction set to the UPDATE library for the affected part of MASTER. This correction set is given a unique identifier, based upon the date that coding of the update was begun. The identifier contains 8 characters. The first 2 characters are the year, the next 3 letters are the month, and the next 2 characters are the day of the month. The last character is a letter chosen to make the identifier unique. The identifier "82 JAN01A" follows this form. The identifier is input to the UPDATE utility with a *IDENT directive. (See Reference 4.)

5.3.2 UPDATE LOG

A log is kept of the system updates, on file UPDATES. This log lists each identifier in chronological order, with the affected part and the date when each version was changed. It is manually maintained by editing the file.

5.3.3 UPDATE TESTING

Updates can be in error either beacuse incorrect line numbers are specified or because incorrect lines are inserted. Should an error be discovered after the update was entered, it must be removed with the *PURGE directive before a corrected update with the same identifier can be entered. (See Reference 4.)

Update jobs should be tested before modifying the system. The REPLACE commands in the job deck can be deactivated by placing a star before the command. A job deck with all the REPLACE statements deactivated can check the job deck and the line numbers without affecting the system files. (A job deck with only the UPDATE-library replacement deactivated can modify an experimental MASTER version for testing purposes.)

5.3.4 UPDATE-JOB TEMPLATES

The system-update job decks have a regular patterns, which are expressed as the template files, JUPSUBS etc. Job decks can easily be edited from copies of the template corresponding to the part of the system being changed, using the global replacement feature of a text editor. The symbol "+" becomes the initial letter for the system version, the phrase "VERSION" becomes the name of the version (and the associated procedure file). The update identifier is formed from other symbols and phrases: "YEAR" becomes the 2-digit year, "MONTH" becomes the 3-letter month, "DAY" becomes the 2-digit day of the month, and "%" becomes the unique letter at the end of the identifier. The phrase "DESCRIBE CHANGES" becomes a short description of the update. In MAIN updates, "PROGRAM" becomes the main-program name. In SUBS or LIBS updates, "ROUTINE" becomes the subroutine name.

The input UPDATE directives are appended to the edited file, completing the record of data input to UPDATE.

5.3.4.1 Procedure Update

File JUPPROC contains the following text:

```
+MONTHDAY%,CM50000,T05,P02. VERSION YEAR MONTH DAY UPDATE %
USER,<userno>,<password>.<name>/<phone>/<mailstop>
CHARGE,<project>,LRC.
*  YEARMONTHDAY%  -  UPDATE  PROC
*
GET,OLDPL=+PLPROC.
UPDATE,N=+PLPROC.
CATALOG,+PLPROC,N,R.
*  REPLACE,+PLPROC.
CATALOG,COMPILE,N,R.
COPYSBF,COMPILE.
REWIND,COMPILE.
GET,OLD=+PROC.
CATALOG,OLD,N,R.
COPYL,OLD,COMPILE,+PROC,,RT.
CATALOG,+PROC,N,R.
*  REPLACE,+PROC.
*
REWIND,INPUT.
SKIPR,INPUT,1.
RETURN,COMPILE.
UPDATE,D,8. NO SEQUENCE NUMBERS FOR EXECUTION FILE
CATALOG,COMPILE,N,R.
COPYSBF,COMPILE.
REWIND,COMPILE.
GET,OLD=VERSION.
CATALOG,OLD,N,R.
COPYL,OLD,COMPILE,VERSION,,RT.
CATALOG,VERSION,N,R.
*  REPLACE,VERSION.

---  END  OF  RECORD  ---
```

```
*IDENT  YEARMONTHDAY%
*/  UPDATE  PROC:  (DESCRIBE CHANGES)
```

Job decks edited from this template do the following:

1. Modify the procedure source on the UPDATE library, writing the modified procedure to file COMPILE, with line numbers.
2. Replace the modified procedure on the listing file.
3. Repeat the UPDATE step, giving a version without line numbers.
4. Replace the modified procedure on the execution file.

5.3.4.2 Data Update

File JUPDATA contains the following:

```
+MONTHDAY%,CM50000,T05,P02.  VERSION YEAR MONTH DAY UPDATE  %
USER,<userno>,<password>.<name>/<phone>/<mailstop>
CHARGE,<project>,LRC.
*
*  YEARMONTHDAY%:  UPDATE  DATA
*
GET,OLDPL=+PLDATA.
CATALOG,OLDPL,N,R.
REWIND,INPUT.
SKIPR,INPUT,1.
UPDATE,F,C=+DATA,N=+PLDATA.
CATALOG,+PLDATA,N,R,U.
*  REPLACE,+PLDATA.  UPDATE  LIBRARY
CATALOG,+DATA,N,R.
*  REPLACE,+DATA.  LISTING WITH SEQUENCE NUMBERS
*
REWIND,INPUT.
SKIPR,INPUT,1.
UPDATE,D,8,C=DATA,N=0.  NO  SEQUENCE  NUMBERS
CATALOG,DATA,N,R.
(COPY RECORDS FROM DATA TO FILES, CATALOG, AND REPLACE)

---  END  OF  RECORD  ---

*IDENT  YEARMONTHDAY%
*/UPDATE  DATA:  (DESCRIBE CHANGES)
```

Job decks edited from this template do the following:

1. Modify the data source on the UPDATE library, writing the modified data to file COMPILE, with line numbers.
2. Replace the modified data on the listing file.
3. Repeat the UPDATE step, giving a version without line numbers, on file DATA.
4. Replace the data file which contains the modified data.

Note that the last line of the control-statement record must be replaced with the appropriate commands to copy the revised information from file DATA and to replace the system files. The contents of DATA will be divided by record marks.

5.3.4.3 Main-Program Update

File JUPMAIN contains the following:

```
+MONTHDAY%,CM70000,T10,P02. VERSION YEAR MONTH DAY UPDATE %
USER,<userno>,<password>.<name>/<phone>/<mailstop>
CHARGE,<project>,LRC.
*
*  YEARMONTHDAY% - UPDATE MAIN
*
GET,OLDPL=+PLMAIN.
CATALOG,OLDPL,N,R,U.
UPDATE,N=+PLMAIN.
CATALOG,+PLMAIN,N,R.
*  REPLACE,+PLMAIN.
REWIND,COMPILE.
FTN,I=COMPILE,OPT=0,R=3,ROUND,B=+PROGRAM.
CATALOG,+PROGRAM,N,R.
*  REPLACE,+PROGRAM.
GET,OLD=+BRMAIN.
CATALOG,OLD,N,R.
COPYL,OLD,+PROGRAM,+BRMAIN,,RAT.
CATALOG,+BRMAIN,N,R.
*  REPLACE,+BRMAIN.

---  END  OF  RECORD  ---

*IDENT  YEARMONTHDAY%
*/  UPDATE  MAIN:  PROGRAM  -  (DESCRIBE  CHANGES)
```

Job decks edited from this template do the following:

1. Modify the source code on the UPDATE library, writing the modified program to file COMPILE.
2. Compile the object code for the modified program.
3. Replace the execution file for the program.
4. Replace the copy of the modified program on the combined file.

5.3.4.4 Subroutine-Library Update

File JUPSUBS contains the following:

```
+MONTHDAY%,CM70000,T2,P02. VERSION YEAR MONTH DAY UPDATE %
USER,<userno>,<password>.<name>/<phone>/<mailstop>
CHARGE,<project>,LRC.
*
*  YEARMONTHDAY% - UPDATE SUBS
*
GET,OLDPL=+PLSUBS.
CATALOG,OLDPL,N,R,U.
UPDATE,N=+PLSUBS.
CATALOG,+PLSUBS,N,R.
*  REPLACE,+PLSUBS.
REWIND,COMPILE.
FTN,I=COMPILE,OPT=0,R=3,ROUND,B=LGO.
CATALOG,LGO,N,R.
GET,OLD=+BRSUBS.
CATALOG,OLD,N,R.
COPYL,OLD,LGO,+BRSUBS,,RAT.
```

```
CATALOG,+BRSUBS,N,R.
*  REPLACE,+BRSUBS.
GET,OLDBIN=+MASBIN.
CATALOG,OLDBIN,N,R.
REWIND,LGO.
COPYL,OLDBIN,LGO,+MASBIN,,RAT.
CATALOG,+MASBIN,N,R.
*  REPLACE,+MASBIN.
LIBGEN,F=+MASBIN,P=+MASLIB,N=MASTLIB.
CATALOG,+MASLIB,N,R,U.
*  REPLACE,+MASLIB.

--- END OF RECORD ---

*IDENT YEARMONTHDAY%
*/ UPDATE SUBS: ROUTINE - (DESCRIBE CHANGES)
```

Job decks edited from this template do the following:

1. Modify the source code on the UPDATE library, writing the modified subroutine to file COMPILE.
2. Compile the object code for the modified subroutine.
3. Replace the copy of the modified subroutine, on file +BRSUBS.
4. Replace the copy of the modified subroutine, on file +MASBIN.
5. Regenerate and replace the subroutine library, on file +MASLIB.

File JUPLIBS contains a similar template, except that "LIBS" appears in the place of "SUBS" everywhere.

5.4 ERROR LOGS

An error package is appended to the error-log file whenever a MASTER procedure aborts. These packages contain considerable information to analyze the errors, including the data which was input.

Error packages are used to automatically capture data on errors. The error log should be cleared periodically of old packages, to reduce file-storage charges.

6.0 PROCEDURES

This section describes the control-language procedures which execute the MASTER programs. First the features which are shared by all procedures are explained, then the details of specific procedures are shown.

6.1 GENERAL FEATURES

This section explains the general features which all the procedures share.

6.1.1 CCL PROCEDURE ENTRY AND EXIT

Cyber Control Language (CCL) procedures are used. (See Reference 2.) They are called by a BEGIN statement and exited with a REVERT statement. The procedure file consists of separate records for each procedure. The first line of each procedure is a PROC statement. The PROC statement defines symbolic names within the procedure. The BEGIN statement has fields corresponding to each symbolic name. The fields which contain arguments cause the arguments to be substituted for the corresponding symbolic names those symbolic names that correspond to empty fields are used as the default arguments.

6.1.2 CCL REGISTERS

CCL supports a set of registers that are local to each procedure; a pair of these registers are used. Register R1 contains the field length before the procedure was entered. This field length is restored when the procedure is left. Register R2 is used as an error indicator; it has a nonzero value only when an error has been detected.

6.1.3 CCL DECISIONS

CCL supports decisions through IFE statements. Such decisions are controlled by conditions, which are evaluated during execution. The condition OT=TXO is used to indicate that the current job has an interactive, rather than batch, origin. The condition FILE(<filename>,AS) is used to indicate that a file exists with the name specified. The condition FILE(<filename>,TT) is used to indicate that the named file is associated with an interactive terminal. (This test is used to avoid attempts to backspace such files.)

A similar type of decision is the trapping of an error exit. This uses an EXIT statement, which is prevented from affecting normal execution by a SKIP statement.

The range of an IFE statement is marked by an ENDIF statement, and possibly by an ELSE statement, with the same identifier as the IFE statement. The range of a SKIP statement is marked with an ENDIF statement with the same identifier as the SKIP statement. The following identifiers are used:

1. The test for existance of a file uses that file's name for an identifier.
2. Identifiers containing a file name followed by "TT" are used to test that file for being an interactive terminal.
3. "UNKNOWN" is used to trap errors in the filename tests. (No such errors are expected, but they could happen if the condition symbols are modified.)
4. "PROCEED" is used to proceed after successful file name tests.
5. "ACCESS" is used to trap errors made attempting to access MASTER system files.
6. "REQUEST" is used to proceed after system-file accesses. (The next step is to request the desired field length.)
7. "FIELD" is used to trap errors made requesting field length. (A MFL extension is attempted before aborting the procedure.
8. "MFL" is used to trap errors made requesting a MFL extension.
9. "EXECUTE" is used to proceed to program execution after a successful field-length request.
10. "PROGERR" is used to trap program-execution errors.
11. "ERRPACK" is used to decide whether to assemble an error package or to assemble the usual listing output.
12. "LISTING" is used to decide which file name (OUT or OUTPUT) to use for the listing file.

The basic structure of these decisions is the following:

```
        Check Data Files (and Possibly Set Error Flag)

        Skip  UNKNOWN
        .       EXIT.
        .       Set  Error  Flag
        Endif  UNKNOWN

        If  (No  Error)  PROCEED
        .
        .       Access  System  Files
        .
        .       Skip  ACCESS
        .       .       EXIT.
        .       .       Set  Error  Flag
        .       Endif  ACCESS
        .
        .       If  (No  Error)  REQUEST
        .       .
        .       .       Request  Field  Length
        .       .
        .       .       Skip  FIELD
        .       .       .
        .       .       .       EXIT.
        .       .       .       Request  MFL  Extension
        .       .       .       Request  Field  Length
        .       .       .
        .       .       .       Skip  MFL
        .       .       .       .       EXIT.
        .       .       .       .       Abort  Procedure,
<-------+-------+-------+-------+-------and  Revert  to  Calling  Job
        .       .       .       Endif  MFL
        .       .       .
        .       .       Endif  FIELD
        .       .
        .       .       If  (No  Error)  EXECUTE
        .       .       .
        .       .       .       Execute  Program
        .       .       .
        .       .       .       Skip  PROGERR
        .       .       .       .       EXIT.
        .       .       .       .       Set  Error  Flag
        .       .       .       Endif  PROGERR
        .       .       .
        .       .       Endif  EXECUTE
        .       .
        .       Endif  REQUEST
        .
        Endif  PROCEED
```

```
        If (Error) ERRPACK
        .
        .       Prepare Error Package
        .
        .       If (Interactive Job) LISTING
        .       .
        .       .       Copy Error Package to OUT
        .       .
        .       Else LISTING
        .       .
        .       .       Copy Error Package to OUTPUT
        .       .
        .       Endif LISTING
        .
<-------+-------Abort Procedure, and Revert to Calling Job
        .
        Else ERRPACK
        .
        .       Prepare Listing File
        .
        .       If (Interactive Job) LISTING
        .       .
        .       .       Copy Listing File to OUT
        .       .
        .       Else LISTING
        .       .       .       .       Copy Listing File to OUTPUT
        .       .
        .       Endif LISTING
        .
<-------+-------Revert to Calling Job
        .
        Endif ERRPACK
```

6.1.4 DATA FILES

The symbolic names of a procedure appear in place of the names for the user's input and output data files. If no file name appears in the corresponding field of the BEGIN statement, the default name is used; otherwise the user-specified file name is substituted. (Throughout this manual, these data files are referred to by their default names.)

The data formats named in this manual are defined in Section 3 of the MASTER user's manual.

If input data can appear on a file, it is rewound before program execution, otherwise the file is returned before execution. All data files are rewound after execution.

6.1.5 LISTING FILE

All listing information is written to a special file, called the listing file. This file is never rewound and assumed to always be positioned at the end of information, which is compatible with the batch-job handling of file OUTPUT. For batch jobs, this file is OUTPUT, so the listing is printed. For interactive jobs,the listing file is OUT; this prevents the terminal from being swamped with data. The interactive user must then dispose OUT to get a printed copy. The listing from some procedures also contains a listing of the output data. The listing file is packed into a single record. The procedures for interactive programs write the interactive prompts to OUTPUT, rather than the listing file. This allows the user to respond to the prompts.

6.1.6 TEMPORARY FILES

When a procedure uses files other than the user's input and output files, they are hidden from the user. Such files are returned before and after execution. They are given names which start with "ZZZ". (This is an extension of the NOS convention of using names starting with "ZZZZZ" for temporary files; see Reference 2.) By avoiding such names, users can avoid interference from the procecure with their other local files.

All the program's files are explictly named in the procedure. This changes the temporary files from the natural names which appear on the PROGRAM statement to the procedure's names.

6.1.7 ERROR PACKAGES

Error packages are intended to include all the available information about an aborted procedure, placing the most useful information first. The listing for an error package starts with the program-execution output. This is followed by the results of an ENQUIRE command. (This command lists local file names and another general information about job status.) Next comes a pair of DAYFILE versions. The "OP=M" DAYFILE, which comes first, is limited to the procedure. (The IFE-ELSE-ENDIF decision with identifier "DAYFILE" is used to recognized the last BEGIN statement, which appears differently in dayfiles from batch and interactive jobs.) The following DAYFILE covers the entire job up to this time. Now the data files are cataloged and listed. This is followed by the load map. Finally all the binary files, including program and any subroutine libraries, are cataloged. The listing for the error package is packed into a single record.

The listing for the error package is copied to the user's listing file. Then the procedure record and the data which generated the error are added, as separate records. This expanded package is appended to the error log, on the system account. This process automatically enables the programmer to duplicate the user's problem.

At the beginning of each procedure, files ZZZPROC and ZZZPAGE are copied from the system account. ZZZPAGE is simply a page header, to start components of an error package on new pages. ZZZPROC is a copy of the procedure file which is being called, to extract the procedure record for an error package.

6.2 PROCEDURE DRAWIT

DRAWIT is used to display geometric data graphically. It can show surface models, curve models, and intersection normals. It uses a Tektronix 4010-series interactive graphics terminal, with a storage screen.

6.2.1 DATA FILES

DRAWIT uses files INPUT and OUTPUT to interact with the terminal and to display views of the geometric data. The format of the data read from INPUT is text. The format of the data written to OUTPUT is dependent on the Tektronix graphics implementation. These files do not appear in the procedure parameter list.

The BEGIN statement for DRAWIT has the following list of data files: SRF, CUR, NRM, and OUT. Surface-model data to be viewed is input on file SRF. Curve-model data to be viewed is input on file CUR. Intersection-normal data to be viewed is input on file NRM. OUT is the listing file for interactive executions. (This file is only used when an error package is prepared.)

Program DRAWIT is executed with the following list of input files: INPUT, OUTPUT, SRF, CUR, NRM, ZZZSRF, ZZZCUR, ZZZNRM, and ZZZREAD. ZZZSRF, ZZZCUR, and ZZZNRM are used within the program for random-access data storage. (ZZZREAD is not used.)

6.2.2 RESTRICTIONS

Procedure DRAWIT must be executed from an interactive job and from a Tektronix 4010-series storage-tube graphics terminal. It requires a field length of 100000 (octal).

6.2.3 SCRATCH FILES

File ZZZPROG is the program, ZZZMAST is the MASTER library, and ZZZP10 is the PLOT-10 graphics library. Files ZZZPAT, ZZZCUR, and ZZZNRM are random-access files which store data for use within the program; ZZZDAT is a file used to read data into the program and to hold data for appending to an error log. ZZZPAGE is a page header for an error package; ZZZLOAD is a load map; ZZZLIST holds listing for an error package. ZZZPROC is a copy of the procedure file, for appending DRAWIT procedure to an error log.

6.3 PROCEDURE GENTRN

GENTRN is used to form elementary rotation matrices, to multiply them, and to combine them with translation vectors in TRN format.

6.3.1 DATA FILES

The BEGIN statement for GENTRN has the following list of data files: INPUT, TRN, and OUT. GENTRN uses files INPUT and OUTPUT to interact with the terminal. Text messages requesting input are written to OUTPUT, then numeric data (with comment lines possible) are read from INPUT. File TRN receives the output data, in TRN format. File OUT receives the listing output from interactive executions.

Program GENTRN is executed from interactive jobs with the following list of files: INPUT, OUTPUT, TRN, ZZZTMP, and ZZZLIN. It is executed from batch jobs with the following list of files: INPUT, ZZZLIST, ZZZTRN, TRN, ZZZTMP, and ZZZLIN; in this case, ZZZLIST holds the program output, which is copied to the listing file. In both cases, a catalog of the TRN output and a listing of it are copied to the listing file after execution. ZZZTMP and ZZZLIN are temporary files which are only used within the program.

6.3.2 RESTRICTIONS

GENTRN requires a field length of 50000 (octal).

6.3.3 SCRATCH FILES

File ZZZPROG is the program, and ZZZMAST is the MASTER library. ZZZLIN holds the latest line of input data, and ZZZTMP holds the transformations to be output. ZZZLIST holds the output from a batch execution, and possibly an error package; ZZZPAGE is a page header for an error package; ZZZLOAD is a load map. ZZZDAT holds data for appending to the error log; ZZZPROC is the procedure file, for appending the GENTRN procedure to an error log.

6.4 PROCEDURE MSHNRM

MSHNRM is used to intersect a coordinate mesh with a surface model.

6.4.1 DATA FILES

The BEGIN statement for MSHNRM has the following list of data files: MSH, SRF, NRM, CUR, and OUT. Mesh data must be input from file MSH, in MSH format. Surface-model data must be input from file SRF, in SRF format. Intersection-normal data is output to file NRM, in NRM format. (Intersection curves, an intermediate result, are output to file CUR, in CUR format.) The listing file for interactive executions is file OUT.

Program MSHNRM is executed with the following list of files specified: MSH, SRF, CUR, NRM, ZZZLIST, ZZZPSMS, ZZZPCMS, ZZZRSMS, and ZZZRCMS. ZZZLIST holds the program output, which is copied to the listing file. The last 4 files are temporary files, which are only used within the program.

6.4.2 RESTRICTIONS

MSHNRM requires a field length of 160000 (octal). Files MSH and SRF must exist as local disk files.

6.4.3 SCRATCH FILES

File ZZZPROG is the program, and ZZZMAST is the MASTER library. ZZZPSMS stores the patches for use within the program ZZZPCMS holds intersection curves within the program; ZZZRSMS holds radius-interpolating surface data from rectangular-coordinate surface models ZZZRCMS holds radius-interpolating intersection-curve data from rectangular-coordinate surface models. ZZZLIST holds the listing output, and possibly an error package; ZZZPAGE is a page header for an error package ZZZLOAD is a load map. ZZZDAT holds data for appending to an error log ZZZPROC is the procedure file, for appending the MSHNRM procedure to an error log.

6.5 PROCEDURE NRMCFD

NRMCFD is used to combine the complete set of normals and the complete coordinate mesh for a configuration, giving a CFD analysis input file.

6.5.1 DATA FILES

The BEGIN statement for NRMCFD has the following list of data files: MSH, NRM, OLDCFD, NEWCFD, and OUT. Mesh data must be input from file MSH, in MSH format. Intersection normal data must be input from file NRM, in NRM format. Header data can be input from file CFD, in text format; such data is optional. A complete file of input for CFD analysis is output to file NEWCFD, in CFD format. The listing file for interactive executions is output to file OUT.

Program NRMCFD is executed with the following list of files specified: MSH, NRM, OLDCFD, NEWCFD, and ZZZLIST. ZZZLIST holds the program output, which is copied to the listing file.

6.5.2 RESTRICTIONS

NRMCFD requires a field length of 160000. Files MSH and NRM must exist as local disk files.

6.5.3 SCRATCH FILES

File ZZZPROG is the program, and ZZZMAST is the MASTER library. ZZZLIST holds listing output, possibly an error package; ZZZPAGE is a page header for an error package; ZZZLOAD is a load map. ZZZDAT holds data for appending to an error log; ZZZPROC is the procedure file, for appending procedure NRMCFD to an error log.

6.6 PROCEDURE NRMREV

NRMREV is used to reverse the direction of all the normals in a file.

6.6.1 DATA FILES

The BEGIN statement for NRMREV has the following list of data files: OLDNRM and NEWNRM. (There is no listing file.) Intersection normals must be input on file OLDNRM, in NRM format. Reversed normals are output on file NEWNRM, also in NRM format.

Program NRMREV is executed with the following list of files: OLDNRM, NEWNRM, and ZZZLIST. ZZZLIST holds the program output, which is copied to the listing file.

6.6.2 RESTRICTIONS

NRMREV requires a field length of 20000 (octal). File OLDNRM must exist as a local disk file.

6.6.3 SCRATCH FILES

File ZZZPROG is the program, and ZZZMAST is the MASTER library. ZZLIST holds listing data, possibly an error package ZZZPAGE is a page header for an error package; ZZZLOAD is a load map. ZZZDAT holds data for appending to an error log; ZZZPROC is the procedure file, for appending procedure NRMREV to an error log.

6.7 PROCEDURE REGSIL

REGSIL regulates the point spacing within input curves. It reads a simplified SIL section specification set, and it writes a complete SIL file. REGSIL is very expensive to use, so SELECT should be used instead. (See Appendix E of Reference 7.)

6.7.1 DATA FILES

The BEGIN statement for REGSIL has the following list of file names: OPTION, OLDSIL, NEWSIL. (Interactive execution is forbidden, so no listing file for interactive execution is needed.) Program-control selections must be input from file OPTION, in the format described in Section 7.4.1.5 of the MASTER user's manual. Simplified SIL sections must be input on file OLDSIL, in the format described in Section 7.3.2 of the user's manual. Surface-description data is output to file NEWSIL, in SIL format; the form of this surface description is explained in Section 7.3.3 of the user's manual.

Program REGSIL is executed with the following list of files specified: OLDSIL, OPTION, NEWSIL, and ZZZLIST. ZZZLIST holds the program output, which is copied to the listing file. The NEWSIL output is cataloged and listed on the listing file.

6.7.2 RESTRICTIONS

REGSIL must be executed from a batch job. It requires a field length of 160000 (octal). File OLDSIL must exist as a local disk file.

6.7.3 SCRATCH FILES

File ZZZPROG is the program, and ZZZMAST is the MASTER library. ZZZLIST holds listing output, possibly an error package; ZZZPAGE is a page header for an error package; ZZZLOAD is a load map. ZZZDAT holds data for appending to an error log; ZZZPROC is the procedure file, for appending procedure REGSIL to an error log.

6.8 PROCEDURE SILSRF

SILSRF is used to model surfaces.

6.8.1 DATA FILES

The BEGIN statement for SILSRF has the following list of data files: SIL, SRF, and OUT. Surface-description data must be input from file SIL, in SIL format. The resulting surface model is output to file SRF, in SRF format. The listing file from interactive execution is output to file OUT.

Program SILSRF is executed with the following list of files: SIL, ZZZNEWS, SRF, and ZZZLIST. File ZZZNEWS is a rewritten version of the data from SIL after execution, it replaces the original form of this data on file SIL. ZZZLIST is the program output, which is copied to the listing file. The reformatted surface description is cataloged and listed on the listing file.

6.8.2 RESTRICTIONS

SILSRF required a field length of 160000 (octal). File SIL must exist as a local disk file.

6.8.3 SCRATCH FILES

File ZZZPROG is the program, and ZZZMAST is the MASTER library. ZZZSDMS is the SDMS data-manager library; ZZZSIL is the SDMS data definition; ZZZSIL1, ZZZSIL2, ZZZSIL3, and ZZZSIL4 hold the SDMS database used within the program. ZZZNEWS is the reformatted version of the SIL input data. ZZZLIST holds printed output, possibly an error package; ZZZPAGE is a page header for an error package; ZZZLOAD is a load map. ZZZDAT holds data for appending to an error log; ZZZPROC is the procedure file, for appending procedure SILSRF to an error log.

The database files are temporary direct access files on the user's account. File ZZZSIL is stored on the user's account as a direct access permanent file. If this file is not found from a previous execution, it is produced from the source file SILDBD by DDP. (This feature deals with the frequent archiving of small direct-access files on the ACD system.)

6.9 PROCEDURE SRFINT

SRFINT computes the intersection of two surface models, giving SIL section specifications.

6.9.1 DATA FILES

The BEGIN statement for SRFINT has the following list of data files: SRF1, SRF2, SEC, OPTION, and OUT. The pair of surface models to be intersected must be input to files SRF1 and SRF2, in SRF format. Non-default program control options can be input from file OPTION, in the format described in Section 7.4.2.5 of the MASTER user's manual. The curves of intersection are output to file SEC, in the modified SIL format described in Section 7.3.1 of the user's manual. The listing file for interactive execution is output to file OUT.

Program SRFINT is executed with the following list of files: SEC, OPTION, ZZZLIST, SRF1, SRF2, ZZZSRF1, and ZZZSRF2. ZZZLIST is the program output, which is copied to the listing file. The SEC output is cataloged and listed on the listing file. The last pair of files are temporary; they are only used within the program.

6.9.2 RESTRICTIONS

SRFINT requires a field length of 160000. Files SRF1 and SRF2 must exist as local disk files.

6.9.3 SCRATCH FILES

File ZZZPROG is the program, and ZZZMAST is the MASTER library. ZZZSDMS is the SDMS library; ZZZIBR is the SDMS data definition; ZZZIBR1, ZZZIBR2, ZZZIBR3, and ZZZIBR4 hold the SDMS database for use within the program. ZZZSRF1 and ZZZSRF2 hold the surface data for use within the program. ZZZLIST holds the listing output, possibly an error package; ZZZPAGE is a page header for the error package; ZZZLOAD is a load map. ZZZDAT holds data for appending to an error log; ZZZPROC is the procedure file, for appending procedure SRFINT to an error log.

The database files are temporary direct access files on the user's account. File ZZZIBR is stored on the user's account as a direct access permanent file. If this file is not found from a previous execution, it is produced from the source file IBRDBD by DDP. (This feature deals with the frequent archiving of small direct-access files on the ACD system.)

6.10 PROCEDURE TRNSIL

TRNSIL is used either to convert a surface description from cylindrical coordinates to rectangular coordinates or to rotate and translate a surface description in rectangular coordinates as if it were a rigid object or to convert a surface description from rectangular coordinates to cylindrical coordinates.

6.10.1 DATA FILES

The BEGIN statement for TRNSIL has the following list of file names: TRN, OLDSIL, NEWSIL, INPUT, and OUT. Surface description data must be input to file OLDSIL, in SIL format. Those executions which rotate and translate rectangular-coordinate data must have a list of coordinate transformations input from file TRN, in TRN format. The choice of a transformation is read from file INPUT. The transformed surface-description data is output to file NEWSIL, in SIL format. The listing file from interactive execution is output to file OUT.

Program TRNSIL is executed from interactive jobs with the following list of files: INPUT, OUTPUT, OLDSIL, NEWSIL, and TRN. It is executed from batch jobs with the following list of files: INPUT, ZZZLIST, OLDSIL, NEWSIL, and TRN; ZZZLIST is the program output, which is copied to the listing file. In both cases, the NEWSIL output is cataloged and listed to the listting file.

6.10.2 RESTRICTIONS

TRNSIL requires a field length of 40000 (octal). File OLDSIL must exist as a local disk file; file TRN can only be a local disk file.

6.10.3 SCRATCH FILES

Files ZZZPROG is the program, and ZZZMAST is the MASTER library. ZZZLIST holds listing output, possibly an error package; ZZZPAGE is a page header for an error package; ZZZLOAD is a load map. ZZZDAT holds data for appending to an error log; ZZZPROC is the procedure file, for appending procedure TRNSIL to an error log.

7.0 MAIN PROGRAMS

This section describes the MASTER code contained in the MAIN part. For most programs, this is simply the main-program module. Programs DRAWIT and TRNSIL have the associated subroutines kept within this part, rather than using the MASTER system library.

7.1 DRAWIT

DRAWIT is a program which displays 3-D views of MASTER geometric data, using interactive graphics. It uses the PLOT-10 graphics library to control a Tektronix 4010-series storage-screen terminal. Surface and curve models can be displayed, as can intersection normals. Individual data elements can be selected for display, and the elements can be labelled.

7.1.1 FILES

The file name list for DRAWIT is the following: INPUT, OUTPUT, SRF, CUR, NRM, TAPE1, TAPE2, and TAPE3. INPUT and OUTPUT are connected to the graphics terminal. INPUT reads text, mostly menu selections. OUTPUT writes both menu text and graphic-display data. SRF is the input file for surface patches. CUR is the input file for PC-curve segments. NRM is the input file for intersection normals. Files TAPE1, TAPE2, and TAPE3 are used for random-access storage of the input data. (This uses READMS and WRITMS.) TAPE1 stores patch data, TAPE2 stores PC-curve data, and TAPE3 stores intersection-normal data.

Within the program, INPUT and OUTPUT are respectively referred to as units 5 and 6.

7.1.2 ROUTINES CALLED

The main program for DRAWIT calls the following subroutines: CLOSMS, DRWPIC, FINITT, INITT, MENU, MOVABS, ORIENT, TINPUT, and TSEND. Subroutine CLOSMS is from the Fortran library; it is documented in the Fortran manual. Subroutines FINITT, INITT, MOVAABS, TINPUT, and TSEND are from the PLOT-10 graphics library; they are documented in the PLOT-10 manual. Subroutines DRWPIC (see Section 7.1.8 below), MENU (7.1.15), and ORIENT (7.1.46) are from the MAIN group of MASTER; they are found in Update deck DRAWIT.

7.1.3 METHOD

DRAWIT has two main branches: menu interaction and graphic display. The menu-interaction branch copies data from input files to random-access files and selects which data elements are to be displayed. The graphic-display branch reads the selected data elements from random-access storage and draws a 3-D view of them. A grid of constant-parameter lines is displayed for each patch.

7.1.4 MAIN-PROGRAM ROUTINE

The PLOT-10 graphics interface is initialized. A cycle is repeated until the menu-interaction branch returns a selection to stop the program. This cycle contains the following steps: Subroutine MENU is called, giving the user a series of menus through which to interact with the program. Unless and exit is requested, subroutine DRAWPIC draws a 3-D display. Subroutine ORIENT draws the 3 axes. PLOT-10 subroutine TINPUT then waits for a carriage return before the cycle is repeated. Finally, the graphics interface and the random-access data storage are closed.

The following common blocks are declared:

```
COMMON  /BLK1/  IDUM1(3)
COMMON  /BLK2/  IDUM2(3)
COMMON  /BLK3/  IDUM3(3)
COMMON  /BLK4/  IDUM4
```

COMMON /QPATCH/ IDUM12(507)
COMMON /QPC/ IDUM7(1006)
COMMON /QVEC/ IDUM8(1005)
COMMON /SCRAT/ IDUM9(6000)
COMMON /STATUS/ IDUM10(3)

7.1.5 SUBROUTINE ADD

Adds a single data element to the list of selected elements

7.1.5.1 Parameter List

1. K [In/Out, INTEGER(100)]
   The list of elements currently selected: This is a list of indices, sorted in ascending order.
2. N [In/Out, INTEGER scalar]
   The number of elements currently selected
3. IPICK [In, INTEGER scalar]
   The index for the element to add

7.1.5.2 Common Data

none

7.1.6 SUBROUTINE DRWPAT

Draws a set of patches

7.1.6.1 Parameter List

1. LUNPAT [In, INTEGER scalar]
   Logical unit for patch random-access storage
2. NPAT [In, INTEGER scalar]
   Number of patches to be drawn
3. IPAT [In, INTEGER(500)]
   List of patches to be drawn: This is a list of indices, sorted in ascending order.
4. NUPAT [In, INTEGER scalar]
   Number of constant-U lines to be drawn for each patch: Possible values are 2 to 10.
5. NVPAT [In, INTEGER scalar]
   Number of constant-V lines to be drawn for each patch: Possible values are 2 to 10.
6. LABPAT [In, INTEGER scalar]
   Flag to draw patch labels: If 1, labels are drawn. Otherwise, they are not drawn.
7. ICYL [In, INTEGER, scalar]
   Flag for cylindrical coordinates: If 1, cylindrical coordinates are used. Otherwise, rectangular coordinates are used.
8. SCRMAX [In, REAL, scalar]
   Maximum screen size

7.1.6.2 Common Data

COMMON /SCRAT/ X(2000), Y(2000), Z(2000)
COMMON /INDEXX/ ICMPAT(501), ICMPC(1001), ICMVEC(1001), CMOPEN(3)

Common block /SCRAT/ holds the coordinates to be plotted.

7.1.7 SUBROUTINE DRWPC

Draws a set of PC curves

7.1.7.1 Parameter List

1. LUNPC [In, INTEGER scalar]
   Logical unit for PC-curve random-access storage
2. NPC [In, INTEGER scalar]
   Number of PC curves to draw
3. IPC [In, INTEGER(1000)]
   List of PC curves to draw: This is a list of indices, sorted in ascending order.
4. NUPC [In, INTEGER scalar]
   Number of points to draw along each PC curve: Possible values are 2 to 10.
5. LABPC [In, INTEGER scalar]
   Flag to draw curve labels: If 1, labels are drawn. Otherwise, they are not drawn.
6. ICYL [In, INTEGER scalar]
   Flag for cylindrical coordinates: If 1, cylindrical coordinates are used. Otherwise, rectangular coordinates are used.
7. IADD [In, INTEGER scalar]
   Flag to add to an existing picture: If 1, the patches are added to an existing display. Otherwise, the screen is cleared and screen limits are reset before drawing curves.
8. SCRMAX [In, REAL scalar]
   Maximum screen size

7.1.7.2 Common Data

```
COMMON /SCRAT/ X(2000), Y(2000), Z(2000)
COMMON /INDEXX/ ICMPAT(501), ICMPC(1001), ICMVEC(1001)
```

Common block /SCRAT/ holds the coordinates to be plotted.

7.1.8 SUBROUTINE DRWPIC

Draws a display containing patches, PC curves, and/or normals

7.1.8.1 Parameter List

1. ICYL [In, INTEGER, scalar]
   Flag for cylindrical coordinates: If 1, cylindrical coordinates are used. Otherwise, rectangular coordinates are used.

7.1.8.2 Common Data

```
COMMON /QPATCH/ IDRPAT, NPATT, LUNPAT, NUPAT, NVPAT, LBLPAT, NPAT,
  IPAT(500)
COMMON /QPC/ IDRPC, NPCT, LUNPC, NUPC, LABPC, NPC, IPC(1000)
COMMON /QVEC/ IDRVEC, NVECTT, LUNVEC, LABVEC, NVEC, IVEC(1000)
COMMON /SCRAT/ DUMMY(6000)
```

7.1.9 SUBROUTINE DRWVEC

Draws a set of intersection normals, represented by vectors

7.1.9.1 Parameter List

1. LUNVEC [In, INTEGER scalar]
   Logical unit for vector random-access storage
2. NVEC [In, INTEGER scalar]
   Number of vectors to draw

3. IVEC [In, INTEGER(1000)]
   List of vectors to draw: This is a list of indices, sorted in ascending order.
4. LABVEC [In, INTEGER scalar]
   Flag to draw vector labels: If 1, labels are drawn. Otherwise, labels are not drawn.
5. ICYL [In, INTEGER scalar]
   Flag for cylindrical coordinates: If 1, cylindrical coordinates are used. Otherwise, rectangular coordinates are used.
6. IADD [In, INTEGER scalar]
   Flag to add to an existing picture: If 1, the vectors are added to an existing display. Otherwise, the screen is cleared and screen limits are reset before drawing vectors.
7. SCRMAX [In, REAL scalar]
   Maximum screen size

7.1.9.2 Common Data

COMMON /SCRAT/ X1(1000), Y1(1000), Z1(1000), X2(1000), Y2(1000), Z2(1000)
COMMON /INDEXX/ ICMPAT(501), ICMPC(1001), ICMVEC(1001)

Common block /SCRAT/ holds the coordinates to be plotted.

7.1.10 SUBROUTINE LBLPAT

Draws labels on a patch display

7.1.10.1 Parameter List

1. LUNPAT [In, INTEGER scalar]
   Logical unit for patch random-access storage
2. NPAT [In, INTEGER scalar]
   Number of patches to be drawn
3. IPAT [In, INTEGER(500)]
   List of patches to be drawn: This is a list of indices, sorted in ascending order.
4. ICYL [In, INTEGER, scalar]
   Flag for cylindrical coordinates: If 1, cylindrical coordinates are used. Otherwise, rectangular coordinates are used.

7.1.10.2 Common Data

COMMON /INDEXX/ ICMPAT(501), ICMPC(1001), ICMVEC(1001)

7.1.11 SUBROUTINE LBLPC

Draws labels on a PC curve display

7.1.11.1 Parameter List

1. LUNPC [In, INTEGER scalar]
   Logical unit for PC curve random-access storage
2. NPC [In, INTEGER scalar]
   Number of curves to be drawn
3. IPC [In, INTEGER(500)]
   List of curves to be drawn: This is a list of indices, sorted in ascending order.
4. ICYL [In, INTEGER, scalar]
   Flag for cylindrical coordinates: If 1, cylindrical coordinates are used. Otherwise, rectangular coordinates are used.

7.1.11.2 Common Data

COMMON /INDEXX/ ICMPAT(501), ICMPC(1001), ICMVEC(1001)

7.1.12 SUBROUTINE LIM3D

Determines the window bounding the display and determines the maximum screen size

7.1.12.1 Parameter List

1. X [In, REAL(NPTS)]
   X coordinates of the points to be plotted
2. Y [In, REAL(NPTS)]
   Y coordinates of the points to be plotted
3. Z [In, REAL(NPTS)]
   Z coordinates of the points to be plotted
4. NPTS [In, INTEGER scalar]
   Number of points to be plotted
5. SCRMAX [Out, REAL scalar]
   Maximum screen size: This is the diagonal length for a box bounding the points to be plotted.

7.1.12.2 Common Data

COMMON /WINDOW/ XMIN, XMAX, YMIN, YMAX, ZMIN, ZMAX, FACTOR

The bounding box definition and the screen size are stored in common Block /WINDOW/, for use by subroutines LABL3D, GCUR3D, GVIEW, and ORIENT.

7.1.13 SUBROUTINE LOADTX

Converts a ranges of indices to an array of text: This is used for displaying selection menus.

7.1.13.1 Parameter List

1. I1 [In, INTEGER scalar]
   Minimum index in range
2. I2 [In, INTEGER scalar]
   Maximum index in range

7.1.13.2 Common Data

COMMON /SCRAT/ TEXT(100)

Common block /SCRAT/ holds the text array.

7.1.14 SUBROUTINE MARKTX

Marks each selected element in a range within the text array of indices: This is used for displaying selection menus.

7.1.14.1 Parameter List

1. I1 [In, INTEGER scalar]
   Minimum subscript in range
2. I2 [In, INTEGER scalar]
   Maximum subscript in range

3. K [In, INTEGER(100)]
   List of elements selected
4. NK [In, INTEGER scalar]
   Number of elements selected: This is a list of indices.

7.1.14.2 Common Data

COMMON /SCRAT/ TEXT(100)

Common block /SCRAT/ contains the text array, which came from subroutine LOADTX.

7.1.15 SUBROUTINE MENU

Displays the main menu and handles the user's selection

7.1.15.1 Parameter List

1. ICYL [Out, INTEGER scalar]
   Flag for cylindrical coordinates: If 1, cylindrical coordinates are used. Otherwise, rectangular coordinates are used.
2. IEXIT [Out, INTEGER scalar]
   Flag to stop execution: If 1, the main program will stop.

7.1.15.2 Common Data

COMMON /QPATCH/ IDRPAT, NPATT, LUNPAT, NUPAT, NVPAT, LABPAT, NPAT, IPAT(500)
COMMON /QPC/ IDRPC, NPCT, LUMPC, NUPC, LABPC, NPC, IPC(1000)
COMMON /QVEC/ IDRVEC, NVECTT, LUNVEC, LABVEC, NVEC, IVEC(1000)

Common block /QPATCH/ contains the following: IDRPAT, the flag to display patches; NPATT, the number of patches read; LUNPAT, the logical unit for patch input, which is initialized to 1; NUPAT and NVPAT, the number of grid lines to display for a patch, which are each initialized to 5; LABPAT, the flag to display patches with labels; NPAT, the number of patches selected; and IPAT, the array of selected patches.

Common block /QPC/ contains the following: IDRPC, the flag to display PC curves; NPCT, the number of PC curves read; LUNPC, the logical unit for PC curve input, which is initialized to 2; NUPC, the number of points to display for a PC curve, which is initialized to 5; LABPC, the flag to display PC curves with labels; NPC, the number of PC curves selected; and IPC, the array of selected PC curves.

Common block /QVEC/ contains the following: IDRVEC, the flag to display vectors; NVECTT, the number of vectors read; LUNVEC, the logical unit for vector input, which is initialized to 3; LABVEC, the flag to display vectors with labels; NVEC, the number of vectors selected; and IVEC, the array of selected vectors.

7.1.16 SUBROUTINE OPTMEN

Displays option menu and handles user's response (i.e., a choice of coordinates)

7.1.16.1 Parameter List

1. ICYL [Out, INTEGER scalar]
   Flag for cylindrical coordinates: If 1, cylindrical coordinates are used. Otherwise, rectangular coordinates are used.

7.1.16.2 Common Data

none

7.1.17 SUBROUTINE PATIN

Reads an input file of patches

7.1.17.1 Parameter List

empty

7.1.17.2 Common Data

```
COMMON  /BLK1/  IDUM1,I1,I2
COMMON  /QPATCH/  IDUM2(1),  NPATT,  LUNPAT,  IDUM3(3),  NPAT,  IPAT(500)
COMMON  /SCRAT/  DUMMY(250),  PAT(48)
COMMON  /STATUS/  ISTAT1
COMMON  /INDEXX/  ICMPAT(501),  ICMPC(1001),  ICMVEC(1001),  CMOPEN(3)
```

Common block /INDEXX/ holds CMOPEN(1), which indicates whether patch data is available in random-access storage; it is set when new patches are read. This block also holds ICMPAT, the key array for patch random-access storage.

Common block /QPATCH/ holds the following: IPAT, the array of selected patches, which is emptied when new patches are read; LUNPAT, the logical unit for patch input; NPATT, the number of patches read, which is set; and NPAT, the number of patches selected, which is set to 0.

Common block /BLK1/ contains I1 and I2, the range of patch indices to display, which are both set to zero.

Common block /STATUS/ holds ISTAT1, which indictates that some patches were read.

7.1.18 SUBROUTINE READPT

Reads a single patch

7.1.18.1 Parameter List

1. PAT [Out, REAL(48)]
   Geometric patch representation: Effectively allocated PAT(4,4,3). PAT(*,*,K) is the K-th component of a vector. PAT(1,1,*) is the (U,V) = (0,0) position; PAT(1,2,*) is the (1,0) position; PAT(2,1,*) is the (0,1) position; and PAT(2,2,*) is the (1,1) position. PAT(3,1,*), PAT(3,2,*), PAT(4,1,*), and PAT(4,2,*) are the corresponding V-derivatives. PAT(1,3,*), PAT(1,4,*), PAT(2,3,*), and PAT(2,4,*) are the corresponding U-derivatives. PAT(3,3,*), PAT(3,4,*), PAT(4,3,*), and PAT(4,4,*) are the corresponding cross-derivatives.
2. IERR [Out, INTEGER scalar]
   Flag for end-of-file: If an EOF is encountered, IERR is set to 1. Otherwise, it is set to 0.

7.1.18.2 Common Data

none

7.1.19 SUBROUTINE PATOPT

Displays patch-option menu and handles the user's responses until the main menu is reentered

7.1.19.1 Parameter List

empty

7.1.19.2 Common Data

```
COMMON /BLK1/ TITL, I1, I2
COMMON /QPATCH/ IDRPAT, NPATT, LUNPAT, NUPAT, NVPAT, LABPAT,
  NPAT, IPAT(500)
COMMON /SCRAT/ SBTEXT(10,24)
COMMON /STATUS/ ISTAT1
```

7.1.20 SUBROUTINE PCGMAL

Converts a PC curve from geometric representation, end position and tangent vectors, to algebraic representation, polynomial coefficients.

7.1.20.1 Parameter List

1. A [In, REAL(4,3)]
   Geometric representation of PC curve: A(1,J) and A(2,J) are the J-th components of the initial and final positions. A(3,J) and A(4,J) are the corresponding components of the tangents.
2. B [Out, REAL(4,3)]
   Algebraic representation of PC curve: B(I,J) is the coefficient of $U^{(4-I)}$ in the cubic polynomial for the J-th position component.

7.1.20.2 Common Data

none

7.1.21 SUBROUTINE PCIN

Reads an input file of PC curves

7.1.21.1 Parameter List

empty

7.1.21.2 Common Data

```
COMMON /BLK2/ IDUM1, I1, I2
COMMON /QPC/ IDUM2(1), NPCT, LUNPC, IDUM3(2), NPC, IPC(1000)
COMMON /SCRAT/ DUMMY(250), PC(12)
COMMON /STATUS/ ISTAT(2)
COMMON /INDEXX/ ICMPAT(501), ICMPC(1001), ICMVEC(1001), CMOPEN(3)
```

7.1.22 SUBROUTINE READPC

Reads a single PC curve

7.1.22.1 Parameter List

1. PC [Out, REAL(12)]
   PC curve: Effectively allocated PC(4,3). PC(*,J) is a parametric cubic function for the J-th spatial component. PC(1,*) and PC(2,*) are the initial and final position components. PC(3,*) and PC(4,*) are the initial and final tangent components.
2. IERR [Out, INTEGER scalar]
   Flag for end-of-file: If an EOF encountered, IERR is set to 1. Otherwise, it is set to 0.

7.1.22.2 Common Data

none

7.1.23 SUBROUTINE PCOPT

Displays PC-curve-option menu and handles the user's responses until the main menu is reentered

7.1.23.1 Parameter List

empty

7.1.23.2 Common Data

```
COMMON  /BLK2/  TITL,  I1,  I2
COMMON  /QPC/  IDRPC,  NPCT,  LUNPC,  NUPC,  LABPC
COMMON  /SCRAT/  SBTEXT(10,24)
COMMON  /STATUS/  IDUM1(1),  ISTAT2
```

7.1.24 SUBROUTINE PC2NPT

Computes a set of points on a PC curve at evenly-spaced parameter values

7.1.24.1 Parameter List

1. PC [In, REAL(4,3)]
   PC curve: PC(*,J) is a parametric cubic function for the J-th spatial component. PC(1,*) and PC(2,*) are the initial and final position components. PC(3,*) and PC(4,*) are the initial and final tangent components.
2. NPT [In, INTEGER scalar]
   A sign times the number of points to compute: If positive, PC is a geometric representation. If negative, PC is an algebraic representation.
3. PTS [Out, REAL(3,*)]
   Points computed along the curve

7.1.24.2 Common Data

none

7.1.25 SUBROUTINE PSGMAL

Converts a patch from geometric representation to algebraic representation

7.1.25.1 Parameter List

1. GEO [In, REAL(4,4,3)]
   Geometric patch representation: GEO(*,*,K) is the K-th component of a vector. GEO(1,1,*) is the (U,V) = (0,0) position; GEO(1,2,*) is the (1,0) position; GEO(2,1,*) is the (0,1) position; and GEO(2,2,*) is the (1,1) position. GEO(3,1,*), GEO(3,2,*), GEO(4,1,*), and GEO(4,2,*) are the corresponding V-derivatives. GEO(1,3,*), GEO(1,4,*), GEO(2,3,*), and GEO(2,4,*) are the corresponding U-derivatives. GEO(3,3,*), GEO(3,4,*), GEO(4,3,*), and GEO(4,4,*) are the corresponding cross-derivatives.
2. ALG [Out, REAL(4,4,3)]
   Algebraic representation of patch: ALG(I,J,K) is the coefficient of $U^{(4-J)} * V^{(4-I)}$ in the bicubic polynomial for the K-th position component.

7.1.25.2 Common Data

none

7.1.26 SUBROUTINE SBMENU

Displays the menu to select which elements from an input file to display

7.1.26.1 Parameter List

1. TITL [In, REAL scalar]
   Title characters for the kind of data element being selected
2. I1 [In, INTEGER scalar]
   Initial index value in the range being displayed for selection
3. I2 [In, INTEGER scalar]
   Final index value in the range being displayed for selection
4. NTOT [In, INTEGER scalar]
   Number of data elements which were read from the input file
5. K [In/Out, INTEGER(100)]
   List of selected elements: This is a list of indices, sorted in ascending order.
6. NK [In/Out, INTEGER scalar]
   The number of elements currently selected

7.1.26.2 Common Data

COMMON /BLK4/ IADD
COMMON /SCRAT/ TEXT(5,24)

7.1.27 SUBROUTINE SUB

Removes a single data element from the list of selected elements

7.1.27.1 Parameter List

1. K [In/Out, INTEGER(100)]
   The list of elements currently selected: This is a list of indices, sorted in ascending order.
2. N [In, INTEGER scalar]
   The number of elements currently selected
3. IPICK [In, INTEGER scalar]
   The index for the element to remove

7.1.27.2 Common Data

none

7.1.28 SUBROUTINE TEXBOT

Adds the bottom to an element-selection menu display

7.1.28.1 Parameter List

empty

7.1.28.2 Common Data

COMMON /SCRAT/ TEXT(5,24)

7.1.29 SUBROUTINE UVXYZ

Computes the position on a patch corresponding to an ordered pair of parameter values

7.1.29.1 Parameter List

1. ALG [In, REAL(4,4,3)]
   Algebraic representation of patch: ALG(I,J,K) is the coefficient of $U^{(4-J)} * V^{(4-I)}$ in the bicubic polynomial for the K-th position component.
2. U [In, REAL scalar]
   Patch parameter
3. V [In, REAL scalar]
   Patch parameter
4. X [Out, REAL scalar]
   Position component
5. Y [Out, REAL scalar]
   Position component
6. Z [Out, REAL scalar]
   Position component

7.1.29.2 Common Data

none

7.1.30 SUBROUTINE UXYZ

Computes the position on a PC curve corresponding to a parameter value

7.1.30.1 Parameter List

1. PCGEO [In, REAL(4,3)]
   PC curve: PCGEO(*,J) is a parametric cubic function for the J-th spatial component. PCGEO(1,*) and PCGEO(2,*) are the inital and final position components. PCGEO(3,*) and PCGEO(4,*) are the inital and final tangent components.
2. U [In, REAL scalar]
   Parameter
3. X [Out, REAL scalar]
   Position component
4. Y [Out, REAL scalar]
   Position component
5. Z [Out, REAL scalar]
   Position component

7.1.30.2 Common Data

none

7.1.31 SUBROUTINE VECIN

Reads an input file of vectors (i.e., intersection normals)

7.1.31.1 Parameter List

empty

7.1.31.2 Common Data

COMMON /BLK3/ IDUM1, I1, I2
COMMON /QVEC/ IDUM2(1), NVECTT, LUNVEC, IDUM3(1), NVEC, IVEC(1000)
COMMON /SCRAT/ DUMMY(250), VEC(6)
COMMON /STATUS/ IDUM4(2), ISTAT3
COMMON /INDEXX/ ICMPAT(501), ICMPC(1001), ICMVEC(1001), CMOPEN(3)

7.1.32 SUBROUTINE READV

Reads a single vector

7.1.32.1 Parameter List

1. VEC [Out, REAL(6)]
   Vector: The first 3 elements are the position components. The remaining 3 elements are components of the vector's direction.
2. IERR [Out, INTEGER scalar]
   Flag for end-of-file: If an EOF is encountered, IERR is set to 1. Otherwise, it is set to 0.

7.1.32.2 Common Data

none

7.1.33 SUBROUTINE VECOPT

Displays vector-option menu and handles user's responses until the main menu is reentered

7.1.33.1 Parameter List

empty

7.1.33.2 Common Data

COMMON /BLK3/ TITL, I1, I2
COMMON /QVEC/ IDRVEC, NVECTT, LUNVEC, LABVEC, NVEC, IVEC(1000)
COMMON /SCRAT/ SBTEXT(5,24)
COMMON /STATUS/ IDUM1(2), ISTAT3

7.1.34 SUBROUTINE VIEW

Displays the orientation menu and handles the user's selection

7.1.34.1 Parameter List

empty

7.1.34.2 Common Data

COMMON /QCVIEW/ ALPHA, BETA, GAMMA, X, Y, Z
COMMON /QXDXVL/ QXVAL(30)
COMMON /WINDOW/ XMIN, XMAX, YMIN, YMAX, ZMIN, ZMAX, FACTOR

7.1.35 SUBROUTINE LABL3D

Writes a text label at a 3-D location

7.1.35.1 Parameter List

1. TEXT [In, REAL scalar]
   Character representation of the label
2. NC [In, INTEGER scalar]
   Number of characters in the label
3. X [In, REAL scalar]
   Position coordinate
4. Y [In, REAL scalar]
   Position coordinate
5. Z [In, REAL scalar]
   Position coordinate

7.1.35.2 Common Data

COMMON /WINDOW/ XMIN, XMAX, YMIN, YMAX, ZMIN, ZMAX, FACTOR
COMMON /MATRIX/ ROTA(4,4)

7.1.36 SUBROUTINE MENUD

Displays a menu and returns the user's selection

7.1.36.1 Parameter List

1. ITITLE [In, INTEGER(7)]
   Title for the menu: This is character data.
2. NCT [In, INTEGER scalar]
   Number of characters in the title
3. LINE [In, INTEGER(*)]
   Character data for display on menu lines: Effectively, the subscripts (word,line) are used, corresponding to an INTEGER(NW,NL) allocation.
4. NW [In, INTEGER scalar]
   Number of words to display per line: This can be up to 7.
5. NL [In, INTEGER scalar]
   Number of lines to display
6. IAREA [Out, INTEGER scalar]
   Index for the line selected by the user

7.1.36.2 Common Data

none

7.1.37 SUBROUTINE LABAXS

Displays a line of text at a 2-D screen position

7.1.37.1 Parameter List

1. NULL [Not used]
2. ILINE [In, INTEGER(7)]
   Character text to display
3. NC [In, INTEGER scalar]
   Number of characters to display
4. IX [In, INTEGER scalar]
   Horizontal screen coordinate: This can range from 1 to 780.
5. IY [In, INTEGER scalar]
   Vertical screen coordinate: This can range from 1 to 780.

7.1.37.2 Common Data

none

7.1.38 SUBROUTINE DATADT

Displays a set of REAL variables and allows the user to selectively modify their values.

7.1.38.1 Parameter List

1. TITLE [In, REAL(*)]
   Title to be displayed: This is character data.
2. NC [In, INTEGER scalar]
   Number of title characters
3. TEXT [In, REAL(*)]
   Text describing the variables: This is packed character data, with the text for each variable immediately following the previous variable's text.
4. NW [In, INTEGER(*)]
   Number of text characters for each variable
5. NX [In, INTEGER scalar]
   Number of variables to display
6. X [In/Out, REAL(*)]
   The variables to be displayed and modified
7. NULL [Not used]

7.1.38.2 Common Data

none

7.1.39 SUBROUTINE GCUR3D

Displays a 3-D curve, by drawing lines connecting the input points in order

7.1.39.1 Parameter List

1. X [In, REAL(*)]
   Array of X-coordinate values
2. Y [In, REAL(*)]
   Array of Y-coordinate values
3. Z [In, REAL(*)]
   Array of Z-coordinate values
4. NPTS [In, INTEGER scalar]
   Number of points

7.1.39.2 Common Data

COMMON /MATRIX/ ROTA(4,4)
COMMON /WINDOW/ XMIN, XMAX, YMIN, YMAX, ZMIN, ZMAX, FACTOR

7.1.40 SUBROUTINE GVIEW

Computes a transformation matrix to rotate the display about its center

7.1.40.1 Parameter List

1. IX [In, INTEGER scalar]
   Ordinal for rotation about the X-axis: IX, IY, and IZ must all have different magnitudes from the set 1, 2, and 3. With a positive sign the rotation is counterclockwise; with a negative sign it is clockwise.
2. IY [In, INTEGER scalar]
   Ordinal for rotation about the Y-axis
3. IZ [In, INTEGER scalar]
   Ordinal for rotation about the Z-axis
4. ALPHA [In, REAL scalar]
   Angle for rotation about the X-axis, in degrees
5. BETA [In, REAL scalar]
   Angle for rotation about the Y-axis, in degrees
6. GAMMA [In, REAL scalar]
   Angle for rotation about the Z-axis, in degrees

7.1.40.2 Common Data

COMMON /WINDOW/ XL, XH, YL, YH, ZL, ZH, FACTOR

7.1.41 SUBROUTINE INPUTA

Reads in character data from the keyboard

7.1.41.1 Parameter List

1. IFILE [Out, INTEGER(*)]
   Input character data
2. NW [Not used]
3. IX [In, INTEGER scalar]
   Horizontal screen coordinate where the cursor is placed before reading data: This can range from 1 to 780.
4. IY [In, INTEGER scalar]
   Vertical screen coordinate where the cursor is placed before reading data: This can range from 1 to 780.

7.1.41.2 Common Data

none

7.1.42 SUBROUTINE INPUTI

Reads in integer data from the keyboard

7.1.42.1 Parameter List

1. IFILE [Out, INTEGER(*)]
   Input integer data
2. NW [Not used]
3. IX [In, INTEGER scalar]
   Horizontal screen coordinate where the cursor is placed before reading data: This can range from 1 to 780.
4. IY [In, INTEGER scalar]
   Vertical screen coordinate where the cursor is placed before reading data: This can range from 1 to 780.

7.1.42.2 Common Data

none

7.1.43 SUBROUTINE IDEN

Initalizes transformation matrix to an identity

7.1.43.1 Parameter List

empty

7.1.43.2 Common Data

COMMON /MATRIX/ ROTA(4,4)

7.1.44 SUBROUTINE XLAT3

Revises the transformation matrix, by applying a 3-D translation

7.1.44.1 Parameter List

1. X [In, REAL scalar]
   Translation component
2. Y [In, REAL scalar]
   Translation component
3. Z [In, REAL scalar]
   Translation component

7.1.44.2 Common Data

COMMON /MATRIX/ ROTA(4,4)

7.1.45 SUBROUTINE ROTAD

Revises the transformation matrix, by applying a rotation about one of the coordinate axes

7.1.45.1 Parameter List

1. DEG [In, REAL scalar]
   Angle, in degrees
2. IAXIS [In, INTEGER scalar]
   Axis selection: 1 selects the X-axis, 2 selects the Y-axis, and 3 selects the Z-axis.

7.1.45.2 Common Data

COMMON /MATRIX/ ROTA(4,4)

7.1.46 SUBROUTINE ORIENT

Draws a set of 3 reference axes

7.2.46.1 Parameter List

empty

7.1.46.2 Common Data

```
COMMON  /QCVIEW/  A,  B,  G,  X,  Y,  Z
COMMON  /WINDOW/  W(6)
COMMON  /MATRIX/  ROTA(4,4)
```

7.2 GENTRN

GENTRN is an interactive program which prompts the user for transformation data and combines it to define coordinate transformations.

7.2.1 FILES

The file list for GENTRN is the following: INPUT, OUTPUT, TRAN, TEMP, and LINEIN. INPUT and OUTPUT are connected to the terminal. Each line from INPUT is copied to file LINEIN before being processed. (This is done to simulate a BACKSPACE of terminal input, which can not be moved backwards.) Coordinate transformations are accumulated on TEMP, a temporary file. The transformations are written to file TRAN. The data on files TEMP and TRAN are in TRN format.

Within the program, INPUT and OUTPUT are respectively referred to as units 0 and 1. TEMP is referred to as unit 3. LINEIN is referred to as unit 4. TRAN is referred to as unit 2.

7.2.2 ROUTINES CALLED

Program GENTRN calls the following routines: COS, DATE, SIN, TIME, TRMCHK, and TXTCPY. Functions COS and SIN and Subroutines DATE and TIME are from the Fortran library; they are documented in the Fortran manual. Subroutines TRMCHK (see Section 8.89 below) and TXTCPY (8.90) are from the SUBS group of MASTER.

7.2.3 METHOD

GENTRN starts operation by writing an introductory message. It continues by defining transformations until an empty line is input instead of starting the next transformation. The number of transformations created is written to file TRAN. The transformations, which have been accumulated on file TEMP, are copied to file TRAN.

An individual transformation is created by reading an initial translation vector, then a sequence of rotations about the coordinate axes, and finally another translation vector.

7.2.4 MAIN-PROGRAM PROCEDURE

If file TRAN contains input transformations, they are copied to file TEMP. Transformations are assembled in core and then written to TEMP. Finally TEMP and TRAN are rewound, the output transformation count is written to TRAN, and the text on TEMP is copied to TRAN.

A transformation is assembled in the following way: Null initial and final translations and an identity matrix are initialized. The user is asked for an initial translation. Rotation input is explained to the user. Rotations are input, as an axis code and then an angle. Each rotation is set up and multiplied into the accumulated rotation matrix. A choice of zero for the axis code ends rotation input. Finally the user is asked for a final translation.

Entering an empty line at any stage is indicated by a false value for DATA. This causes the current transformation to be written to TEMP and then the TRAN output is prepared. (An empty line when an initial translation is requested will not write an identity transformation to TEMP.)

7.2.5 MAIN-PROGRAM ERROR CONDITIONS

none

7.3 MSHNRM

MSHNRM is a program which computes intersection normals where the mesh lines defined by a set of coordinate values touches a surface model.

7.3.1 FILES

The file list for MSHNRM is the following: MSH, SRF, INT, NRM, OUTPUT, PATMS, TEMPPC, TEMPRS, and TEMPRC. A mesh description is read from file MSH, and a set of patches is read from file SRF. Intersection curves, which are intermediate data, are written to file INT. Normals are written to file NRM. The mesh description is in MSH format, and the patch data is in SRF format. The output intersection curves are in CUR format, with system comments added. (There are 6 components to these curves, rather than the usual 3; components 4 through 6 are the surface-normal direction, interpolated along the curve.) The normals are output in NRM format. PATMS is a random-access file which stores the patches; each patch is stored as a 48-word binary record. TEMPPC is a sequential-access file which stores the intersection-curve segments for the current orientation of intersection planes as 24-word binary records. Files TEMPRS and TEMPRC are used only when intersecting rectangular coordinate patches with a cylindrical coordinate mesh. TEMPRS is a sequential-access file which stores patches with RADIUS appended as the fourth coordinate; each patch is stored as a 64-word binary record. TEMPRC is a sequential-access file which stores intersection curves with RADIUS appended as the seventh coordinate; each curve segment is stored as a 28-word binary record.

Within the program, MSH is referred to as unit 1. SRF is referred to as unit 2. INT is referred to as unit 3. NRM is referred to as unit 4. Files PATMS and TEMPPC are respectively referred to as units 5 and 7. OUTPUT is referred to as unit 6. Files TEMPRS and TEMPRC are respectively referred to as units 8 and 9.

7.3.2 ROUTINES CALLED

MSHNRM uses the following routines: COS, DATE, GETMSH, GETPAT, MSHOPT, PLNCUR, PLNSRF, RADCUR, RADCUT, RADINT, RADSRF, SIN, SYSTEM, and TIME. Functions COS and SIN and Subroutines DATE, SYSTEM and TIME are from the Fortran library; they are documented in the Fortran manual. Subroutines GETMSH (see Section 8.30 below), GETPAT (8.31), MSHOPT (8.42), PLNCUR (8.55), PLNSRF (8.56), RADCUR (8.61), RADCUT (8.62), RADINT (8.63), and RADSRF (8.64) are part of MASTER, from the SUBS group.

7.3.3 METHOD

MSHNRM starts by writing an introductory message, reading the mesh input, and then reading the patch input. The normals are calculated by first fitting intersection curves to the surface, where a first coordinate is held constant at mesh values, and then cutting the intersection curves where a second coordinate is held constant at mesh values. The intersection curves contain the 3 surface-normal components as well as the position coordinates.

7.3.4 MAIN-PROGRAM PROCEDURE

Variables NXINT, KXINT, NXCUT and KXCUT control the sequence of intersections and cuts. Each orientation is identified by the coordinate which is held constant. NXINT intersections are used: KXINT(1) to KXINT(NXINT). The current intersection orientation is stored as IXINT. For each IXINT value, NXCUT(IXINT) cutting orientations are used: KXCUT(1,IXINT) to KXCUT(NXCUT(IXINT), IXINT).

For each orientation, a plane is defined for each mesh value by requiring that a linear function of the coordinates must equal zero. This linear function is defined by a 4-element array: PLNINT or PLNCUT. The plane is defined such that

$$PLN(1)*X(1) + PLN(2)*X(2) + PLN(3)*X(3) + PLN(4) = 0.$$

The accuracy of the planar intersector is controlled by the tolerance value, TOLINT. This is set by default to $10.^{-4}$, which is tighter than the value listed in the user's manual by a factor of 10. This default value can be overridden by including a namelist in the mesh input options on file MSH.

Variable LSTOPT controls detailed listing output which is build into the code. The greater value LSTOPT takes, the more details are listed:

1. LSTOPT = 0 gives just a summary of the number of PC-segments representing intersection curves from each intersection orientation and the number of normals resulting from cutting these curves.
2. LSTOPT = 1 is the default. It adds a table of the intersection normals, listing the position, surface-normal direction, and indices for the normal, for the PC-curve which generated this normal, and for the patch which generated the PC-curve.
3. LSTOPT = 2 adds to the intersection-normal table a column listing which root of the cubic equation for the cut generated each normal. It also adds totals of PC-curves and normals for each interseection location.
4. LSTOPT = 3 adds a listing of each cut location.
5. LSTOPT = 4 adds internal details of the intersection computations whenever a possible error is detected.
6. LSTOPT = 10 adds these internal details for all intersections.
7. LSTOPT = 11 adds the cubic polynomial for each cut.
8. LSTOPT = 100 adds internal details of the approximation of each intersection by a set of PC curves.

The intersection curves are an interpolation from the discrete points on the intersection which are given by subroutine PLINT. The tangent direction at these points is well defined, but the tangent magnitudes are not. A pair of methods are implemented for scaling these tangents: The Ferguson-Phillips method, the default, determines tangent magnitudes to minimize the mean 2nd derivative magnitude along the curve. Locally-explicit scaling makes the tangent component at one endpoint pointing towards the other endpoint have a magnitude equal to the distance between the points; this makes the locally-rotated position component along this line a linear function of the parameter, so the remaining locally-rotated components are explicitly cubic functions of the first component. Locally explicit scaling can be selected by setting TANSCL from the default of 2 to 1, in the namelist for MSH option input.

7.3.5 MAIN-PROGRAM ERROR CONDITIONS

The following conditions will cause the main program to abort:
1. Bad input data
2. Over 100 failures to intersect an individual patch with a plane
3. A request to intersect a cylindrical-coordinate mesh with rectangular-coordinate patches (This capability is planned, but not yet implemented.)

7.4 NRMCFD

NRMCFD is a program which combines mesh and intersection-normal data to form a CFD input file. It also sorts, aligns and removes duplications from the input data.

7.4.1 FILES

The file list for NRMCFD is the following: MSH, NRM, OLDCFD, NEWCFD, and OUTPUT. A mesh description is read from file MSH, and a set of intersection normals is read from file NRM. The mesh description is in MSH format, and the normals are in NRM format. The first 80 columns of text data from file OLDCFD are read, until a mesh or normal keyword from CFD format are detected or until the data ends. The combined data is written to file NEWCFD, in CFD format.

Within the program, OUTPUT is referred to as unit 6. MSH, NRM, and OLDCFD are respectively referred to as units 1, 2, and 3. NEWCFD is referred to as unit 4.

7.4.2 ROUTINES CALLED

NRMCFD uses the following routines: CHECK, CHKNRM, DATE, EOF, GETMSH, KOMSTR, MSHOPT, SYSTEM, TIME, UNIQRN, and VERRN. Function EOF and Subroutines DATE, SYSTEM, and TIME are

from the Fortran library; they are documented in the Fortran manual. Subroutines CHECK (see Section 8.6 below), CHKNRM (8.7), GETMSH (8.30), KOMSTR (8.38), MSHOPT (8.42), UNIQRN (8.91), and VERRN (8.94) are from the SUBS group of MASTER.

7.4.3 METHOD

NRMCFD starts operation by writing an introductory message. The header data is copied from OLDCFD to NEWCFD. The mesh data is read, the mesh values are sorted, duplicate values are removed, and the mesh data is written to NEWCFD. The normals are read into core. The range of THETA is corrected to agree with the mesh values. Coordinates within a tolerance of a corresponding mesh value are shifted to the exact mesh value. Normals with duplicate positions are removed. Normals which do not match at least two mesh values are removed. The remaining normals are sorted and written to NEWCFD.

7.4.4 MAIN-PROGRAM PROCEDURE

All mesh and normal data are handled as in-core arrays.

7.4.5 MAIN-PROGRAM ERROR CONDITIONS

Missing mesh or normal input are the only error conditions detected by the main program. (Bad input data and sorting errors can be detected by its subroutines.) These conditions will abort the program.

7.5 NRMREV

NRMREV is a program which reverses the normal direction of a file of intersection normals.

7.5.1 FILES

The file list for NRMREV is the following: OLDNRM, NEWNRM, and OUTPUT. Intersection normals in NRM format are read from file OLDNRM, and normals are written to file NEWNRM in NRM format. A message introducing the program is written to file OUTPUT.

Within the program, OUTPUT is referred to as unit 6. OLDNRM and NEWNRM are respectively referred to as units 1 and 2.

7.5.2 ROUTINES CALLED

NRMREV calls the following routines: CHECK, DATE, and TIME. Subroutines DATE and TIME are from the Fortran library they are documented in the Fortran manual. Subroutine CHECK (see Section 8.6 below) is from the SUBS group of MASTER.

7.5.3 METHOD

An individual normal is read, all 3 normal components are negated, and the normal is written. This is repeated until the input ends.

7.5.4 MAIN-PROGRAM PROCEDURE

OLDNRM is rewound and any input comments are listed to OUTPUT. OLDNRM is rewound again and the normals are copied, with reversal, to NEWNRM. (Comments are copied with the normals.)

7.5.5 MAIN-PROGRAM ERROR CONDITIONS

NRMREV will not abort under any known conditions.

7.6 REGSIL

REGSIL is a program which reads a set of point strings representing curves, computes a new set of point strings representing the same curves, and writes a surface description based upon the new curve set. Each string in the new set has the same relative spacing, which is selected to minimize interpolation errors. The new strings have the least number of points required to preserve the curve shapes, within a tolerance.

REGSIL is very expensive to execute. SELECT should be used instead. (See Appendix E.3 in Reference 7.)

7.6.1 FILES

The file list for REGSIL is the following: OLDSIL, OPTION, NEWSIL, and OUTPUT. Section curves in the simplified SIL format described in Section 7.3.2 of the MASTER user's manual are read from file OLDSIL. Program-control options are read from file OPTION, as described in Section 7.4.1.5 of the user's manual. A surface description in SIL format is written to file NEWSIL. Printer-formatted listing data is written to file OUTPUT.

File OLDSIL is referred to as unit 2 within the program. OPTION is referred to as unit 3. NEWSIL is referred to as unit 4. OUTPUT is referred to as unit 6.

7.6.2 ROUTINES CALLED

REGSIL uses the following routines: DATE, ENRICH, EOF, ORDER, READ, SYSTEM, TIME, VARKNT, and XTRACT. Function EOF and subroutines DATE, SYSTEM, and TIME are from the Fortran library; they are documented in the Fortran manual. Subroutines ENRICH (see Section 8.26 below), ORDER (8.47), READ (8.66), and XTRACT (8.100) are from the SUBS group of MASTER; subroutine VARKNT (8.93) is from the LIBS group.

7.6.3 METHOD

SIL sections are input first. Optimal placements of a given number of knots are found for successive numbers of interior knots, until the input tolerance is satisfied. Finally a SIL block based upon the new knot placements is output.

7.6.4 MAIN-PROGRAM PROCEDURE

Subroutine READ is called to read the input sections. In a loop over the number of interior knots, Subroutine VARKNT is called to compute optimal knot placements. (Subroutine ENRICH can be used to add more points to the input curves and thus remove interlacing deficiencies.) This loop is exited when the maximum error is within the tolerance. Subroutine ORDER sorts the knot locations. Subroutine XTRACT writes the output SIL data.

7.6.5 MAIN-PROGRAM ERROR CONDITIONS

This program runs for a long time, so time limits are the most frequent error encountered.

7.7 SILSRF

SILSRF is a program which reads a surface description and computes a set of patches which models the described surface.

7.7.1 FILES

The file list for SILSRF is the following: OLDSIL, NEWSIL, SRF, and OUTPUT. A surface description is read from file OLDSIL, in SIL format. This data is copied to file NEWSIL, rewritten in SIL format with system comments. The surface model is output on file SRF, in SRF format. An introductory message, a patch count from each SIL block input, and optional dump output forms the printer information, which is written to OUTPUT.

Within the program, OLDSIL is referred to as unit 1. NEWSIL is referred to as unit 2. SRF is referred to as unit 3. OUTPUT is referred to as unit 6.

7.7.2 ROUTINES CALLED

SILSRF uses the following routines: CHECK, CURPAR, CURSLP, DATE, DBCLOS, DBOPEN, DSMAP, ENDMAP, ESGET, ESPUT, ISDMS, SILCPY, SILOPT, SVMAP, SYSTEM, TIME, and WRIPAT. Subroutines DATE, SYSTEM, and TIME are from the Fortran libarary; they are documented in the Fortran manual. Subroutines DBCLOS, DBOPEN, DSMAP, ENDMAP, ESGET, ESPUT, ISDMS, and SVMAP are part of the SDMS data-manager library; they are documented in the SDMS manual. Subroutines CHECK (see Section 8.6 below), CURPAR (8.18), CURSLP (8.19), SILCPY (8.73), SILOPT (8.74), and WRIPAT (8.98) are from the SUBS group of MASTER.

7.7.3 METHOD

Input options are processed first, the SIL input is copied with system comments added, and rewound. The input options are read past again. Then input SIL blocks are used to create patches until there is no more input. Each block is processed as it appears: section group, then member group, and finally patch group.

7.7.4 SDMS DATA MANAGEMENT

The data structure is predefined, according to the text in file SILDBD. SDMS program DDP uses this text to create a data-definition binary file, ZZZSIL. During execution, the data is stored on direct-access files ZZZSIL1, ZZZSIL2, ZZZSIL3, and ZZZSIL4. The SDMS working space, array BUFR, is initialized by subroutine ISDMS. A new database is opened by subroutine DBOPEN for each SIL block. Map "KNOTS" is associated with data set "KNOTSET", for section-knot data; map "CPTS" is associated with data set "CPSET", for corner-point data. These maps are defined by a sequence of subroutine calls: DSMAP, SVMAP, and then ENDMAP. The maps link in-core variables with corresponding fields of an arbitrary data-set element and the key values identifying the element. The key and data values are placed in the proper in-core variables, and then elements are added to the data set with a call to Subroutine ESPUT. When data is to be retreived, the key values are placed in their in-core variables, and ESGET is called; the data values then appear in their in-core variables. Subroutine DBCLOS closes the database. The data-definition options cause the database to be deleted upon closing.

7.7.5 MAIN-PROGRAM PROCEDURE

SILSRF begins operation with an introductory message. The SIL input is copied, checked, and reformatted. The input is rewound, and the SIL options are read. A SDMS database, to store section-knot and corner-point (i.e., member knot) data, is initialized. Patches are created independently from each block of SIL data.

For each block, the section-point positions are fitted with parametric cubic tension splines to give tangent vectors at the section knots. The position and tangent data is stored in the database. Next the member points are processed. Position and section-tangent vectors are retreived from the database. The positions are fitted with splines to give tangent vectors along the members, at the corner points. The corresponding section tangents are fitted with splines to give twist vectors (i.e., cross derivatives). The member tangent and twist data is stored along with section-knot index data in the database. Finally the patch specifications are processed. Data for the corner points is retreived, along with the corresponding section-knot data. The derivatives are scaled, and the patch is written out.

Before splines are fit, parameter values are assigned to each curve, in proportion to the chord length between adjacent points. This parameterization ranges from 0 to 1 over a complete curve. The parameter values are stored in the database. The derivatives which are retreived are scaled to give a parameter difference of 1.0 between adjacent knots, increasing their values. (This causes the opposite sides of a patch to be scaled differently, but a shared patch boundary is scaled identically in both patches.)

7.7.6 MAIN-PROGRAM ERROR CONDITIONS

A premature end to the SIL input will cause SILSRF to abort, as will a SDMS-detected error.

7.8 SRFINT

SRFINT is a program which computes the curves where a pair of surface models intersect each other.

7.8.1 FILES

The file list for SRFINT is the following: TAPE7, INPUT, OUTPUT, TAPE1, TAPE2, TAPE3, and TAPE4. The pair of surface models are read from TAPE1 and TAPE2; they are in SRF format. These surface models are stored on TAPE3 and TAPE4 respectively; they are sequential-access binary files containing 48-word records. Program-control options can be read from INPUT, in the format described in Section 7.4.2.5 of the MASTER user's manual. (Otherwise the default options are used.) Printer-formatted listing data is written to OUTPUT. The intersection curves are written to TAPE7, in the format described in Section 7.3.1 of the user's manual.

Files INPUT and OUTPUT are respectively referred to within the program as units 5 and 6.

7.8.2 ROUTINES CALLED

SRFINT calls the following routines: CLOSCV, DATE, DBCLOS, DSMAP, ENDMAP, ESSOPN, ESSPUT, GETPAT, ISDMS, OPENCV, SETTOL, SUFFER, SUFINT, SVMAP, SYTEM, TIME, and WRTCRV. Subroutines DATE, SYSTEM and TIME are from the Fortran library; they are documented in the Fortran manual. Subroutines DBCLOS, DSMAP, ENDMAP, ESSOPN, ESSPUT, ISDMS, and SVMAP are from the SDMS library; they are documented in the SDMS manual. Subroutines CLOSCV (see Section 8.8 below), GETPAT (8.31), OPENCV (8.46), SETTOL (8.72), SUFFER (8.84), and WRTCRV (8.98) are from the SUBS part of MASTER; Subroutine SUFINT (8.85) is from the LIBS part.

7.8.3 METHOD

SRFINT starts operation with an introductory message. The working space for the database manager is initialized, the database is opened, and the map from the database to corresponding in-core variables is defined. The patch sets representing both surfaces are read. A double loop computes the intersection of each combination of a patch from one surface and a patch from the other surface. Each intersection consists of one or more branches, which are disconnected from each other. Each branch is stored in the database, and its endpoints are stored within an in-core table. Next, connections between branches are added to the table, by matching endpoint positions. Finally, the connected curves, both open and closed, are output.

7.8.4 SDMS DATA MANAGEMENT

The data structure is predefined, according to the text in file IBRDBD. SDMS program DDP uses this text to create a data-definition binary file, ZZZIBR. During execution, the data is stored on direct-access files ZZZIBR1, ZZZIBR2, ZZZIBR3, and ZZZIBR4. The SDMS working space, array WORK, is initialized by subroutine ISDMS. Map "BRANCHMAP" is associated with data set "BRANCHES". This map is defined by a series of subroutine calls: DSMAP, SVMAP, and then ENDMAP. The map links an in-core array with an arbitrary data-set element sequence and the key values identifying the sequence. The key and data values are placed in the proper in-core variables, and then an element sequence is added to the data set with a call to subroutine ESSPUT. When data is to be retrieved, the key values are placed in their in-core variables, and ESSGET is called; the sequence then appears in its in-core array. Subroutine DBCLOS closes the database. The data-definition options cause the data base to be deleted upon closing.

7.8.5 MAIN-PROGRAM PROCEDURE

The procedure is explained in the main-program comments.

7.8.6 MAIN-PROGRAM ERROR CONDITIONS

At most 200 insersection branches can be handled by the endpoint table; overflow will cause the program to abort. Errors in reading patch data and SDMS-detected errors will also cause it to abort.

7.9 TRNSIL

TRNSIL is a program which transforms the coordinates within a surface description. It can convert cylindrical coordinates to or from rectangular coordinates. It can perform rotations and translations on rectangular coordinates.

The source code for TRNSIL was precompiled from TRANSFOR structured pseudocode.

7.9.1 FILES

The file list for TRNSIL is the following: INPUT, OUTPUT, G, TRAN, and NEWG. Files INPUT and OUTPUT are connected to the terminal (or batch input and output). Transformation selections are read from INPUT after prompting messages are written to OUTPUT. The surface description is read from file G, in SIL format. The transformed surface description is written to file NEWG in SIL format and with system comments.

Within the program, G is referred to as unit 1. NEWG is referred to as unit 2. TRAN is referred to as unit 3. INPUT and OUTPUT are respectively referred to as units 5 and 6.

7.9.2 ROUTINES CALLED

The main program for TRNSIL uses the following routines: DATE, GETRAN, MEMGRP, PATGRP, SECGRP, SYSTEM, and TIME. Subroutines DATE, SYSTEM, and TIME are from the Fortran library; they are documented in the Fortran Manual. Subroutines GETRAN (see Section 7.9.5 below), MEMGRP (7.9.10), PATGRP (7.9.11), and SECGRP (7.9.9) are from the MAIN group of MASTER; they are found in Update deck TRNSIL.

7.9.3 METHOD

TRNSIL begins operation with an introductory message. It gets the transformation selection. It transforms the 3 parts of a SIL block in the order that they appear: first the section group, the member group, and then the patch-specification group.

7.9.4 MAIN-PROGRAM ROUTINE

7.9.4.1 Description

TRNSIL calls subroutines to perform each step of operation. It will abort only if a subroutine detects an error.

7.9.4.2 Common Data

none

7.9.5 SUBROUTINE GETRAN

Gets the transformation selection from the user and reads the selected transformation from file TRAN

7.9.5.1 Parameter List

1. LIN [In, INTEGER scalar]
   Logical unit for file INPUT

2. LOUT [In, INTEGER scalar]
   Logical unit for file OUTPUT
3. LTRAN [In, INTEGER scalar]
   Logical unit for file TRAN
4. OK [Out, LOGICAL scalar]
   Error flag

7.9.5.2 Common Data

COMMON /TRNDEF/ RECPOL, POLREC, DELTAI(3), ROT(3,3), DELTAO(3)

7.9.6 SUBROUTINE RDTRN

Reads a selected transformation from file TRAN

7.9.6.1 Parameter List

1. LOUT [In, INTEGER scalar]
   Logical unit for file OUTPUT
2. LTRAN [In, INTEGER scalar]
   Logical unit for file TRAN
3. ITRAN [In, INTEGER scalar]
   Index for the desired transformation
4. DELTAI [Out, REAL(3)]
   Inital translation
5. ROT [Out, REAL(3,3)]
   Rotation matrix
6. DELTAO [Out, REAL(3)]
   Final translation
7. OK [Out, LOGICAL scalar]
   Error flag

7.9.6.2 Common Data

none

7.9.7 SUBROUTINE FNDTRN

Finds the desired transformation, by skipping past the preceding ones

7.9.7.1 Parameter List

1. LOUT [In, INTEGER scalar]
   Logical unit for file OUTPUT
2. ITRAN [In, INTEGER scalar]
   Index for the desired transformation
3. OK [Out, LOGICAL scalar]
   Error flag

7.9.7.2 Common Data

none

7.9.8 SUBROUTINE CHKROT

Checks a rotation matrix for orthogonality, to within a tolerance

7.9.8.1 Parameter List

1. LOUT [In, INTEGER scalar]
   Logical unit for OUTPUT
2. N [In, INTEGER scalar]
   Matrix size
3. ROT [In, REAL(3,3)]
   Rotation matrix
4. OK [Out, LOGICAL scalar]
   Flag for orthogonality of the matrix

7.9.8.2 Common Data

none

7.9.9 SUBROUTINE SECGRP

Transforms the section group of a SIL block

7.9.9.1 Parameter List

1. LOUT [In, INTEGER scalar]
   Logical unit for file OUTPUT
2. LGIN [In, INTEGER scalar]
   Logical unit for file G
3. LGOUT [In, INTEGER scalar]
   Logical unit for file NEWG
4. OK [Out, LOGICAL scalar]
   Error flag

7.9.9.2 Common Data

none

7.9.10 SUBROUTINE MEMGRP

Transforms the member group of a SIL file

7.9.10.1 Parameter List

1. LOUT [In, INTEGER scalar]
   Logical unit for file OUTPUT
2. LGIN [In, INTEGER scalar]
   Logical unit for file G
3. LGOUT [In, INTEGER scalar]
   Logical unit for file NEWG
4. OK [Out, LOGICAL scalar]
   Error flag

7.9.10.2 Common Data

none

7.9.11 SUBROUTINE PATGRP

Copies the patch group of a SIL file

7.9.11.1 Parameter List

1. LOUT [In, INTEGER scalar]
   Logical unit for file OUTPUT
2. LGIN [In, INTEGER scalar]
   Logical unit for file G
3. LGOUT [In, INTEGER scalar]
   Logical unit for file NEWG
4. OK [Out, LOGICAL scalar]
   Error flag

7.9.11.2 Common Data

none

7.9.12 SUBROUTINE SEC

Transforms a single section

7.9.12.1 Parameter List

1. LOUT [In, INTEGER scalar]
   Logical unit for file OUTPUT
2. LGIN [In, INTEGER scalar]
   Logical unit for file G
3. LGOUT [In, INTEGER scalar]
   Logical unit for file NEWG
4. ISEC [In, INTEGER scalar]
   Section index
5. OK [Out, LOGICAL scalar]
   Error flag

7.9.12.2 Common Data

none

7.9.13 SUBROUTINE MEM

Transforms a single member

7.9.13.1 Parameter List

1. LOUT [In, INTEGER scalar]
   Logical unit for file OUTPUT
2. LGIN [In, INTEGER scalar]
   Logical unit for file G
3. LGOUT [In, INTEGER scalar]
   Logical unit for file NEWG
4. IMEM [In, INTEGER scalar]
   Member index
5. OK [Out, LOGICAL scalar]
   Error flag

7.9.13.2 Common Data

COMMON /SECKNT/ NKNOT(27), XKNOT(3,29,27)

7.9.14 SUBROUTINE READEC

Reads the end condition for either a section or a member

7.9.14.1 Parameter List

1. LOUT [In, INTEGER scalar]
   Logical unit for file OUTPUT
2. LGIN [In, INTEGER scalar]
   Logical unit for file G
3. LGOUT [In, INTEGER scalar]
   Logical unit for file NEWG
4. IENDF [Out, INTEGER scalar]
   End-condition code
5. ENDD [Out, REAL(3,2)]
   Inital and final tangent vectors
6. OK [Out, LOGICAL scalar]
   Error flag

7.9.14.2 Common Data

none

7.9.15 SUBROUTINE WRITEC

Transforms and writes the end condition for either a section or a member

7.9.15.1 Parameter List

1. LOUT [In, INTEGER scalar]
   Logical unit for file OUTPUT
2. LGOUT [In, INTEGER scalar]
   Logical unit for file NEWG
3. IENDF [In, INTEGER scalar]
   End-condition code
4. ENDD [In, REAL(3,2)]
   Initial and final tangent vectors
5. XFIRST [In, REAL(3)]
   Inital position
6. XLAST [In, REAL(3)]
   Final position
7. OK [Out, LOGICAL scalar]
   Error flag

7.9.15.2 Common Data

none

7.9.16 SUBROUTINE READSP

Reads the string of points for a section

7.9.16.1 Parameter List

1. LOUT [In, INTEGER scalar]
   Logical unit for file OUTPUT
2. LGIN [In, INTEGER scalar]
   Logical unit for file G

3. LGOUT [In, INTEGER scalar]
   Logical unit for file NEWG
4. NPTS [Out, INTEGER scalar]
   Number of points
5. XPT [Out, REAL(3,53)]
   Coordinates
6. TENSE [Out, REAL(53)]
   Tension values
7. ISKNOT [Out, INTEGER(53)]
   Knot flags
8. OK [Out, LOGICAL scalar]
   Error flag

7.9.16.2 Common Data

none

7.9.17 SUBROUTINE READMP

Reads the string of points for a member

7.9.17.1 Parameter List

1. LOUT [In, INTEGER scalar]
   Logical unit for file OUTPUT
2. LGIN [In, INTEGER scalar]
   Logical unit for file G
3. LGOUT [In, INTEGER scalar]
   Logical unit for file NEWG
4. NPTS [Out, INTEGER scalar]
   Number of points
5. ISKNOT [Out, INTEGER(27)]
   Section-knot indices
6. ISEC [Out, INTEGER(27)]
   Section indices
7. TENSE [Out, REAL(27)]
   Tension values
8. IMKNOT [Out, INTEGER(27)]
   Knot flags
9. OK [Out, LOGICAL scalar]
   Error flag

7.9.17.2 Common Data

none

7.9.18 SUBROUTINE WRITSP

Transforms and writes the string of points for a section

7.9.18.1 Parameter List

1. LOUT [In, INTEGER scalar]
   Logical unit for file OUTPUT
2. LGOUT [In, INTEGER scalar]
   Logical unit for file NEWG

3. NPTS [In, INTEGER scalar]
   Number of points
4. XPT [In, REAL(3,53)]
   Untransformed coordinates
5. TENSE [In, REAL(53)]
   Tension values
6. JSKNOT [In, INTEGER(53)]
   Knot flags
7. OK [Out, LOGICAL scalar]
   Error flag

7.9.18.2 Common Data

COMMON /TRNDEF/ RECPOL, POLREC, DUMMY1(3), DUMMY2(3,3), DUMMY3(3)

COMMON /BRANCH/ THOLD

7.9.19 SUBROUTINE WRITMP

Copies the point string for a member

7.9.19.1 Parameter List

1. LOUT [In, INTEGER scalar]
   Logical unit for file OUTPUT
2. LGOUT [In, INTEGER scalar]
   Logical unit for file NEWG
3. NPTS [In, INTEGER scalar]
   Number of points
4. ISKNOT [In, INTEGER(27)]
   Section-knot indices
5. ISEC [In, INTEGER(27)]
   Section indices
6. TENSE [In, REAL(27)]
   Tension values
7. JMKNOT [In, INTEGER(27)]
   Knot flags
8. OK [Out, LOGICAL scalar]
   Error flag

7.9.19.2 Common Data

none

7.9.20 SUBROUTINE SEQKNT

Adjusts the knot flags for either a section or a member curve to show the ordering of the knots

7.9.20.1 Parameter List

1. NPTS [In, INTEGER scalar]
   Number of points in curve
2. ISKNOT [In, INTEGER(53)]
   Unsequenced knot flags
3. JSKNOT [Out, INTEGER(53)]
   Sequenced knot flags

7.9.20.2 Common Data

none

7.9.21 SUBROUTINE SAVKNT

Saves the knot coordinates from a section, for use when transforming member end directions

7.9.21.1 Parameter List

1. LOUT [In, INTEGER scalar]
   Logical unit for file OUTPUT
2. ISEC [In, INTEGER scalar]
   Index for this section
3. NPTS [In, INTEGER scalar]
   Number of points in this section
4. XPT [In, REAL(3,53)]
   Section-point coordinates
5. JSKNOT [In, INTEGER(53)]
   Knot flags for this section
6. OK [Out, LOGICAL scalar]
   Error flag

7.9.21.2 Common Data

COMMON /SECKNT/ NKNOT(27), XKNOT(3,29,27)

7.9.22 SUBROUTINE CHECK

Finds the next data line in the input file: System comment lines are skipped, and user comment lines are copied to an output file.

7.9.22.1 Parameter List

1. LIN [In, INTEGER scalar]
   Logical unit for input file
2. LOUT [In, INTEGER scalar]
   Logical unit for output file to which the user comments are copied
3. DATA [Out, LOGICAL scalar]
   Flag indicating that a data line was found before the end-of-file was reached

7.9.22.2 Common Data

none

7.9.23 SUBROUTINE TRAN

Transforms position vectors and (optionally) tangent vectors

7.9.23.1 Parameter List

1. XI [In, REAL(3)]
   Untransformed position vector
2. DXI [In, REAL(3)]
   Untransformed tangent vector
3. SLOPES [In, LOGICAL scalar]
   Flag to transform the tangent vector

4. XO [Out, REAL(3)]
   Transformed position vector
5. DXO [Out, REAL(3)]
   Transformed tangent vector
6. OK [Out, LOGICAL scalar]
   Error flag

7.9.23.2 Common Data

COMMON /TRNDEF/ RECPOL, POLREC, DUMMY1(3), DUMMY2(3,3), DUMMY3(3)

7.9.24 SUBROUTINE LINEAR

Performs linear transformation of position and (optionally) tangent

7.9.24.1 Parameter List

1. XI [In, REAL(3)]
   Untransformed position vector
2. DXI [In, REAL(3)]
   Untransformed tangent vector
3. SLOPES [In, LOGICAL scalar]
   Flag to transform tangent
4. XO [Out, REAL(3)]
   Transformed position vector
5. DXO [Out, REAL(3)]
   Transformed tangent vector

7.9.24.2 Common Data

COMMON /TRNDEF/ DUMMY(2), DELTAI(3), ROT(3,3), DELTAO(3)

7.9.25 SUBROUTINE RTOP

Converts 2-D rectangular coordinates to polar coordinates

7.9.25.1 Parameter List

1. X [In, REAL scalar]
   The rectangular coordinate that points in the THETA = 0 degrees direction
2. Y [In, REAL scalar]
   The rectangular coordinate that points in the THETA = 90 degrees direction
3. R [Out, REAL scalar]
   Radial polar coordinate
4. THETA
   [Out, REAL scalar]
   Angular polar coordinate, in degrees

7.9.25.2 Common Data

none

7.9.26 SUBROUTINE PTOR

Converts 2-D polar coordinates to rectangular coordinates

7.9.26.1 Parameter List

1. R [In, REAL scalar]
   Radial polar coordinate
2. T [In, REAL scalar]
   Angular polar coordinate, in degrees
3. X [Out, REAL scalar]
   The rectangular coordinate that is proportional to COS(THETA)
4. Y [Out, REAL scalar]
   The rectangular coordinate that is proportional to SIN(THETA)

7.9.26.2 Common Data

none

7.9.27 SUBROUTINE ROTATE

2-D rotation of rectangular coordinates

7.9.27.1 Parameter List

1. XIN [In, REAL scalar]
   Unrotated rectangular coordinate
2. YIN [In, REAL scalar]
   Unrotated rectangular coordinate
3. ANGLE [In, REAL scalar]
   Angle of rotation, in degrees

7.9.27.2 Common Data

none

8.0 SYSTEM LIBRARY

This section describes the code in the MASTER system library. Each routine is found in either the SUBS part of MASTER or the LIBS part, in an Update deck with the same name as the routine.

8.1 SUBROUTINE ARRSRT (from LIBS)

Sorts the elements of a real array in ascending order and creates a key array

8.1.1 PARAMETER LIST

1. N [In, INTEGER scalar]
   Number of values to sort
2. A [In/Out, REAL(*)]
   Array of real values to be sorted in place
3. KEY [In/Out INTEGER(*)]
   Array of integer values which is rearranged to match A

8.1.2 COMMON DATA

none

8.2 SUBROUTINE BICCOF (from LIBS)

Converts a bicubic function from geometric representation to algebraic representation

8.2.1 PARAMETER LIST

1. P [In, REAL(16)]
   Geometric representation of the bicubic function: Effectively allocated REAL(4,4). P(1,1) is the (U,V) = (0,0) value; P(1,2) is the (1,0) value; P(2,1) is the (0,1) value; and P(2,2) is the (1,1) value. P(3,1), P(3,2), P(4,1), and P(4,2) are the corresponding V-derivatives. P(1,3), P(1,4), P(2,3), and P(2,4) are the corresponding U-derivatives. P(3,3), P(3,4), P(4,3), and P(4,4) are the corresponding cross-derivatives.
2. BC [Out, REAL(4,4)]
   Algebraic representation of the bicubic function: BC(I,J) is the coefficient of $U^{(4-J)} * V^{(4-I)}$.

8.2.2 COMMON DATA

none

8.3 SUBROUTINE BRCOEF (from LIBS)

From Subroutine SUFINT, generates Bernstein coefficients for a patch (or for a part of a subdivided patch): One spatial component is processed with each call to BRCOEF.

8.3.1 PARAMETER LIST

1. SVAL [In, REAL(2)]
   U values bounding the subpatch
2. TVAL [In, REAL(2)]
   V values bounding the subpatch
3. PATCH [In, REAL(16)]
   Geometric representation of the bicubic function (Actually a single bicubic function): Effectively allocated REAL(4,4). P(1,1) is the (U,V) = (0,0) value; P(1,2) is the (1,0) value; P(2,1) is the (0,1) value; and P(2,2) is the (1,1) value. P(3,1), P(3,2), P(4,1), and P(4,2) are the corresponding V-derivatives. P(1,3), P(1,4), P(2,3), and P(2,4) are the corresponding U-derivatives. P(3,3), P(3,4), P(4,3), and P(4,4) are the corresponding cross-derivatives.

4. VET [Out, REAL(16)]
   Bicubic coefficients for Bernstein representation: This is called VERT in Subroutine SUFINT.

8.3.2 COMMON DATA

none

8.4 SUBROUTINE BRINT (from LIBS)

Within Subroutine PLINT, determines the S values in the region of interest at which the bicubic has a double root, by using utility Subroutine HSZERO to root the resultant function that is calculated by Function DISVAL

8.4.1 PARAMETER LIST

1. TOL [In, REAL(2)]
   Array containing the user-specified tolerance: TOL(1) is the input tolerance. TOL(2) is computed by PLINT and is used as a lower limit for the stepsize.
2. IRT [In, INTEGER(3)]
   Array containing the subscripts of the roots of a cubic equation: U(IRT(1)) is the real root found via a Newton iterative scheme; U(IRT(2)) and U(IRT(3)) are the roots found from the quadratic factor.
3. VBND [In, REAL(2)]
   Vertical (S) boundaries to the region of interest
4. ANS [Out, REAL(15)]
   Array containing the S values at which the bicubic has a double root for T: This is defined only when NDEG is 3.
5. NA [Out, INTEGER scalar]
   Number of values in ANS
6. BB [Out, REAL(3)]
   Array containing S values where the bicubic is degenerate (i.e., The polynomial in T at these S values has degree less than 3.)
7. NBB [Out, INTEGER(3)]
   The degree of the polynomial in T obtained by evaluating the bicubic at the degenerate S values
8. NB [Out, INTEGER scalar]
   Number of values in BB
9. NDEG [Out, INTEGER scalar]
   The maximum degree of the polynomial in T obtained by evaluating the bicubic at an arbitrary S value
10. IER [In/Out, INTEGER scalar]
    Success/error code: BRINT detects IER = -6 and returns IER values from Subroutines HSZERO (whose error codes are shifted in value) and DEGNER.

    (-6) is returned when the patch is recognized as having more then the theoretical limit of fifteen S values where T has a double root.

8.4.2 COMMON DATA

COMMON /IRP1/ A(4,4)

The bicubic polynomial is input from A, in algebraic form.

8.5 SUBROUTINE CBSPN (from LIBS)

Evaluates the set of cubic B-spline functions for a set of knots at a single point

8.5.1 PARAMETER LIST

1. X [In, REAL(*)]
   Array of knots: They must be monotonically increasing.
2. N [In, INTEGER scalar]
   Number of knots: This must be at least 8.
3. T [In, REAL scalar]
   Value where the B-spline functions are evaluated: T must be within the closed interval [X(4), X(N-4)].
4. VAL [Out, REAL(*)]
   B-spline function values: There are N-4 functions.
5. IER [Out, INTEGER scalar]
   Error code: 0 if successful; -5 if T is out of range.

8.5.2 COMMON DATA

COMMON /CNTRL/ IL, EPS, DELX, MODE, MAXFN, KF, KORD

IL is the only active common variable; it is set so that T lies in the half-open interval [X(IL), X(IL+1)).

8.6 SUBROUTINE CHECK (from SUBS)

Finds the next data line on an input file: System comments are skipped, and user comments are copied to an output file.

8.6.1 PARAMETER LIST

1. INPUT [In, INTEGER scalar]
   Logical unit for the input file
2. OUTPUT [In, INTEGER scalar]
   Logical unit for the output file to which user comments are copied
3. DATA [Out, LOGICAL scalar]
   Indicates that a data line was found before the end-of-file was reached

8.6.2 COMMON DATA

none

8.7 SUBROUTINE CHKNRM (from SUBS)

Checks that a set of intersection normals each lie on a mesh line: Any normals which do not are removed from the set.

8.7.1 PARAMETER LIST

1. OUTPUT [In, INTEGER scalar]
   Logical unit for listing file: Any removed normals are listed.
2. NMSH [In, INTEGER(3)]
   Number of mesh values in each set
3. MAXMSH [In, INTEGER scalar]
   Length allocated to XMSH array
4. XMSH [In, REAL(MAXMSH,3)]
   Mesh value sets for each coordinate
5. NNRM [In/Out, INTEGER scalar]
   Number of intersection normals in the set
6. XNRM [In/Out, REAL(6,NNRM)]
   Set of intersection normals

8.7.2 COMMON DATA

none

8.8 SUBROUTINE CLOSCV (from SUBS)

Used by Program SRFINT to connect a closed curve: The endpoint table entries for the intersection branches are linked together.

8.8.1 PARAMETER LIST

1. L6 [In, INTEGER scalar]
   Logical unit for OUTPUT
2. I [In, INTEGER scalar]
   Index of the beginning endpoint in the table
3. NEND [In, INTEGER scalar]
   Number of endpoints in the table
4. ICON [In, INTEGER(201)]
   Where each endpoint connects: This is a table of endpoint indices, whose subscripts are also endpoint indices.
5. IDONE [In/Out, INTEGER(201)]
   Flags to show whether each endpoint has been connected yet: Initially 0, this is set to 1 when a connection is made.
6. ICURVE [Out, INTEGER(201)]
   The connected list of endpoints making up the output curve.
7. NP [Out, INTEGER scalar]
   The number of endpoints in the output curve

8.8.2 COMMON DATA

COMMON /ENDTBL/ X(201), Y(201), Z(201), IP1(201), IP2(201), NBR(201)
KOUNT(201), IPOINT(201)

8.9 SUBROUTINE CONCUV (from LIBS)

Called by Subroutines REACUV and FINCUV to store surface/surface intersection branches

8.9.1 PARAMETER LIST

1. IS [In, INTEGER scalar]
2. ITYP [In, INTEGER scalar]
   Type of curve to store
3. JCRV1 [In, INTEGER scalar]
   Curve index for PONT: This is for the output location and for a type-1 curve input location.
4. JCRV2 [In, INTEGER scalar]
   Curve index for PONT: This in the input location for a non-type-1 curve.
5. MXJPT [In, INTEGER scalar]
   Maximum number of points to be stored in a single intersection branch
6. NJCRV [In, INTEGER scalar]
   Number of curves in array PONT
7. NMPT [In/Out, INTEGER(*)]
   Number of points stored on each intersection branch
8. PONT [In/Out, REAL(3,MXJPT,NJCRV)]
   Intersection branches, also called curves
9. IER [Out, INTEGER scalar]
   Error code: This is zero for a normal exit. See Subroutine SUFINT for more details.

8.9.2 COMMON DATA

none

8.10 SUBROUTINE CONHUL (from LIBS)

Computes the convex hull bounding a patch

8.10.1 PARAMETER LIST

1. TOL [In, REAL scalar]
   Tolerance value
2. NV [In, INTEGER scalar]
   Number of Bernstein coefficients: This is typically 16.
3. VERT [In, REAL(3,NV)]
   Bernstein coefficients for the patch
4. NUM [Out, INTEGER scalar]
   Number of planes in the convex hull
5. NORM [Out, REAL(4,30)]
   The convex hull bounding the patch: This is a set of planes, each of the form NORM(*,J). Each plane is represented by the coefficients of its equation:
   NORM(1,J)*X(1) + NORM(2,J)*X(2) + NORM(3,J)*X(3) + NORM(4,J) + 0.
6. IER [Out, INTEGER scalar]
   Error code: This is zero for a normal exit. See Subroutine SUFINT for more details.

8.10.2 COMMON DATA

none

8.11 SUBROUTINE CONINT (from LIBS)

Called by Subroutine SUFINT, determines whether a patch from the second surface intersects the convex hull bounding a patch from the first surface

8.11.1 PARAMETER LIST

1. TOL [In, REAL scalar]
   Tolerance value
2. NUM [In, INTEGER scalar]
   Number of planes in the convex hull
3. NORM [In, REAL(4,NUM)]
   The convex hull bounding the first-surface patch: This is a set of planes, each of the form NORM(*,J). Each plane is represented by the coefficients of its equation:
   NORM(1,J)*X(1) + NORM(2,J)*X(2) + NORM(3,J)*X(3) + NORM(4,J) + 0.
4. NV [In, INTEGER scalar]
   Number of Bernstein coefficients for the second-surface patch
5. VERT [In, REAL(16,3)]
   Bernstein coefficients representing the second-surface patch
6. INT [Out, INTEGER scalar]
   Intersection code: This is 1 if the second-surface patch intersects the convex hull of the first surface. Otherwise this is 0, and there is no possibility of the patches intersecting each other.

8.11.2 COMMON DATA

none

8.12 SUBROUTINE CON2HL (from LIBS)

Called by Subroutine CONHUL

8.12.1 PARAMETER LIST

1. TOL [In, REAL scalar]
   Tolerance value
2. VERT [In, REAL(3,16)]
   Bernstein patch coefficients
3. TVER [Temporary, REAL(3,16)]
   Array to store and manipulate VERT data
4. NV [In, INTEGER scalar]
   Number of Bernstein coefficients
5. NF [In, INTEGER scalar]
6. NORM [In/Out, REAL(3)]
   Normal to a polygon face
7. TNRM [In, REAL(4)]
   Coefficients for a plane
8. KEY [In/Out, INTEGER(16)]
   Key to rearrange convex hull data
9. INT [Out, INTEGER scalar]
10. NXTRM [In/Out, INTEGER scalar]

8.12.2 Common Data

none

8.13 SUBROUTINE CRSPRD (from LIBS)

Computes the normalized cross product of a pair of vectors and the component of a third vector in this direction

8.13.1 PARAMETER LIST

1. VEC1 [In, REAL(3)]
   First vector for cross product
2. VEC2 [In, REAL(3)]
   Second vector for cross product
3. VEC3 [In, REAL(3)]
   Vector whose component in cross-product direction is computed
4. NORM [Out, REAL(4)]
   NORM(1), NORM(2), and NORM(3) are the components of the normalized cross product. NORM(4) is the component of VEC3 in this direction.

8.13.2 COMMON DATA

none

8.14 SUBROUTINE CR1PRM (from LIBS)

Within Program REGSIL, computes parameter values along a curve

8.14.1 PARAMETER LIST

1. CRV [In, REAL(NDIMC,*)]
   Coordinates of the input points and the chord lengths along the curve.

2. NPTS [In, INTEGER scalar]
   Number of points along the curve
3. NDIM [In, INTEGER scalar]
   Number of spatial coordinates; set to 3.
4. NDIMC [In, INTEGER scalar]
   Number of curve dimensions
5. P [Out, REAL(*)]
   Parameter values: These are monotonically increasing. The increase is proportional to the chord length between adjacent points. The values range from 0 to 1.
6. IER [Out, INTEGER scalar]
   Error code: 0 for success; 1 if NPTS is 1; 2 if adjacent points on the curve are coincident. (When all the points are coincident, an equally-spaced parametrization is returned.)
7. CYL [In, LOGICAL scalar]
   Flag for cylindrical coordinates
8. IXAXI [In, INTEGER scalar]
   Index for the axial cylindrical coordinate
9. IXRAD [In, INTEGER scalar]
   Index for the radial cylindrical coordinate
10. IXANG [In, INTEGER scalar]
    Index for the angular cylindrical coordinate

8.14.2 COMMON DATA

none

8.15 SUBROUTINE CUBCOF (from LIBS)

Computes the bicubic polynomial for the distance from a point on a patch to the nearest point on a plane: This function's zero curves are the plane/patch intersection slices.

8.15.1 PARAMETER LIST

1. PLN [In, REAL(4)]
   Coefficients for the plane:
   PLN(1)*X(1) + PLN(2)*X(2) + PLN(3)*X(3) + PLN(4) = 0.
2. P [In, REAL(16,3)]
   Geometric patch representation: Effectively allocated P(4,4,3). P(*,*,K) is the K-th component of a vector. P(1,1,*) is the (U,V) = (0,0) position; P(1,2,*) is the (1,0) position; P(2,1,*) is the (0,1) position; and P(2,2,*) is the (1,1) position. P(3,1,*), P(3,2,*), P(4,1,*), and P(4,2,*) are the corresponding V-derivatives. P(1,3,*), P(1,4,*), P(2,3,*), and P(2,4,*) are the corresponding U-derivatives. P(3,3,*), P(3,4,*), P(4,3,*), and P(4,4,*) are the corresponding cross-derivatives.
3. BC [Out, REAL(4,4)]
   Coefficients for the output bicubic function, in algebraic form: BC(I,J) is the coefficient of $U^{(4-J)} * V^{(4-I)}$.

8.15.2 COMMON DATA

none

8.16 SUBROUTINE CUBIC (from LIBS)

Finds the roots of a cubic polynomial

8.16.1 PARAMETER LIST

1. A [In, REAL(4)]
   Array containing the coefficients of the cubic:
   $A(1) * X^3 + A(2) * X^2 + A(3) * X + A(4) = 0.$

2. TOL [In, REAL(2)]
   Array containing the user-specified tolerance
3. IGS [In, INTEGER scalar]
   Guess code: If positive, an initial approximation to a root is input as U(IT(1)), otherwise Subroutine CUBIC computes the initial approximation.
4. IT [In, INTEGER(3)]
   Array containing the subscripts where the roots of a cubic equation are to be stored: U(IT(1)) is the real root found via a Newton iterative scheme; U(IT(2)) and U(IT(3)) are the roots found from the quadratic factor.
5. NR [In, INTEGER scalar]
   The number of roots desired: If NR is 1, only the initial root found via the Newton scheme is returned; otherwise all 3 roots are returned.
6. IC [Out, INTEGER scalar]
   The sign of the discriminant for the quadratic equation obtained by deflating the cubic with the first root. (The quadratic equation has IC + 1 roots, so the cubic equation has IC + 2 roots.)
7. U [In/Out, REAL(3)]
   The real parts of roots to the cubic equation: The roots are returned here. If IGS is 1, an initial approximation to start the Newton iteration is input as U(IT(1)).
8. IER [Out, INTEGER scalar]
   Success/error code: 0 for a successful solution; 1 if the iterative scheme to find the first root did not converge to the tolerance within MXIT iterations.

8.16.2 COMMON DATA

COMMON /CPLINT/ MXIT

MXIT is the upper limit on the number of iterations used to find the first root of the cubic polynomial.

8.17 SUBROUTINE CURDIF (from SUBS)

Computes derivatives along a curve, according to parametric cubic tension spline interpolation

8.17.1 PARAMETER LIST

1. XYZ [In, REAL(3,NPT)]
   Point coordinates at each point
2. DERIV [Out, REAL(3,NPT)]
   Parametric derivatives at each point: These are the tangent vectors.
3. GNU [In, REAL(NPT)]
   Tension value at each point
4. TT [In, REAL(NPT)]
   Parameter value at each point
5. NPT [In, INTEGER scalar]
   Number of points: This can be from 1 to 175.
6. IBDY [In, INTEGER scalar]
   End-condition flag: 0 indicates natural (called "unknown" in the user's manual) conditions at each end; 1 indicates a specified derivative at the initial end and a natural condition at the final end; 2 indicates a natural condition at the initial end and a specified derivative at the final end; 3 indicates specified derivatives at both ends; 4 indicates a periodic curve.
7. VLEFT [In, REAL(3)]
   Input derivative value at the inital end: This is used only when IBDY is 1 or 3.
8. VRIGHT [In, REAL(3)]
   Input derivative value at the final end: This is used only when IBDY is 2 or 3.
9. IERR [Out, INTEGER scalar]
   Error code: 0 is returned from a successful call; 4 is returned if IBDY is not a proper value; 8 is returned if NPT is too large.

8.17.2 COMMON DATA

none

8.18 SUBROUTINE CURPAR (from SUBS)

Calculates parameter values along a curve: The parametrization is based upon chord length.

8.18.1 PARAMETER LIST

1. XYZ [In, REAL(3,N)]
   Coordinates at each point
2. CYL [In, LOGICAL scalar]
   Flag for cylindrical coordinates
3. IXAXI [In, INTEGER scalar]
   Index for the axial cylindrical coordinate
4. IXRAD [In, INTEGER scalar]
   Index for the radial cylindrical coordinate
5. IXANG [In, INTEGER scalar]
   Index for the angular cylindrical coordinate
6. PVAL [Out, REAL(N)]
   Parameter values: This is a monotonic list. An unnormalized parametrization has values from 0 to TLGTH. A normalized one has values from 0 to 1.
7. TLGTH [Out, REAL scalar]
   Total chord length along the curve
8. N [In, INTEGER scalar]
   Number of points: This must be at least 2.
9. NORM [In, LOGICAL scalar]
   Flag to return a normalized parametrization

8.18.2 COMMON DATA

none

8.19 SUBROUTINE CURSLP (from SUBS)

An interface to the spline-fit subroutine CURDIF: This scales end conditions before calling CURDIF and copies the slopes at knots after calling.

8.19.1 PARAMETER LIST

1. NPTS [In, INTEGER scalar]
   Number of points
2. POINTS [In, REAL(3,NPTS)]
   Coordinate values
3. NKNOTS [In, INTEGER scalar]
   Number of points that are knots
4. IKNOTS [In, INTEGER(NKNOTS)]
   Point indices for the knots
5. TENS [In, REAL(NPTS)]
   Tension values
6. IENDF [In, INTEGER scalar]
   End-condition flag: 0 indicates natural (called "unknown" in the user's manual) conditions at each end; 1 indicates a specified derivative at the initial end and a natural condition at the final end; 2 indicates a natural condition at the initial end and a specified derivative at the final end; 3 indicates specified derivatives at both ends; 4 indicates a periodic curve.

7. ENDD [In, REAL(3,2)]
   ENDD(*,1) is the tangent direction at the initial end: This is used only when IENDF is 1 or 3.
   ENDD(*,2) is the tangent direction at the final end: This is used only when IENDF is 2 or 3.
8. PARM [In, REAL(NPTS)]
   Parameter values
9. CHORD [In, REAL scalar]
   Total chord length along the curve
10. CYL [In, LOGICAL scalar]
    Flag for cylindrical coordinates
11. IXAXI [In, INTEGER scalar]
    Index for the axial cylindrical coordinate
12. IXRAD [In, INTEGER scalar]
    Index for the radial cylindrical coordinate
13. IXANG [In, INTEGER scalar]
    Index for the angular cylindrical coordinate
14. SLOPE [Out, REAL(3,NKNOTS)]
    Tangent values: These are computed by a parametric cubic spline fit.

8.19.2 COMMON DATA

none

8.20 SUBROUTINE CYCBEN (from LIBS)

Within Subroutine PLINT, determines whether a double root of the bicubic exists at a given S value: This is used to locate the extreme S values for a cycle, which is a loop-shaped zero curve.

8.20.1 PARAMETER LIST

1. TR [In, REAL)3,2)]
   Array containing the roots of the polynomial in T obtained by evaluating the bicubic at the current S value.
2. TOL [In, REAL scalar]
   The user-specified tolerance
3. HBND [In, REAL(2)]
   Horizontal (T) boundaries to the region of interest
4. ICY [Out, INTEGER scalar]
   Double-root code: This is 1 if a double root exists; otherwise it is 0.
5. IT [In/Out, INTEGER (3)]
   Array containing the subscripts of the roots of a cubic equation: U(IT(1)) is the real root found via a Newton iterative scheme; U(IT(2)) and U(IT(3)) are the roots found from the quadratic factor.
6. IR1 [Out, INTEGER scalar]
   Lower limit for a range of curves used by subroutine PROCUV when initializing for tracing
7. IR2 [Out, INTEGER scalar]
   Upper limit for a range of curves used by subroutine PROCUV when initializing for tracing

8.20.2 COMMON DATA

none

8.21 SUBROUTINE DEGNER (from LIBS)

Within subroutine PLINT, determines the general degree of the bicubic in a vertically-bounded region and those S values where the bicubic is degenerate

8.21.1 PARAMETER LIST

1. TOL [In, REAL(2)]
   Array containing the user-specified tolerance: TOL(1) is the input tolerance. TOL(2) is computed by PLINT and is used as a lower limit for the stepsize.
2. VBND [In, REAL(2)]
   Vertical (S) boundaries to the region of interest
3 IRT [In, INTEGER(3)]
   Array containing the subscripts of the roots of a cubic equation: U(IRT(1)) is the real root found via a Newton iterative scheme; U(IRT(2)) and U(IRT(3)) are the roots found from the quadratic factor.
4. NDG [Out, INTEGER scalar]
   Number of degenerate S values
5. DGBND [Out, REAL(3)]
   Array containing S values where the polynomial in T is degenerate
6. IDGS [Out, INTEGER(3)]
   Array containing the degree of degeneracy of the bicubic at the corresponding DGBND values
7. NDEG [Out, INTEGER scalar]
   The maximum degree of the polynomial in T obtained by evaluating the bicubic at an arbitrary S value
8. IER [In/Out, INTEGER scalar]
   Success/error code: The only possible error code is a 1, which is passed along from Subroutine CUBIC.

8.21.2 COMMON DATA

COMMON /IRP1/ BICOF(4,4)

The bicubic polynomial is input from BICOF.

8.22 SUBROUTINE DEGTST (from LIBS)

Within Subroutine PLINT, stores any super-degenerate curves and determines the U-values at which double roots exist in the region of interest

8.22.1 PARAMETER LIST

1. BC [In, REAL(4,4)]
   Coefficients for the output bicubic function, in algebraic form: BC(I,J) is the coefficient of $U^{(4-J)} * V^{(4-I)}$.
2. VBND [In, REAL(2)]
   Vertical (S) boundaries to the region of interest
3. HBND [In, REAL(2)]
   Horizontal (T) boundaries to the region of interest
4. TOL [In, REAL(2)]
   Array containing the user-specified tolerance: TOL(1) is the input tolerance. TOL(2) is computed by PLINT and is used as a lower limit for the stepsize.
5. ID [In, INTEGER(2)]
   Array specifying the renaming of U and V as S and T. ID(1) is the index value for S in the ordered pair (U,V), and ID(2) is the index value for T. (e.g., If ID(1) is 2, then S is the name for V within the planar intersector.)
6. IRT [In, INTEGER(3)]
   Array containing the subscripts of the roots of a cubic equation: U(IRT(1)) is the real root found via a Newton iterative scheme; U(IRT(2)) and U(IRT(3)) are the roots found from the quadratic factor.
7. TCT [Out, REAL(13)]
   Array containing S values at which the polynomial in T has a double root. (Subroutine BRINT uses 15, rather than 13, as the limit for the number of double roots.)
8. NC [Out, INTEGER scalar]
   The number of S values at which the polynomial in T has a double root

9. NCI [Out, INTEGER scalar]
   The number of S values at which the polynomial in T has a double root between the horizontal boundaries
10. NDG [Out, INTEGER scalar]
    Number of degenerate S values
11. NDEG [Out, INTEGER scalar]
    The maximum degree of the polynomial in T obtained by evaluating the bicubic at an arbitrary S value
15. MXPT [In, INTEGER scalar]
    Upper limit on the number of points desired on a single output curve: If a particular curve generates more than MXPT points, the subroutine will stop processing that curve and the remaining points will be lost. MXPT must be at least 2. (This feature guarantees that each intersection curve will not exceed the space allocated for it.)
16. MXCRV [In, INTEGER scalar]
    Upper limit on the number of curves generated from a single call: If a call generates more than MXCRV curves, the subroutine will finish processing any curves and will return only MXCRV curves. (This feature guarantees that the intersection curves will not exceed the space allocated for them.) MXCRV must be positive and at most 12.
17. NCRV [In/Out, INTEGER scalar]
    The current number of curves generated
18. CURVE [In/Out, REAL(2,MXPT,MXCRV)]
    Array containing (U,V) pairs for points on the patch, along intersection curves. CURVE(1,1,K) is the number of points in the K-th curve; CURVE(*,J+1,K) is the J-th (U,V) pair on this curve.
19. IER [In/Out, INTEGER scalar]
    Success/error DEGTST can generate error codes -5, -6, and 4. It also returns error codes from utility Subroutine HSZERO (whose error codes are shifted in value by Subroutine BRINT), Subroutine BRINT, and Subroutine DEGNER.

    -5 is returned when the patch is recognized as having more than the theoretical limit of three degenerate U values.

    -6 is returned when the patch is recognized as having more than the theoretical limit of twelve U values where V has a double root. (This code can also come from Subroutine BRINT, which allows a limit of 15 (not 12) double roots.)

    4 is returned when the number of intersection curves exceeds MXCRV. (This error code is detected when storing super-degenerate curves.)

8.22.2 COMMON DATA

COMMON /IRP1/ BCT(4,4)

The bicubic polynomial is input from BCT.

8.23 SUBROUTINE DETLOC (from LIBS)

Called by Subroutine SUFINT, trims a plane/patch intersection slice to just those points which are in a triangular region of the plane

8.23.1 PARAMETER LIST

1. TOL [In, REAL scalar]
   Tolerance value
2. LN [Out, REAL(3,4)]
3. NORM [In, REAL(4,3)]
4. TEMP1 [In/Out, REAL(3)]
   A point to be tested

5. IRD [In, INTEGER(3,2)]
6. NPT [In, INTEGER scalar]
   Number of points in CURVE
7. CURVE [In, REAL(2,NPT)]
8. BICOF [In, REAL(4,4,3)]
   Patch coefficients
9. IC [In, INTEGER(3)]
10. PAT [In, REAL(16,3)]
    Patch coefficients
11. PLN [In, REAL(4)]
    Plane coefficients
12. NJCRV [In/Out, INTEGER scalar]
13. MXJPT [In, INTEGER scalar]
14. MXJCRV [In, INTEGER scalar]
    Number of intersection slices to trim
15. NMPT [Out, INTEGER(MXJCRV)]
16. POINT [In/Out, REAL(3,MXJPT,MXJCRV)]
    Intersection slices: They are trimmed.
17. IER [Out, INTEGER scalar]
    Error code: This is zero for a normal exit. See Subroutine SUFINT for more details.

8.23.2 COMMON DATA

none

8.24 REAL FUNCTION DISVAL (from LIBS)

Within Subroutine PLINT, computes the resultant for the bicubic: This is a polynomial in S of degree 15 whose zeros correspond to the degeneracies of the bicubic function.

8.24.1 PARAMETER LIST

1. X [In, REAL scalar]
   S value

8.24.2 COMMON DATA

COMMON /IRP1/ A

[Not used]

8.25 REAL FUNCTION DOTPRD (from LIBS)

Computes the scalar product of a pair of vectors and subtracts a constant

8.25.1 PARAMETER LIST

1. TOL [In, REAL scalar]
   Tolerance: If the magnitude of the result is less than this value, it is set exactly to zero.
2. VEC1 [In, REAL(3)]
   First vector
3. VEC2 [In, REAL(4)]
   VEC2(1), VEC2(2), and VEC2(3) are the components of the second vector. VEC2(4) is the constant which is subtracted from the result.

8.25.2 COMMON DATA

none

8.26 SUBROUTINE ENRICH (from SUBS)

Interpolates points on a curve, with enriched density

8.26.1 PARAMETER LIST

1. NSSLP [In, INTEGER(17)]
   End condition codes for each curve
2. DXI [In, REAL(17)]
   Initial tangent X-components for each curve
3. DYI [In, REAL(17)]
   Initial tangent Y-components for each curve
4. DZI [In, REAL(17)]
   Initial tangent Z-components for each curve
5. DXO [In, REAL(17)]
   Final tangent X-components for each curve
6. DYO [In, REAL(17)]
   Final tangent Y-components for each curve
7. DZO [In, REAL(17)]
   Final tangent Z-components for each curve
8. X [In/Out, REAL(17,175)]
   Position X-components for each point on each curve
9. Y [In/Out, REAL(17,175)]
   Position Y-components for each point on each curve
10. Z [In/Out, REAL(17,175)]
    Position Z-components for each point on each curve
11. CYL [In, LOGICAL scalar]
    Flag for cylindrical coordinates
12. NPMAX [In, INTEGER scalar]
    Upper limit on curve length
13. IXAXI [In, INTEGER scalar]
    Index for axial cylindrical coordinate
14. IXRAD [In, INTEGER scalar]
    Index for radial cylindrical coordinate
15. IXANG [In, INTEGER scalar]
    Index for angular cylindrical coordinate

8.26.2 COMMON DATA

COMMON /MODSTF2/ NCURV, NDIM, NDIMC, N1, N2, IFAIL
COMMON /MODSTF3/ LEN(17)

LEN [In/Out REAL(17)] is the number of points on each curve.

8.27 SUBROUTINE ERRCHK (from SUBS)

Evaluate the spatial errors along a curve between the original point spacing and a revised spacing: The distance between each interior point from the original spacing and a point interpolated at the same parameter value from the revised spacing is written to unit 6. (No data values are returned from this subroutine.)

8.27.1 PARAMETER LIST

1. XYZ [In, REAL(3,*)]
   Coordinate values at the original points
2. XYZK [In, REAL(3,*)]
   Coordinate values at the revised points

3. END [In, REAL(*,*)]
   End tangent values
4. CYL [In, LOGICAL scalar]
   Flag for cylindrical coordinates
5. IX [In, INTEGER scalar]
   Index for the radial cylindrical coordinate
6. IR [In, INTEGER scalar]
   Index for the radial cylindrical coordinate
7. IA [In, INTEGER scalar]
   Index for the angular cylindrical coordinate
8. NP [In, INTEGER scalar]
   Number of interior points for the revised spacing: This does not count the 2 endpoints.
9. NPTS [In, INTEGER scalar]
   Number of points for the original spacing: The 2 endpoints are counted.
10. NSSLP [In, INTEGER scalar]
    End condition code
11. PVALO [In, REAL(*)]
    Parameter values at the interior points from the initial spacing

8.27.2 COMMON DATA

none

8.28 SUBROUTINE FINCUV (from LIBS)

Called by Subroutine SUFINT, to finish connecting plane/patch intersection slices to form patch/patch intersection branches

8.28.1 PARAMETER LIST

1. TOL [In, REAL scalar]
   Tolerance value
2. NCRV [In/Out, INTEGER scalar]
   Number of intersection branches: These are formed by connecting the slices.
3. MXPT [In, INTEGER scalar]
   Maximum number of points to be calculated for an intersection branch
4. MXCRV [In, INTEGER scalar]
   Maximum number of intersection branches to be calculated
5. NMPT [Out, INTEGER(*)]
   Count of matches, for each intersection branch
6. CURVE [In/Out, REAL(*)]
   Connected intersection branches
7. IER [Out, INTEGER scalar]
   Error code: This is zero for a normal exit. See Subroutine SUFINT for more details.

8.28.2 COMMON DATA

none

8.29 SUBROUTINE FSORT (from LIBS)

Sorts in place the elements of a one-dimensional real array in ascending algebraic order: The Shell algorithm is used.

8.29.1 PARAMETER LIST

1. FA [In/Out, REAL(*)]
   The array to be sorted
2. N [In, INTEGER scalar]
   The number of elements in FA

8.29.2 COMMON DATA

none

8.30 SUBROUTINE GETMSH (from SUBS)

Reads coordinate value sets from MSH format data

8.30.1 PARAMETER LIST

1. MSH [In, INTEGER scalar]
   Logical unit for mesh input
2. OUTPUT [In, INTEGER scalar]
   Logical unit for printer output
3. NCLABM [In, INTEGER(3)]
   Number of characters in mesh coordinate labels
4. LABELM [In, INTEGER(3)]
   Mesh coordinate labels: This is character data.
5. MAXMSH [In, INTEGER scalar]
   Number of spaces allocated for mesh coordinate values
6. NMSH [Out, INTEGER(3)]
   Number of mesh coordinate values for each component
7. XMSH [Out, REAL(MAXMSH,3)]
   Mesh coordinate values
8. QERR [Out, LOGICAL scalar]
   Error flag

8.30.2 COMMON DATA

none

8.31 SUBROUTINE GETPAT (from SUBS)

Reads patches from SRF format data: The patches are stored on a sequential-access binary file for later use.

8.31.1 PARAMETER LIST

1. SRF [In, INTEGER scalar]
   Logical unit for patch input
2. OUTPUT [In, INTEGER scalar]
   Logical unit for printer output
3. PATMS [In, INTEGER scalar]
   Logical unit for patch mass storage
4. NPAT [Out, INTEGER scalar]
   Number of patches read
5. QERR [Out, LOGICAL scalar]
   Error flag

8.31.2 COMMON DATA

none

8.32 SUBROUTINE HGERR (from LIBS)

Handles error exits from some utility routines. (The source code to this routine is proprietary to Boeing Computer Services.)

8.32.1 PARAMETER LIST

1. MODE [In, INTEGER scalar]
   Code for the mode of operation: (-4) to restore the default I/O unit value to IEUNIT; (-3) to set IEFLAG, the countdown limit for printing messages; (-2) to set IEUNIT to a non-default I/O unit; (-1) for a null call to force initialization of the COMMON variables; (0) to print a warning message; (1) to print an input error message; (2) to print a working-storage error message; (3) to print a processing error message; (4) to print a traceback line; and (5) to print a traceback line and state that the error was unexpected.
2. SUBNAM [In, REAL(2)]
   Subroutine name: This is character data.
3. IER [In, INTEGER scalar]
   Error code from the calling routine
4. N [In, INTEGER scalar]
   Various usages, depending on the value of MODE: If MODE = -3, N is the countdown value to be input. If MODE = -2, N is the I/O unit value to be input. If MODE = 2, N is the number of additional real words of working storage required by the calling routine. For all other values of MODE, N is ignored.

8.32.2 COMMON DATA

COMMON /HGERRC/ IEUNIT, IEFLAG, NERRS, IXPAND(3)

IEUNIT is the logical I/O unit for error message output, which is specified by the function value JHMCON(4). IEFLAG is the number of messages to be printed. The call which reduces IEFLAG to 0 will only print if a traceback message is requested; calls after this number reaches 0 do not print. NERRS contains the total number of calls to HGERR. IXPAND reserves space in the COMMON block for future expansion.

8.33 REAL FUNCTION HSMCON (from LIBS)

Hardware-dependent REAL constants (The source code for this routine is proprietary to Boeing Computer Services):

1. SCLOBR
   Clobber constant: A number used when errors are detected to abort continued computations. This is -INDEFINITE in this case.
2. SRANGE
   Symmetric range: The largest positive X such that X, -X, 1/X, and -1/X are floating-point numbers. This is approximately .319 * 10^{294} in this case.
3. SOVFLO
   Overflow threshold: The largest positive X such that X and -X are floating-point numbers. This is approximately .127 * 10^{323} in this case.
4. SUNFLO
   Underflow threshold: The smallest positive X such that X and -X are floating-point numbers. This is approximately .313 * 10^{293} in this case.

5. SRELSP
   Relative spacing: The maximum relative spacing between adjacent floating-point numbers; this is $B^{(1-N)}$ where B is the base, HSMCON(7), and N is the mantissa length, HSMCON(8). This is the conservative definition of "machine epsilon." This is approximately .711 * 10^{-14} in this case.
6. SRELPR
   Relative precision: The smallest positive floating-point X such that 1-X and 1+X are distinct from 1. This is the IFIP definition of "machine epsilon." This is approximately .355 * 10^{-14} in this case.
7. SRADIX
   Radix: The base of the floating-point number system. This is 2 in this case.
8. SDIGIT
   Mantissa length: The number of base-RADIX digits in the mantissa of a floating point number. (If an implicit or hidden digit is present, it is included.) This is 48 in this case.
9. SEOVFL
   Exponential overflow: The largest positive floating-point integer X such that e^X is a floating-point number without overflow. This is 741 in this case.
10. SEUNFL
    Exponential underflow: The negative integer X with the largest magnitude such that e^X is a floating-point number without underflow. This is -675 in this case.
11. SMXINT
    Maximum floatable integer: The largest positive floating-point integer such that it, and all smaller integers can be truncated correctly to INTEGER and converted back to the same floating-point number. This is approximately .281 * 10^{15} in this case.
12. SPI
    PI: Approximately 3.14159
13. SEXPE
    The base of natural logarithms: Approximately 2.71828
14. SEULER
    Euler's constant: Approximately .57721
15. SRADEG
    Number of radians in 1 degree (PI/180): Approximately .01745
16. SDEGRA
    Number of degrees in 1 radial (180/PI): Approximately 57.295

8.33.1 PARAMETER LIST

1. I [In, INTEGER scalar]
   Index to the desired constant: This can range from 1 to 16.

8.33.2 COMMON DATA

none

8.34 SUBROUTINE HSZERO (from LIBS)

(The source code for this routine is proprietary to Boeing Computer Services) Finds one zero of an arbitrary real-valued function of a real variable, in a specified interval. The function values at the two ends of the interval must be of opposite signs. Either the secant rule or inverse quadratic interpolation is used.

8.34.1 PARAMETER LIST

1. AX [In, REAL scalar]
   Left endpoint of the specified interval
2. BX [In, REAL scalar]
   Right endpoint of the specified interval: This must be greater than AX.
3. TOL [In, REAL scalar]
   Tolerance value: TOL must be positive. The solution, ZEROIN, will be given within TOL + 2*EPS*ABS(ZEROIN) of its exact value, where EPS is the value of HSMCON(5).

4. F [User-supplied Routine]
   A REAL FUNCTION subprogram which calculates F(X).
5. ZEROIN [Out, REAL scalar]
   The desired zero
6. IER [Out, INTEGER scalar]
   Success/error code: 0 for a successful call; 1 when AX is not less than BX; 2 when TOL is not positive; 3 when F(AX) and F(BX) have the same sign.

8.34.2 COMMON DATA

none

8.35 SUBROUTINE INTERP (from SUBS)

Interpolates the locations of a new point on a curve, from its parameter value in the interval between the surrounding old points

8.35.1 PARAMETER LIST

1. XYZ [In, REAL(3,175)]
   Positions of old points
2. SLOPE [In, REAL(3,175)]
   Tangent vectors at old points
3. LOD [In, INTEGER scalar]
   Interval to be used for interpolation: This is the index for the inital old point in the interval
4. XYZK [Out, REAL(3)]
   Positions at new point
5. PDIF [In, REAL scalar]
   Parameter value locating the new point in the selected interval
6. PSCALE [In, REAL scalar]
   Scale factor for tangents at old points: This is the curve length for which tangent values were computed, divided by the length of the selected interval.

8.35.2 COMMON DATA

none

8.36 INTEGER FUNCTION JHMCON (from LIBS)

Machine-dependent INTEGER constants (The source code for this routine is proprietary to Boeing Computer Services):

1. ICLOBR
   Clobber constant: A number used when errors are detected to abort continued computations. This is approximately .576 * 10^{18} in this case.
2. IRANGE
   Symmetric range: The largest integer I such that a full arithmetic capability is supported for all integers in [-I,I]. This is approximately .281 * 10^{15} in this case.
3. IOVFLO
   Overflow threshold: The largest integer I such that all integers in [-I,I] are machine integers. This is approximately .576 * 10^{18} in this case.
4. NERR
   Default error message unit: The logical I/O unit number for standard error-message output. This is the left-justified Hollerith value 6LOUTPUT in this case. (It is conventionally the same as the default output unit.)
5. NIN
   Default input unit: The logical I/O unit number that is preconnected to the standard input unit of the system. This is the left-justified Hollerith value 5LINPUT in this case.

6. NOUT
   Default output unit: The logical I/O unit number that is preconnected to the standard output unit of the system. This is the left-justified Hollerith value 6LOUTPUT in this case.
7. NCNIN
   Number of input characters: The default number of characters, including blanks, which can be read from a single record on the default input unit. This is 80 in this case.
8. NCNOUT
   Number of output characters: The default number of characters, including blanks and carriage control, which can be output to a single record on the standard output unit of the system. This is 133 in this case.
9. NIDEC
   Number of decimal digits for INTEGER: The largest number of decimal digits always allowed and converted correctly by the compiler when compiling INTEGER constants. This is 17 in this case.
10. NSDEC
    Number of decimal digits for REAL: The largest number of decimal digits always allowed and converted correctly by the compiler when compiling REAL constants. This is 15 in this case.
11. NDDEC
    Number of decimal digits for DOUBLE PRECISION: The largest number of decimal digits always allowed and converted correctly by the compiler when compiling DOUBLE PRECISION constants. This is 29 in this case.
12. NIPAGE
    Page size: The size of a page, in INTEGER storage units, on virtual machines. On non-paging machines, as in this case, it has the value 1.
13. NCHAR
    Characters in INTEGER: The maximum number of characters that will fit into one INTEGER storage unit. This is 10 in this case.
14. NSISU
    Size of a REAL word: The maximum number of INTEGER storage units that will fit on one REAL storage unit. This is 1 in this case.
15. NDISU
    Size of a DOUBLE PRECISION word: The maximum number of INTEGER storage units that will fit into one DOUBLE PRECISION storage unit. This is 2 in this case.

8.36.1 PARAMETER LIST

1. I [In, INTEGER scalar]
   Index to the desired constant: This can range from 1 to 15.

8.36.2 COMMON DATA

none

8.37 SUBROUTINE KN1CHK (from LIBS)

Within Program REGSIL, checks to see whether a set of knots interlaces with the input data

8.37.1 PARAMETER LIST

1. X [In, REAL(101)]
   Knot values: They are monotonically increasing.
2. NX [In, INTEGER scalar]
   Number of knot values
3. K [In, INTEGER scalar]
   The order of the splines for which interlacing conditions are being tested: This is the polynomial degree, plus 1.
4. T [In, REAL(101)]
   Input data values: They are monotonically increasing.

5. NT [In, INTEGER scalar]
   Number of input data values
6. IFAIL [Out, INTEGER scalar]
   Result code: 0 if interlacing requirements are satisfied; 10000 + I if interlacing fails at knot I; -1 if T(1) is less than X(1); -2 if T(NT) is greater than X(NX); -10000 - I if the multiplicity of the knot at X(I) exceeds K.

8.37.2 COMMON DATA

none

8.38 COMPASS INTEGER FUNCTION KOMSTR (from LIBS)

Compares two strings: A value of -1, 0, or +1 is returned when the first string respectively is less than, equal to, or greater than the second.

8.38.1 PARAMETER LIST

1. SA [In, Array]
   String 1
2. LA [In, INTEGER scalar]
   Location of the first character in string SA: This is 1 for the leftmost character position in the array.
3. NA [In, INTEGER scalar]
   The number of characters in string SA and in string SB
4. SB [In, Array]
   String 2
5. LB [In, INTEGER scalar]
   Location of the first character in string SB: This is 1 for the leftmost chracter position in the array.

8.38.2 COMMON DATA

none

8.39 COMPASS INTEGER FUNCTION LSTRNG (from LIBS)

Locates the first occurance of a character string as a substring of another string: If found, the location of the inital character of the substring is returned; 0 is returned if no match is found; -1 is returned if an error in the inputs was detected.

8.39.1 PARAMETER LIST

1. S1 [In, Array]
   The string to be searched for
2. I1 [In, INTEGER scalar]
   Location of the first character in string S1: This is 1 for the leftmost character position in the array.
3. N1 [In, INTEGER scalar]
   The length of string S1
4. S2 [In, Array]
   The string being searched for a substring
5. I2 [In, INTEGER scalar]
   Location of the first character in string S2: This is 1 for the leftmost character in the array.
6. N2 [In, INTEGER scalar]
   The length of S2

8.39.2 COMMON DATA

none

8.40 SUBROUTINE MEMPAT (from SUBS)

Given a set of section specifications, writes member and patch specifications to complete a SIL file for a regular region

8.40.1 PARAMETER LIST

1. NSECT [In, INTEGER scalar]
   Number of section curves
2. NMEM [In, INTEGER scalar]
   Number of member curves: This equals the number of knots on each of the existing sections
3. CYL [In, LOGICAL scalar]
   Flag for cylindrical coordinates

8.40.2 COMMON DATA

none

8.41 SUBROUTINE MODEL (from LIBS)

Within program REGSIL, computes the error for a trial knot placement.

8.41.1 PARAMETER LIST

1. PKNOT [In, REAL(*)]
   Parameter values to place knots
2. NP [In, INTEGER scalar]
   Number of PKNOT values
3. ERRMAX [Out, REAL scalar]
   Maximum position error between the original points and the corresponding interpolated points on the new curve, for each curve.
4. IER [Out, INTEGER scalar]
   Error code: 0 for success; -4 if some curve has less points than NP; -5 for an error return from CBSPN; -6 for an error return from SQRSL; -7 for a failure while checking knots; -8 if NP is not positive ; and 1000+I for an interlacing failure, which is a local deficiency in the number of input points.

8.41.2 COMMON DATA

```
COMMON  /CNTRL/  IL,  EPS,  DELX,  MODE,  MAXFN,  KF,  KORD
COMMON  /MODSTF1/  CRV(800,4),  WORK(9000)
COMMON  /MODSTF2/  NCURV,  NDIM,  NDIMC,  NPRT1,  NPRT2,  IFAIL
COMMON  /MODSTF3/  LEN(17)
```

Common block /CNTRL/ primarily controls the optimization routine NWO21. IL is the location parameter used by subroutine CBSPN. EPS is the error tolerance used by NWO21. DELX is the step size used by NWO21 for numerical differentation. MODE is used by NWO21. MAXFN is the limit on the number of error function evalutations to be made by NWO21. KF controls the listing output from NWO21. KORD is the order of the splines used; it is 4, giving cubic splines.

CRV stores the coordiates of the input points and the chord lengths along the curves. WORK is used as temporary storage. NCURV is the number of curves. NDIM is the number of spatial dimensions; it is set to 3. NDIMC equals the length of CRV. LEN is the number of input points for each curve.

8.42 SUBROUTINE MSHOPT (from SUBS)

Reads option declarations from a file of MSH data for use by programs MSHNRM or NRMCFD

8.42.1 PARAMETER LIST

1. MSH [In, INTEGER scalar]
   Logical unit for MSH-format input
2. OUTPUT [In, INTEGER scalar]
   Logical unit for listing output
3. CYLSRF [In/Out, LOGICAL scalar]
   Flag for cylindrical surface coordinates: This is FALSE upon entry, but it can be set TRUE before exit.
4. CYLMSH [In/Out, LOGICAL scalar]
   Flag for cylindrical mesh coordinates: This is FALSE upon entry, but it can be set TRUE before exit.
5. LSTOPT [In/Out, INTEGER scalar]
   Level of MSHNRM listing output selected: This is 1 upon entry, but it can be set to any value before exit.
6. IXAXIM [In/Out, INTEGER scalar]
   Index for the axial cylindrical coordinate: This is 1 upon entry, but it can be set to 2 or 3 before exit. (IXAXIM, IXRADM, and IXANGM must be a permutation of the set 1, 2, and 3.)
7. IXRADM [In/Out, INTEGER scalar]
   Index for the radial cylindrical coordinate: This is 2 upon entry, but it can be set to 1 or 3 before exit. (IXAXIM, IXRADM, and IXANGM must be a permutation of the set 1, 2, and 3.)
8. IXANGM [In/Out, INTEGER scalar]
   Index for the angular cylindrical coordinate: This is 3 upon entry, but it can be set to 1 or 2 before exit. (IXAXIM, IXRADM, and IXANGM must be a permutation of the set 1, 2, and 3.)
9. IXSAXI [In/Out, INTEGER scalar]
   Index for the rectangular axial coordinate: This is 1 upon entry, but it can be set to 2 or 3 before exit. (IXSAXI, IXSSIN, and IXSCOS must be a permutation of the set 1, 2, and 3.)
10. IXSSIN [In/Out, INTEGER scalar]
    Index for the rectangular coordinate whose axis points in the THETA = 90 degrees (horizontal) direction: This is 1 upon entry, but it can be set to 2 or 3 before exit. (IXSAXI, IXSSIN, and IXSCOS must be a permutation of the set 1, 2, and 3.)
11. IXSCOS [In/Out, INTEGER scalar]
    Index for the rectangular coordinate whose axis points in the THETA = 0 degrees (vertical) direction: This is 1 upon entry, but it can be set to 2 or 3 before exit. (IXSAXI, IXSSIN, and IXSCOS must be a permutation of the set 1, 2, and 3.)
12. LABELR [In/Out, INTEGER(3)]
    Labels for rectangular coordinates: This is character data. The default labels are input, but the labels can be changed before exit.
13. NCLABR [In/Out, INTEGER(3)]
    The number of characters in the rectangular-coordinate labels: This can range from 1 to 10.
14. LABELC [In/Out, INTEGER(3)]
    Labels for cylindrical coordinates: This is character data. The default labels are input, but the labels can be changed before exit.
15. NCLABC [In/Out, INTEGER(3)]
    The number of characters in the cylindrical-coordinate labels: This can range from 1 to 10.
16. LABELX [Out, INTEGER(3)]
    Labels for surface coordinates: This is character data. It is either the labels for cylindrical or rectangular coordinates.
17. NCLABX [Out, INTEGER(3)]
    The number of characters in the surface coordinate labels: This can range from 1 to 10.
18. LABELM [Out, INTEGER(3)]
    Labels for mesh coordinates: This is character data. It is either the labels for cylindrical or rectangular coordinates.
19. NCLABM [Out, INTEGER(3)]
    The number of characters in the mesh coordinate labels: This can range from 1 to 10.
20. NXINT [In/Out, INTEGER scalar]
    The number of intersection orientations for MSHNRM mesh/surface intersection: A value of 0 is input, but this can be changed before exit. If the value remains 0, the default values of NXINT, KXINT, NXCUT, and KXCUT for the choice of mesh coordinates is copied into these variables. The default output value is 2. Values of 1 or 3 are allowed.

21. KXINT [Out, INTEGER(3)]
MSHNRM intersection orientations: This is the index for the coordinate held constant. Either default values or new values input from file MSH are returned. Default values for cylindrical mesh coordinates define KXINT(1) = IXANGM and KXINT(2) = IXAXIM; default values for rectangular mesh mesh coordiates define KXINT(1) = IXSAXI and KXINT(2) = IXSSIN.
22. NXCUT [Out, INTEGER(3)]
The number of cut orientations for each MSHNRM intersection orientation in mesh/surface intersection: Either default values or new values input from file MSH are returned. These are listed by the intersection orientation, which is a coordinate index. Default values for cylindrical mesh coordinates define NXCUT(IXANGM) = 2 and NXCUT(IXAXIM) = 1; default values for rectangular mesh coordinates define NXCUT(IXSAXI) = 2 and NXCUT(IXSSIN) = 1. Possible values for NXCUT(I) are 1 or 2 whenever I is declared as an intersection orientation.
23. KXCUT [Out, INTEGER(3,2)]
Cut orientations for each intersection MSHNRM orientation in mesh/surface intersection: These are index values for the coordinate to be held constant, and they are stored by the intersection orientation, which is a similar coordinate index. Either default values or new values input from file MSH are returned. Default values for cylindrical mesh coordinates are KXCUT(IXANGM,1) = IXAXIM AND KXCUT(IXANGM,2) = KXCUT(IXAXIM,1) = IXRADM; default values for rectangular mesh coordinates are KXCUT(IXSAXI,1) = IXSSIN and KXCUT(IXSAXI,2) = KXCUT(IXSSIN,1) = IXSCOS. Possible values for KXCUT(I,J) exclude I, provided I is declared as an intersection orientation with at least J cut orientations declared; they are from the set 1, 2, and 3.
24. IQSCAL [In/Out, INTEGER scalar]
Code to select a method to scale tangents for parametric cubic interpolation along MSHNRM intersection curves: The default value is present upon entry (2), but a different value can be present upon exit. Possible values are 1 (which gives locally explicit scaling) and 2 (which gives Ferguson-Phillips scaling, minimizing the integral of 2nd derivatives along the arc).
25. MXIT [In/Out, INTEGER scalar]
Limit on iterations for MSHNRM cubic-polynomial solution: The default value (50) is present upon entry, but a different value can be present upon exit.
26. TOLINT [In/Out, REAL]
Parametric tolerance for MSHNRM plane/patch intersection: The default value (.0001) is present upon entry, but a different value can be present upon exit.
27. TOLDIS [In/Out, REAL]
Distance tolerance for NRMCFD resolving distinct normal locations: The default value (.0005) is present upon entry, but a different value can be present upon exit. The value is in distance units, typical inches.
28. TOLANG [In/Out, REAL]
Angular tolerance for NRMCRD resolving distinct normal locations: The default value (.01) is present upon entry, but a different value can be present upon exit. The value is in units of degrees.
29 QERR [Out, LOGICAL scalar]
Error flag

8.42.2 COMMON DATA

none

8.43 COMPASS INTEGER FUNCTION NSCAN (from LIBS)

Scans a string of characters for the first occurance of a character that is not contained in a specified pool: If such a character is found, its position in the string is returned; otherwise 0 is returned; -1 is returned if an error is detected in the input.

8.43.1 PARAMETER LIST

1. S1 [In, Array]
The string to be scanned.

2. I1 [In, INTEGER scalar]
   Location of the first character in pool S1: This is 1 for the leftmost character position in the array.
3. N1 [In, INTEGER scalar]
   The length of string S1
4. S2 [In, Array]
   The pool of characters
5. I2 [In, INTEGER scalar]
   Location of the first character in pool S2: This is 1 for the leftmost character position in the array.
6. N2 [In, INTEGER scalar]
   The number of characters in pool S2

8.43.2 COMMON DATA

none

8.44 SUBROUTINE NW021 (from LIBS)

Finds a local minimum of a sum of squares of M nonlinear functions of N variables: $R1(X1,X2,...,XN)^2 + ... + RM(X1,X2,...,XN)^2$. It is adapted from a subroutine called VA07A. The method used is a combination of Gauss-Newton and steepest descent, called modified Marquardt.

8.44.1 PARAMETER LIST

1. RESID [User-Supplied Routine]
   This routine calculates the R-functions at a given value of the X-variables. It must act in the following way:

```
      SUBROUTINE RESID(M,N,X,R,IFL)
      INTEGER  M,N,IFL
      REAL  X(N),R(M)

C
      IFL  =  0
C
      Using the input values for X,
      compute values for R. If an error
      prevents this, set IFL to a
      nonzero value.
C
      RETURN
      END
```

2. LSQ [User-Supplied Routine]
   This routine calculates the coefficients for the least-squares normal equations at a given value of the X-variables. There are A and V coefficients. A(I,J) is required for all I from 1 to N and for all J from 1 to N. (A is symmetric.) It is defined by: A(I,J) = DR1/DXI * DR1/DXJ + ... + DRM/DXI * DRM/DXJ. V(I) is required for all I from 1 to N. It is defined by: V(I) = R1 * DR1/DXI + ... + RM * DRM/DXI. The routine must act in the following way:

```
      SUBROUTINE LSQ(M.N.X.R.A.V)
      INTEGER  M,N,IFL
      REAL  X(N),R(M),A(N,N),V(N)
C
      Using the input values for X,
      compute values for A and for V
      (This can be done by first
      computing the partial derivatives
      and then using the above formulas).
```

```
C
      RETURN
      END
```

3. M [In, INTEGER scalar]
   The number of dependent R-functions. M must be at least N; it can be at most 200.
4. N [In, INTEGER scalar]
   The number of independent X-variables. N must be at least 2; it can be at most 25.
5. X [In/Out, REAL(N)]
   The initial approximation to the desired minimum is input, and the final approximation is returned.
6. R [Out, REAL(M)]
   The values for the R-functions at the final approximation to the minimum is returned.
7. SS [Out, REAL scalar]
   The final approximation to the minimum value for the sum of squares is returned
8. A [Temporary, REAL(N,N)]
9. D [In/Out, REAL(N)]
   Nonnegative scaling factors: These values are input only if MODE is 3. If so, their values should approximate the square roots of the corresponding X-values at the expected minimum. (See the discussion of MODE below.)
10. EPS [In, REAL(N)]
    The absolute accuracies to which the X-variables are computed.
11. MAXFN [In, INTEGER scalar]
    Maximum number of calls to RESID before abandoning the problem: This must be positive
12. MODE [In, INTEGER scalar]
    Scaling code: This can be 1, 2, or 3. It controls the handling of the D-array, which is used to scale the A-matrix
    If MODE is 1, then the subroutine estimates values for the D-array. IF A(I,I) happens to be zero, D(I) is set to zero; otherwise, D(I) is set to A(I,I).
    If MODE is 2, then no scaling is used. The subroutine sets the D-array to ones.
    If MODE is 3, then the input D-array values are used.
13. KF [In/Out, INTEGER scalar]
    Input print-control code: Intermediate steps are listed to file OUTPUT, using PRINT statements. If KF is zero, no listing is generated; otherwise every KF-th iteration lists the following: the number of calls to RESID, SS, and X. If KF is positive, R is also listed.

    Output success/error code: 0 is returned from a successful call; 1 is returned when the MAXFN limit is reached; 2 is returned when N is input less than two; 3 is returned when M is input less than N; 4 is returned when MODE = 3 is input and some D element is not positive; 5 is returned when some EPS element input is not positive; 6 is returned when MAXFN input is not positive. (When codes 2 through 6 are returned, X is unchanged and no values are set for R and SS.); 7 is returned when the RESID subroutine returns a nonzero error code; 8 is returned when an element on the diagonal of the A-matrix is negative.

8.44.2 COMMON DATA

```
COMMON  /NWO2B/  S(25)
COMMON  /NWO2C/  T(25)
COMMON  /NWO2D/  U(25)
COMMON  /NWO2E/  V(25)
COMMON  /NWO2F/  W(200)
```

8.45 SUBROUTINE NWO93 (from LIBS)

Cholesky decomposition of a symmetric matrix and back-substitution solution.

8.45.1 PARAMETER LIST

1. A [In/Out, REAL(NR,*)]
   The input N-by-N symmetric matrix: This is replaced by the triangular factor from the decomposition.
2. NR [In, INTEGER scalar]
   Length of columns allocated for A.
3. N [In, INTEGER scalar]
   Matrix size
4. S [In/Out, REAL(*)]
   Input N-long right-hand-side vector: This is replaced by intermediate results from the substitution.
5. IER [Out, INTEGER scalar]
   Error flag: This is set to 0 for a successful call. It is set to 1 if a zero value is encountered on the diagonal during factorization; in this case the input matrix is not positive-definite.

8.45.2 COMMON DATA

none

8.46 SUBROUTINE OPENCV (from SUBS)

Used by Program SRFINT to connect an open curve: The endpoint table entries for the intersection branches are linked together.

8.46.1 PARAMETER LIST

1. L6 [In, INTEGER scalar]
   Logical unit for printer output
2. I [In, INTEGER scalar]
   Index of the beginning open endpoint in the table
3. NEND [In, INTEGER scalar]
   Number of endpoints in the table
4. ICON [In, INTEGER]
   Where each endpoint connects: This is a table of endpoint indices, whose subscripts are also endpoint indices.
5. IDONE [In/Out, INTEGER(201)]
   Flags to show whether each endpoint has been connected yet: Initally 0, this is set to 1 when a connection is made.
6. ICURVE [Out, INTEGER(201)]
   The connected list of endpoints making up the output curve.
7. NP [Out, INTEGER scalar]
   The number of endpoints in the output curve

8.46.2 COMMON DATA

COMMON /ENDTBL/ X(201), Y(201), Z(201), IP1(201), IP2(201), NBR(201), KOUNT(201), IPOINT(201)

8.47 SUBROUTINE ORDER (from SUBS)

Checks parameter values along a curve for range and duplicates

8.47.1 PARAMETER LIST

1. PKNOT [In/Out, REAL(*)]
   Parameter values: Duplicated values are removed.
2. NP [In/Out, REAL]
   Number of points: This is reduced to reflect the removed duplicates.
3. QERR [Out, LOGICAL scalar]
   Error flag

8.47.2 COMMON DATA

none

8.48 SUBROUTINE PARTAL (from LIBS)

Within Subroutine SUFINT, computes partial derivatives at a point on a patch.

8.48.1 PARAMETER LIST

1. BC [In, REAL(4,4)]
   Bicubic function, algebraic representation: BC(I,J) is the coefficient of $U^{(4-J)} * V^{(4-I)}$.
2. SVAL [In, REAL scalar]
   U value
3. TVAL [In, REAL scalar]
   V value
4. PATCH [Out, REAL(16)]
   Function values and derivatives

8.48.2 COMMON DATA

none

8.49 SUBROUTINE PARTA1 (from LIBS)

Within Subroutine REGSIL, computes partial derivatives for NW021 optimization.

8.49.1 Parameter List

1. M [In, INTEGER scalar]
2. NP [In, INTEGER scalar]
   Number of knots
3. P [In, INTEGER scalar]
   Trial knot placements
4. R [Out, REAL(*)]
   Residuals
5. A [Out, REAL(NP, *)]
6. V [Out, REAL(*)]

8.49.2 COMMON DATA

```
COMMON  /CNTRL/  IL, EPS, DELX, MODE, MAXFN, KF, KORD
COMMON  /MODSTF1/  CRV(800,4), WORK(9000)
COMMON  /MODSTF2/  NCURV, NDIM, NDIMC, NPRT1, NPRT2, IFAIL
COMMON  /MODSTF3/  LEN(17)
```

8.50 SUBROUTINE PATNRM (from SUBS)

Evaluates the position and surface normal at a point, along with their directions of change along a plane/patch intersection curve.

8.50.1 PARAMETER LIST

1. OUTPUT [In, INTEGER scalar]
   Logical unit for listing output
2. NCLABX [In, INTEGER(3)]
   Number of characters in coordinate labels
3. LABELX [In, INTEGER(3)]
   Coordinate labels: This is character data.
4. LSTOPT [In, INTEGER scalar]
   Code for the level of listing output
5. CYLSRF [In, INTEGER scalar]
   Flag for cylindrical surface coordinates
6. IXRAD [In, INTEGER scalar]
   Index for the radial cylindrical coordinate
7. INXANG [In, INTEGER scalar]
   Index for the angular cylindrical coordinate
8. ALG [In, REAL(16,3)]
   Patch coefficients, in algebraic format: Effectively allocated REAL(4,4,3). ALG (I,J,K) is the coefficient of $U^{(4-J)} * V^{(4-I)}$ in the bicubic function interpolating the K-th spatial component.
9. PLN [In, REAL(3)]
   Direction normal to the plane of intersection
10. U0 [In, REAL(2)]
    Parametric location of the desired point on the patch: This is a (U,V) pair.
11. X [Out, REAL(3)]
    Position
12. DXDT [Out, REAL(3)]
    Tangent along curve (i.e., the direction of position change along the curve)
13. SN [Out, REAL(3)]
    Surface normal direction: This is scaled to a unit length.
14. DSNDT [Out, REAL(3)]
    Direction of surface-normal change along the curve: This is scaled consistently with DXDT and SN.

7.50.2 COMMON DATA

none

8.51 SUBROUTINE PATVAL (from SUBS)

Computes position and parametric derivatives at a parametric location on a patch

8.51.1 PARAMETER LIST

1. OUTPUT [In, INTEGER scalar]
   Logical unit for listing output
2. NCLABX [In, INTEGER(3)]
   Number of characters in coordinate labels
3. LABELX [In, INTEGER(3)]
   Coordinate labels: This is character data.
4. LSTOPT [In, INTEGER scalar]
   Code for the level of listing output
5. CYLSRF [In, LOGICAL scalar]
   Flag for cylindrical surface coordinates

6. IXRAD [In, INTEGER scalar]
   Index for the radial cylindrical coordinate
7. IXANG [In, INTEGER scalar]
   Index for the angular cylindrical coordinate
8. ALG [In, REAL(4,4,3)]
   Patch coefficients, in algebraic format: ALG(I,J,K) is the coefficient of $U^{(4-J)} * V^{(4-I)}$ in the bicubic function interpolating the K-th spatial component.
9. UV [In, REAL(2)]
   Parametric location: This is a (U,V) pair.
10. X [Out, REAL(3)]
    Position vector
11 DXDU [Out, REAL(3)]
   First U-derivative vector
12. DXDV [Out, REAL(3)]
    First V-derivative vector
13. D2XDU2 [Out, REAL(3)]
    Second U-derivative vector
14. D2XDUV [Out, REAL(3)]
    Mixed second-derivative vector
15. D2XDV2 [Out, REAL(3)]
    Second V-derivative vector

8.51.2 COMMON DATA

none

8.52 SUBROUTINE PCGMAL (from SUBS)

Converts a parametric cubic curve segment from a geometric representation to an algebraic representation

8.52.1 PARAMETER LIST

1. B [In, REAL(4,3)]
   Geometric representation of PC curve: B(1,J) is the inital value of the J-th coordinate, B(2,J) is its final value, B(3,J) is its inital derivative, and B(4,J) is its final derivative.
2. A [Out, REAL(4,3)]
   Algebraic representation of PC curve: A(I,J) is the coefficient of $U^{(4-I)}$ in the cubic polynomial for the J-th coordinate.

8.52.2 COMMON DATA

none

8.53 SUBROUTINE PERSTO (from LIBS)

Cyclically permutes the storage of 3 values

8.53.1 PARAMETER LIST

1. ARRAY [In/Out, REAL(3)]
   The values to be permuted

8.53.2 COMMON DATA

none

8.54 SUBROUTINE PLINT (from LIBS)

Computes the intersections of a plane and a patch: These intersections are called curves here, in some contexts they are called intersection slices. This is done by finding the zero curves of a bicubic function.

8.54.1 PARAMETER LIST

1. PAT [In, REAL(16,3)]
   Geometric patch representation: Effectively allocated PAT(4,4,3). PAT(*,*,K) is the K-th component of a vector. PAT(1,1,*) is the (U,V) = (0,0) position; PAT(1,2,*) is the (1,0) position; PAT(2,1,*) is the (0,1) position; and PAT(2,2,*) is the (1,1) position. PAT(3,1,*), PAT(3,2,*), PAT(4,1,*), and PAT(4,2,*) are the corresponding V-derivatives. PAT(1,3,*), PAT(1,4,*), PAT(2,3,*), and PAT(2,4,*) are the corresponding U-derivatives. PAT(3,3,*), PAT(3,4,*), PAT(4,3,*), and PAT(4,4,*) are the corresponding cross-derivatives.
2. PLN [In, REAL(4)]
   Coefficients for the plane:
   $$PLN(1)*X(1) + PLN(2)*X(2) + PLN(3)*X(3) + PLN(4) = 0.$$
3. TOL [In/Out, REAL(2)]
   Array containing the user-specified tolerance: TOL(1) is the input tolerance. Any point on a parametric cubic curve interpolating the output (U,V) points is within TOL(1) distance (in parametric units) of a straight line between the surrounding output points. TOL(2) is computed by the subroutine and is used as a lower limit for the stepsize.
4. MXIT [In, INTEGER scalar]
   Upper limit on the number of iterations used to find a real root of a cubic polynomial via a Newton scheme: MXIT must be positive.
5. MXPT [In, INTEGER scalar]
   Upper limit on the number of points desired on a single output curve: If a particular curve generates more than MXPT points, the subroutine will stop processing that curve and the remaining points will be lost. MXPT must be at least 2. (This feature guaruntees that each intersection curve will not exceed the space allocated for it.)
6. MXCRV [In, INTEGER scalar]
   Upper limit on the number of curves generated from a single call: If a call generates more than MXCRV curves, the subroutine will finish processing any curves and will return only MXCRV curves. (This feature guaruntees that the number of intersection curves will not exceed the number of spaces allocated for them.) MXCRV must be positive and at most 12.
7. WORK [Temporary, REAL(2,MXPT,3)]
   A work array that is passed to Subroutine PROCUV (as array C)
8. NCRV [Out, INTEGER scalar]
   The number of curves generated
9. CURVE [Out, REAL(2,MXPT,MXCRV)]
   Array containing (U,V) pairs for points on the patch, along intersection curves. CURVE(1,1,K) is the number of points in the K-th curve; CURVE(*,J+1,K) is the J-th (U,V) pair on this curve.
10. IER [Out, INTEGER scalar]
    Success/error code: Zero for a successful call. For negative values of IER, Subroutine PLINT will stop processing immediately and all its calculations will be lost. For positive values of IER, calculations will continue but the results may be suspect.

    -1 is returned if MXPT is less than two. (This error code originates in Subroutine PLINT.)

    -2 is returned if MXCRV is either less than 1 or greater than 12. (This error code originates in Subroutine PLINT.)

    -3 is returned if MXIT is not positive. (This error code originates in Subroutine PLINT.)

    -4 is returned if TOL(1) is less than the relative precision of the machine. (This error code originates in Subroutine PLINT.)

-5 is returned if the patch is recognized as having more than the theoretical limit of three degenerate S values. (This error code originates in Subroutine PLINT.)

-6 is returned if the patch is recognized as having more than the theoretical limit of twelve or fifteen S values where T has a double root. (This error code can originate either in Subroutine DEGTST or in Subroutine BRINT.) NOTE: Subroutine PLNSRF changes PLINT error code -6 to a value of 6, which in a nonfatal error.

-7 is returned if the total number of boundary intersections recognized exceeds thetheoretical limit of tewenty-three. (This error code has been deactivated.)

-8 is returned if the number of intersection boundaries recognized on the upper or the lower edge exceeds the theoretical limit of three. (This error code can originate either in Subroutine PLINT or in Subroutine PROCUV.)

-9 is returned during curve processing if more than the expected number of double roots was found. (This error code originates in Subroutine PLINT.)

-10 is returned during curve processing if more than the expected number of boundary roots was found. (This error code has been deactivated.)

-11 is returned if the bicubic is degenerate at a boundary value. (i.e., Evaluating the bicubic at this boundary gives a polynomial of degree less than 3.) (This error code has been deactivated.)

-12, -13, and -14 are returned if Subroutine HSZERO failed while being called by Subrutine BRINT. The HSZERO error code is IER +11.

1 is returned if more than MXIT iterations were required for Subroutine CUBIC.

2 is returned if the step size required to follow the curve is less than TOL(2). NOTE: This case tends to indicate that some of the returned (S,T) data lies outside the unit square, which is theoretically impossible. (This error code originates in Subroutine TRACE.)

3 is returned if the number of points calculated on a curve exceeds MXPT. (This error code can originate either in Subroutine PROCUV or in Subroutine STRCUV.)

4 is returned if the number of intersection curves exceeds MXCRV. (This error code can originate either in Subroutine STRCUV or in Subroutine DEGTST.)

5 is returned if internal conditions indicate that either a curve is missing or that values outside the unit square are returned. (This error code can either originate either in Subroutine PLINT or in Subroutine PROCUV.)

8.54.2 COMMON DATA

COMMON /CPLINT/ MAXIT

MAXIT stores the value of MXIT

8.55 SUBROUTINE PLNCUR (from SUBS)

Used by MSHNRM to intersect a plane with a set of PC curves: This performs the cutting step upon plane/patch intersection curves. These curves contain 6 components, the 3 position components plus 3 surface-normal components.

8.55.1 PARAMETER LIST

1. OUTPUT [In, INTEGER scalar]
   Logical unit for listing output
2. TEMPPC [In, INTEGER scalar]
   Logical unit for mass storage of intersection curves: Intersection slices (Connected plane/patch intersections are stored, in the following form: patch index, slice index (within the intersection of this particular plane/patch pair), number of PC segments, initial PC-segment count for the slice, and finally the PC segments (in geometric form, with surface-normal components)
3. NRM [In, INTEGER scalar]
   Logical unit for intersection-normal output
4. NCLABX [Not used]
5. LABELX [Not used]
6. PLNCUT [In, REAL(4)]
   Coefficients for cutting plane:
   PLN(1)*X(1) + PLN(2)*X(2) + PLN(3)*X(3) + PLN(4) = 0.
7. MXPT [In, INTEGER scalar]
   Maximum number of segments in an intersection curve
8. INTCUR [Temporary, REAL(4,6,MXPT)]
   Array to hold an intersection curve from mass storage
9. NSLINT [In, INTEGER scalar]
   Number of intersection slices stored on file TEMPPC
10. TOL [In, REAL scalar]
    Tolerance for cubic polynomial solution
11. NNVCUM [In/Out, INTEGER scalar]
    Cumulative count of intersection normals
12. LSTOPT [In, INTEGER scalar]
    Level of listing output
13. QERR [Out, LOGICAL scalar]
    Error flag

8.55.2 COMMON DATA

none

8.56 SUBROUTINE PLNSRF (from SUBS)

Used by MSHNRM to intersect a plane with a set of patches: This performs the intersection step that gives curves, which are cut later. These curves contain 6 components, the 3 position components plus 3 surface-normal components.

8.56.1 PARAMETER LIST

1. OUTPUT [In, INTEGER scalar]
   Logical unit for listing output
2. PATMS [In, INTEGER scalar]
   Logical unit for patch mass storage
3. TEMPPC [In, INTEGER scalar]
   Logical unit for mass storage of intersection curves: Intersection slices (Connected plane/patch intersections are stored, in the following form: patch index, slice index (within the intersection of this particular plane/patch), number of PC segments, initial, PC segments (in geometric form, with surface-normal components).
4. INT [In, INTEGER scalar]
   Logical unit for listing of intersection curves: The curves are listed in CUR format (basically geometric representation), with surface-normal components and with system comments labelling each intersection slice and each PC segment within the slice.

5. NCLABX [In, INTEGER (3)]
   Number of characters in the coordinate labels
6. LABELX [In, INTEGER(3)]
   Coordinate labels: This is character data
7. PLNINT [In, REAL(4)]
   Coefficients for intersection plane:
   PLNINT(1)*X(1) + PLNINT (2)*X(2) + PLNINT(3)*X(3) + PLNINT(4) = 0.
8. NPAT [In, INTEGER scalar]
   Number of patches
9. MXIT [In, INTEGER scalar]
   Limit on the number of iterations for cubic polynomial solution
10. MXPT [In, INTEGER scalar]
    Limit on the number of points to be returned on an intersection slice
11. MXCRV [In, INTEGER scalar]
    Limit on the number of intersection slices to be returned from a plane/patch combination
12 TOLINT [In, REAL(2)]
   Plane/patch intersection tolerance: The tolerance is input as TOLINT(1), which is a tolerance of (U,V) pairs interpolated between the returned set of such pairs. TOLINT(2) is set to approximately the mean proprotional of the input tolerance and the precision of floating-point representation for this machine.
13. CURPAR [Temporary, REAL(2, MXPT, MXCRV)]
    Array to hold parametric points computed along plane/patch intersection curves
14. INTWRK [Temporary, REAL(2, MXPT, 3)]
    Array used by subroutine PLINT to compute plane/patch intersection curves
15. INTCUR [Temporary, REAL(4, 6, MXPT)]
    Array to hold a parametric cubic curve interpolating and intersection slice
16. LSTOPT [In, INTEGER scalar]
    Level of listing output
17. CYLSRF [In, LOGICAL scalar]
    Flag for cylindrical surface coordinates
18. IXRADM [In, INTEGER scalar]
    Index for the radial cylindrical coordinate
19. IXANGM [In, INTEGER scalar]
    Index for the angular cylindrical coordinate
20. IQSCAL [In/Out, INTEGER scalar]
    Code to select a method to scale tangents for parametric cubic interpolation along MSHNRM intersection curves: Possible values are 1 (which gives locally explicit scaling) and 2 (which gives Ferguson-Phillips scaling, minimizing the integral of 2nd derivatives along the arc).
21. NSLINT [Out, INTEGER scalar]
    Number of slices in the intersection of this plane with all the patches
22. NPCCUM [In/Out, INTEGER scalar]
    Cumulative count of PC segments
23. IERINT [Out, INTEGER scalar]
    Error code returned from Subroutine PLINT
24. QERR [Out, LOGICAL scalar]
    Error flag

8.56.2 COMMON DATA

none

8.57 SUBROUTINE PPTCUR (from SUBS)

Fits PC curves in physical space to parametric points computed along plane/surface intersection curves

8.57.1 PARAMETER LIST

1. OUTPUT [In, INTEGER scalar]
   Logical unit for listing output
2. NCLABX [In, INTEGER(3)]
   Number of characters in coordinate labels
3. LABELX [In, INTEGER(3)]
   Coordinate labels: This is character data.
4. LSTOPT [In, INTEGER scalar]
   Level of listing output
5. CYLSRF [In, LOGICAL scalar]
   Flag for cylindrical coordinates
6. IXRAD [In, INTEGER scalar]
   Index for the radial cylindrical coordiante
7. IXANG [In, INTEGER scalar]
   Index for the angular cylindrical coordinate
8. IQSCAL [In, INTEGER scalar]
   Code to select a method to scale tangents for parametric cubic interpolation along MSHNRM intersection curves: Possible values are 1 (which gives locally explicit scaling) and 2 (which gives Ferguson-Phillips scaling, minimizing the integral of 2nd derivatives along the arc).
9. PAT [In, REAL(16,3)]
   Geometric patch representation: Effectively allocated PAT(4,4,3). PAT(*,*,K) is the K-th component of a vector. PAT(1,1,*) is the (U,V) = (0,0) position; PAT(1,2,*) is the (1,0) position; PAT(2,1,*) is the (0,1) position; and PAT(2,2,*) is the (1,1) position. PAT(3,1,*), PAT(3,2,*), PAT(4,1,*), and PAT(4,2,*) are the corresponding V-derivatives. PAT(1,3,*), PAT(1,4,*), PAT(2,3,*), and PAT(2,4,*) are the corresponding U-derivatives. PAT(3,3,*), PAT(3,4,*), PAT(4,3,*), and PAT(4,4,*) are the corresponding cross-derivatives.
10. PLN [In, REAL(4)]
    Coefficients for intersection plane:
    $$PLN(1)*X(1) + PLN(2)*X(2) + PLN(3)*X(3) + PLN(4) = 0.$$
11. NPT [In, INTEGER scalar]
    Number of points input in array CURVE
12. CURVE [In, REAL(2,NPT)]
    Points on the plane/patch intersection curve: These are parametric (U,V) pairs.
13. INTCUR [Out, REAL(4,6,NPT)]
    Parametric curves interpolating along the intersection
14. QERR [Out, LOGICAL scalar]
    Error flag

8.57.2 COMMON DATA

none

8.58 SUBROUTINE PROCUV (from LIBS)

Within Subroutine PLINT, controls the tracing of curves (After the patch has been divided into vertical strips in which the number of roots to trace is constant)

8.58.1 PARAMETER LIST

1. BC [In, REAL(4,4)]
   Coefficients for the output bicubic function, in algebraic form: BC(I,J) is the coefficient of $U^{(4-J)} * V^{(4-I)}$.
2. NDEG [In, INTEGER scalar]
   The maximum degree of the polynomial in T obtained by evaluating the bicubic at an arbitrary S value

3. HBND [In, REAL(2)]
   Horizontal (T) boundaries to the region of interest
4. BBND [In, REAL(NB)]
   Array containing S values at which the number of T roots can change for one of the following reasons:
   1. An zero curve leaves the region of interest via an horizontal or vertical boundary.
   2. The quadratic factor in the polynomial in T has a double root.
   3. The polynomial in T is degenerate.
5. JB [In/Temporary, INTEGER scalar]
   BBND(JB) is the S value at which a zero curve enters the region of interest.
6. NBDG [In, INTEGER(NB)]
   The degree of the polynomial obtained by evaluating the bicubic at an S value from array BBND
7. NB [In, INTEGER scalar]
   Number of S values where the bicubic is degenerate
8. TOL [In, REAL(2)]
   Array containing the user-specified tolerance: TOL(1) is the input tolerance. Any point on a parametric cubic curve interpolating the output (U,V) points is within TOL(1) distance (in parametric units) of a straight line between the surrounding output points. TOL(2) is computed by PLINT and is used as a lower limit for the stepsize.
9. TRT [In, REAL(4,2)]
   Array containing S values at which a zero curve enters or exits the region of interest by a horizontal boundary
10. TCT [In, REAL(NC)]
    Array containing S values at which the polynomial in T has a double root
11. NC [In, INTEGER scalar]
    The number of S values at which the polynomial in T has a double root
12. IPT [In/Out, INTEGER(3)]
    The number of points calculated on a zero curve: If IPT(I) = -1, then the curve passing through the root U(IT(I)) will not be traced.
13. IPTT [In, INTEGER(3)]
    A temporary version of array IPT used to initially determine which zero curves will be traced.
14. IT [In/Out, INTEGER(3)]
    Array containing the subscripts of the roots of a cubic equation: U(IT(1)) is the real root found via a Newton iterative scheme; U(IT(2)) and U(IT(3)) are the roots found from the quadratic factor.
15. ID [In, INTEGER(2)]
    Array specifying the renaming of U and V as S and T. ID(1) is the index value for S in the ordered pair (U,V), and ID(2) is the index value for T. (e.g., If ID(1) is 2, then S is the name for V within the planar intersector.)
16. NTR [In, INTEGER(2)]
    Array containing the number of values in TRT
17. TR [In/Temporary, REAL(3,2)]
    Array containing the roots of T corresponding to the ends of the interval in S for the current tracing step. TR(*,1) at the left of the patch is input to start the tracing.
18. RPT [In/Temporary, REAL(3,2)]
    Array containing the partial derivative of the bicubic with respect to T at the 3 T roots at the current tracing step.
19. RPS [In/Temporary, REAL(3,2)]
    Array containing the partial derivative of the bicubic with respect to S at the 3 T roots at the current tracing step.
20. RPTT [In/Temporary REAL(3,2)]
    Array containing the second partial derivative of the bicubic with respect to T at the 3 T roots at the current tracing step.
21. RPSS [In/Temporary, REAL(3,2)]
    Array containing the second partial derivative of the bicubic with respect to S at the 3 T roots at the current tracing step.
22. RPST [In/Temporary, REAL(3,2)]
    Array containing the mixed second partial derivative of the bicubic at the 3 T roots at the current tracing step.

23. C [Temporary, REAL(2,MXPT,3)]
    Array used to accumulate points on the intersection curves while they are traced
24. MXPT [In, INTEGER scalar]
    Upper limit on the number of points desired on a single output curve: If a particular curve generates more than MXPT points, the subroutine will stop processing that curve and the remaining points will be lost. MXPT must be at least 2. (This feature guaruntees that each intersection curve will not exceed the space allocated for it.)
25. MXCRV [In, INTEGER scalar]
    Upper limit on the number of curves generated from a single call: If a call generates more than MXCRV curves, the subroutine will finish processing any curves and will return only MXCRV curves. (This feature guaruntees that the number of intersection curves will not exceed the number of spaces allocated for them.) MXCRV must be positive and at most 12.
26. NCRV [In/Out, INTEGER scalar]
    The number of curves generated
27. CURVE [In/Out, REAL(2,MXPT,MXCRV)]
    Array containing (U,V) pairs for points on the patch, along intersection curves. CURVE(1,1,K) is the number of points in the K-th curve; CURVE(*,J+1,K) is the J-th (U,V) pair on this curve.
28. IER [In/Out, INTEGER scalar]
    Success/error code: PROCUV detects error codes -9, -8, and 5. It also returns error codes from Subroutines TRACE and PROCUV.

    -8 is returned when the number of intersection boundaries recognized on the upper or the lower edge exceeds the theoretical limit of three.

    -9 is returned when during curve processing more than the expected number of double roots was found.

    5 is returned when the JP indices to the TRT array lose alignment with the JB and JB1 indices to the BBND array.

8.58.2 COMMON DATA

none

8.59 SUBROUTINE PSGMAL (from SUBS)

Converts a patch from geometric representation to algebraic representation

8.59.1 PARAMETER LIST

1. GEO [In, REAL(4,4,3)]
   Geometric patch representation: GEO(*,*,K) is the K-th component of a vector. GEO(1,1,*) is the (U,V) = (0,0) position; GEO(1,2,*) is the (1,0) position; GEO(2,1,*) is the (0,1) position; and GEO(2,2,*) is the (1,1) position. GEO(3,1,*), GEO(3,2,*), GEO(4,1,*), and GEO(4,2,*) are the corresponding V-derivatives. GEO(1,3,*), GEO(1,4,*), GEO(2,3,*), and GEO(2,4,*) are the corresponding U-derivatives. GEO(3,3,*), GEO(3,4,*), GEO(4,3,*), and GEO(4,4,*) are the corresponding cross-derivatives.
2. ALG [Out, REAL(4,4,3)]
   Algebraic representation: ALG(I,J,K) is the coefficient of $U^{(4-J)} * V^{(4-I)}$ in the bicubic polynomial for the K-th coordinate.

8.59.2 COMMON DATA

none

8.60 SUBROUTINE QUAD (from LIBS)

Within Subroutine PLINT, finds the roots of a quadratic equation

8.60.1 PARAMETER LIST

1. A [In, REAL(3)]
   Array containing the coefficients of the equation:
   $$A(1) * X^2 + A(2) * X + A(3) = 0.$$
2. TOL [In, REAL(2)]
   The user-specified tolerance
3. IC [Out, INTEGER scalar]
   The sign of the discriminant for the quadratic equation. (The equation has IC + 1 roots.)
4. LR [Out, REAL scalar]
   The real part of the larger root of a quadratic equation
5. SR [Out, REAL scalar]
   The real part of the smaller root of a quadratic equation

8.60.2 COMMON DATA

none

8.61 SUBROUTINE RADCUR (from SUBS)

Stub for use by the planned MSHNRM capability to intersect a cylindrical-coordinate mesh with rectangular-coordinate patches: This subroutine will transform parametric cubic intersection curves, adding radius-squared as a 7th coordinate. (The surface normal is represented as the 4th-6th coordinates.)

8.61.1 PARAMETER LIST

1. OUTPUT [In, INTEGER scalar]
   Logcial unit for listing output
2. PATMS [Not used]
3. TEMPRC [Not used]
4. NCLABX [Not used]
5. LABELX [Not used]
6. PLNCUT [Not used]
7. NCRV [Not used]
8. QERR [Out, LOGICAL scalar]
   Error flag

8.61.2 COMMON DATA

none

8.62 SUBROUTINE RADCUT (from SUBS)

Stub for use by the planned MSHNRM capability to intersect a cylindrical-coordinate mesh with rectangular-coordinate patches: This subroutine will cut intersection curves at specific values of radius-squared, which is interpolated as the 7th coordinate. This cutting will produce intersection normals.

8.62.1 PARAMETER LIST

1. OUTPUT [In, INTEGER scalar]
   Logical unit for listing output
2. TEMPPC [Not used]

3. TEMPRS [Not used]
4. NRM [Not used]
5. NCLABX [Not used]
6. LABELX [Not used]
7. RADIUS [Not used]
8. NCRV [Not used]
9. QERR [Out, LOGICAL scalar]
   Error flag

8.62.2 COMMON DATA

none

8.63 SUBROUTINE RADINT (from SUBS)

Stub for use by the planned MSHNRM capability to intersect a cylindrical-coordinate mesh with rectangular-coordinate patches: This subroutine will intersect patches at specific values of radius-squared, which is interpolated as a 4th coordinate. This process will produce the usual intersection curves.

8.63.1 PARAMETER LIST

1. OUTPUT [In, INTEGER scalar]
   Logcial unit for listing output
2. PATMS [Not used]
3. TEMPPC [Not used]
4. TEMPRS [Not used]
5. NCLABX [Not used]
6. LABELX [Not used]
7. RADIUS [Not used]
8. MXIT [Not used]
9. MXPT [Not used]
10. NXCRV [Not used]
11. TOLINT [Not used]
12. NCRV [Not used]
13. INTCUR [Not used]
14. IERINT [Not used]
15. INTWRK [Not used]
16. QERR [Out, LOGICAL scalar]
    Error flag

8.63.2 COMMON DATA

none

8.64 SUBROUTINE RADSRF (from SUBS)

Stub for use by the planned MSHNRM capability to intersect a cylindrical-coordinate mesh with rectangular-coordinate patches: This subroutine will transform patches, adding radius-squared as a 4th coordinate.

8.64.1 PARAMETER LIST

1. OUTPUT [In, INTEGER scalar]
   Logical unit for listing output
2. PATMS [Not used]
3. TEMPRS [Not used]
4. NCLABX [Not used]
5. LABELX [Not used]

6. NPAT [Not used]
7. QERR [Out, LOGICAL scalar]
   Error flag

8.64.2 COMMON DATA

none

8.65 SUBROUTINE REACUV (from LIBS)

Called from Subroutine SUFINT to begin to connect intersection slices

8.65.1 PARAMETER LIST

1. MXJPT [In, INTEGER scalar]
2. NJCVT [In, INTEGER scalar]
3. NJCRV [In, INTEGER scalar]
4. MXJCRV [In, INTEGER scalar]
5. NHLD [In, INTEGER scalar]
6. NMPT [In, INETGER(NHLD)]
7. PONT [In/Out, REAL(3,MXJPT,NJCRV)]
8. TOL [In, REAL scalar]
   Tolerance value
9. MXPT [In, REAL scalar]
   Maximum number of points to be calculated in each output intersection branch
10. MXCRV [In, REAL scalar]
    Maximum number of intersection branches to be calculated
11. NCRV [In/Out, INTEGER scalar]
    Number of intersection branches calculated
12. CURVE [In/Out, INTEGER scalar]
    Intersection branches: This is the type of data output from SUFINT.
13. IER [Out, INTEGER scalar]
    Error code: This is zero for a normal exit. (See Subroutine SUFINT for more details.)

8.65.2 COMMON DATA

none

8.66 SUBROUTINE READ (from SUBS)

Within Program REGSIL, reads a SIL-format set of sections from unit 2, copying SIL options to unit 4

8.66.1 PARAMETER LIST

1. NSSLP [Out, INTEGER(17)]
   End-condition codes for each section
2. DXIS [Out, REAL(17)]
   X-components of initial end directions for each section
3. DYIS [Out, REAL(17)]
   Y-components of initial end directions for each section
4. DZIS [Out, REAL(17)]
   Z-components of initial end directions for each section
5. DXOS [Out, REAL(17)]
   X-components of final end directions for each section
6. DYOS [Out, REAL(17)]
   Y-components of final end directions for each section
7. DZOS [Out, REAL(17)]
   Z-components of final end directions for each section

8. X [Out, REAL(17,175)]
   X-coordinates of points for each section, for each point
9. Y [Out, REAL(17,175)]
   Y-coordinates of points for each section, for each point
10. Z [Out, REAL(17,175)]
    Z-coordinates of points for each section, for each point
11. LABELX [In/Out, INTEGER(3)]
    Coordinate labels: The default labels are input, but SIL options can change the labels. This is character data.
12. NCLABX [In/Out, INTEGER(3)]
    Number of characters in the coordinate labels
13. CYL [In/Out, LOGICAL scalar]
    Flag for cylindrical coordinates: The default is input as rectangular coordinates.
14. IXAXI [In/Out, INTEGER scalar]
    Index for the axial cylindrical coordinate: (IXAXI, IXRAD, and IXANG must be a permutation of the set 1, 2, and 3.)
15. IXRAD [In/Out, INTEGER scalar]
    Index for the radial cylindrical coordinate: (IXAXI, IXRAD, and IXANG must be a permutation of the set 1, 2, and 3.)
16. IXANG [In/Out, INTEGER scalar]
    Index for the angular cylindrical coordinate: (IXAXI, IXRAD, and IXANG must be a permutation of the set 1, 2, and 3.)

8.66.2 COMMON DATA

COMMON /MODSTF2/ NSECT, NDIM, NDIMC, N1, N2, IFAIL
COMMON /MODSTF3/ NPTS(17)

NSECT is the number of sections and NPTS is the number of points for each section. Values for these variables are read by this subroutine.

8.67 SUBROUTINE RESID (from LIBS)

Used by subroutine NWO21 to compute residuals.

8.67.1 PARAMETER LIST

1. M [not used]
2. NP [In, INTEGER scalar]
   Number of knots per curve
3. PKNOT [In, REAL(*)]
   Trial knot placements
4. R [Out, REAL(*)]
   Residuals
5. IFL [Out, INTEGER scalar]
   Error code

8.67.2 COMMON DATA

none

8.68 SUBROUTINE ROOTS (from LIBS)

Within Subroutine PLINT, determines the roots of the cubic polynomial in T given by by evaluating the bicubic at a given S value. (Error codes from Subroutine CUBIC are not passed along.)

8.68.1 PARAMETER LIST

1. BC [In, REAL(4,4)]
   Coefficients for the output bicubic function, in algebraic form: BC(I,J) is the coefficient of $U^{(4-J)} * V^{(4-I)}$.
2. SVAL [In, REAL scalar]
   The S value
3. IRT [In, INTEGER(3)]
   Array containing the subscripts of the roots of a cubic equation: U(IRT(1)) is the real root found via a Newton iterative scheme; U(IRT(2)) and U(IRT(3)) are the roots found from the quadratic factor.
4. TOL [In, REAL(2)]
   Array containing the user-specified tolerance: TOL(1) is the input tolerance. Any point on a parametric cubic curve interpolating the output (U,V) points is within TOL(1) distance (in parametric units) of a straight line between the surrounding output points. TOL(2) is computed by PLINT and is used as a lower limit for the step size.
5. TRVAL [Out, REAL(3)]
   Array containing the roots of T at the S value in SVAL
6. IC [Out, INTEGER scalar]
   The sign of the discriminant for the quadratic equation obtained by deflating the cubic with the first root. (The quadratic equation has IC + 1 roots, so the cubic equation has IC + 2 roots.)

8.68.2 COMMON DATA

none

8.69 SUBROUTINE SAXPY (from LIBS)

BLAS (LINPACK Basic Linear Algebra Subprogram) to compute a constant times a vector plus a vector (see Reference 7).

8.69.1 PARAMETER LIST

1. N [In, INTEGER scalar]
   Number of elements of SX and SY
2. SA [In, REAL scalar]
   Constant
3. SX [In, REAL(*)]
   The vector which is multiplied by constant
4. INCX [In, INTEGER scalar]
   The storage increment between the elements of SX
5. SY [In/Out, REAL(*)]
   The other input vector, also the result
6. INCY [In, INTEGER scalar]
   The storage increment between the elements of SY

8.69.2 COMMON DATA

none

8.70 SUBROUTINE SCOPY (from LIBS)

BLAS (LINPACK Basic Linear Algebra Subprogram) to copy a vector (see Reference 7).

8.70.1 PARAMETER LIST

1. N [In, INTEGER scalar]
   The number of elements in the vector

2. SX [In, REAL(*)]
   The original vector
3. INCX [In, INTEGER scalar]
   The storage increment between the elements of SX
4. SY [Out, REAL(*)]
   The copy
5. INCY [In, INTEGER scalar]
   The storage increment between the elements of SY

8.70.2 COMMON DATA

none

8.71 REAL FUNCTION SDOT (from LIBS)

BLAS (LINPACK Basic Linear Algebra Subprogram) to compute the scalar product of two vectors (see Reference 7).

8.71.1 PARAMETER LIST

1. N [In, INTEGER scalar]
   The number of elements in SX and in SY
2. SX [In, REAL(*)]
   Vector 1
3. INCX [In, INTEGER scalar]
   The storage increment between elements of SX
4. SY [In, REAL(*)]
   Vector 2
5. INCY [In, INTEGER scalar]
   The storage increment between elements of SY

8.71.2 COMMON DATA

none

8.72 SUBROUTINE SETTOL (from SUBS)

Sets the tolerances used by Program SRFINT

8.72.1 PARAMETER LIST

1. L5 [In, INTEGER scalar]
   Logical unit number for input
2. L6 [In, INTEGER scalar]
   Logical unit number for output
3. TOLPT [Out, REAL(3)]
   Tolerances for each coordinate for matching end points between intersection slices: The default values are 1.0 distance units (typically inches) for each component.
4. TOL [Out, REAL(4)]
   Tolerances for computing the intersection of a pair of patches: TOL(1) is a gluing tolerance for joining planar-intersector results; TOL(2) is a flatness tolerance for the approximation of a patch by polygon faces; TOL(3) is the planar-intersector tolerance. TOL(4) is set within the planar intersector. TOL(1), TOL(2), and TOL(3) have default values of .01, .001, and .00001 respectively.

8.72.2 COMMON DATA

none

8.73 SUBROUTINE SILCPY (from SUBS)

Copies a file of SIL data, rewriting it with system comments added

8.73.1 PARAMETER LIST

1. OLDSIL [In, INTEGER scalar]
   Logical unit for SIL input
2. NEWSIL [In, INTEGER scalar]
   Logical unit for SIL output
3. NCLABX [In/Out, INTEGER(3)]
   Number of characters in the coordinate labels
4. LABELX [In/Out, INTEGER(3)]
   Coordinate labels: This is character data.
5. XPT [Temporary, REAL(3,175)]
   Array to hold point coordinates for a section
6. IQPT [Temporary, INTEGER(2,120)]
   Array to hold section-knot and section indices for a member
7. TEN [Temporary, REAL(175)]
   Array to hold tension values for either a section or a member
8. TENDD [Temporary, REAL(3,2)]
   Inital and final end directions for either a section or a member
9. IU0V0 [Temporary, INTEGER(2)]
   Array to hold member-knot and member indices for the (0,0) patch corner
10. IU0V1 [Temporary, INTEGER(2)]
    Array to hold member-knot and member indices for the (0,1) patch corner
11. IU1V0 [Temporary, INTEGER(2)]
    Array to hold member-knot and member indices for the (1,0) patch corner
12. IU1V1 [Temporary, INTEGER(2)]
    Array to hold member-knot and member indices for the (1,1) patch corner

8.73.2 COMMON DATA

none

8.74 SUBROUTINE SILOPT (from SUBS)

Reads option declaractions at the beginning of a SIL file and copies them

8.74.1 PARAMETER LIST

1. OLDSIL [In, INTEGER scalar]
   Logical unit for SIL input
2. NEWSIL [In, INTEGER scalar]
   Logical unit for SIL option-declaration output: If negative, copying is suppressed.
3. OUTPUT [In, INTEGER scalar]
   Logical unit for listing output: If negative, listing is suppressed
4. CYL [In/Out, LOGICAL scalar]
   Flag for cylindrical coordinates: Rectangular coordinates are input as the default.
5. IXAXI [In/Out, INTEGER scalar]
   Index for the axial cylindrical coordinate. A default value of 1 is input. (IXAXI, IXANG, and IXRAD must be a permutation of the set 1, 2, and 3.)
6. IXRAD [In/Out, INTEGER scalar]
   Index for the axial cylindrical coordinate. A default value of 2 is input. (IXAXI, IXANG, and IXRAD must be a permutation of the set 1, 2, and 3.)
7. IXANG [In/Out, INTEGER scalar]
   Index for the axial cylindrical coordinate. A default value of 3 is input. (IXAXI, IXANG, and IXRAD must be a permutation of the set 1, 2, and 3.)

8. LABELR [In/Out, INTEGER(3)]
   Labels for rectangular coordinates: This is character data. The default labels are input, but the labels can be changed before exit.
9. NCLABR [In/Out, INTEGER(3)]
   The number of characters in the rectangular-coordinate labels: This can range from 1 to 10.
10. LABELC [In/Out, INTEGER(3)]
    Labels for cylindrical coordinates: This is character data. The default labels are input, but the labels can be changed before exit.
11. NCLABC [In/Out, INTEGER(3)]
    The number of characters in the cylindrical-coordinate labels: This can range from 1 to 10.
12. LABELX [Out, INTEGER(3)]
    Labels for surface coordinates: This is character data. It is either the labels for cylindrical or rectangular coordinates.
13. NCLABX [Out, INTEGER(3)]
    The number of characters in the surface coordinate labels: This can range from 1 to 10.
14. DUMP [Out, LOGICAL scalar]
    Flag to list details of the surface-modeling calculations

8.74.2 COMMON DATA

none

8.75 REAL FUNCTION SNRM2 (from LIBS)

BLAS (LINPACK Basic Linear Algebra Subprogram) to compute the Euclidean norm of a vector (see (see Reference 7).

8.75.1 PARAMETER LIST

1. N [In, INTEGER scalar]
   The number of elements in SX
2. SX [In, REAL(*)]
   The vector
3. INCX [In, INTEGER scalar]
   The storage increment between the elements of SX

8.75.2 COMMON DATA

none

8.76 SUBROUTINE SORTRN (from SUBS)

Sorts N-tuples of real values, using quicksort

8.76.1 PARAMETER LIST

1. LFERR [In, INTEGER scalar]
   Logical unit to write error messages
2. NKEY [In, INTEGER scalar]
   The number of components in the sorting key: This must be positive.
3. IKEY [In, INTEGER(NKEY)]
   Key to precedence of components in sorting: These must either be between 1 and N or be between -N and -1. IKEY(I) controls the sorting of all N-tuples which have the previous I-1 components equal. If IKEY(I) is positive, these values are sorted in ascending order on their I-th component. If IKEY(I) is negative, these values are sorted in descending order on their I-th component.
4. N [In, INTEGER scalar]
   The number of components in each N-tuple: This must be positive.

5. LONG [In, INTEGER scalar]
   The number of N-tuples to sort: From 1 to 2^{20} are permitted.
6. A [In/Out, REAL(N,LONG)]
   Array containing the N-tuples
7. QERR [Out, LOGICAL scalar]
   Error Flag

8.76.2 COMMON DATA

none

8.77 SUBROUTINE SQRDC (from LIBS)

LINPACK routine to compute the QR decomposition of a matrix: The matrix, X, is factored into two matrices, Q and R. Q is orthogonal and R is upper triangular. (This decomposition can be used by LINPACK routine SQRSL to perform coordinate transformations, projections, and least-squares solutions.) (see Reference 7.)

8.77.1 PARAMETER LIST

1. X [In/Out, REAL(LDX,P)]
   On entry this array holds the input matrix, X. On exit it holds in its upper triangle the upper-triangular factor, R; below its diagonal, X contains information from which the orthogonal factor, Q, can be recovered.
2. LDX [In, INTEGER scalar]
   The leading dimension allocated for the array X
3. N [In, INTEGER scalar]
   The number of rows in the matrix X. (N should be greater than P, or the least-squares problem will be underdetermined.)
4. P [In, INTEGER scalar]
   The number of columns in the matrix X.
5. QRAUX [Out, REAL(P)]
   Further information from which (combined with the data returned below the diagonal in array X) the orthogonal factor, Q, can be recovered.
6. JPVT [In/Out, INTEGER(P)]
   If JOB = 1, pivoting control information is input and the pivoting ordering used is returned.
   The sign of each input value controls the treatment of the corresponding matrix column: A positive value indicates an initial column;a zero value indicates a free column; a negative value indicates a final column.(Initial columns are moved to the leading part of X, and final columns are moved to the trailing part of X. During the decomposition, only free columns are moved. The information from the leading part of the matrix, as pivoted, stays in the leading part of R, along with all of Q.)
   The output values from an execution with pivoting show the index of the column that was moved into the J-th location.
7. WORK [Temporary, REAL(P)]
   Array used while pivoting: This is not referenced if JOB = 0.
8. JOB [In, INTEGER scalar]
   Pivoting code: If 0, pivoting is not done; if 1, pivoting is done according to the JPVT input. A pivoting execution factors a rearrangement of X, so the rearrangement information that is output in JPVT is needed to use the factorization.

8.77.2 COMMON DATA

none

8.78 SUBROUTINE SQRSL (from LIBS)

LINPACK routine to manipulate the QR decomposition of a matrix, as computed by Subroutine SQRDC: (The matrix, X, has been factored into two matrices, Q and R. Q is orthogonal and R is upper triangular.) This manipulation can solve least-squares problems, and it can perform coordinate transformations and projections (See Reference 7.)

8.78.1 Parameter List

1. X [In, REAL(LDX,P)]
   The contents of array X after returning from Subroutine SQRDC
2. LDX [In, INTEGER scalar]
   The leading dimension allocated for the array X
3. N [In, INTEGER scalar]
   The number of rows in the matrix X. (N should be greater than P, or the least-squares problem will be underdetermined.)
4. K [In, INTEGER scalar]
   The number of leading columns of the matrix X to manipulate: K must not exceed either N or P.
5. QRAUX [In, REAL(P)]
   The contents of array QRAUX after returning from Subroutine SQRDC
6. Y [In, REAL(N)]
   A vector to be manipulated, of length N
7. QY [Out, REAL(N)]
   If requested, the matrix product of Q before Y. Otherwise this array is not referenced.
8. QTY [Out, REAL(N)]
   If requested, the matrix product of Q-transpose before Y. Otherwise this array is not referenced.
9. B [Out, REAL(P)]
   If requested, the solution of the least-squares problem minimizing the norm of the N-vector difference "Y-XB". Otherwise this array is not referenced.
10. RSD [Out, REAL(N)]
    If requested, the residual vector "Y-XB" for the least-squares solution. Otherwise this array is not referenced. (This is also the orthogonal projection of Y onto the orthogonal complement to the column space of X.)
11. XB [Out, REAL(N)]
    If requested, the matrix product of X before B, which is the least-squares approximation to Y. Otherwise this array is not referenced. (This is also the orthogonal proection of Y onto the column space of X.)
12. JOB [In, INTEGER scalar]
    Parameter to request specific results: This is written as a decimal number with 5 digits: ABCDE. Digit A is nonzero to request the computation of QY; one of digits B, C, D, or E are nonzero to request the computation of QTY; digit C is nonzero to request the computation of B; digit D is nonzero to request the computation of RSD; and digit E is nozero to request the computation of XB.
13. INFO [Out, INTEGER scalar]
    Error code: This is zero for the case when R is nonsingular. Otherwise it is the index of the first zero on the diagonal of R, and B is not changed.

8.78.2 COMMON DATA

none

8.79 SUBROUTINE SSCAL (from LIBS)

BLAS (LINPACK Basic Linear Algebra Subprogram) to scale a vector by a constant (see Reference 7).

8.79.1 PARAMETER LIST

1. N [In, INTEGER scalar]
   The number of elements in SX
2. SA [In, REAL scalar]
   The constant
3. SX [In/Out, REAL(*)]
   The vector
4. INCX [In, INTEGER scalar]
   The storage increment between the elements of SX

8.79.2 COMMON DATA

none

8.80 SUBROUTINE SSWAP (from LIBS)

BLAS (LINPACK Basic Linear Algebra Subprogram) to swap two vectors (see Reference 7).

8.80.1 PARAMETER LIST

1. N [In, INTEGER scalar]
   The number of elements in SX and in SY
2. SX [In/Out, REAL(*)]
   Vector 1
3. INCX [In, INTEGER scalar]
   The storage increment between the elements of SX
4. SY [In/Out, REAL(*)]
   Vector 2
5. INCY [In, INTEGER scalar]
   The storage increment between the elements of SY

8.80.2 COMMON DATA

none

8.81 SUBROUTINE STRCUV (from LIBS)

Within Subroutine PLINT, stores a plane/patch intersection curve

8.81.1 PARAMETER LIST

1. C [In/Out, REAL(2,MXPT,3)]
   Array used to accumulate points on the intersection curves while they are traced: Curves are input here and then cleared from the array.
2. IRT [In, INTEGER scalar]
   Index to the curve to be stored
3. IRT1 [In, INTEGER scalar]
   Index to the remaining part of a loop-shaped curve to be stored, also the index to where an ITYP = 4 call stores a curve in C.
4. ID [In, INTEGER(2)]
   Array specifying the renaming of U and V as S and T. ID(1) is the index value for S in the ordered pair (U,V), and ID(2) is the index value for T. (e.g., If ID(1) is 2, then S is the name for V within the planar intersector.)
5. ITYP [In, INTEGER scalar]
   The type of zero curve: (1) Simple curve, (2) Open loop, (3) Closed loop, (4) Indicates a call to Subroutine STRCUV to reverse a curve, moving it from C(*,*,IRT) to C(*,*,IRT1).

6. IPT [In/Out, INTEGER(3)]
   The number of points on a zero curve in C(*,*,K): If IPT(I) = -1, then the curve passing through the root U(IT(I)) will not be traced.
7. MXPT [In, INTEGER scalar]
   Upper limit on the number of points desired on a single output curve: If a particular curve generates more than MXPT points, PLINT will stop processing that curve and the remaining points will be lost. MXPT must be at least 2. (This feature guarantees that the each of intersection curves will not exceed the number of spaces allocated for them.)
8. MXCRV [In, INTEGER scalar]
   Upper limit on the number of curves generated from a single call: If a call generates more than MXCRV curves, PLINT will finish processing any curves and will return only MXCRV curves. (This feature guarantees that the number of intersection curves will not exceed the number of spaces allocated for them.) MXCRV must be positive and at most 12.
9. NCRV [In/Out, INTEGER scalar]
   The number of curves stored in array CURVE
10. CURVE [In/Out, REAL(2,MXPT,MXCRV)]
    Array containing (U,V) pairs for points on the patch, along the stored intersection curves. CURVE(1,1,K) is the number of points in the K-th curve; CURVE(*,J+1,K) is the J-th (U,V) pair on this curve.
11. IER [In/Out, INTEGER scalar]
    Success/error code: STRCUV can return error codes 3 and 4. 3 is returned when the number of points calculated on a curve exceeds MXPT. 4 is returned when the number of intersection curves exceeds MXCRV.

8.81.2 COMMON DATA

none

8.82 COMPASS SUBROUTINE STRMOV

Copies a character string.

8.82.1 PARAMETER LIST

1. SA [In, Array]
   Input string
2. LA [In, INTEGER scalar]
   Location of the first charcter in string SA: This is 1 for the leftmost character position in the array.
3. NA [In, INTEGER scalar]
   Number of characters in string SA
4. SB [Out, Array]
   String copy
5. LB [In, INTEGER scalar]
   Location of the first character in string SB: This is 1 for the leftmost character position in the array.

8.82.2 COMMON DATA

none

8.83 SUBROUTINE SUFDEG (from LIBS)

Within Subroutine PLINT, determines if the bicubic is degenerate at a given S value, also computes the roots at the S value and the partial derivatives at these roots: This is called by PLINT to root patch boundaries and to initalize roots for tracing.

8.83.1 PARAMETER LIST

1. SVAL [In, REAL scalar]
   S value
2. BC [In, REAL(4,4)]
   Coefficients for the output bicubic function, in algebraic form: BC(I,J) is the coefficient of $U^{(4-J)} * V^{(4-I)}$.
3. HBND [In, REAL(2)]
   Horizontal (T) boundaries to the region of interest
4. IDEG [Out, INTEGER scalar]
   Degree of the polynomial in T obtained when the bicubic is evaluated at the S value
5. NDEG [In, INTEGER scalar]
   The maximum degree of the polynomial in T obtained by evaluating the bicubic at an arbitrary S value
6. IRT [In, INTEGER(3)]
   Array containing the subscripts of the roots of cubic equation: U(IRT(1)) is the real root found via a Newton iterative scheme; U(IRT(2)) and U(IRT(3)) are the roots found from the quadratic factor.
7. TOL [In, REAL(2)]
   Array containing the user-specified tolerance: TOL(1) is the input tolerance. Any point on a parametric cubic curve interpolating the output (U,V) points is within TOL(1) distance (in parametric units) of a straight line between the surrounding output points. TOL(2) is computed by PLINT and is used as a lower limit for the stepsize.
8. IC [Out, INTEGER(2)]
   The sign of the discriminant for the quadratic equation obtained by deflating the cubic with the first root. (The quadratic equation has IC + 1 roots, so the cubic equation has IC + 2 roots.)
9. TRVAL [Out, INTEGER scalar]
   Array containing the roots of the cubic polynomial in T obtained by evaluating the bicubic at SVAL
10. RPS [In, REAL(3)]
    Array containing the partial derivative with respect to S of the cubic polynomial in T obtained by evaluating the bicubic at the 3 T roots as SVAL
11. RPT [Out, REAL(3)]
    Array containing the partial derivative with respect to T of the cubic polynomial in T obtained by evaluating the bicubic at the 3 T roots as SVAL
12. RPSS [Out, REAL(3)]
    Array containing the second partial derivative with respect to S of the cubic polynomial in T obtained by evaluating the bicubic at the 3 T roots at SVAL
13. RPST [Out, REAL(3)]
    Array containing the mixed second partial derivative of the cubic polynomial in T obtained by evaluating the bicubic at the 3 T roots at SVAL
14. RPTT [Out, REAL(3)]
    Array containing the second partial derivative with respect to T of the cubic polynomial in T obtained by evaluating the bicubic at the 3 T roots at SVAL
15. IER [In/Out, INTEGER scalar]
    Success/error code: The only possible error code is 1, which is passed along from Subroutine CUBIC.

8.83.2 COMMON DATA

none

8.84 SUBROUTINE SUFFER (from SUBS)

Reports errors from subroutine SUFINT

8.84.1 PARAMETER LIST

1. L6 [In, INTEGER scalar]
   Logical unit for error-message output

2. IER [In, INTEGER scalar]
   Error code, returned from SUFINT
3. IPAT1 [In, INTEGER scalar]
   Index to current first-surface patch
4. IPAT2 [In, INTEGER scalar]
   Index to current second-surface patch
5. MXJPT [In, INTEGER scalar]
   This must be at least 3.
6. NLEV [In, INTEGER scalar]
   Maximum depth for the binary patch-subdivision tree: This must be at least 1.
7. MXJCRV [In, INTEGER scalar]
   This must be at least 2.
8. TOL [In, INTEGER(4)]
   Tolerance values: TOL(1) is used to control the gluing of trimmed slices together to make intersection branches. TOL(2) is used to determine when a subpatch is effectively flat. TOL(3) is the input tolerance for the plane/patch intersector. TOL(4) is the second tolerance computed by the plane/patch intersector.
9. MXPT [In, INTEGER scalar]
   The maximum number of points to be computed on an intersection branch: This must be at least 2.
10. MXCRV [In, INTEGER scalar]
    The maximum number of intersection branches to be computed: This must be at least 1.

8.84.2 COMMON DATA

none

8.85 SUBROUTINE SUFINT (from LIBS)

In Program SRFINT, computes the intersection of a single patch from the first surface with a single patch from the second surface: The first patch is subdivided until its components are within a tolerance of being flat; it is represented by planar regions approximating these components. The planes corresponding to these regions are intersected with the second patch, giving slices. These slices are trimmed at the edges of the planar regions, then they are glued together at their endpoints. Each connected string of trimmed slices is called an intersection branch.

8.85.1 PARAMETER LIST

1. ICC [In/Out, INTEGER scalar]
   The input ICC value indicates the type of call: ICC is negative for an initial call, when the convex hull of the first patch is not yet computed; ICC is positive for a continue call, when the convex hull was previously computed. The output ICC value is set to zero if an error occured while computing the convex hull; otherwise it is the absolute value of the input value.
2. PATCH [In, REAL(16,3,2)]
   The pair of patches to be intersected: PATCH (*,J,K) is the geometric-form representation of the bicubic for the J-th spatial coordinate for the K-th patch.
3. TOL [In, INTEGER(4)]
   Tolerance values: TOL(1) is used to control the gluing of trimmed slices together to make intersection branches. TOL(2) is used to determine when a subpatch is effectively flat. TOL(3) is the input tolerance for the plane/patch intersector. TOL(4) is the second tolerance computed by the plane/patch intersector.
4. IWRK [Temporary, INTEGER(*)]
   An array used during patch subdivision
5. NWRK [In, INTEGER scalar]
   The amount of space allocated for array IWRK: This must be at least NLEV + MXCRV + MXJCRV.
6. NLEV [In, INTEGER scalar]
   Maximum depth for the binary patch-subdivision tree: This must be at least 1.

7. PONT [Temporary, REAL(3, MXJPT, MXJCRV)]
   Array to hold spatial coordinates for plane/patch intersection results
8. MXJPT [In, INTEGER scalar]
   This must be at least 3.
9. MXJCRV [In, INTEGER scalar]
   This must be at least 2.
10. WORK2 [Temporary, REAL(3,MXIPT,MXICRV)]
    Array used by PLINT, the plane/patch intersector: The parametric representation of intersection curves computed by PLINT are stored in part of WORK2.
11. MXIT [In, INTEGER scalar]
    The maximum number of iterations for cubic polynomial solution within PLINT, the plane/patch intersector.
12. MXIPT [In, INTEGER scalar]
    Maximum number of points to be computed on an intersection slice: This must be at least 2.
13. MXICRV [In, INTEGER scalar]
    This must be at least 4. (MXICRV - 3 is the maximum number of intersection slices to be computed by a single call to PLINT, the plane/patche intersector.
14. HOLD [In/Out, INTEGER(*)]
    The planes defining the convex hull of the first patch: At least 120 values must be allocated for this array. This is input when ICC is positive, and it is output when the input value of ICC was negative.
15. MXPT [In, INTEGER scalar]
    The maximum number of points to be computed on an intersection branch: This must be at least 2.
16. MXCRV [In, INTEGER scalar]
    The maximum number of intersection branches to be computed: This must be at least 1.
17. NCRV [Out, INTEGER scalar]
    The number of intersection branches computed
18. CURVE [Out, REAL(3, MXPT,MXCRV)]
    The intersection branches: CURVE(1,1,K) is the number of points on the K-th branch. CURVE(I,J+1,K) is the I-th spatial coordinate of the J-th point on the K-th curve.
19. IER [Out, INTEGER scalar]
    Error code: This is zero for a successful execution. It is positive is the results are suspect. It is negative if a fatal error stopped processing.

    Fatal error codes are : (-1) An error was detected while finding the convex hull of patch 1, and ICC was set to zero; (-2) One of the three input tolerance values was less than the relative precision for this machine; (-3) MXJPT is less than three; (-4) MXJCRV is less than 2; (-5) MXICRV is less than four; (-6) NWRK is less than NLEV + MXCRV + MXJCRV; (-7) NLEV is less than 1; (-8) MXPT is less than 2; (-9) MXCRV is less than 1; (-10) Bad normals were produced for the convex hull; (-11) No extreme face could be found to initialize the convex hull algorithm; (-12) The convex hull contains more normals than the theoretical limit; (-13) The convex hull contains more edges than the theoretical limit; (-14) the convex hull is planar; (-15 to -28) PLINT produced an error code equal to IER + 13.

Nonfatal error codes are: (1) A subpatch does not satisfy the flatness tolerance after NLEV successive subdivisions.

8.85.2 COMMON DATA

none

8.86 SUBROUTINE SUFROT (from LIBS)

When tracing plane/patch intersections within Subroutine PLINT, computes the roots of the polynomial in T at the next S value and the partial derivatives of the bicubic at these roots.

8.86.1 PARAMETER LIST

1. SVAL [In, REAL(2)]
   Array containing the S values bounding the current interval for tracing roots: Roots at SVAL(2) are being calculated, and the previous roots are at SVAL(1)
2. HVAL [In, REAL scalar]
   Step size: This is the amount that S increases. (It is used to initialize the iteractive scheme in Subroutine CUBIC.
3. BC [In, REAL(4,4)]
   Coefficients for the output bicubic function, in algebraic form: BC(I,J) is the coefficient of $U^{(4-J)} * V^{(4-I)}$
4. HBND [In, REAL(2)]
   Horizontal (T) boundaries to the region of interest
5. IDEG [In, INTEGER scalar]
   Degree of the polynomial in T obtained when the bicubic is evaluated at the S value
6. NDEG [In, INTEGER scalar]
   The maximum degree of the polynomial in T obtained by ebaluating the bicubic at an arbitrary S value
7. IRT [In/Out, INTEGER(3)]
   Array containing the subscripts of the roots of a cubic equaton: U(IRT(1)) is the real root found via a Newton iterative scheme; U(IRT(2)) and U(IRT(3)) are the roots found from the quadratic factor.
8. IPT [In, INTEGER(3)]
   The number of points calculated on a zero curve: If IPT(I) = -1, then the curve passing through the root U(IT(I)) will not be traced.
9. TOL [In, REAL(2)]
   Array containing the user-specified tolerance: TOL(1) is the input tolerance. Any point on a parametric cubic curve interpolating the output (U,V) points is within TOL(1) distance (in parametric units) of a straight line between the surrounding output points. TOL(2) is computed by PLINT and is used as a lower limit for the stepsize.
10. IC [Out, INTEGER scalar]
    The sign of the discriminant for the quadratic equation obtained by deflating the cubic with the first root. (The quadratic equation has IC + 1 roots, so the cubic equation has IC + 2 roots.)
11. TRVAL [Out, REAL(3,2)]
    Array containing the values of the 3 T roots at the 2 S values from array SVAL
12. RPS [In/Out, REAL(3,2)]
    Array containing the partial derivative of the bicubic with respect to S at the 3 T roots at the 2 S values from array SVAL
13. RPT [In/Out, REAL(3,2)]
    Array containing the partial derivative of the bicubic with respect to T at the 3 T roots at the 2 S values from array SVAL
14. RPSS [In/Out, REAL(3,2)]
    Array containing the second partial derivative of the bicubic with respect to S at the 3 T roots at the 2 S values from arry SVAL
15. RPST [In/Out, REAL(3,2)]
    Array containing the mixed second partial derivative of the bicubic at the 3 T roots at the 2 S values from array SVAL
16. RPTT [In/Out, REAL(3,2)]
    Array containing the second partial derivative of the bicubic with respect to T at the 3 T roots at the 2 S values from array SVAL
17. IER [In/Out, INTEGER scalar)]
    Success/error code: The only possible error code is 1, which is passed along from Subroutine CUBIC.

8.86.2 COMMON DATA

none

8.87 SUBROUTINE TANSCL (from SUBS)

In Program MSHNRM, scales position ans surface-normal derivatives for parametric cubic interpolation along intersection curves.

8.87.1 PARAMETER LIST

1. OUTPUT [In, INTEGER scalar]
   Logical unit for listing output
2. X0 [In, REAL(3)]
   Initial position
3. DXDU0 [In/Out, REAL(3)]
   Initial position-derivative: The input direction is not changed, only the scale.
4. SN0 [In, REAL(3)]
   Initial surface-normal direction
5. DSNDU0 [In/Out, REAL(3)]
   Initial surface-normal derivative: The input direction is not changed, only the scale.
6. X1 [In, REAL(3)]
   Final position
7. DXDU1 [In/Out, REAL(3)]
   Final position-derivative: The input direction is not changed, only the scale.
8. SN1 [In, REAL(3)]
   Final surface-normal direction
9. DSNDU1 [In/Out, REAL(3)]
   Final surface-normal derivative: The input direction is not changed, only the scale.
10. IQSCAL [In, INTEGER scalar]
    Code to select a method to scale tangents for parametric cubic interpolation along MSHNRM intersection curves: Possible values are 1 (which gives locally explicit scaling) and 2 (which gives Ferguson-Phillips scaling, minimizing the integral of 2nd derivatives along the arc).
11. QERR [Out, LOGICAL scalar]
    Error flag

8.87.2 COMMON DATA

none

8.88 SUBROUTINE TRACE (from LIBS)

Within Subroutine PLINT, determines the step size required to satisfy the tolerance and performs a single step of tracing the roots from one S value to another.

8.88.1 PARAMETER LIST

1. BICOF [In, REAL(4,4)]
   Coefficients for the output bicubic function, in algebraic form: BC(I,J) is the coefficient of $U^{(4-J)} * V^{(4-I)}$.
2. TOL [In, REAL(2)]
   Array containing the user-sepcified tolerance: TOL(1) is the input tolerance. Any point on a parametric cubic curve interpolating the output (U,V) points is within TOL(1) distance (in parametric units) of a straight line between the surrounding output points. TOL(2) is computed by PLINT and is used as a lower limit for the stepsize.
3. BBNDY [In, REAL scalar]
   Maximum S value for the current tracing step

4. IDET [In, INTEGER scalar]
   The degree of the polynomial in T when the bicubic is evaluated at BBNDY
5. HBND [In, REAL(2)]
   Horizontal (T) boundaries to the region of interest
6. HBEPS [In, REAL(2)]
   Array containing horizontal boundaries slightly wider than the region of interest
7. NDEG [In, INTEGER scalar]
   The maximum degree of the polynomial in T obtained by evaluating the bicubic at an arbitrary S value
8. IPT [In, INTEGER(3)]
   The number of points calculated on a zero curve: If IPT(I) = -1, then the curve passing through the root U(IT(I)) will not be traced.
9. IT [In/Out, INTEGER(3)]
   Array containing the subscripts of the roots of a cubic equation: U(IT(1)) is the real root found via a Newton iterative scheme; U(IT(2)) and U(IT(3)) are the roots found from the quadratic factor.
10. SVAL [In/Out, REAL(2)]
    Array containing the S values bounding the current interval for tracing roots: Roots and partial derivatives at SVAL(2) are being calculated, and the previously-calculated roots and partial derivatives are at SVAL(1).
11. TRVAL [In/Out, REAL(3,2)]
    Array containing the roots of T at the S value in SVAL
12. IFG [Out, INTEGER scalar]
    Code for completion of tracing: 1 if SVAL(2) has reached BBNDY, 0 otherwise.
13. RPS [In/Out, REAL(3,2)]
    Array containing the partial derivative of the bicubic with respect to S at the 3 T roots at the 2 S values from array SVAL
14. RPT [In/Out, REAL(3,2)]
    Array containing the partial derivative of the bicubic with respect to T at the 3 T roots at the 2 S values from array SVAL
15. RPST [In/Out, REAL(3,2)]
    Array containing the mixed second partial derivative of the bicubic at the 3 T roots at the 2 S values from array SVAL
16. RPSS [In/Out, REAL(3,2)]
    Array containing the second partial derivative of the bicubic with respect to S at the 3 T roots at the 2 S values from array SVAL
17. RPTT [In/Out, REAL(3,2)]
    Array containing the second partial derivative of the bicubic with respect to T at the 3 T roots at the 2 S values from array SVAL
18. IC [Out, INTEGER scalar]
    The sign of the discriminant for the quadratic equation obtained by deflating the cubic with the first root. (The quadratic equation has IC + 1 roots, so the cubic equations has IC + 2 roots.)
19. IER [In/Out, INTERGER scalar]
    Success/error code: 1 is passed along from Subroutine CUBIC; 2 is returned when the step size required to follow the curve is less than TOL(2). (This case tends to indicate that some of the returned (S,T) data lies outside the unit square, which is not valid.)

8.88.2 COMMON DATA

none

8.89 LOGICAL FUNCTION TRMCHK (from SUBS)

Finds the next data line on an input file: System comments are skipped, and user comments are copied to an output file.

8.89.1 PARAMETER LIST

1. INPUT [In, INTEGER scalar]
   Logical unit for the input file
2. LINEIN [In, INTEGER scalar]
   Logical unit where the current data line is copied for reading within the calling program: 80 columns are copied.
3. OUTPUT [In, INTEGER scalar]
   Logical unit for the output file to which user comments are copied
4. DATA [Out, LOGICAL scalar]
   Indicates that a data line was found before the end-of-file was reached

8.89.2 COMMON DATA

none

8.90 SUBROUTINE TXTCPY (from SUBS)

Copies text lines: 80 characters are copied per line.

8.90.1 PATAMETER LIST

1. INPUT [In, INTEGER scalar]
   Logical unit for input
.2. OUTPUT [In, INTEGER scalar]
   Logical unit for output

8.90.2 COMMON DATA

none

8.91 SUBROUTINE UNIQRN (from SUBS)

Removes duplicates from a set of real N-tuples: The N-tuples are sorted by SORTRN and then adjacent pairs are tested for duplication within a tolerance.

8.91.1 PARAMETER LIST

1. LFERR [In, INTEGER scalar]
   Logical unit for error message output
2. NKEY [In, INTEGER scalar]
   Number of sorting keys and tolerances
3. IKEY [In, INTEGER(NKEY)]
   Key to precedence of components in sorting: These must either be between 1 and N or be between -N and -1. IKEY(I) controls the sorting of all N-tuples which have the previous I-1 components equal. If IKEY(I) is positive, these values are sorted in ascending order on their I-th component. If IKEY(I) is negative, these values are sorted in descending order on their I-th component.
4. TOL [In, REAL(NKEY)]
   Tolerances for recognizing effectively equal components: These must be positive. The I-th tolerance corresponds to the IKEY(I) component.
5. N [In, INTEGER scalar]
   Number of components in the N-tuples
6. NN [In, INTEGER scalar]
   Number of input N-tuples
7. LONG [Out, INTEGER scalar]
   Number of N-tuples after removing duplicates

8. A [In/Out, REAL(N,NN)]
   Array of N-tuples
9. QERR [Out, LOGICAL scalar]
   Error flag

8.91.2 COMMON DATA

none

8.92 REAL FUNCTION UPOLC (from LIBS)

Evaluates a coefficient of the cubic polynomial in T obtained by evaluating the bicubic at a given S value.

8.92.1 PARAMETER LIST

1. J [In, INTEGER scalar]
   The coefficient of $T^{(4-J)}$ in the polynomial is to be evaluated
2. X [In, REAL scalar]
   The S value where the polynomial is to be evaluated

8.92.2 COMMON DATA

COMMON /IRP1/ A(4,4)

The bicubic polynomial is input from A.

8.93 SUBROUTINE VARKNT (from LIBS)

Within Program REGSIL, optimizes knot placement on a set of spline curves: The curves are fit with first-degree continuous splines. The number of points is incremented until a parametric knot placement can keep the position error at the input points within a tolerance.

8.93.1 PARAMETER LIST

1. TOL [In, REAL scalar]
   Tolerance value
2. NPMAX [In, INTEGER scalar]
   Maximum number of knots
3. PKNOT [Out, REAL(*)]
   Knot placements: This in normalized arclength; it increases monotonically from 0 to 1.
4. ERRMAX [Out, ReAL(*)]
   Errors from the final knot placement: This is the maximum error for each curve.
5. IER [Out, INTEGER scalar]
   Error code: 0 for success; -1 for negative NCURV, NDIM, or NDIMC; -2 for an input curve with not enough points to define NBEG knots; -3 for an error in subroutine NW021.
6. NBEG [In, INTEGER scalar]
   Initial number of knots to try
7. NP [Out, INTEGER scalar]
   Number of knots
8. CYL [In, LOGICAL scalar]
   Flag for cylindrical coordinates
9. IXAXI [In, INTEGER scalar]
   Index for the axial cylindrical coordinate
10. IXRAD [In, INTEGER scalar]
    Index for the radial cylindrical coordinate
11. IXANG [In, INTEGER scalar]
    Index for the angular cylindrical coordinate

8.93.2 COMMON DATA

COMMON /CNTRL/ IL, EPS, DELX, MODE, MAXFN, KF, KORD
COMMON /MODSTF1/ CRV(800,4), WORK(9000)
COMMON /MODSTF2/ NCURV, NDIM, NDMIC, NPRT1, NPRT2, IFAIL
COMMON /MODSTF3/ LEN(17)

Common block /CNTRL/ primarily controls the optimization routine NW021. IL is the location parameter used by subroutine CBSPN. EPS is the error tolerance used by NW021. DELX is the step size used by NW021 for numerical differentation. MODE is used by NW021. MAXFN is the limit on the number of error function evaluations to be made by NW021. KF controls the listing output from NW021. KORD is the order of the splines used; it is 4, giving cubic splines.

CRV stores the coordinates of the input points and the chord lengths along the curves. WORK is used as temporary storage. NCURV is the number of curves. NDIM is the number of spatial dimensions; it is set to 3. NDIMC equals the length of CRV. LEN is the number of input points for each curve.

8.94 SUBROUTINE VERRN (from SUBS)

Verifies (i.e., aligns) a set of real N-tuples to a set of discrete values for each component: They are sorted before verification.

8.94.1 PARAMETER LIST

1. LFERR [In, INTEGER scalar]
   Logical unit for error message output
2. NKEY [In, INTEGER scalar]
   Number of components to be verified
3. IKEY [In, INTEGER(NKEY)]
   Key to precedence of components in sorting: These must either be between 1 and N or be between -N and -1. IKEY(I) controls the sorting of all N-tuples which have the previous I-1 components equal. If IKEY(I) is positive, these values are sorted in ascending order on their I-th component. If IKEY(I) is negative, these values are sorted in descending order on their I-th component.
4. TOL [In, REAL(4)]
   Tolerances for recognizing components near to verifying values: These must be positive. The I-th tolerance corresponds to the IKEY(I) component.
5. NSET [In, INTEGER(N)]
   Number of values in the verification set for each component: The index corresponds to the position of the component in the N-tuple.
6. MAXSET [In, INTEGER scalar]
   Number of variables allocated for each verification set
7. SET [In, REAL(MAXSET,N)]
   Sets of values for aligning components: The second index corresponds to the position of the component in the N-tuple.
8. N [In, INTEGER scalar]
   Number of components in each N-tuple
9. LONG [In, INTEGER scalar]
   Number of N-tuples
10. A [In/Out, REAL(N,LONG)]
    Array of N-tuples
11. QERR [Out, LOGICAL scalar]
    Error flag

8.94.2 COMMON DATA

none

8.95 REAL FUNCTION VIP (from LIBS)

Calculates the inner product of two vectors

8.95.1 PARAMETER LIST

1. A [In, REAL(*)]
   Vector 1
2. INCA [In, INTEGER scalar]
   The storage increment between the elements of A
3. B [In, REAL(*)]
   Vector 2
4. INCB [In, INTEGER scalar]
   The storage increment between the elements of B
5. N [In, INTEGER scalar]
   The number of elements in A and in B
6. C [Out, REAL scalar]
   Result

8.95.2 COMMON DATA

none

8.96 REAL FUNCTION VIPA (from LIBS)

Adds the inner product of two vectors to the input value of a variable

8.96.1 PARAMETER LIST

1. A [In, REAL(*)]
   Vector 1
2. INCA [In, INTEGER scalar]
   The storage increment between the elements of A
3. B [In, REAL(*)]
   Vector 2
4. INCB [In, INTEGER scalar]
   The storage increment between the elements of B
5. N [In, INTEGER scalar]
   The number of elements in A and in B
6. C [In/Out, REAL scalar]
   The variable to which the inner-product value is added

8.96.2 COMMON DATA

none

8.97 DOUBLE PRECISION FUNCTION VIPDS (from LIBS)

Calculates the inner product of two vectors with double-precision accumulation and subtracts this inner product from the input value of a variable

8.97.1 PARAMETER LIST

1. A [In, REAL(*)]
   Vector 1
2. INCA [In, INTEGER scalar]
   The storage increment between the elements of A

3. B [In, REAL(*)]
   Vector 2
4. INCB [In, INTEGER scalar]
   The storage increment between the elements of B
5. N [In, INTEGER scalar]
   The number of elements in A and in B
6. C [In/Out, REAL scalar]
   The variable from which the scalar-product value is subtracted

8.97.2 COMMON DATA

none

8.98 SUBROUTINE WRIPAT (from SUBS)

Writes a patch

8.98.1 PARAMETER LIST

1. LUNOUT [In, INTEGER scalar]
   Logical unit for patch output
2. PATCH [In, REAL(16,3)]
   Patch coefficients

8.98.2 COMMON DATA

none

8.99 SUBROUTINE WRTCRV (from SUBS)

In Program SRFINT, writes a surface/surface intersection curve: These are SIL-format section curves.

8.99.1 PARAMETER LIST

1. L6 [In, INTEGER scalar]
   Logical unit for listing output
2. L7 [In, INTEGER scalar]
   Logical unit for curve output
3. ICURVE [In, INTEGER(NP)]
   Indices to the endpoint table for the branches making up the curve: Each location appears twice, as a final endpoint for one branch and then as the initial endpoint for the next branch.
4. NP [In, INTEGER scalar]
   Number of endpoints for the branches making up the curve: This is an even number.
5. CURVE [Temporary, REAL(3,100)]
   Array to hold a branch of the curve
6. KEND [In, INTEGER scalar]
   SIL end-condition code to be written with the section. (End directions are not output.)

8.99.2 COMMON DATA

COMMON /DBVALU/ IPAT1, IPAT2, IBR, XYZ(3)
COMMON /ENDTBL/ X(201), Y(201), Z(201), IP1(201), IP2(201) NBR(201), KOUNT(201), IPOINT(201)
COMMON /SDMSER/ NERR

Common block /DBVALU/ contains the mapped variables for SDMS database keys and data values. Common block /SDMSER/ is used to report retreival of the end of an SDMS element set sequence (i.e., the end of a curve), which is handled like an error in SDMS. Common block /ENDTBL/ holds the endpoint table.

8.100 SUBROUTINE XTRACT (from SUBS)

Used by Program REGSIL to compute and write new points on a curve, from their parametric values and the old points

8.100.1 PARAMETER LIST

1. NSSLP [In, INTEGER(17)]
   End-condition flags for each curve
2. DXI [In, REAL(17)]
   X-components of initial end directions for each curve
3. DYI [In, REAL(17)]
   Y-components of initial end directions for each curve
4. DZI [In, REAL(17)]
   Z-components of initial end directions for each curve
5. DX0 [In, REAL(17)]
   X-components of final end directions for each curve
6. DY0 [In, REAL(17)]
   Y-components of final end directions for each curve
7. DZ0 [In, REAL(17)]
   Z-components of final end directions for each curve
8. X [In, REAL(17,175)]
   X-components of input points for each curve, for each point
9. Y [In, REAL(17,175)]
   Y-components of input points for each curve, for each point
10. Z [In, REAL (17.175)]
    Z-components of input points for each curve, for each point.
11. PKNOT [In, REAL(50)]
    Parameter values where output points will be interpolated on the interior of each curve
12. NP [In, INTEGER scalar]
    Number of interior points on the output curves
13. LABX [In, INTEGER(3)]
    Coordinates labels: This is character data
14. NLABX [In, INTEGER(3)]
    Number of characters in the coordinate labels
15. CYL [In, LOGICAL scalar]
    Flag for cylindrical coordinates
16. IXAXI [In, LOGICAL scalar]
    Index for the axial cylindrical coordinate
17. IXRAD [In, INTEGER scalar]
    Index for the radial cylindrical coordinate
18. IXANG [In, INTEGER scalar]
    Index for the angular cylindrical coordinate

8.100.2 COMMON DATA

```
COMMON /MODSTF2/ NCURV, NDIM, NDIMC, N1, N2, IFAIL
COMMON /MODSTF3/ LEN(17)
```

NCURV is the number of curves. LEN is the number of input points on each curve.

9.0 REFERENCES

1. Gibson, S.G., "User's Manual for MASTER: Modeling of Aerodynamic Surfaces by Three-Dimensional Explicit Representation," NASA CR166056, January 1983.
2. "NOS Version 1 Reference Manual," CDC Document 60435400, Rev. N, November 1981.
3. "FORTRAN Extended Version 4 Reference Manual," CDC Document 6049780, Rev. F, August 1980.
4. "UPDATE 1 Reference Manual," CDC Document 60449900, Rev. B, March 1978.
5. Baruah, P.K., et al., "PANAIR, Volume 4—Maintenance Document (Version 1.1)," NASA CR 3254, 1980.
6. "PLOT-10 Terminal Control System User's Manual," Tektronix Document 062-1474-00, 1972.
7. Dongerra, J.J., Moler, C.B., Bunch, J.R., and Stewart, G.W., "LINPACK User's Guide," Society for Industrial and Applied Mathematics (Philadelphia, PA), 1979.

1. Report No.	2. Government Accession No.	3. Recipient's Catalog No.
NASA CR-172244		

4. Title and Subtitle	5. Report Date
System Maintenance Manual for Master: Modeling of Aerodynamic Surfaces by Three-Dimensional Explicit Representation	April 1983
	6. Performing Organization Code: B-8411
7. Author(s): S. G. Gibson	8. Performing Organization Report No.: D6-51089
9. Performing Organization Name and Address: Boeing Commercial Airplane Company, P.O. Box 3707, Seattle, WA 98124	10. Work Unit No.: 4.3.3
	11. Contract or Grant No.: NAS1-15325
12. Sponsoring Agency Name and Address: National Aeronautics and Space Administration, Washington, D.C. 20546	13. Type of Report and Period Covered: Contractor Final Report
	14. Sponsoring Agency Code

15. Supplementary Notes

Langley Technical Monitor: David E. Reubush

16. Abstract

A system of computer programs has been developed to model general three-dimensional surfaces. Surfaces are modeled as sets of parametric bicubic patches. There are also capabilities to transform coordinates, to compute mesh/surface intersection normals, and to format input data for a transonic potential flow analysis (NASA CR-3514). A graphical display of surface models and intersection normals is available. There are additional capabilities to regulate point spacing on input curves and to compute surface/surface intersection curves.

Internal details of the implementation of this system are explained, and maintenance procedures are specified. (Data formats and operating instructions are referenced from NASA CR-166056, which is the user's manual for the system.)

17. Key Words		18. Distribution Statement
bicubic patch	parametric	Unclassified - Unlimited
database manager	surface model	
interactive graphics	three-dimensional	Subject Category 61
	version control	

19. Security Classif. (of this report)	20. Security Classif. (of this page)	21. No. of Pages	22. Price
Unclassified	Unclassified	124	A06

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