FINAL REPORT

SOFTWARE TO MODEL AXAF IMAGE QUALITY

Contract Number
NAS8-38609
Delivery Order Number
39

Prepared for:

NASA/MSFC
Huntsville, Alabama 35812

Prepared by:

CENTER FOR APPLIED OPTICS
University of Alabama in Huntsville
Huntsville, Alabama 35899

June 1993

(NASA-CR-192574) SOFTWARE TO MODEL AXAF IMAGE QUALITY Final Report, 1 May 1992 - 29 Jun. 1993 (Alabama Univ.) 318 p
N93-31637
Unclas
G3/61 0172860

TABLE OF CONTENTS

1. ABSTRACT

This draft final report describes the work performed under this delivery order from May 1992 through June 1993. The purpose of this contract was to enhance and develop an integrated optical performance modeling software for complex X-ray optical systems such as AXAF. The GRAZTRACE program developed by the MSFC Optical Systems Branch for modeling VETA-I was used as the starting baseline program. The original program was a large single file program and, therefore, could not be modified very efficiently. The original source code has been reorganized, and a "Make Utility" has been written to update the original program. The new version of the source code consists of 36 small source files to make it easier for the code developer to manage and modify the program. A user library has also been built and a "Makelib" utility has been furnished to update the library. With the user library, the users can easily access the GRAZTRACE source files and build a custom library. A user manual for the new version of GRAZTRACE has been compiled.

The plotting capability for the 3-D point spread functions and contour plots has been provided in the GRAZTRACE using the graphics package DISPLAY. The Graphics emulator over the network has been set up for programming the graphics routine. The point spread function and the contour plot routines have also been modified to display the plot centroid, and to allow the user to specify the plot range, and the viewing angle options.

A Command Mode version of GRAZTRACE has also been developed. More than 60 commands have been implemented in a Code-V like format. The functions covered in this version include data manipulation, performance evaluation, and inquiry and setting of internal parameters. The user manual for these commands has been formatted as in Code-V, showing the command syntax, synopsis, and options. An interactive on-line help system for the command mode has also been accomplished to allow the user to find valid commands, command syntax, and command function.

A translation program has been written to convert FEA output from structural analysis to GRAZTRACE surface deformation file (.dfm file). The program can accept standard output files and list files from COSMOS/M and NASTRAN finite analysis programs. Some interactive options are also provided, such as Cartesian or cylindrical coordinate transformation, coordinate shift and scale, and axial length change.

A computerized database for technical documents relating to the AXAF project has been established. Over 5000 technical documents have been entered into the master database. A user can now rapidly retrieve the desired documents relating to the AXAF project.

The summary of the work performed under this contract is shown in Figure 1.

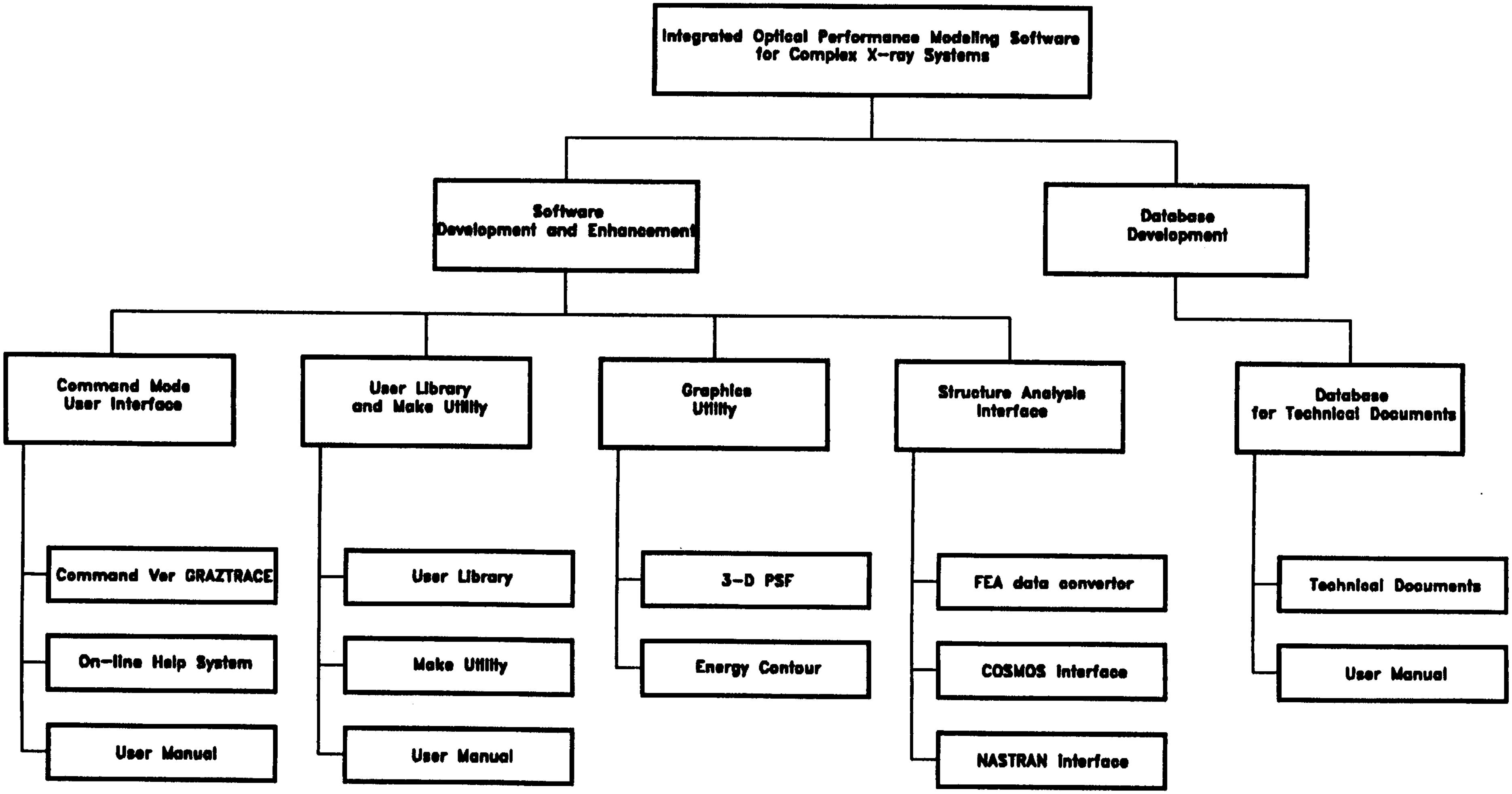


Figure 1 Summary of the Work Performed

2. INTRODUCTION

The work performed under this delivery order by the Center for Applied Optics (CAO) at the University of Alabama in Huntsville includes the development of various software modules to predict the optical performance and image quality of AXAF. The goal of this modeling effort was to take into account the effects of optical, structural and thermal distortions, as well the metrology errors in optical surfaces to predict the performance of a large and complex optical system such as Advanced X-ray Astrophysics Facility (AXAF). The objective was to make the modeling software user-friendly and well documented so that it can be used conveniently by the users, who may not be intimately familiar and experienced in x-ray optical analysis.

A number of meetings were held with the Optical Branch technical staff to discuss the structure and other details of the software to be developed. UAH was assigned to implement this software on Sun workstations, and to document the software, provide graphical output capability, and make it user-friendly.

The GRAZTRACE program was developed by MSFC Optical Systems Branch for modeling VETA-I. As this x-ray optical analysis program had proven to give reliable results, it was decided to use this program as the baseline for the modeling software effort. A direct network link was established between the CAO computers and the Sun workstations at the Optical Branch, using an ethernet card and the network software CUTCP. A separate account for CAO was established on the Sun for the software development work. These arrangements made it possible for CAO to access ZORRO and ZEUS computers at MSFC. Two options for this connection have been established:

1. PC-Ethernet with PC-TCP/IP software

    Directly access ZORRO using PC ethernet card "Elite Plus" and the software CUTCP by typing:

    telnet zorro.msfc.nasa.gov

    or

    telnet 128.158.21.11

2. UAH network

    Access ZORRO through UAH network using serial port and common communication software by logging into UAH network and then calling telnet.

3. GRAZTRACE: X-ray Optical Analysis Program

A significant number of useful features have been incorporated into the GRAZTRACE software to make it more useful and user friendly. A summary of these features is described below.

3.1 The "Make" utility and GRAZTRACE User Library:

The original GRAZTRACE was a single file program. A single large file makes the modification of the program quite inefficient. To improve this with the "Make" utility, the program source file was split into 62 small source files. Then some of the files were combined to produce the new GRAZTRACE source code, which consists of 36 files.

The GRAZTRACE user library has been built. The utility to rebuild the library has also been furnished and is called "makelib". This utility finds all the specified object files and builds a user specific library called "libgtrac.a".

"Make" files for both the GRAZTRACE developer and the user have been developed, with the names: "gtmakefile" and "ugtmakefile". The users can simply modify their own routine "main.f" and "user.f", and then make and execute the program.

The complete description of the "Make" utility and the "User Library" is attached as Appendix 1.

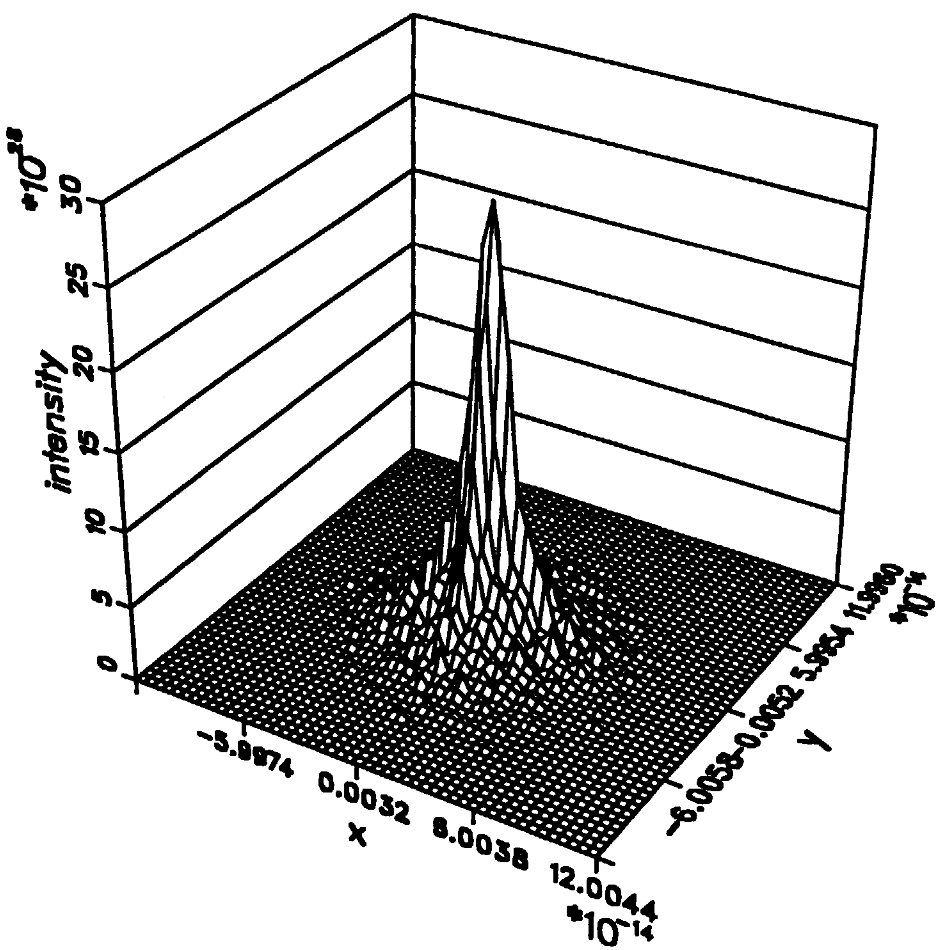
3.2 The User Manual for GRAZTRACE Library:

The user manual for GRAZTRACE library has been written. The manual provides a sample session for a new user to get a quick start. It also contains information about the compiler, the Make utility and the data format. The manual covers a total of over 60 user accessible routines and the code developer accessible routines.

This user manual is attached as Appendix 2.

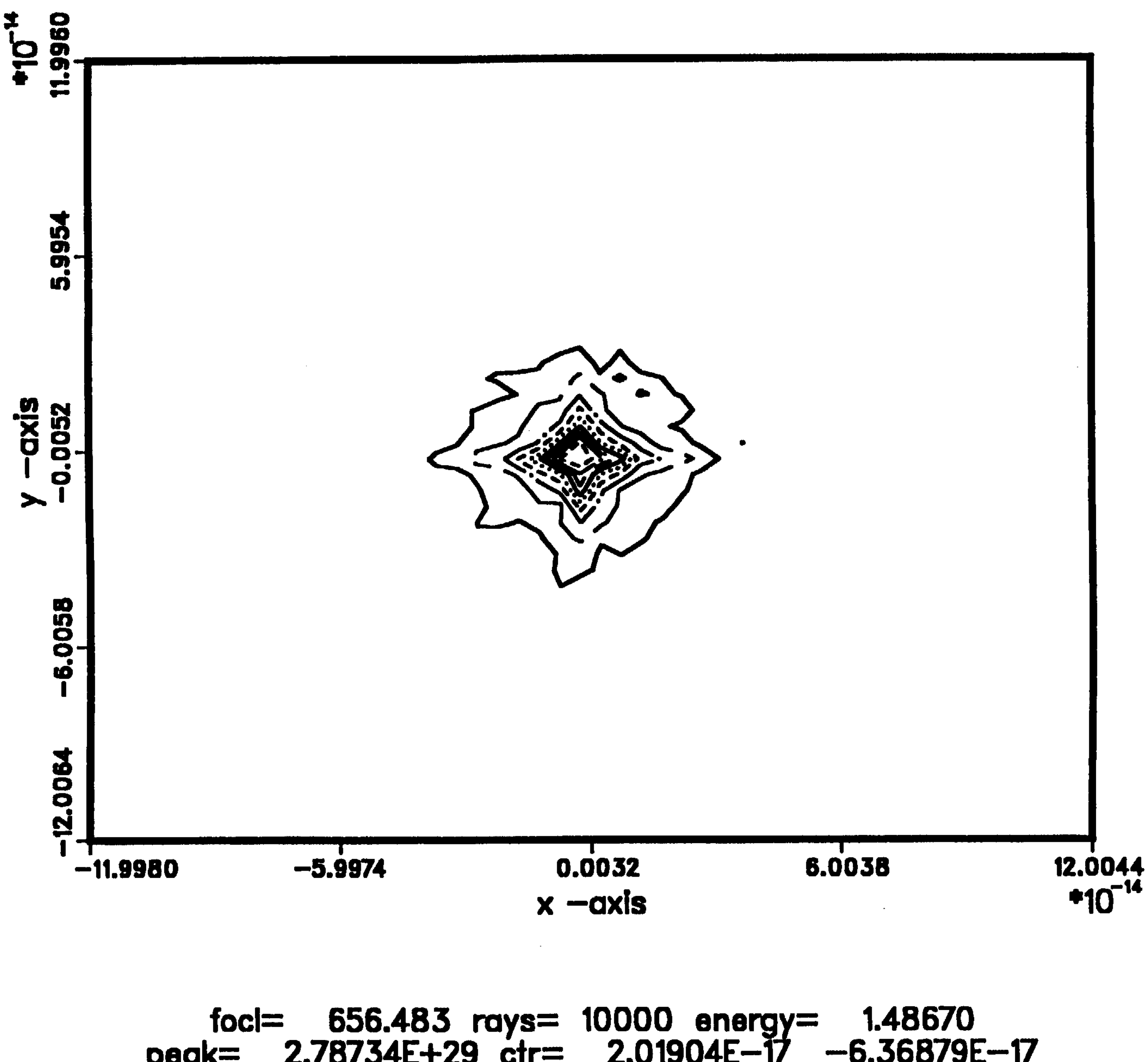
3.3 The Graphics Features:

The plotting capability for the 3-D point spread functions and contour plots has been provided in the GRAZTRACE using the graphics package DISPLAY. The graphics emulator over the network has been set up for programming the graphics routine. The point spread function and the contour routines have also been modified to display the plot centroid, and to allow the user to specify the plot range, and the viewing angle options. Figures 2 and 3 are sample 3-D point spread function and contour plots. The plot routines for these graphic features are attached as Appendix 3.



focl= 656.483 rays= 10000 energy= 1.48670
peak= 2.78734E+29 ctr= 2.01904E-17 -6.36879E-17

Figure 2 3-D Point Spread Function



*10^-14
11.9960
5.9954
-0.0052
-6.0058
-12.0064
y -axis
-11.9980
-5.9974
0.0032
6.0038
12.0044
*10^-14
x -axis
focl= 656.483 rays= 10000 energy= 1.48670
peak= 2.78734E+29 ctr= 2.01904E-17 -6.36879E-17

4. THE COMMAND MODE GRAZTRACE

UAH was asked by the Optical Systems Branch to implement the "command mode" input, similar to CodeV software, to improve the user interface for the GRAZTRACE software. The command mode user interface has been implemented and tested. The commands have CodeV-like structure. Most commands have exact format and function as those in CodeV. Using the command interface, a user can input, change, and inquire all the system parameters as well as perform the analysis. The command error process and correct command syntax prompts are also included.

To accomplish the command mode user interface, an interpreter program has been written, and the main GRAZTRACE program has also modified to accommodate the command mode. This program consists of extensive FORTRAN code (more than 2000 lines). The page layout for the user manual is also like that in CodeV. Each page includes the command summary, mnemonic and input option description. The detail explanation of each option and the examples of inputs have also been added to this documentation.

This highly structured program allows the code developer to easily modify or upgrade the command mode. Some modifications have also been made to enhance the program, such as various ray trace patterns, ray trace data save, and array variable inquiries.

An interactive on-line help system has also been furnished to allow the user to find the valid command, command syntax, and command function.

The user manual for command mode GRAZTRACE is attached as Appendix 4. The complete source code and documentation are included as Appendix 5.

5. STRUCTURAL ANALYSIS INTERFACE

One of the goals of this project was to develop a convenient method for inputting the structural deformation data into the GRAZTRACE to predict the effects of structural distortions on the image quality. A general purpose translation program has been written to convert the outputs from finite element analysis (FEA) programs to GRAZTRACE deformation file (.dmf file). The program can accept deformation data from most commonly used FEA programs such as COSMOS/M and NASTRAN (standard file from NASTRAN and list file from COSMOS/M). The translation program is structured in such a way that other deformation file formats can be also be integrated easily to accept FEA outputs from other stuctural analysis programs. This translation program is quite flexible, and does not require the structural analysis data to be uniformly spaced, or to have a fixed number of grid points. The grid points can also be in a random order. Therefore, the structural analyst has complete freedom to select the number, spacing and the order of grid points to optimize the structural analysis.

The translation program provides some interactive options also, such as cartesian or cylindrical coordinate translation, coordinate shift and scaling, and the axial length change. After the FEA output file is read in, the program interpolates the data to a uniform grid, 201 points in the axial direction and 1001 points along the circumference.

This translation program has been tested to evalute the effects of structural deformation on a sample SXI mirror. Test runs were made to determine the impact of structural distortions on the image quality. Figures 4 and 5 are structural analysis plots modeled by COSMOS/M for the SXI mirror. Figures 6 to 8 show the spot diagrams for the perfect mirror, and the spot diagrams for the deformed mirror as predicted by COSMOS/M, and NASTRAN. These spot diagrams were generated by using the command mode GRAZTRACE. In a similar way, the command mode GRAZTRACE can also accept the deformations from metrology data to predict the image degradation caused by low frequency surface errors.

The source code for this structural analysis interface and the sample deformation files are shown in Appendix 6.

6. AXAF PROJECT DATABASE

The purpose of this task was to establish a project database to enable the users to rapidly access the technical information relating to AXAF project. Initially, the technical data was reviewed by UAH to establish the guidelines for relational database to orchestrate and retrieve the technical programmatic information. Once the guidelines were established, research was done on the software options for the Sun System — UNIX Operating System to make recommendations for the technical data management system. Based on the requirements for the relational database, a decision was made to use the Text Editor on the Sun System — UNIX Operating System for tracking the documents and to use the GREP search command to allow the user to search for the documents.

Documentation was organized according to the categories of the project which serve as the key field for locating the data. For example, there are several subcategories under the AXAF Project such as AXAF-HDOS, AXAF-TRW, AXAF-SAO, and AXAF-SCHOTT. Each category document was labeled with the project subcategory, document title and the location of the file. Table 1 shows all the project files.

After the documents were categorized, over 5000 documents were entered in the master data base called "FILE%." If the technical staff needs to locate a document, the FILE% master file is searched by using the "grep" search command as shown in Table 2 and

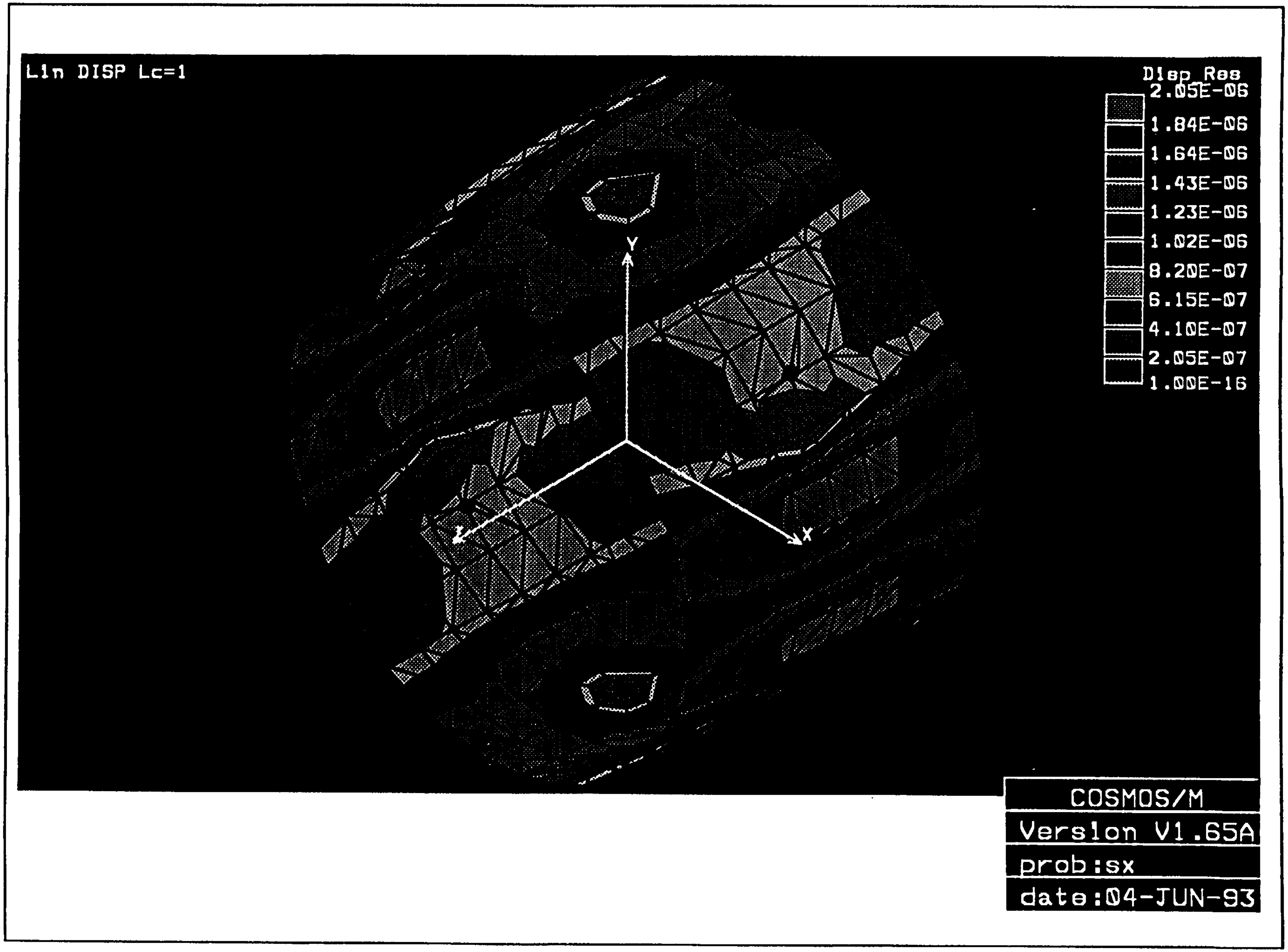
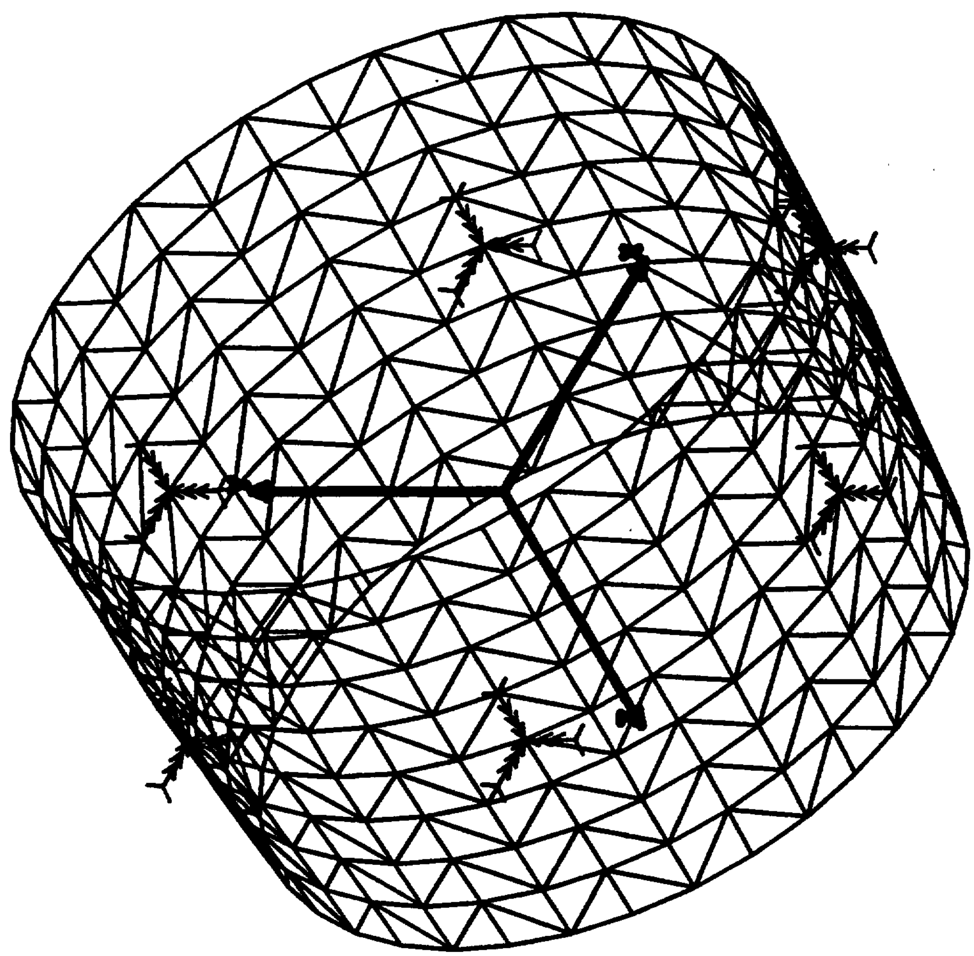


Figure 4 Surface Deformation



Ideal Surface

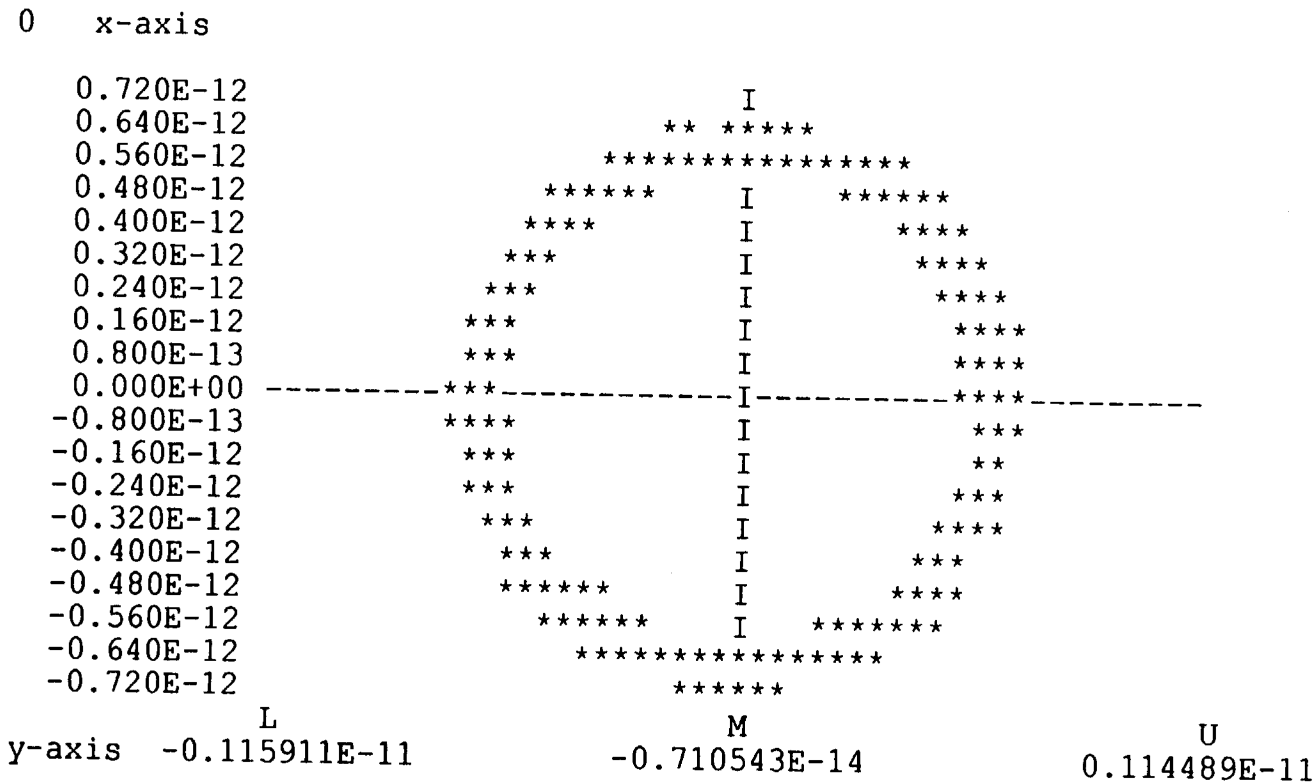


Figure 6 Spot Diagram for the Perfect Mirror Surface

Deformation from COSMOS/M

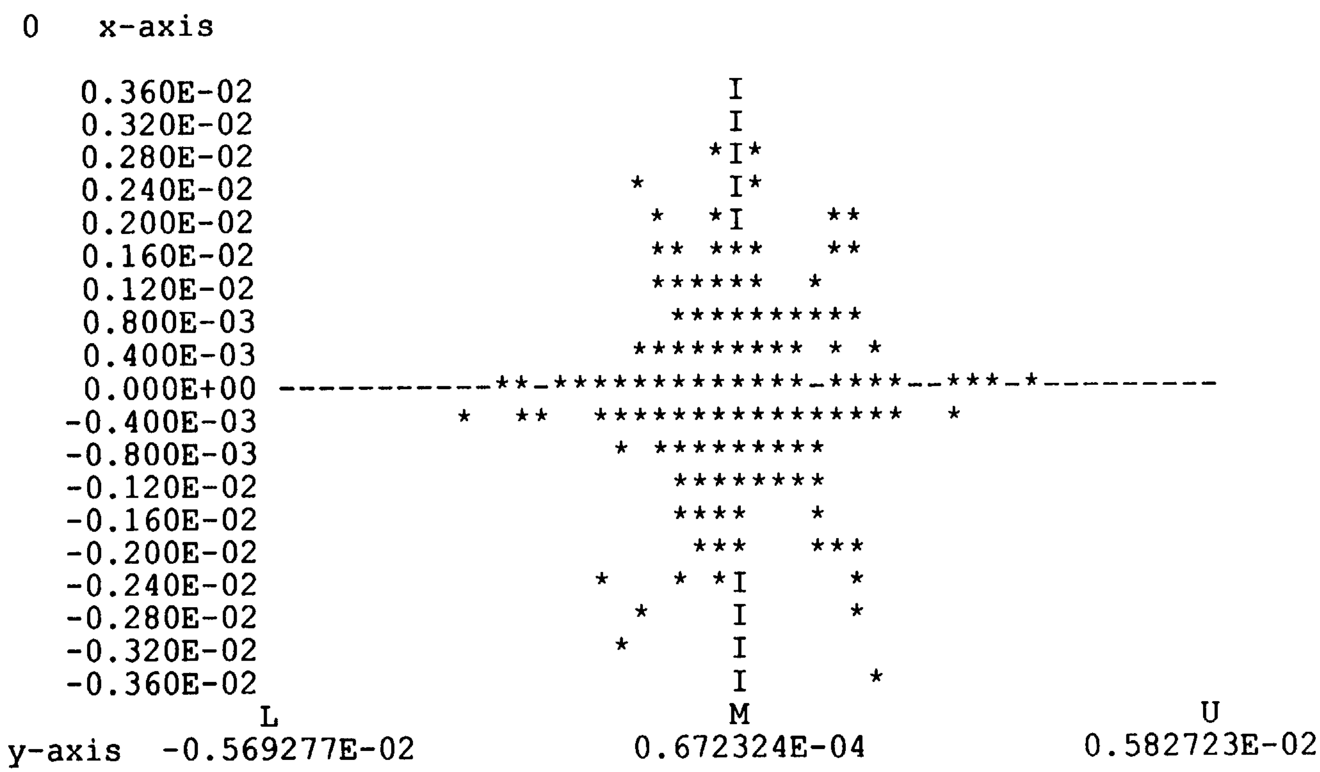


Figure 7 Spot Diagram for the Deformation from COSMOS/M

Deformation from NASTRAN

```
 0   x-axis

    0.540E-02                         I
    0.480E-02                        *I
    0.420E-02                         I
    0.360E-02                    *    I
    0.300E-02                        *I*
    0.240E-02                     ** *I*
    0.180E-02                      ****  **
    0.120E-02                      **** ***
    0.600E-03                     * ***** *
    0.000E+00 ---------------*************_******---------------
   -0.600E-03                 ** ************   *
   -0.120E-02                     * *******
   -0.180E-02                       *** **
   -0.240E-02                     * ***  **
   -0.300E-02                    *    I   *
   -0.360E-02                         I
   -0.420E-02                    *   *I    *
   -0.480E-02                         I
   -0.540E-02                         I
              L                       M                       U
y-axis  -0.845256E-02            0.187438E-03           0.882744E-02
```

Figure 8 Spot Diagram for the Deformation from NASTRAN

Table 3. Basically, what the command asks is to search for documents with the string "REQUIREMENTS REVIEW" in Table 3, "AD-FIG-TECH" in Table 4. Any document with that string is moved into the OUT file or the venus prompt.

The "OUT" file generates the AXAF-TECHNICAL DATABASE LIST in alphabetical order which is placed in the Technical Data Base book list. However, if a list is not needed, at the venus prompt the request for the document will be listed on the computer screen as shown in Table 4. This command is used if you are searching for a particular document, and there is not a need to print a detailed listing of documents.

For an information system to be useful and adequately meet the objectives of easy data retrieval, a user manual is needed and the users must be trained properly. The Optical Branch staff was trained on the layout and management of the data base, and the commands used to retrieve documentation as needed. The files and manuals will be provided to the staff to refer to on the use of the Technical Data Base. The User Manual for the AXAF project documentation database is attached as Appendix 8.

OPTICAL SYSTEMS BRANCH TECHNICAL FILES

2,1	3,1	4,1	5.1	8,1	HALL	
VIEWGRAPHS	AXAF-SCHOTT PROJECT FILES	AXAF-HDOS PROJECT FILES	LAWS PROJECT FILES EOS	AXAF-HDOS SPECS	SLENE PICTURES	
0SSA-OAST	AXAF-SCHOTT CONT. AXAF-PROJECT FILES	AXAF-HDOS PROJECT FILES	LASERS/GEN. AXAF-KODAK VETA	AXAF-HDOS SPECS	AXAF-I AXAF-S	
0SSA-OAST	AXAF-PROJECT FILES	AXAF-HDOS PROJECT FILES	LAWS-GE	AXAF-HDOS SPECS	OPTICS TECHNOLOGY	
0SSA-OAST	AXAF-TRW AXAF-SAO	AXAF-HDOS PROJECT FILES	LAWS LOCKHEED	SMALL PROJECTS	OPTICS TECHNOLOGY LIDAR	
WORLDWIDE TRAVEL INFORMATION	AXAF SAO CONT.	CORPORATION SBIR UNIVERSITY INFORMATION	BACKSCATTER BEST, WINDSTAT	SMALL PROJECTS	SRR	AXAF-TMA

Table 1.

Table 2

OUTPUT LISTED AS A RESULT OF THE UNIX COMMAND

Unix Command:

venus{wanda}45>grep AXAF REQUIREMENTS REVIEW FILE%>>OUT

AXAF TECHNICAL DATABASE

PROJECT	DOCUMENT	FILE
AXAF	AXAF HRC AND REQUIREMENTS REV	AXAF-PJ**
AXAF	SYSTEMS REQUIREMENTS REVIEW	AXAF-PJ
AXAF-I	LEVEL II PROJECT REQUIREMENTS	AXAF-PJ

**NOTE: AXAF-PJ is the acronym for AXAF PROJECT FILES

Table 3

TECHNICAL DATABASE = 'AD-FIG-TECH'

SPECIAL HEADINGS	DOCUMENTS	FILE
	ADAPTIVE OPTICS	TECHNOLOGY
ADV-FIG-TECH	<PACE> PLASMA ASSISTED CHEMICAL ETCHING	TECHNOLOGY
ADV-FIG-TECH	CCP EDGE FIGURING	TECHNOLOGY
ADV-FIG-TECH	ELECTRO FORMED X-RAY MIRRORS	TECHNOLOGY
ADV-FIG-TECH	ION FIGURING - NO INFO IN FILE	TECHNOLOGY

Table 4

Unix Command:

venus{wanda}46>grep AXAF REQUIREMENTS REVIEW FILE%

PROJECT	DOCUMENT	FILE
AXAF	LEVEL II PROJECT REQUIREMENTS	AXAF-PJ
AXAF	SYSTEMS REQUIREMENTS REVIEW	AXAF-PJ
AXAF	HRC AND REQUIREMENTS REV	AXAF-PJ

7. CONCLUSIONS

Significant progress has been made towards completing the tasks identified in the scope of work for this delivery order. The software required to model the AXAF optical performance has been developed. Many useful features such as graphics, "Make" utility, user library, command mode version of the GRAZTRACE analysis program, and structural analysis interface have been developed. All software has been properly organized and documented for user friendliness.

The command mode version of GRAZTRACE has similar command structure as in CODE V, so a user who is familiar with the CODE V optical analysis program will be able to perform optical analysis of X-ray systems such as AXAF. The graphics feature allows the plotting of 3-D point spread functions and energy distribution control plots. The "Make" utility and a user library have been developed to allow the customization of the program for specific applications. The structural analysis interface can extract deformation data from some major FEA program outputs and convert them to GRAZTRACE deformation format. The user manuals for the original GRAZTRACE and for the command mode version of GRAZTRACE have been compiled.

Sample user sessions showing the command mode, the interactive help system and the effects of structural deformations on the image quality are shown in Appendix 7.

The AXAF technical documents have been organized and entered into a SUN database. The technical information has been organized systematically and the file structures were tested. A total of over 5000 AXAF documents have been entered into the database. A user manual, which explains the procedure to search for a particular type of documents, has been prepared. The Optical Systems Branch staff has also been trained in the use of the database, the file structure and the terminology used for the master database.

APPENDIX 1

MAKE UTILITY AND USER LIBRARY

Appendix 1 Make utility and User Library

A1.1 gtmakefile Make file for GRAZTRACE code developer

```
###########################################################################
# makefile: gtmakefile
#
# This is the make file for the graztrace
###########################################################################

GTOBJ=/home/chen/obj
GTEXE=/home/chen/exe

LIBS=
CMPL=f77
OPTS=-g

FILE= ${GTOBJ}/calcdb.o \
     ${GTOBJ}/calwgt.o \
     ${GTOBJ}/coord.o \
     ${GTOBJ}/deltbl.o \
     ${GTOBJ}/dproc.o \
     ${GTOBJ}/eescat.o \
     ${GTOBJ}/encirc.o \
     ${GTOBJ}/focus.o \
     ${GTOBJ}/grid1.o \
     ${GTOBJ}/grid2.o \
     ${GTOBJ}/main.o \
     ${GTOBJ}/mcomm.o \
     ${GTOBJ}/metref.o \
     ${GTOBJ}/misce.o \
     ${GTOBJ}/pfocus.o \
     ${GTOBJ}/rdref.o \
     ${GTOBJ}/red.o \
     ${GTOBJ}/rgene.o \
     ${GTOBJ}/rprint.o \
     ${GTOBJ}/spdiag.o \
     ${GTOBJ}/splot.o \
     ${GTOBJ}/ssrt.o \
     ${GTOBJ}/stat.o \
     ${GTOBJ}/strace.o \
     ${GTOBJ}/user.o \
     ${GTOBJ}/utrace.o \
     ${GTOBJ}/vcalc.o \
     ${GTOBJ}/vignet.o \
     ${GTOBJ}/wray.o \
     ${GTOBJ}/wrayso.o  \
     ${GTOBJ}/wraysv.o \
```

```
     ${GTOBJ}/wspot1.o \
     ${GTOBJ}/wspot2.o \
     ${GTOBJ}/wstat.o \
     ${GTOBJ}/xalign.o \
     ${GTOBJ}/yintp.o

${GTEXE}/gtrac: ${FILE}
     ${CMPL} ${OPTS} ${FILE} ${LIBS} -o ${GTEXE}/gtrac

include gtmakerules

clean: ${GTEXE}/gtrac
      strip ${GTEXE}/gtrac
```

A1.2 gtmakerule Make rule for GRAZTRACE

```
##########################################################################
# makerules: gtmakerules
#
# This is the make rules for the graztrace
##########################################################################

GTSRC=/home/chen/src
GTOBJ=/home/chen/obj

CMPL=f77
OPTS=-g

${GTOBJ}/calcdb.o : ${GTSRC}/calcdb.f
     ${CMPL} -c ${OPTS} ${GTSRC}/calcdb.f
     mv calcdb.o ${GTOBJ}

${GTOBJ}/calwgt.o :${GTSRC}/calwgt.f
     ${CMPL} -c ${OPTS} ${GTSRC}/calwgt.f
     mv calwgt.o ${GTOBJ}

${GTOBJ}/coord.o :${GTSRC}/coord.f
     ${CMPL} -c ${OPTS} ${GTSRC}/coord.f
     mv coord.o ${GTOBJ}

${GTOBJ}/deltbl.o : ${GTSRC}/deltbl.f
     ${CMPL} -c ${OPTS} ${GTSRC}/ deltbl.f
     mv deltbl.o ${GTOBJ}

${GTOBJ}/dproc.o :${GTSRC}/dproc.f
     ${CMPL} -c ${OPTS} ${GTSRC}/dproc.f
     mv dproc.o ${GTOBJ}

${GTOBJ}/eescat.o :${GTSRC}/eescat.f
     ${CMPL} -c ${OPTS} ${GTSRC}/ eescat.f
     mv eescat.o ${GTOBJ}

${GTOBJ}/encirc.o :/${GTSRC}/encirc.f
     ${CMPL} -c ${OPTS} ${GTSRC}/encirc.f
     mv encirc.o ${GTOBJ}

${GTOBJ}/focus.o :${GTSRC}/focus.f
     ${CMPL} -c ${OPTS} ${GTSRC}/focus.f
     mv focus.o ${GTOBJ}

${GTOBJ}/grid1.o :${GTSRC}/grid1.f
     ${CMPL} -c ${OPTS} ${GTSRC}/grid1.f
     mv grid1.o ${GTOBJ}
```

```
${GTOBJ}/grid2.o :${GTSRC}/grid2.f
	${CMPL} -c ${OPTS} ${GTSRC}/grid2.f
	mv grid2.o ${GTOBJ}

${GTOBJ}/main.o :${GTSRC}/main.f
	${CMPL} -c ${OPTS} ${GTSRC}/main.f
	mv main.o ${GTOBJ}

${GTOBJ}/mcomm.o :${GTSRC}/mcomm.f
	${CMPL} -c ${OPTS} ${GTSRC}/mcomm.f
	mv mcomm.o ${GTOBJ}

${GTOBJ}/metref.o :${GTSRC}/metref.f
	${CMPL} -c ${OPTS} ${GTSRC}/metref.f
	mv metref.o ${GTOBJ}

${GTOBJ}/misce.o :${GTSRC}/misce.f
	${CMPL} -c ${OPTS} ${GTSRC}/misce.f
	mv misce.o ${GTOBJ}

${GTOBJ}/pfocus.o :${GTSRC}/pfocus.f
	${CMPL} -c ${OPTS} ${GTSRC}/pfocus.f
	mv pfocus.o ${GTOBJ}

${GTOBJ}/rdref.o :${GTSRC}/rdref.f
	${CMPL} -c ${OPTS} ${GTSRC}/rdref.f
	mv rdref.o ${GTOBJ}

${GTOBJ}/red.o :${GTSRC}/red.f
	${CMPL} -c ${OPTS} ${GTSRC}/red.f
	mv red.o ${GTOBJ}

${GTOBJ}/rgene.o :${GTSRC}/rgene.f
	${CMPL} -c ${OPTS} ${GTSRC}/rgene.f
	mv rgene.o ${GTOBJ}

${GTOBJ}/rprint.o :${GTSRC}/rprint.f
	${CMPL} -c ${OPTS} ${GTSRC}/rprint.f
	mv rprint.o ${GTOBJ}

${GTOBJ}/spdiag.o :${GTSRC}/spdiag.f
	${CMPL} -c ${OPTS} ${GTSRC}/spdiag.f
	mv spdiag.o ${GTOBJ}

${GTOBJ}/splot.o :${GTSRC}/splot.f
	${CMPL} -c ${OPTS} ${GTSRC}/splot.f
	mv splot.o ${GTOBJ}

${GTOBJ}/ssrt.o :${GTSRC}/ssrt.f
	${CMPL} -c ${OPTS} ${GTSRC}/ssrt.f
	mv ssrt.o ${GTOBJ}

${GTOBJ}/stat.o :${GTSRC}/stat.f
```

```
     ${CMPL} -c ${OPTS} ${GTSRC}/stat.f
     mv stat.o ${GTOBJ}

${GTOBJ}/strace.o :${GTSRC}/strace.f
     ${CMPL} -c ${OPTS} ${GTSRC}/strace.f
     mv strace.o ${GTOBJ}

${GTOBJ}/user.o :${GTSRC}/user.f
     ${CMPL} -c ${OPTS} ${GTSRC}/user.f
     mv user.o ${GTOBJ}

${GTOBJ}/utrace.o :${GTSRC}/utrace.f
     ${CMPL} -c ${OPTS} ${GTSRC}/utrace.f
     mv utrace.o ${GTOBJ}

${GTOBJ}/vcalc.o :${GTSRC}/vcalc.f
     ${CMPL} -c ${OPTS} ${GTSRC}/vcalc.f
     mv vcalc.o ${GTOBJ}

${GTOBJ}/vignet.o :${GTSRC}/vignet.f
     ${CMPL} -c ${OPTS} ${GTSRC}/vignet.f
     mv vignet.o ${GTOBJ}

${GTOBJ}/wray.o :${GTSRC}/wray.f
     ${CMPL} -c ${OPTS} ${GTSRC}/wray.f
     mv wray.o ${GTOBJ}

${GTOBJ}/wrayso.o :${GTSRC}/wrayso.f
     ${CMPL} -c ${OPTS} ${GTSRC}/wrayso.f
     mv wrayso.o ${GTOBJ}

${GTOBJ}/wraysv.o :${GTSRC}/wraysv.f
     ${CMPL} -c ${OPTS} ${GTSRC}/wraysv.f
     mv wraysv.o ${GTOBJ}

${GTOBJ}/wspot1.o :${GTSRC}/wspot1.f
     ${CMPL} -c ${OPTS} ${GTSRC}/wspot1.f
     mv wspot1.o ${GTOBJ}

${GTOBJ}/wspot2.o :${GTSRC}/wspot2.f
     ${CMPL} -c ${OPTS} ${GTSRC}/wspot2.f
     mv wspot2.o ${GTOBJ}

${GTOBJ}/wstat.o :${GTSRC}/wstat.f
     ${CMPL} -c ${OPTS} ${GTSRC}/wstat.f
     mv wstat.o ${GTOBJ}

${GTOBJ}/xalign.o :${GTSRC}/xalign.f
     ${CMPL} -c ${OPTS} ${GTSRC}/xalign.f
     mv xalign.o ${GTOBJ}

${GTOBJ}/yintp.o :${GTSRC}/yintp.f
     ${CMPL} -c ${OPTS} ${GTSRC}/yintp.f
```

```
mv yintp.o  ${GTOBJ}
```

A1.3 Make file for GRAZTRACE User

```
##########################################################################
# makefile: ugtmakefile
#
# This is the make file for user to use graztrace
##########################################################################

GTSRC=.
GTOBJ=.
GTEXE=.
GTLIB=/home/chen/lib

LIBS=-L${GTLIB} -lgtrac
CMPL=f77
OPTS=-g

FILE= ${GTOBJ}/main.o    \
     ${GTOBJ}/user.o

${GTEXE}/ugtrac: ${FILE}
     ${CMPL} ${OPTS} ${FILE} ${LIBS} -o ${GTEXE}/ugtrac

${GTOBJ}/main.o: ${GTSRC}/main.f
     ${CMPL} -c ${OPTS} ${GTSRC}/main.f

${GTOBJ}/user.o:${GTSRC}/user.f
     ${CMPL} -c ${OPTS} ${GTSRC}/user.f

clean: ${GTEXE}/ugtrac
      strip ${GTEXE}/ugtrac
```

A1.4 makelib User Library Generator (Unix shell script)

```
GTLIB=/home/chen/lib
GTOBJ=/home/chen/obj

ar r ${GTLIB}/libgtrac.a ${GTOBJ}/calcdb.o   \
     ${GTOBJ}/calwgt.o   \
     ${GTOBJ}/coord.o   \
     ${GTOBJ}/deltbl.o   \
     ${GTOBJ}/dproc.o   \
     ${GTOBJ}/eescat.o   \
     ${GTOBJ}/encirc.o   \
     ${GTOBJ}/focus.o   \
     ${GTOBJ}/grid1.o   \
     ${GTOBJ}/grid2.o   \
     ${GTOBJ}/mcomm.o   \
     ${GTOBJ}/metref.o   \
     ${GTOBJ}/misce.o   \
     ${GTOBJ}/pfocus.o   \
     ${GTOBJ}/rdref.o   \
     ${GTOBJ}/red.o   \
     ${GTOBJ}/rgene.o   \
     ${GTOBJ}/rprint.o   \
     ${GTOBJ}/spdiag.o   \
     ${GTOBJ}/splot.o   \
     ${GTOBJ}/ssrt.o   \
     ${GTOBJ}/stat.o   \
     ${GTOBJ}/strace.o   \
     ${GTOBJ}/utrace.o   \
     ${GTOBJ}/vcalc.o   \
     ${GTOBJ}/vignet.o   \
     ${GTOBJ}/wray.o   \
     ${GTOBJ}/wrayso.o   \
     ${GTOBJ}/wraysv.o   \
     ${GTOBJ}/wspot1.o   \
     ${GTOBJ}/wspot2.o   \
     ${GTOBJ}/wstat.o   \
     ${GTOBJ}/xalign.o   \
     ${GTOBJ}/yintp.o

ranlib ${GTLIB}/libgtrac.a
```

A1.5 user.f Sample User Routine (FORTRAN source code)

```
c
c*****************************************************************
c
c  USER SUBROUTINE FOR SXI TELESCOPE RAY TRACE FOLLOWS
c
c*****************************************************************
c
      subroutine user
c
c  trace sxi system
c
      implicit double precision (a-h,o-z)
c*******************************************************************
      common /syscl/ zrange,elev,azim,foclen,source(3)
     * ,radlim(2,50),dxcirc(50),dycirc(50)
     * ,xwidth(50),ywidth(50),dxrect(50),dyrect(50),threct(50)
     * ,zlim(2,50),adata(25,50)
     * ,tilt(3,50),rmat(3,3,50)
     * ,disp(3,50),thick(50),findex(50)
     * ,sdata(25,50),delta
     * ,sp(3,50),ra(3,50),spi(3),rai(3)
     * ,energy(15),delbet(2,15,50),wgt(15,50),wgtnet(15),effa(15)

     * ,pi
     * ,imove(50),irstr(50),iwgt(50),nsurf
     * ,nnrg,kmax,kprint(51),ichief,itilt(50)
     * ,npass,nvig,nerr
     * ,iaper(50),iobs(50),itype(50),imode(50),ifdfm(50),ihead(20)
      character * 80 ihead,ifdfm
      character * 8 itype,imode,iaper,iobs
c*********************************************************************
      dimension enc(500),frac(100),rad(100),xref(15),yref(15)
c********************************************************************
c  output list file is default to print.gtrace
      open(6,file='print.gtrace')
c*********************************************************************
c  flag for readin to open system input file
      istat=1
c*******************************************************************
c  number of systems to loop through
      nconic=1
c*******************************************************************
      do 900 iel=1,nconic
c  read in the prescription for the first element of the HRMA.
      call readin(1,'presc.sxi.2',istat)
      if(istat.ne.0) go to 900
c********************************************************************
c  modifications
```

```
      ihead(2)=' '

c
c   parabola and hyperbola surface numbers.
      ip=5
      ih=11
c
c   modify parabola and hyperbola surface types.
c     itype(ip)='grzcon03'
c     itype(ih)='grzcon03'
c
c   reflectivity weight flags and number of energies
c     iwgt(ip)=0
c     iwgt(ih)=0
c     nnrg=1
c
      ifrom=6
      ito=3
      delbet(1,ito,ip)=delbet(1,ifrom,ip)
      delbet(2,ito,ip)=delbet(2,ifrom,ip)
      delbet(1,ito,ih)=delbet(1,ifrom,ih)
      delbet(2,ito,ih)=delbet(2,ifrom,ih)
      energy(ito)=energy(ifrom)
      nnrg=3
c   respace.
c   misc. cases
      d=0.d0
c   assume symmetric respace for the time being
c    (surface 7 is the finished end of the parabola)
c    (surface 8 is to be the position of the
c     mid point between the glass ends)
      thick(7)=thick(7)+d/2.d0
      thick(8)=thick(8)+d/2.d0
c   leave the distance between the mid point between the glass ends ar

c   the nominal image plane unchanged.
c    (surface 16 is the image plane)
      thick(15)=thick(15)-d/2.d0
c
c   finite source distance to first surface
c   misc cases
c     zrange=1700.d0*12.d0*25.4d0
c   values from source to center distance and various respace errors
c   (t.casey 910129)
c     zrange=1731.d0*12.d0*25.4d0
c     n1=1
c     n2=7
c     do 600 i=n1,n2
c     zrange=zrange-thick(i)
c 600 continue
c
c  length of element
      size=zlim(2,ip)-zlim(1,ip)
```

```
c
c  elevation of source
c     elev=50.d0/3600.d0*pi/180.d0
c
c  azimuth of source
c     azim=0.d0
c     azim=pi/4.d0
      azim=pi/2.d0
c     azim=0.75d0*pi
c     azim=pi
c     azim=7.d0*pi/8.d0
c  modify distance to last surface
c     thick(nsurf-1)=thick(nsurf-1)+0.010d0
c
c  surface tilts
c     tilt(1,ih)=.15d0/3600.d0*pi/180.d0
c     tilt(2,ih)=.15d0/3600.d0*pi/180.d0
c     tilt(3,ih)=pi/4.d0
c     imove(ih)=1
c     irstr(ih)=1
c     itilt(ih)=213
c
c  hyperbola decenter and compensating tilt
c
c     decenx=0.d0
c     decenx=0.254d0
c     deceny=0.d0
c     deceny=0.254d0
c     n1=ih
c     n2=nsurf-1
c     zoff=10069.21899483571d0
c     comlen=zoff+d/2.d0
c     do 400 i=n1,n2
c     comlen=comlen+thick(i)
c 400 continue
c     comtx=-dasin(decenx/comlen)
c     comty=dasin(deceny/comlen)
c     imove(ih)=1
c     irstr(ih)=1
c     dcomtx=0.d0
c     dcomtx=.15d0/3600.d0*pi/180.d0
c     dcomty=0.d0
c     dcomty=.15d0/3600.d0*pi/180.d0
c     disp(1,ih)=decenx
c     tilt(2,ih)=comtx+dcomtx
c     disp(2,ih)=deceny
c     tilt(1,ih)=comty+dcomty
c
c  sag error
c     sdata(5,ip)=-400.d-7
c     sdata(5,ih)=-400.d-7
c  save minimum radius
      rminsv=radlim(1,1)
```

```
c  save zrange
      zrngsv=zrange
c  modify convergence criterium
c     delta=1.d-7
c
c  ray print flag
c     kprint(1)=2
c     kprint(2)=1
c     kprint(3)=ip
c     kprint(4)=ih
c     kprint(5)=nsurf
c
c  number of field angles
      nfield=2
c
      do 200 kk=1,nfield
c  adjust field angle
c
c     if(kk.eq.2) elev=1.d0/3600.d0*pi/180.d0
c     if(kk.eq.3) elev=50.d0/3600.d0*pi/180.d0
      elev=dble(kk-1)*1.d0/3600.d0*pi/180.d0
c
c  adjust tilt of first surface and zrange
c  to simulate field angle entry
c
c     if(kk.eq.2) then
c     tsv=-1.d0/3600.d0*pi/180.d0
c     tilt(2,1)=tsv
c     zrange=zrngsv/dcos(dabs(tsv))
c     imove(1)=1
c     endif
c
c     if(kk.eq.3) then
c     tsv=-50.d0/3600.d0*pi/180.d0
c     tilt(2,1)=tsv
c     zrange=zrngsv/dcos(dabs(tsv))
c     imove(1)=1
c     endif
c
c  adjust radius limits with field angle and source distance.
      radlim(1,1)=zrange/(zrange+size)*(rminsv
     *            -dtan(dabs(elev))*size)
c     radlim(1,1)=zrngsv/(zrngsv+size)*(rminsv
c    *            -dtan(dabs(tsv))*size)
c***********************************************************************
c  set up the common area.
      call setcom(ierr)
c***********************************************************************
c  print out the system common area
      call rdout(6,idum)
c***********************************************************************
c  do a weighted ray trace with random ray distribution in
c  entrance annulus
```

```
c
      ipat=1
c
      if(ipat.eq.1) then
c
      mspot=10000
      rmin=radlim(1,1)
      rmax=radlim(2,1)
      azmid=0.d0
      delaz=2.d0*pi
      azmin=azmid-delaz/2.d0
      azmax=azmid+delaz/2.d0
      irand=0
      call ranset(irand)
      call wspot1(mspot,irand,rmin,rmax,azmin,azmax)
c
      endif
c***********************************************************************
c  do a weighted ray trace with modified wheel spoke distribution in
c  entrance annulus
c
      ipat=0
c
      if(ipat.eq.1) then
c
      nlong=10000
      naz=1
      rmin=radlim(1,1)
      rmax=radlim(2,1)
      azmid=0.d0
      delaz=2.d0*pi/200.d0
c     delaz=2.d0*pi
      azmin=azmid-delaz/2.d0
      azmax=azmid+delaz/2.d0
      call wspot2(nlong,naz,rmin,rmax,azmin,azmax)
c
      endif
c***********************************************************************
c  do a ray trace with modified spoke wheel distribution.
c  (all weights set to 1)
c  (constant radial and varying azimuthal increments)
c  to compare with subroutine rfocs in vetasag.f
c     nlong=501
c     naz=72
c     rmin=radlim(1,1)*(1.d0+1.d-8)
c     rmax=radlim(2,1)/(1.d0+1.d-8)
c     call grid1(nlong,naz,rmin,rmax)
c***********************************************************************
c  do a ray trace with  spoke wheel distribution.
c  (all weights set to 1)
c  (constant radial and constant azimuthal increments)
c     naz=1
c     nlong=839
```

```
c     rmin=radlim(1,1)*(1.d0+1.d-8)
c     rmax=radlim(2,1)/(1.d0+1.d-8)
c     call grid2(nlong,naz,rmin,rmax)
c*********************************************************************
c
c  loop over energies
c
      do 300 iener=1,nnrg
c*********************************************************************
c  refocus
      call focus(iener,xav,yav,delz)
c*********************************************************************
c  calculate average position and rms
      if(kk.eq.1) then
      xref(iener)=0.d0
      yref(iener)=0.d0
      endif
      call wstat(iener,xav,yav,wav,wtot,xref(iener),yref(iener)
     * ,foclen,elev)
      if(kk.eq.1) then
c  get reference for apparent focal length calculation
      xref(iener)=xav
      yref(iener)=yav
      endif
c*********************************************************************
c  make unweighted spot diagram
      call spdiag(xav,yav,0)
c*********************************************************************
c  calculate encircled energy distribution
c    maximum angle in arc sec for calculation
      amax=2.d0
c    number of calculation points
      na=500
c    number of fractions for radii calculation
      nf=20
      do 100 i=1,nf
      frac(i)=dble(i)/dble(nf)
  100 continue
      frac(nf)=frac(nf)/(1.d0+1.d-8)
      call encirc(iener,xav,yav,foclen,amax,na,frac,rad,nf,enc
     * ,wamax,wtot)
c*********************************************************************
c end of energy loop
  300 continue
c*********************************************************************
c  write out system data and ray data to files
c     call wrayso('ring.gtray')
c*********************************************************************
c end of field angle loop
  200 continue
c*********************************************************************
c end of mirror system loop
  900 continue
```

```
c*****************************************************************
      return
      end
```

APPENDIX 2

GRAZTRACE USER MANUAL

GRAZTRACE

X-ray Optical Analysis Program

USER MANUAL

CONTENTS

Section 1. INTRODUCTION

GRAZTRACE is a high level FORTRAN subroutine library designed for the analysis of X-ray telescope optics. **GRAZTRACE** subroutines are easy to understand and use. Users can generate their own analysis program using **GRAZTRACE** subroutines.

"**GRAZTRACE**" stands for GRAZe ray TRACE. The basic system is a library of over 60 FORTRAN subroutines and functions.

Command mode **GRAZTRACE** is under development which allows the users to interactively use the system.

Section 2. PRIMER

2.1 Summary

This section includes the following topics: Section 2.2, "Quick Start", explains what a basic session **GRAZTRACE** is, and provides a hands-on example to get you started; Section 2.3, "Using **GRAZTRACE**", contains information about using **GRAZTRACE**, FORTRAN COMPILERS, MAKE and data file format.

2.2 Quick Start

To use **GRAZTRACE**, users need to write or modify their own main program and the user program, main.f and user.f.

Programs for a basic sample session can be:

```
c-------------------------------------------------------------
c
c  sample main.f
c
      program main
      implicit double precision (a-h,o-z)
c
c  open output file for the result
c
      open(6,file='sample.gtrace')
      call user
      stop
      end
c
c  sample user.f
c
      subroutine user
      implicit double precision (a-h,o-z)
      common /syscl/ zrange,elev,azim,foclen,source(3)
     * ,radlim(2,50),dxcirc(50),dycirc(50)
     * ,xwidth(50),ywidth(50),dxrect(50),dyrect(50),threct(50)
     * ,zlim(2,50),adata(25,50)
     * ,tilt(3,50),rmat(3,3,50)
     * ,disp(3,50),thick(50),findex(50)
     * ,sdata(25,50),delta
     * ,sp(3,50),ra(3,50),spi(3),rai(3)
     * ,energy(15),delbet(2,15,50),wgt(15,50),wgtnet(15),effa(15)
     * ,pi
     * ,imove(50),irstr(50),iwgt(50),nsurf
     * ,nnrg,kmax,kprint(51),ichief,itilt(50)
     * ,npass,nvig,nerr
     * ,iaper(50),iobs(50),itype(50),imode(50),ifdfm(50),ihead(20)
      character * 80 ihead,ifdfm
```

```
      character * 8 itype,imode,iaper,iobs
      dimension enc(500),frac(100),rad(100),xref(15),yref(15)
      istat=1
c
c readin prescription
c
      call readin(1,'sample',istat)
      call setcom(ierr)
      mspot=10000
      rmin=rradlim(1,1)
      rmax=radlim(2,1)
      azmid=0.d0
      delaz=2.0*pi
      azmin=azmid-delaz/2.d0
      azmax=azmid+delaz/2.d0
      irand=0
      call ranset(irand)
c
c random ray trace
c
      call wspot1(mspot,irand,rmin,rmax,azmin,azmax)
      iener=1
c
c refocus
c
      call focus(iener, xav,yav,delz)
c
c statistical performance analysis
c
      call   wstat(iener, xav, yav, wav, wtot, xref(iener),
     *       yref(iener), foclen,elev)
c
c spot diagram
c
      call spdig(xav,yav,0)
      amax=2.d0
      na=500
      nf=20
      do 100 i=1, nf
      frac(i)=dble(i)/dble(nf)
 100  continue
      frac(nf)=frac(nf)/(1.d0+1.d-8)
c
c encircled energy distribution
c
      call encirc(iener, xav, yav, foclen, amax, na, frac, rad,
     *     nf, enc, wamax, wtot)
c
c save ray information to file
c
      call wrayso('sample.gtray')
      return
      end
```

```
c--------------------------------------------------------------
```

The input data is in data file "sample". Compile or make the program, then run the program. The results are in file "sample.gtrac" and ray data is in "sample.gtray"

2.3 Using GRAZTRACE

GRAZTRACE library is in the system with the name "libgtrac.a" currently in /home/chen/lib. It can be later linked to other directory such as /usr/lib to allow more users to use it.

2.3.1 Compiler

After modifying main.f and user.f, users can compile their program with F77 compiler with -L/home/chen/lib -lgtrac.

f77 main.f user.f -L/home/chen/lib -lgtrac

2.3.2 Make Utility

The other simple way to generate user code is "make utility." After modifying main.f and user.f, the users can make their own program with command:

make -f ugtmakefile

Currently, "ugtmakefile" is in /home/chen/mk. The executable file is generated with the name "ugtrac".

2.3.3 Data File Format

Data files for GRAZTRACE are using the standard FORTRAN NAMELIST format. The first line of the file must be:

$inp

beginning in column two. Do NOT FOLLOW **$inp** with a comma.

The data input can be in any order. The basic format is:

PNEUMONIC = $v_1 v_2 ...., v_n$

where v_i is the ith value. Note that the pneumonic is simply its name regardless of whether or not it is a single variable or an array variable. Trailing comma is required.

The last line of the data file must be:

$end

beginning in column two. DO NOT FOLLOW **$end** WITH A COMMA OR ANYTHING ELSE.

Pneumonic definition.

Pneumonic	Description	Format
zrange	Object distance	Double
elev	Object elevation	Double
azim	Object azimuth	Double
foclen	Focal length	Double
source		
radlim		

...

Section 3. USER LIBRARY ROUTINES

In the **GRAZTRACE** library, only some of the routines are designed to be called by the users directly. It is necessary for users to know only these USER LIBRARY ROUTINES to use the system. Other routines are called indirectly. Only the **GRAZTRACE** program developer needs to access the other routines.

3.1 Data Manipulate Routines

All data in the system common area can be manipulated by calling the following routines:

readin
setcom
redout
wrayso

...

3.1.1 readin

All system data should be read in using **readin**

call readin (i, jpresc, istat)

i, unit number to be opened for the file;

jpresc, data file name;

istat =
- 1 , proceed to open file and read in;
- 0 , reach the end of file;
- else, read error.

Purpose: Read in data to common area from file in **jpresc** using unit **i**.

Subroutines called: **czero**.

3.1.2 setcom

After readin or modification of system data, the data in common area should be set up using **setcom.** **setcom** (a) set up source positions relative to the undisplaced center of first surface in cartesian coordinate(**source(1)** = **x** , **source(2)** = **y**, **source(3)** = **z**), and (b) sets rotation matrices from tilts for surfaces with **imoves** = 1 (which means surface coordinate transformation is subject to the surface).

call setcom(jerr)

jerr = 0, normal; 1, error.

Purpose: Set up common data after readin or modification of common data.

Subroutines called: **rstart**
rdfm

3.1.3 rdout

All system data can be written into a file using **rdout**. The data file has the same namelist format which can be later read into the system by **readin**.

call rdout(i, istat)

i, unit number to be opened for the file;

istat = 0, normal; -1, error.

Purpose: Write out system common data to unit **i**.

Subroutines called: none

3.1.4 wrayso

Ray information from analysis routines can be saved to a file for further analysis using wrayso.

call wrayso(fname)

fname, file name for ray data writing, prefix is .gtray.

Purpose: Write saved ray data to file.

Subroutine called: none

Note: The data is unformatted. The contents are as follows:

nsv,	number of rays saved;
nnrg,	number of energy positions;
zshift,	focal shift;
foclen,	focal length;
nhead,	number of lines for the head description;
xpsv(i),	**x** position of the ith ray, **i**=1..**nsv**;
ypsv(i),	**y** position of the ith ray, **i**=1..**nsv**;
dxdzsv(i),	**x** direction slop of the ith ray, **i**=1..**nsv**;
dydzsv(i),	**y** direction slop of the ith ray, **i**=1..**nsv**;
entx(i),	**x** direction incident angle of the ith ray, **i**=1..**nsv**;
enty(i),	**y** direction incident angle of the ith ray, **i**=1..**nsv**;
wtsv(j,i),	weight for the ith ray at the **j**th energy level, **i**=1..**nsv**, **i**=1..**nnrg**;
energy(i),	the ith energy, **i**=1..**nnrg**;
ihead(i),	the ith head message, **i**=1..**nhead**.

3.2 Math routines

Some math routines are available in the library:

ranset

...

3.2.1 ranset

Before using function **ranf** to generate uniform distribution random number, random number seed should be reset using **ranset**. **wspot1** also needs to initialize the random number seed **irand** using **ranset.**

call ranset(irand)

irand, random number seed (an arbitrary integer).

Purpose: Reinitialize random number seed.

Subroutines called: none

3.3 Performance evaluation routines

Most commonly used optical system analysis routines are included in the **GRAZTRACE** library:

wspot1
wspot2
focus
wstat
spdiag
encirc

...

3.3.1 wspot1

Weighted rays can be traced with rays randomly distributed in the entrance annulus using **wspot1**.

call wspot1(mspot, irand, rmin, rmax, azmin, azmax)

mspot, total number of rays;

irand, random number seed; (reset by **ranset** before using it in this routine)

rmin, minimum radius of entrance annulus;

rmax, maximum radius of entrance annulus;

azmin, minimum azimuth of entrance section of the annulus;

azmax, maximum azimuth of entrance section of the annulus.

Purpose: Randomly trace weighted rays in entrance annulus.

Subroutines called: **ssrti**
wsvi
ranf
cnvout
wray
wraysv
wsvrst

Note: 1. Random rays are traced and effective area weights **effa(i)** are accumulated.

2. The ray information from the last surface is saved for further analysis.

3. The results are printed out.

3.3.2 wspot2

Weighted rays can be traced with modified wheel-spoked distribution in the entrance annulus using **wspot2**.

call wspot2(nlong, naz, rmin, rmax, azmin, azmax)

nlong, number of rays in radius direction;

naz, number of rays in azimuth direction;

rmin, minimum radius of entrance annulus;

rmax, maximum radius of entrance annulus;

azmin, minimum azimuth of entrance section of the annulus;

azmax, maximum azimuth of entrance section of the annulus.

Purpose: Wheel spokely trace the weighted rays in entrance annulus.

Subroutines called: **ssrti**
wsvi
cnvout
wray
wraysv

Note: 1. Wheel-spoked rays are traced and the effective area weights **effa(i)** are accumulated.

2. The ray information from the last surface is saved for further analysis.

3. The results are printed out.

3.3.3 focus

The system can be refocused to the best focal position for a given energy level using **focus**.

call focus(iener, xav, yav, delz)

iener, energy position;

xav, new **x** average;

yav new **y** average;

deltz, difference between best focus **z** value and initial **z** value.

Purpose: Focus spot in storage array for a given energy position.

Subroutines called: **pfocus**

Note: At a given energy position, **focus** finds the best focus and sends back the new centroid coordinates **xav, yav** and the required focal shift **deltz.**

3.3.4 wstat

System performance can be statistically evaluated using **wstat.**

call wstat(iener, xav, yav, wav, wtot, xref, fl, ell)

iener, energy position;

xav, **x** average;

yav, **y** average;

wav, weights average;

wtot, weights total;

xref, **x** reference;

yref, **y** reference;

fl, assumed focal length;

el1, assumed field angle.

Purpose: Calculate the average and rms of stored rays at given energy position.

Subroutines called: **stat**

Note: 1. Input is required are reference center coordinates **xref** and **yref**, assumed focal length **fl**, and assumed field angle **el1.**

2. Results are sent back and printed out.

3.3.5 spdiag

System performance can be checked by spot diagram using **spdiag.**

call spdiag(xcen, ycen, npoint)

xcen, x value of assumed center for diagram;

ycen, y value of assumed center for diagram;

npoint, ray number to be used in diagram.

Purpose: Make up line printer spot diagram from the storage array using first **npoint** rays.

Subroutines called: **splot**

3.3.6 encirc

System performance can be quantitatively examined by encircled energy using **encirc.**

call encirc(iener, xcen, ycen, ft, amax, na, frac, rad, nf, enc, wamax, wtot)

iener, energy pointer;

xcen, x of assumed center of encircled energy distribution;

ycen, y of assumed center of encircled energy distribution;

ft, assumed focal length;

amax, maximum angle considered (arc sec) for encircled energy distribution calculation;

na, number of radius increments for encircled energy distribution calculation;

frac, encircled energy fractions for radii calculations;

red, radii values calculated for **nf** fraction values input;

nf, number of encircled energy fractions;

enc, encircled energy distribution (at na radius values up to amax);

wamax, weight total up to radius **amax;**

wtot, total weight sum.

Purpose: Calculate the encircled energy distribution for energy iener.

Subroutine called: none

Note: 1. Inputs required are **iener, xcen, ycen, ft, amax, na, frac,** and **nf.**

2. Outputs sent back are **rad, enc, wamax,** and **wtot.** The results are also printed out.

Section 4. SYSTEM PROGRAM DEVELOPER LIBRARY ROUTINES

System program developer library routines are not directly called by the users. Only GRAZTRACE program developer need to know those routines in order to debug or upgrade the program.

4.1 Data manipulating routines

4.1.1 czero

call czero

Purpose: Zero the common area.

Subroutines called: none

Note: 1. Set 0.d0 to 6510 double precision variables;

2. Set 0 to 258 integers;

3. Set ' ' to 200 character*8 strings;

4. Set ' ' to 70 character*80 strings;

5. Common name is **sysc1**.

4.1.2 rstart

call rstart(ierr)

ierr =
- 0, normal;
- 1, error.

Purpose: Set up rotation matrices for surface with **imove = 1** (which means surface is subject to tilt).

Subroutines called: matab

Note: 1. Set up rotation matrices following the order given by **itilt** or 1,2,3 (when **itilt = 0**).

2. **itilt** has the format of an integer, each digit presets the tilt order. For example 123 means order 1,2,3.

3. Results in **rmat(i,j,k)**,
   j,k, surface dimensions;
   i, surface number.

4.1.3 rdfm

Deformation information can be read from the file to common area using **rdfm**.

call rdfm(iurdfm)

iurdfm, unit number for deformation file.

Purpose: Read in deformation values to common area from the deformation file.

Subroutine called: **prtdfm**

Note: 1. Deformation file name is in **ifdfm**.

2. Namelist format has been used.

...

4.1.4 prtdfm

call prtdfm(debug, nsurf)

debug =
- true, dump out the derived deformations;
- false, no derived deformations.

nsurf, number of surfaces.

Purpose: Print out deformation storage data.

Subroutines called: none

4.1.5 rprint

call rprint(lsurf, irstat, ktr)

lsurf, surface number to be printed;

= 0, normal;

irstat > 0, vignetting;
< 0, ray error;

ktr, index of print control array kprint.

Purpose: Print out ray surface information.

Subroutines called: none.

4.2 Math routines

4.2.1 ranf

call ranf (irand)

irand, random number seed.

Purpose: Generate random numbers.

Subroutines called: none.

Note: 1. Random numbers are uniformly distributed in the range (0,1).

2. Random seed **irand** should be reset by **ranset(irand)**.

4.2.2 matab

call matab(a,b,c,n1,n2,n3,d)

a (n1, n2), first matrix to be multiplied;

b (n2, n3), second matrix to be multiplied;

c (n1, n3), result matrix;

n1, row number of the first matrix;

n2, column number of the first matrix, row number of the second matrix;

n3, column number of the second matrix;

d (n1, n3), temporary matrix.

Purpose: Multiply the first matrix by the second matrix
c = a X b.

Subroutines called: none

4.2.3 cnvout

call cnvout(sp1, ra1, sp2, ra2, rmat, disp)

sp1, input position;

ra1, input direction consine;

sp2, output position;

ra2, output direction consines;

rmat, transformation matrix;

disp, displacement array.

Purpose: Transform out of location coordinates.

Subroutines called: none.

4.2.4 cnvin

call cnvin(sp1, ra1, sp2, ra2, rmat, disp)

sp1, input position;

ra1, input direction consine;

sp2, output position;

ra2, output direction consines;

rmat, transformation matrix;

disp, displacement array.

Purpose: Transform out of location coordinates.

Subroutines called: none.

4.2.5 trfin

call trfin(is)

is, surface number to be transformed.

Purpose: Transform into local coordinates.

Subroutine called: none.

Note: Ray positions **sp(,is)** and direction cosines **ra(,is)** are updated.

4.2.6 trfout

call trfout(is)

is, surface number to be transformed.

Purpose: Transform out of local coordinates.

Subroutines called: none.

Note: Positions **sp(,is)** and direction cosines **ia(,is)** are updated.

4.2.7 cnvin

call cnvin(sp1, ra1, sp2, ra2, rmat, disp)

sp1, input position;

ra1, input direction;

sp2, output position;

ra2, output direction;

rmat, transformation matrix;

disp, displacement array.

Purpose: Transform into local coordinates.

Subroutines called: none

4.2.8 rotate

call rotate(xp, yp, ang, x, y)

xp, original **x** coordinate;

yp, original **y** coordinate;

ang, angle to be rotated;

x, new **x** coordinate;

y, new **y** coordinate.

Purpose: Rotate coordinates.

Subroutines called: none

4.3 Performance evaluation routines

4.3.1 ssrti

call ssrti

Purpose: Initialize ray counters for pass, vignetting, and error.

Subroutines called: none

Note: **ssrti** set **pass** = 0, **nvig** =0, **nerr** = 0.

4.3.2 nusvi

call nusvi

Purpose: Initialize storage ray counter and **zshift**.

Subroutines called: none.

Note: **nusvi** set **nsv** = 0, **zshift** = 0.

4.3.3 wray

call wray(efact, irstat)

efact, initial effective area weight for ray

irstat = 0, normal; else, ray error.

Purpose: Trace ray and accumulate reflectivity weights and effective area weight for ray.

Subroutines called: **cnvin**
ssrt
calwgt
rprint

4.3.4 ssrt

call ssrt(is, irstat)

is, surface number to be traced to;

0, successful ray;

irstat = 1, vignetted ray;

-1, ray error.

Purpose: Trace a single ray.

Subroutines called: **trfin**
strace
strac02
vignet
trfout

4.3.5 strace

call strace(isterr, is)

0, normal;

isterr =

else, ray error.

is - surface number to be traced to.

Purpose: Trace a ray for reflection or dummy surface.

Subroutines called: **utraci**

Note: Ray positions **sp(,is)** and direction cosine **ra(,is)** are updated.

4.3.6 utrace

call utrace

Purpose: Calculate function **f** and gradient **fx, fy, fz** for surface

Subroutines called: none.

Note: 1. Input:

x, y, z,	position;
isurf or **n,**	surface number
itype(n)	surface parameters
ifcalc,	calculate function value if **ifcalc=1**

2. Output:

f,	interception function value;
fx, fy, fz,	gradient of function
isferr,	non zero if error occurs.

4.3.7 strc02

call strc02(isterr, is)

isterr = 0, normal; else, ray error.

is, surface number to be traced to.

Purpose: Trace ray through deformated surface.

Subroutines called: **utrc02.**

Note: Ray positions **sp(,is)** and ray direction cosines **ra(,is)** are updated.

4.3.8 utrac02

call utrac02

Purpose: Calculate function f and gradient **fx, fy, fz** for deformated surface.

Subroutines called: **dfm02**

Note: 1. Input:

x, y, z,	position;
isurf or **n,**	surface number;
itype(n),	surface type;
sdata(...,n),	surface parameters
ifcalc =	1, calculate function value; 2, calculate gradient; 3, calculate both.

2. Output:

f,	function value;
fx, fy, fz,	gradient of the function.

4.3.9 dfm02

call dfm02

Purpose: Compute contribution of surface errors radius error and gradient of radius error.

Subroutines called: none.

Note: **f** is the difference between the ray position radius value and the surface radius value.

4.3.10 vignet

call vignet(ivig, is)

ivig = 0, normal
else, vignetting.

is, surface number to be checked.

Purpose: Check for surface vignetting.

Subroutines called: **rotate**

4.3.11 calwgt.f

call calwgt.f(lsurf)

lsurf, surface number to be accumulated.

Purpose: Accumulate metal reflectivity weights for applicable surface and update ray effective area weight.

Subroutines called: **metref.**

4.3.12 metref

call metref(anginc, delta, beta, rs, rp)

anginc, incident angle in radians;

delta, beta, reflectivity data;

rs, reflectivity for parallel polarization;

rp, reflectivity for perpendicular polarization.

Purpose: Calculate the reflectivity as a function of incident angle and complex index of refraction for metals.

Subroutines called: none.

Note: 1. Input: **anginc, delta, beta.**

2. Output: **rs, rp**.

4.3.13 wraysv

call wraysv(ifill)

ifill = 1, ray number saved = 200,000;
0, less than 200,000 rays saved.

Purpose: Save last surface ray information.

Subroutine called: **cnvin**

Note: Ray information saved in common rsave1 with format:

xpsv(200,000),	x position;
ypsv(200,000),	y position;
dxdzsv(200,000),	x direction slope;
dydzsv(200,000),	y direction slope;
entx(200,000),	x direction incident angle;
enty(200,000),	y direction incident angle;
wtsv(15,2000,000),	effective area weights.

4.3.14 wsvrst

call wsvrst(factor)

factor, scale factor.

Purpose: Reset the effective area weights

Subroutines called: none.

Note: **wsvrst** loops through rays and energy levels to scale the saved weights.

wtsv(j,i)= wtsv(j,i)*factor

4.3.15 pfocus

call pfocus(x,y,ck,cl,w,n,xloc,yloc,zloc)

x,y,	positions of rays at initial **z** value;
ck,cl,	**dx/dz, dy/dz** for each ray;
n,	number of rays;
w,	ray weights;
xloc, yloc,	positions of best focus in **x-y** plane;
zloc,	delta **z** to best focus from initial **z** value;
x,y,	position of rays at new **z** value.

Purpose: Find weighted planar best focus using rays from geometric ray trace.

Subroutines called: none.

Note: **x, y, ck, cl, w,** and **n** are input.

4.3.16 stat

call stat(x, y, w, n, xav, yav, xrms, yrms, rms, wtot, wav, wrms, xmin, xmax, ymin, ymax, wmin, wmax)

x, y,	ray intercepts;
w,	ray weights;
n,	number of rays;
xav, yav,	**x, y** value at centroid
xrms, yrms,	rms values of **x** and **y** about centroid;

wtot, sum of weights;

wav, average value of wights;

wrms, rms deviation of weights

xmin, minimum **x** value;

xmax, maximum **x** value;

ymin, minimum **y** value;

ymax, maximum **y** value;

wmin, minimum weight value;

wmax, maximum weight value.

Purpose: Calculate the weighted spot average and rms.

Subroutines called: none.

Note: **x, y, w,** and **n** are inputs, the rest are outputs.

4.3.17 splot

call splot(npts,f,x)

npts, ray number to be plotted;

f, x, values of plot data **x, y.**

Purpose: On-line printer plot for spot diagram.

Subroutines called: none.

APPENDIX 3

CONTOUR AND 3-D POINT SPREAD FUNCTION

Appendix 3 Plot Routines for Contour and 3-D Point Spread Function

A3.1 rayplot.f Test Plot (FORTRAN source code)

```
c*****************************************************************
c
c file: rayplot.f
c
c test contour and point spread function plot
c       dis77links rayplot.f pltcnt.f pltpsf.f
c
c plot ray file "ray.gtray"
c
c*****************************************************************
      program main
      call wrayri('ray.gtray')
      call pltcnt(10000,0.0,0    )
      call pltpsf(10000,0.0,0,60.,30.)
      stop
      end
      subroutine wrayri(fname)
c
c  read in ray save data from ray file
c
c  fname is the file prefix for the .gtray file.
c
c
      implicit double precision (a-h,o-z)
c*****************************************************************
      common /sysc1/ zrange,elev,azim,foclen,source(3)
     * ,radlim(2,50),dxcirc(50),dycirc(50)
     * ,xwidth(50),ywidth(50),dxrect(50),dyrect(50),threct(50)
     * ,zlim(2,50),adata(25,50)
     * ,tilt(3,50),rmat(3,3,50)
     * ,disp(3,50),thick(50),findex(50)
     * ,sdata(25,50),delta
     * ,sp(3,50),ra(3,50),spi(3),rai(3)
     * ,energy(15),delbet(2,15,50),wgt(15,50),wgtnet(15),effa(15)
     * ,pi
     * ,imove(50),irstr(50),iwgt(50),nsurf
     * ,nnrg,kmax,kprint(51),ichief,itilt(50)
     * ,npass,nvig,nerr
     * ,iaper(50),iobs(50),itype(50),imode(50),ifdfm(50),ihead(20)
      character * 80 ihead,ifdfm
      character * 8 itype,imode,iaper,iobs
c*****************************************************************
      common /rsave1/ xpsv(200000),ypsv(200000),dxdzsv(200000)
     * ,dydzsv(200000),entx(200000),enty(200000),wtsv(15,200000)
     * ,zshift,nsv
c*****************************************************************
      character * 80 fname
c  open the ray file
```

```
      iuray=7
c     call fildf(iuray,fname,'gtray ','unformatted ')
      open(iuray,file=fname,form='unformatted')
c  read in   the ray data
      nhead=20
      read (iuray)nsv,nnrg,zshift,foclen,nhead
      read (iuray)(xpsv(i),ypsv(i),dxdzsv(i),dydzsv(i),entx(i)
     * ,enty(i),(wtsv(j,i),j=1,nnrg),i=1,nsv),(energy(i),i=1,nnrg)
     * ,(ihead(i),i=1,nhead)
c
c    print check
c
c     write(*,*)  nsv,nnrg,zshift,foclen,nhead
c     write(*,*)   (xpsv(i),ypsv(i),dxdzsv(i),dydzsv(i),entx(i)
c    * ,enty(i),(wtsv(j,i),j=1,nnrg),i=1,nsv),(energy(i),i=1,nnrg)
c    * ,(ihead(i),i=1,nhead)
c
c
c
      return
c
      end
```

A3.2 pltcnt.f Contour Plot (FORTRAN source code)

```
c*************************************************************
      subroutine pltcnt(mspot,size,ngrid)
c
c  plot intensity contour
c
c  mspot, total number of rays
c  size, half width of the plot region in arc sec
c  ngrid, grid number of the plot
c
c*************************************************************
      real rmat1(250,250)
      if (ngrid .eq. 0) ngrid = 50
      call pltcnt1(mspot,size,ngrid,rmat1)
      return
      end
c*************************************************************
      subroutine pltcnt1(mspot,size,ngrid,rmat1)
c
      implicit double precision (a-h,o-z)
c*******************************************************************
      common /syscl/ zrange,elev,azim,foclen,source(3)
     * ,radlim(2,50),dxcirc(50),dycirc(50)
     * ,xwidth(50),ywidth(50),dxrect(50),dyrect(50),threct(50)
     * ,zlim(2,50),adata(25,50)
     * ,tilt(3,50),rmat(3,3,50)
     * ,disp(3,50),thick(50),findex(50)
     * ,sdata(25,50),delta
     * ,sp(3,50),ra(3,50),spi(3),rai(3)
     * ,energy(15),delbet(2,15,50),wgt(15,50),wgtnet(15),effa(15)
     * ,pi
     * ,imove(50),irstr(50),iwgt(50),nsurf
     * ,nnrg,kmax,kprint(51),ichief,itilt(50)
     * ,npass,nvig,nerr
     * ,iaper(50),iobs(50),itype(50),imode(50),ifdfm(50),ihead(20)
      character * 80 ihead,ifdfm
      character * 8 itype,imode,iaper,iobs
c*******************************************************************
      common /rsavel/ xpsv(200000),ypsv(200000),dxdzsv(200000)
     * ,dydzsv(200000),entx(200000),enty(200000),wtsv(15,200000)
     * ,zshift,nsv
c*******************************************************************
c
c  single presion for plot routine
c
      real datmin, datmax, xmin, xmax, ymin, ymax, xcen, ycen
     *  , xlen, ylen, xstp, ystp, xphy, yphy, xarea, yarea, zincr
     * ,enr ,peak ,size, focl
      character * 80 capt
       real rmat1(ngrid,ngrid)
       COMMON /PAKRAY/ IPKRAY(400)
       COMMON /MYCONX/ DATMIN,DATMAX
c      data ixdim/100/,iydim/100/
```

```
c
      ixdim=ngrid
      iydim=ngrid
c**********************************************************************
c
c loop over energies
c
c**********************************************************************
      do 100, n=1, nnrg
c
c
c  centroid
c
      swx=0
      swy=0
      sw=0
      do k=1,nsv
      swx=swx+wtsv(n,k)*xpsv(k)
      swy=swy+wtsv(n,k)*ypsv(k)
      sw=sw+wtsv(n,k)
      end do
      xcen=swx/sw
      ycen=swy/sw
c
c  find plot range
c
      xlen=2*foclen*size*4.84813e-6
      if (size .lt. 1.0e-20) then
      xmin=xpsv(1)
      xmax=xpsv(1)
      ymin=ypsv(1)
      ymax=ypsv(1)
      do i=1, nsv
      xmin=dmin1(xmin,xpsv(i))
      xmax=dmax1(xmax,xpsv(i))
      ymin=dmin1(ymin,ypsv(i))
      ymax=dmax1(ymax,ypsv(i))
      end do
c
c  round to good number
c
      xlen=2.*amax1( abs(xmin-xcen), abs(xmax-xcen)
     * , abs(ymin-ycen), abs(ymax-ycen))
c     size=xlen/(2*foclen*4.84813e-6)
c     del=size/dble(ixdim)
c     k=dlog10(del)
c     if (del .lt . 1.d0) k=k-1
c     q=10.d0**k
c     c=del/q
c     ic = c
c     cp = ic
c     if (cp.lt.c) cp = cp +1.d0
c     size = ixdim * cp*q
      end if
c
      ylen = xlen
      xmin=xcen-0.5*xlen
      xmax=xcen+0.5*xlen
```

```
        ymin=ycen-0.5*ylen
        ymax=ycen+0.5*ylen
c
c  tick interval
c
        xstp = xlen / 8.
        ystp = ylen / 8.
c
c prepare plot matrix
c
      do i=1,ixdim
      do j=1,iydim
      rmat1(i,j)=0.
      end do
      end do
c
c accumulate energy
c
      do k=1, nsv
      i=(xpsv(k)-xmin)*(ixdim-1)/xlen + 1
      j=(ypsv(k)-ymin)*(iydim-1)/ylen + 1
      if ((i .ge. 1) .and. (i .le. ixdim)
     *       .and. (j .ge. 1) .and. (j .le. iydim)) then
      rmat1(i,j)=rmat1(i,j)+wtsv(n,k)*energy(n)
      end if
      end do
c
c scale to intensity
c
      datmin=rmat1(1,1)*ixdim*iydim/(xlen*ylen)
      datmax=datmin
      do i=1,ixdim
      do j=1,iydim
      rmat1(i,j)=rmat1(i,j)*ixdim*iydim/(xlen*ylen)
      datmax=amax1(datmax,rmat1(i,j))
      datmin=amin1(datmin,rmat1(i,j))
      end do
      end do
      peak=datmax
      if (energy(n) .lt. 0) peak = datmin
c
c*****************************************************************
c
c plot
c
        CALL VT240
C                       NOMINATE A DEVICE
        CALL PAGE(11.,8.5)
C                       ASSIGN PLOT PAGE SIZE
      CALL HWROT('AUTO')
C                       WANT HORIZONTAL LAYOUT FOR HARDCOPY
c       XPHY = 1.4
      xphy = 2.25
        YPHY = 1.0
C                       DEFINE PHYSICAL ORIGIN
c       XAREA = 8.2
      xarea = 6.5
        YAREA = 6.5
```

```
C                       DEFINE SUBPLOT DIMENSIONS  (AREA2D)
C
C** DRAW THE CAPTION
C
        CALL HEIGHT(.14)
C                       SET CHARACTER HEIGHT
        CALL DUPLX
C                       SET CHARACTER STYLE
        MAXLIN = LINEST(IPKRAY,400,80)
C                       INIT PACK ARRAY
c
      focl=foclen
      enr=energy(n)
        write(capt,*)'focl=',focl,'  rays=',mspot
     * ,'  energy=',enr,'$'
      call lines(capt,ipkray,1)
      write(capt,*)'peak=',peak,'  ctr=',xcen,ycen,'$'
      call lines(capt,ipkray,2)
c
c       CALL LINES('graztrace$',IPKRAY,1)
        NLINES = 2
C                       NUMBER OF LINES IN CAPTION
        YPHY = YPHY+1.5
C                       INCREMENT Y PHYSICAL ORIGIN
        YAREA = YAREA-1.5
C                       DECREMENT Y AREA TO FIT CAPTION
        CALL PHYSOR(XPHY,YPHY)
C                       DEFINE PHYSICAL ORIGIN
        CALL AREA2D(XAREA,YAREA)
C                       DEFINE PLOT AREA (VIEWPORT)
        CALL ALNSTY(.5,.5)
C                       CAPTION ALIGNMENT IS CENTER,CENTER
        CALL STORY(IPKRAY,NLINES,XAREA/2.,-YPHY/2.)
C                       PLOT CAPTION TEXT
C
C** GET DATA MINIMUM AND MAXIMUM
C
c       DATMIN = RMAT1(1,1)
c       DATMAX = RMAT1(1,1)
C                       INITIALIZE DATA MIN AND MAX
c       DO 80 J=1,IYDIM
C                       LOOP THROUGH EACH COLUMN
c          DO 70 I=1,IXDIM
C                       LOOP THROUGH EACH ROW OF THIS COLUMN
c             DATMIN = AMIN1(DATMIN,RMAT1(I,J))
C                       NEW MINIMUM?
c             DATMAX = AMAX1(DATMAX,RMAT1(I,J))
C                       NEW MAXIMUM?
c  70      CONTINUE
c  80   CONTINUE
C                       END DATA SCAN LOOP
C
        CALL FRAME
C                       FRAME THE SUBPLOT AREA
c       XMIN = -1
c       XSTP = .25
c       XMAX = 1
c       YMIN = -1
```

```
c       YSTP = .25
c       YMAX = 1
C                       DEFINE AXES MIN, STEP, MAX
C                       ESTABLISH AXES LIMITS
        CALL XNAME('x -axis$',100)
C                       FORCE X AXIS TO BE DRAWN, LABEL IT
        CALL YNAME('y -axis$',100)
C                       FORCE Y AXIS TO BE DRAWN, LABEL IT
        CALL GRAF(XMIN,XSTP,XMAX,YMIN,YSTP,YMAX)
C                       AXES SET-UP  (WINDOW)
C                       BRING DISSPLA TO LEVEL 3
c       ZINCR = 1
      zincr = (datmax-datmin)/ 10.
c
C                       USER-SUPPLIED Z-LEVEL INCREMENT
C                       (CONTOUR INTERVAL)
chen    CALL RASPLN(2.5)
C                       SMOOTH CONTOUR LINES
        CALL CONMAK(RMAT1,IXDIM,IYDIM,zincr)
C                       GENERATE CONTOUR LINES FROM SURFACE DATA.
c
      call conlin(0,'solid','nolabels',2,10)
      call conlin(1,'chndsh','nolabels',1,4)
      call conlin(2,'chndot','nolabels',1,5)
      call conlin(3,'dash','nolabels',1,4)
      call conlin(4,'dot','nolabels',1,3)
      call contur(5,'nolabels','draw')
c
c       CALL CONLIN(0,'MYCNLN','LABELS',3,10)
C                       SET CONTOUR LINE ATTRIBUTES FOR HIGHEST
C                       PRIORITY (MAJOR) LINES
c       DO 500 I=1,3
c 500      CALL CONLIN(I,'MYCNLN','NOLABELS',1,9)
C                       SET LINE ATTRIBUTES FOR REMAINING 3, LOWER
C                       PRIORITY (MINOR) LINES
c       CALL CONANG(90.)
C                       SET MAXIMUM ANGLE OF LINE CURVATURE FOR WHICH
C                       LABELS WILL NOT BE OMITTED
c       CALL CONTUR(4,'LABELS','DRAW')
C                       DRAW THE CONTOUR LINES WITH BLANKED LABELS
        CALL ENDPL(0)
C                       TERMINATE THE PLOT
C                       CALL THE APPLICATION SUBROUTINE
        CALL RESET('ALL')
        CALL DONEPL
C                       END DISSPLA
C
c end of energy loop
c
100     continue
        return
      end
c
        SUBROUTINE MYCNLN(RARAY,IARAY,LCHAR)
c         implicit double precision (a-h,o-z)
C**********************************************************
C
C  USER-SUPPLIED ESCAPE ROUTINE WILL BE CALLED BY DISSPLA
```

```
C  FOR EACH CONTOUR LEVEL -- USED FOR COLOR CONTROL OF THE
C  CONTOUR LINES.
C
C**************************************************************
        DIMENSION RARAY(2),IARAY(9)
      CHARACTER*20 LCHAR
        COMMON /MYCONX/ DATMIN,DATMAX
        HUE = (RARAY(1)-DATMIN)/(DATMAX-DATMIN)
C                       GET THIS Z-LEVEL'S PERCENTAGE OF TOTAL SCALE
        HUE = HUE*2.5+0.5
C                       SCALE BY HUE RANGE (.5 --> 3)
C                       AND ADD TO HUE BASE (0.5)
        CALL HWHSI(HUE,1.,1.)
C                       SET COLOR FOR THIS CONTOUR LEVEL
        RETURN
        END
```

A3.3 pltpsf 3-D Point Spread Function Plot (FORTRAN source code)

```
c*************************************************************
      subroutine pltpsf(mspot,size,ngrid,azm,ele)
c
c plot point spread function
c
c mspot, total number of rays
c size, half width of the plot regin in arc sec
c ngrid, grid number of the plot
c azm, azimuth of the view (60.0 yields good perspective)
c ele, elevation of the view (30.0 recommended)
c
c*************************************************************
      real rmat1(250,250)
      if (ngrid .eq. 0) ngrid = 50
      call pltpsf1(mspot,size,ngrid,azm,ele,rmat1)
      return
      end
c*************************************************************
      subroutine pltpsf1(mspot,size,ngrid,azm,ele,rmat1)
c
      implicit double precision (a-h,o-z)
c*******************************************************************
      common /syscl/ zrange,elev,azim,foclen,source(3)
     * ,radlim(2,50),dxcirc(50),dycirc(50)
     * ,xwidth(50),ywidth(50),dxrect(50),dyrect(50),threct(50)
     * ,zlim(2,50),adata(25,50)
     * ,tilt(3,50),rmat(3,3,50)
     * ,disp(3,50),thick(50),findex(50)
     * ,sdata(25,50),delta
     * ,sp(3,50),ra(3,50),spi(3),rai(3)
     * ,energy(15),delbet(2,15,50),wgt(15,50),wgtnet(15),effa(15)
     * ,pi
     * ,imove(50),irstr(50),iwgt(50),nsurf
     * ,nnrg,kmax,kprint(51),ichief,itilt(50)
     * ,npass,nvig,nerr
     * ,iaper(50),iobs(50),itype(50),imode(50),ifdfm(50),ihead(20)
      character * 80 ihead,ifdfm
      character * 8 itype,imode,iaper,iobs
c*******************************************************************
      common /rsavel/ xpsv(200000),ypsv(200000),dxdzsv(200000)
     * ,dydzsv(200000),entx(200000),enty(200000),wtsv(15,200000)
     * ,zshift,nsv
c*******************************************************************
c
      real datmin, datmax, xmin, xmax, ymin, ymax, xcen, ycen
     *  , xlen, ylen, xstp, ystp, xphy, yphy, xarea, yarea
     *  , work,lshad,phi,theta,radius,xvol,yvol,zvol
     *  , enr, peak, size, focl, azm, ele
      character *80 capt
      real rmat1(ngrid,ngrid)
c       DIMENSION WORK(2),XP(1000),YP(1000)
     dimension work(2)
```

```
        DIMENSION LSHAD (2)
        COMMON /PAKRAY/ IPKRAY(400)
C
c       DATA PHI/-60./, THETA/30./, RADIUS/60./
      DATA RADIUS/60./
        DATA LSHAD /45150, 135150/
C                       SHADE PATTERNS FOR THE FRONT & SIDE
C                       OF THE "SLAB"
        DATA IBELOW/'BELO'/
c      data ixdim/100/,iydim/100/
c
      phi = -azm
      theta = ele
      ixdim = ngrid
      iydim = ngrid
c*********************************************************************
c
c loop over energies
c
c*********************************************************************
c
       do 100, n=1, nnrg
c
c centroid
c
      swx=0
      swy=0
      sw=0
      do k=1,nsv
      swx=swx+wtsv(n,k)*xpsv(k)
      swy=swy+wtsv(n,k)*ypsv(k)
      sw=sw+wtsv(n,k)
      end do
      xcen=swx/sw
      ycen=swy/sw
c
c find plot range
c
       xlen=2*foclen*size*4.84813e-6
       if (size .lt. 1.0e-20) then
       xmin=xpsv(1)
       xmax=xpsv(1)
       ymin=ypsv(1)
       ymax=ypsv(1)
       do i=1, nsv
       xmin=dmin1(xmin,xpsv(i))
       xmax=dmax1(xmax,xpsv(i))
       ymin=dmin1(ymin,ypsv(i))
       ymax=dmax1(ymax,ypsv(i))
       end do
c
c round to good number
c
       xlen=2.*amax1(abs(xmin-xcen),abs(xmax-xcen)
      * ,abs(ymin-ycen),abs(ymax-ycen))
c      del = xlen/dble(ixdim)
c      k=dlog10(del)
c      if (del .lt. 1.d0) k=k-1
```

```
c     q=10.d0**k
c     c=del/q
c     ic=c
c     cp = ic
c     if (cp.lt.c) cp = cp +1.d0
c     xlen=ixdim*cp*q
      end if
c
      ylen=xlen
      xmin=xcen-.5*xlen
      xmax=xcen+.5*xlen
      ymin=ycen-.5*ylen
      ymax=ycen+.5*ylen
c
c tick interval
c
      xstp = xlen / 8.
      ystp = ylen / 8.
c
c prepare plot matrix
c
      do i=1,ixdim
      do j=1,iydim
      rmat1(i,j)=0.
      end do
      end do
c
c accumulate energy
c
      do k=1, nsv
      i=(xpsv(k)-xmin)*(ixdim-1)/xlen + 1
      j=(ypsv(k)-ymin)*(iydim-1)/ylen + 1
      if ((i .ge. 1) .and. (i .le. ixdim)
     *      .and. (j .ge. 1) .and. (j .le. iydim)) then
      rmat1(i,j)=rmat1(i,j)+wtsv(n,k)*energy(n)
c
      end if
      end do
c
c scale to intensity
c
      datmin=rmat1(1,1)*ixdim*iydim/(xlen*ylen)
      datmax=datmin
      do i=1,ixdim
      do j=1,iydim
      rmat1(i,j)=rmat1(i,j)*ixdim*iydim/(xlen*ylen)
      datmax=amax1(datmax,rmat1(i,j))
      datmin=amin1(datmin,rmat1(i,j))
      end do
      end do
      peak = datmax
      if (energy(n) .lt. 0) peak = datmin
c
c*******************************************************************
c
c plot
c
        CALL VT240
```

```
C                       NOMINATE A DEVICE
        CALL PAGE(11.,8.5     )
C                       ASSIGN PLOT PAGE SIZE
      CALL HWROT('AUTO')
C                       WANT HORIZONTAL LAYOUT FOR HARDCOPY
c        CALL PLOT(RMAT1,NX,NY)
chen
C
C** SET UP PLOTTING ENVIRONMENT
C
        XPHY = 1.4
        YPHY = 1.
C                       DEFAULT PHYSICAL ORIGIN
        XAREA = 8.2
        YAREA = 6.5
C                       DEFAULT PHYSICAL AREA
        XVOL = 12.
        YVOL = 12.
        ZVOL = 12.
C                       WORKBOX DIMENSIONS ARE EQUAL,
C                       RESULTING IN A CUBE SHAPED WORKBOX.
C                       CHANGE THESE AS NECESSARY TO CONFORM
C                       TO THE DATA.a
c
c draw caption
c
      call height(.14)
      call duplx
      maxlin = linest(ipkray,400,80)
      focl = foclen
      enr = energy(n)
      write(capt,*)'focl=',focl,'  rays=',mspot
     *    ,'  energy=',enr,'$'
      call lines(capt,ipkray,1)
      write(capt,*)'peak=',peak,'  ctr=',xcen,ycen,'$'
         call lines(capt,ipkray,2)
      nlines = 2
      yphy=yphy+1.5
      yarea=yarea-1.5
c
         CALL PHYSOR(XPHY,YPHY)
C                       DEFINE PHYSICAL ORIGIN
         CALL AREA2D(XAREA,YAREA)
C                       DEFINE PLOT AREA (VIEWPORT)
c
      call alnsty(.5,.5)
      call story(ipkray,nlines,xarea/2.,-yphy/2.)
c
c
c        XMIN = 0
c        XSTP = 1
c        XMAX = 10
c        YMIN = 0
c        YSTP = 1
c        YMAX = 10
C                       DEFINE AXES MIN, STEP, MAX
C
C** OBTAIN THE LOWEST AND HIGHEST VALUES FROM THE DATA GIVEN..
```

```
C
c       ZMIN = rMAT1(1,1)
c       ZMAX = rMAT1(1,1)
C                           INIT STARTING VARS FOR SEARCH
c       DO 20 II = 1,IXDIM
C                           DO FOR EACH X
c          DO 10 JJ = 1,IYDIM
C                           DO FOR EACH Y
c             ZMAX = AMAX1(ZMAX,rMAT1(II,JJ))
C                           RETAIN HIGHEST VALUE
c             ZMIN = AMIN1(ZMIN,rMAT1(II,JJ))
C                           RETAIN LOWEST VALUE
c10        CONTINUE
C                           NEXT Y
c20     CONTINUE
C                           NEXT X
C
C** DETERMINE SUITABLE Z AXIS MIN, MAX, STEP FROM THE HIGH/LOW
C** VALUES EXTRACTED FROM THE DATA.
C
        WORK(1) = datMAX
        WORK(2) = datMIN
        CALL RNDLIN(WORK,2,ZVOL/2.,datMIN,ZSTP,datMAX)
C                           DETERMINE LIMITS, STEP
C
C** PREPARE TO DRAW THE PLOT
C
        CALL VOLM3D(XVOL,YVOL,ZVOL)
C                           SET UP VOLUME PROPORTIONS
        CALL VUANGL(PHI,THETA,RADIUS)
C                           SET THE VIEWPOINT
        CALL BLSUR
C                           BLANK THE SURFACE AFTER IT IS DRAWN
        CALL HEIGHT(.15)
        CALL DUPLX
C                           SET CHAR HEIGHT & STYLE
        CALL INTAXS
C                           INTEGERIZE AXIS LABELS
        CALL ZAXANG(0.)
C                           Z AXIS LABELS ARE HORIZONTAL
C
C** DRAW THE SURFACE ..
C
        CALL X3NAME('x$',100)
C                           FORCE X AXIS TO BE DRAWN, LABEL IT
        CALL Y3NAME('y$',100)
C                           FORCE Y AXIS TO BE DRAWN, LABEL IT
        CALL Z3NAME(' intensity$',100)
C                           FORCE Z AXIS TO BE DRAWN, LABEL IT
        CALL GRAF3D(XMIN,XSTP,XMAX,YMIN,YSTP,YMAX,datMIN,ZSTP,datMAX)
C                           DEFINE USER AXIS SYSTEM (WINDOW)
        CALL SETCLR('CYAN')
C                           SET COLOR OF SURFACE
        CALL SURVIS('TOP')
C                           ONLY DRAW TOP OF SURFACE
        IXPTS = 1
        IYPTS = 1
C                           DRAW ONE SURFACE LINE PER DATA POINT
```

```
      CALL SURMAT(rMAT1,IXPTS,IXDIM,IYPTS,IYDIM,0)
C                         DRAW THE SURFACE
C
C** SHADE THE FRONT FACE OF THE "SLAB"
C
c     XINC = (XMAX-XMIN)/FLOAT(IXDIM-1)
C                         GET X INCREMENT SIZE
c     XX = XMIN
c     DO 25 II = 1,IXDIM
C                         DO FOR EACH X POINT
c        XP(II) = XX
c        YP(II) = rMAT1(II,1)
c        XX = XX+XINC
C                         BUMP X INCREMENT
c25   CONTINUE
C                         GET POINTS THAT DEFINE THE CURVE
c     CALL GRFITI(0.,0.,0.,XVOL,0.,0.,0.,0.,ZVOL)
C                         DEFINE GRFITI PLANE FOR FRONT FACE
c     CALL AREA2D(XVOL,ZVOL)
C                         SET SUBPLOT DIMENSIONS (VIEWPORT)
c     CALL GRAF(XMIN,XSTP,XMAX,datMIN,ZSTP,datMAX)
C                         DEFINE USER AXIS SYSTEM (WINDOW)
c     CALL SHDPAT(LSHAD(1))
C                         SET SHADE PATTERN
c     CALL SHDCRV(XP,YP,IXDIM,0,0,IBELOW)
C                         SHADE THE AREA
c     CALL END3GR(0)
C                         END THIS GRFITI PLANE
C
C** BLANK THE FRONT FACE OF THE SLAB WHERE WE JUST SHADED..
C
c     XX = XMIN
c     CALL RELPT3(XMAX,YMIN,ZMIN,XP(1),YP(1))
C                         PROJECT COORDS OF LOWER-CENTER POINT
C                         OF THE WORKBOX
c     CALL RELPT3(XMIN,YMIN,ZMIN,XP(2),YP(2))
C                         PROJECT COORDS OF LOWER-LEFT POINT
C                         OF THE WORKBOX
c     DO 30 II = 1,IXDIM
C                         DO FOR EACH X POINT IN MATRIX
c        CALL RELPT3(XX,YMIN,rMAT1(II,1),XP(II+2),YP(II+2))
C                         PROJECT EACH POINT FROM 3D -> 2D
c        XX = XX+XINC
C                         BUMP X INCREMENT
c30   CONTINUE
C                         NEXT X POINT
c     CALL BLPOLY(XP,YP,IXDIM+2,1.)
C                         BLANK THE FRONT AREA
C                         (AND OUTLINE IT)
C
C** SHADE THE RIGHT SIDE OF THE "SLAB"
C
c     XINC = (YMAX-YMIN)/FLOAT(IYDIM-1)
C                         GET X INCREMENT SIZE
c     XX = YMIN
c     DO 40 II = 1,IYDIM
C                         DO FOR EACH Y POINT
c        XP(II) = XX
```

```
c           YP(II) = rMAT1(IXDIM,II)
c           XX = XX+XINC
C                            BUMP X INCREMENT
c40     CONTINUE
C                            GET POINTS THAT DEFINE THE CURVE
c       CALL GRFITI(XVOL,0.,0.,XVOL,YVOL,0.,XVOL,0.,ZVOL)
C                            DEFINE GRFITI PLANE FOR RIGHT SIDE
c       CALL AREA2D(YVOL,ZVOL)
C                            SET SUBPLOT DIMENSIONS (VIEWPORT)
c       CALL GRAF(YMIN,YSTP,YMAX,datMIN,ZSTP,datMAX)
C                            DEFINE USER AXIS SYSTEM (WINDOW)
c       CALL SHDPAT(LSHAD(2))
C                            SET SHADE PATTERN
c       CALL SHDCRV(XP,YP,IYDIM,0,0,IBELOW)
C                            SHADE THE AREA
c       CALL END3GR(0)
C                            END THIS GRFITI PLANE
C
C** BLANK THE RIGHT SIDE OF THE SLAB WHERE WE JUST SHADED..
C
c       YY = YMIN
c       CALL RELPT3(XMAX,YMAX,datMIN,XP(1),YP(1))
C                            PROJECT COORDS OF LOWER RIGHT POINT
C                            OF THE WORKBOX
c       CALL RELPT3(XMAX,YMIN,datMIN,XP(2),YP(2))
C                            PROJECT COORDS OF LOWER CENTER POINT
C                            OF THE WORKBOX
c       DO 50 II = 1,IYDIM
C                            DO FOR EACH POINT IN Y
c          CALL RELPT3(XMAX,YY,rMAT1(IXDIM,II),XP(II+2),YP(II+2))
C                            PROJECT EACH POINT FROM 3D -> 2D
c          YY = YY+XINC
C                            BUMP Y INCREMENT
c50     CONTINUE
C                            NEXT Y POINT
c       CALL BLPOLY(XP,YP,IYDIM+2,1.)
C                            BLANK THE RIGHT SIDE
C
C** DRAW GRID LINES ON THE LEFT SIDE OF THE PLOT..
C
        CALL NEWCLR('FORE')
C                            SET COLOR TO FOREGROUND
        CALL GRFITI(0.,0.,0.,0.,YVOL,0.,0.,0.,ZVOL)
C                            DEFINE GRFITI PLANE ON LEFT SIDE
        CALL AREA2D(YVOL,ZVOL)
C                            SET SUBPLOT DIMENSIONS (VIEWPORT)
        CALL GRAF(0.,1.,1.,datMIN,ZSTP,datMAX)
C                            DEFINE USER AXIS SYSTEM (WINDOW)
        CALL GRID(0,1)
C                            GRID (IN Y DIR ONLY)
        CALL END3GR(0)
C                            END Y-Z PLANE (LEFT SIDE)
C
C** DRAW GRID LINES ON THE RIGHT SIDE OF THE PLOT..
C
        CALL GRFITI(0.,YVOL,0.,XVOL,YVOL,0.,0.,YVOL,ZVOL)
C                            DEFINE GRFITI PLANE ON RIGHT SIDE
        CALL AREA2D(XVOL,ZVOL)
```

```
C                         SET SUBPLOT DIMENSIONS (VIEWPORT)
        CALL GRAF(0.,1.,1.,datMIN,ZSTP,datMAX)
C                         DEFINE USER AXIS SYSTEM (WINDOW)
        CALL GRID(0,1)
C                         GRID (IN Y DIR ONLY)
        CALL END3GR(0)
C                         END THE PLOT ON RIGHT SIDE
        CALL RESET('BLNKS')
C                         TURN OFF ALL BLANKED AREAS
        CALL ENDPL(0)
C                         END THE CURRENT PLOT
chen
C                         CALL THE APPLICATION SUBROUTINE
        CALL RESET('ALL')
        CALL DONEPL
C                         END DISSPLA
C
C IF HERE, HAD ERROR OPENING DATA FILE
C
c 777    CONTINUE
c       WRITE(6,100)
c100    FORMAT(1X,'ERROR OPENING DATA FILE')
C                          WRITE TO TERMINAL
c       GOTO 9999
C
C IF HERE, HIT EOF
C
c888    CONTINUE
c       WRITE(6,200)
c200    FORMAT(1X,'END OF FILE')
C                          WRITE TO TERMINAL
c       GOTO 9999
C
C IF HERE, HAD ERROR READING FILE
C
c999    CONTINUE
c       WRITE(6,300)
c300    FORMAT(1X,'ERROR READING DATA FILE')
C                          WRITE TO TERMINAL
C
C PROGRAM END
C
c9999   CONTINUE
c       STOP
100     continue
        return
     END
```

A3.4 pltmakefile Makefile for Plot Routines

```
########################################################################
# makefile: pltmakefile
#
# This is the makefile for test plot routines
########################################################################

FILE= rayplot.o pltcnt.f pltpsf.f
testplot: ${FILE}; dis77links ${FILE} -o testplot

rayplot.o: rayplot.f; f77 -c rayplot.f
pltcnt.o: pltcnt.f; f77 -c pltcnt.f
pltpsf.o: pltpsf.f; f77 -c pltpsf.f
```

APPENDIX 4

THE COMMAND MODE GRAZTRACE MANUAL

GRAZTRACE

X-ray Optical Analysis Program

COMMAND MODE

User Manual

CONTENTS

Section 1. INTRODUCTION

Command Mode **GRAZTRACE** allows the users to interactively use the program. Command structure and format are similar to CODEV. Commands cover data manipulating, performance evaluation, internal parameter inquiry, and the utility commands.

Built in editor **EDI** allows the users to edit the system prescription without leaving the program. The operating system command shell **SYS** is also available.

Section 2. COMMAND FORMAT

The format of the command line is the same as in CODEV.

2.1 Command Syntax

Command lines have the general form:

COMMAND QUALIFIERS DATA ! COMMENT

particularly:

COMMAND

COMMAND DATA

COMMAND INDEX(INDICES) DATA.

Blanks are delimiters. Leading blanks and extra spaces are acceptable.

Command lines may be:

- Stacked on one lines by putting semi-colons(;) between the command lines
- Commented by using an exclamation mark(!); all characters following the (!) on that line will be ignored.

2.1.1 Command

First 3 characters of the string are recognized. Additional characters may be added to 3 character commands as desired for readability. The command string is not case sensitive.

2.1.2 Qualifiers

Index qualifiers are used to specify the surface number, field number, etc. Up to three dimensional array indices are allowed.

2.1.3 Data

- Numeric data
  Integers or floating point values, with or without leading sign (+,-) or leading zeros or power of ten exponents are accepted.

- Character string
  Any number of alphanumeric characters with or without being enclosed in single (') or double (") quotes.

- Question mark (?)
  Using question mark (?) in data field allows the user to check the current value of the variable.

2.1.4 Comment

- Any string of characters beginning with an exclamation mark (!).

- A comment may be an entire line starting with !

2.1.5 Error processing

The command interpreter will prompt for the correct command format when any error has been input in the command line.

2.1.6 Some examples of the Syntax are:

```
ZRA 2.0E20
ZRAnge 2.0E20
ZRA ?

SOU 1 1
Source 1 1
Source ?

! This is a Comment

SAV script1.tmp
SAVE script1.tmp

RES script1.tmp
Restore script1.tmp
```

SPO;GO

EDI
EDIT

LIS
list

LEN ! clear data buffer

2.2 Command Summary

More than 60 commands have been furnished in the command interpreter. Command sets consists of excutable commands, single field data commands, one dimensional array data commands, two dimensional array data commands, three dimensional array data commands, and one dimmensional character string data commands.

2.2.1 List of commands

Executable commands

LEN clear data buffer for new system

RES restore data buffer from file

SAV save data from data buffer to file

RSV save raytrace data to file

LIS list prescription

EDI screen edit data buffer

WSP random weighted ray trace

WS2 modified wheel spoke raytrace

GRI trace rays on a modified grid

GR2 trace rays on a grid

RSI single ray trace

FCS refocus

WST average position and rms

SPO spot diagram

RAD encircled energy distribution

GO excute the option
CAN cancel all inputs to this option

HEL help (typing ? in command will also get the same help)

SYS system command

EXI exiting program

Single field data commands

ZRA zrange (source distance)

ELE elev (source elevation)

AZI azmi (source azimuth)

FOC foclen (system focal length)

DET delta (convergence criterion)

SUR nsurf (number of total surfaces)

NRG nnrg (number of total energy levels)

MAX kmax (maximum iterations for intercept)

PAS npass (number of rays passed)

VIG nvig (number of rays vignetted)

ERR nerr (number of rays failed)

AZM azmid (azimuth middle point)

DAZ delaz (range of azimuth)

NRA mspot (number of rays)

IEN iener (current energy level)

XCE xav (x center)

YCE yav (y center)

AMA amax (maximum angle in RAD analysis)

NFR nf (number of fractions in RAD analysis)

One dimensional array data commands

SOU source (source position)

DXC dxcirc (obscuration radius x)

DYC dycirc (obscuration radius y)

XWI xwidth (square aperture width x)

YWI ywidth (square aperture height y)

DXR dxrect (rectangular obscuration width x)

DYR dyrect (rectangular obscuration height y)

THR threct (rectangular obscuration angle)

THI thick (surface saparation)

IND findex (surface index)

ENE energy (energy value)

EFF effa (effective area accumulation)

MOV imove (surface tilt flag)

RST irstr (surface restore flag)

WGT iwgt (surface reflectivity flag)

PRI kprint (surface ray print flag)

ITI itilt (surface tilt sequence)

Two dimensional array data commands

RLI radlim (minimum and maximum radiii of the surface)

ADA adata

TIL tilt (surface tilt data)

DIS disp (surface displacement data)

SDA sdata (surface data)

Three dimensional data array commands

MAT rmat (rotation matrix)

DEB delbet (reflectivity data)

One dimensional character string data commands

TIT title (surface header information)

APE aperture (surface frame type)

OBS obscuration (surface obscuration type)

TYP type (surface type)

MOD mode (surface ray trace mode)

FDF ifdfm (deformation file name)

More commands will be incorporated later on, based on the feedback from the users.

Section 3 QUICK START

This section contains a detailed and realistic "sample session" in **GRAZTRACE** command mode. This sample session will give user a quick start to get familiar with the **GRAZTRACE** program.

3.1 Using Command mode **GRAZTRACE**

The program can be invoked by typing **GT2**. The command mode prompt GTRACE> will show up. Key in any command interactively, followed by a carriage return <Enter>. To quit the program, use the **Exit** command. The program will prompt the user to confirm before exiting the program.

3.2 A Sample Session

zorro{chen}44> **GT2**

```
************************
*                      *
*      GrazTrace       *
*                      *
************************
```

GTRACE>**RES sample !** restore from file "sample"

GTRACE>**WSP !** random ray trace option

WSP>**GO !** execute the option

1 1000 successful rays in wspot1,
random ray distribution on first surface annulus
rmin = 0.7505025549956299E+02, rmax = 0.7640170861803300E+02
azmin (radians) = -0.3141592653589793E+01, azmax (radians) = 0.3141592653589793E+01
field angle (radians) = 0.0000000000000000E+00
azimuth (radians) = 0.0000000000000000E+00

0 rays were vignetted or obscured
0 rays failed in ssrt

```
energy( 1)= -0.1000000000000000E+01, effective area= 0.6430219044059306E+03
energy( 2)=  0.2770000000000000E+00, effective area= 0.4770593063472806E+03
energy( 3)=  0.5728000000000000E+00, effective area= 0.4832782607539592E+03

GTRACE>FCS ! refocus option

   FCS>GO ! execute the option

 weighted planar focus: energy( 1)= -0.1000000000000000E+01
 number of rays=    1000

   ***  stored rays modified   ***

 delta z  = -0.5466777139285604E-11,  net zshift= -0.5466777139285604E-11
 new   x   average=      -0.8423862330255359E-16,        new   y   average=
0.8400834138655648E-15

GTRACE>WST ! average position and rms option

  WST>GO ! execute the option

 length statistics for: energy( 1)= -0.1000000000000000E+01
 number of rays =    1000,  field angle (radians)      = 0.0000000000000000E+00
 net zshift      = -0.5466777139285604E-11
 x average         -0.8423862330255359E-16, y average = 0.8400834138655648E-15
 xrms            =  0.2011815720717621E-13,  yrms      = 0.1974284384504887E-13
          rms  =  0.2818723350211297E-13
 xmin= -0.7117447022322689E-13,  xmax=  0.7377058711339266E-13
 ymin= -0.7081536115041014E-13,  ymax=  0.6300754746166225E-13
    weight sum=  0.6430219044059306E+03
 weight average=  0.6430219044059307E+00
    weight rms=  0.0000000000000000E+00
 wmin=  0.6430219044059191E+00,  wmax=  0.6430219044059191E+00


arc sec statistics for: energy( 1)= -0.1000000000000000E+01
assumed focal length=  0.6564832312844800E+03,  number of rays   1000
x average (arc sec)  = -0.2646748993119960E-13
y average (arc sec)  =  0.2639513613368916E-12
xrms (arc sec)       =  0.6321056807905900E-11
yrms (arc sec)       =  0.6203134621577281E-11
rms (arc sec)        =  0.8856333231207158E-11
```

GTRACE>**SPO !** spot diagram option

SPO>**NRA 1000 !** set ray number 1000

SPO>**GO !** excute the option

1 spot diagram: first 1000 rays of 1000 stored
assumed center: x = -0.8423862330255359E-16, y = 0.8400834138655648E-15

Press <Enter> to continue ......

```
0    x-axis

    0.810E-13                         I
    0.720E-13                         *
    0.630E-13                       * I    *
    0.540E-13                      *  ***
    0.450E-13                       * * ** *  *
    0.360E-13                   ** **********
    0.270E-13                 ** ************** **
    0.180E-13                * ************** ***
    0.900E-14                 *********************
    0.000E+00 --------------*********************--------------
   -0.900E-14            *   *******************
   -0.180E-13               * ********************
   -0.270E-13                 ** ***************
   -0.360E-13                 * ****************
   -0.450E-13              *    *  ********* **
   -0.540E-13                 *    ***I* ** *  *
   -0.630E-13                       * *
   -0.720E-13                         I* **
   -0.810E-13                         I
              L                       M                        U
y-axis  -0.134344E-12          -0.474399E-14          0.124856E-12
```

GTRACE>**ZRA ? !** check z range

zrange = 1.0000000000000D+50

GTRACE>**ZRA 10000 !** try to change z range

GTRACE>**ZRA ? !** check it again

zrange = 10000.0000000000

GTRACE>**FOC ? !** check focal length

foclen = 656.48323128448

GTRACE>**EXI** exit the program

EXITING THE PROGRAM ? (Y/N)**Y**

zorro{chen}45>

Section 4 COMMAND REFERENCE

4.1 Data Manipulating Commands

ENTERING/CHANGING DATA

Manipulate system structural data.

COMMAND MNEMONICS (alphabetical)

AZI	APE	DAZ	DEB	DIS	DXC	DYC	DXR	DYR	ELE	ENE	FDF
FOC	IND	ITI	MOD	MOV	NRG	OBS	RLI	RST	SDA	SOU	SUR
TIL	TIT	TYP	THR	THI	WGT	XWI	YWI	ZRA			

THE TASK — **Re-starting for New Lens**

Command Syntax	
Screen Prompt	Explanation
LEN	
	Declares that the following entries are for a new system, rather than a modification to the old. Initializes defaults for a new system. All old system data are destroyed. LEN is not necessary prior to restoring a lens from the file.

THE TASK — Entering/Changing Data

Command Syntax	
Screen Prompt	Explanation
AZI azim	
	Set source azimuth angle
APE surf_num iaper	
	Declare surface frame type iaper — character string (*80) surf_num — surface number
DAZ delaz	
	Set azimuth range
DEB delb_num iener surf_num delb_val	
	Input surface reflectivity data (α,β) delb_num — reflectivity number = $\begin{cases} 1,\alpha \\ 2,\beta \end{cases}$ iener — energy level surf_num — surface number delb_val — delbet value
DIS dec_num surf_num dec_value	
	Set displacement data dec_num — decenter number = 1, X dec; 2, Y dec; 3, Z dec surf_num — surface number dec_val — decenter value
DXC surf_num radius_X	
	Set obscuration radius X surf_num — Surface number radius_x — radius X
DYC surf_num radius_Y	
	Set obscuration radius Y surf_num — surface number radius_y — radius y

DXR surf_num rect_X	
	Set obscuration width X surf_num — surface number rect_x — width X
DYR surf_num rect_y	
	Set obscuration height y surf_num — surface number rect_y — height y
ELE elev	
	Set source elevation elev — source elevation angle
ENE iener ener_val	
	Set energy levels iener — energy level number ener_val — energy level value
FDF surf_num ifdfm	
	Define deformation file name ifdfm — deformation file name
FOC foclen	
	Check or overwrite focal length foclen — system focal length
IND surf_num findex	
	Input surface index findex — surface index
ITI itilt	
	Define tilt sequence itilt — surface tilt sequence (e.g., 123 for 1,2,3)
MOD imode	
	Define surface ray trace mode imode — surface ray trace mode

MOV surf_num imove	
	Set surface tilt flag surf_num — surface number imove — surface tilt flag = 1, tilt; 0, not tilt
NRG nnrg	
	Declare total energy level number nnrg — total energy level number
OBS surf_number iobs	
	Define surface obscuration type surf_num — surface number iobs — surface obscuration type
RLI surf_num radlim_num radlim_val	
	Set minimum and maximum radii of the surface surf_num — surface number radlim_num — radii numbers = 1 minimum radius; 2 maximum radius radlim_val — radlim value
ADA surf_num adata_num data	
	Input surface error surf_num — surface number adata_num — surface error number adata — surface error
RST surf_num irstr	
	Set surface restore flag surf_num — surface number irstr — surface restore flag = 0, not restore; 1, restore
SDA surf_num sdata_num sdata	
	Input surface data surf_num — surface number sdata_num — surface data number sdata — surface data

SOU source_num source_pos	
	Define source position relative to undisplaced center of first surface source_num — source number = 1 , x / 2 , y / 3 , z source_pos — source position value
SUR nsurf	
	Define total number of surfaces nsurf — total number of surfaces
TIL tilt_num surf_num tilt_val	
	Input surface tilt data tilt_num — tilt number surf_num — surface number tilt_val — surface tilt value
TIT surf_num ihead	
	Set surface description surf_num — surface number ihead — surface head information
TYP surf_num itype	
	Define surface type surf_num — surface number itype — surface type
THR surf_num threct	
	Set angle of obscuration rectangle surf_num — surface number threct — angle of obscuration rectangle
THI surf_num thick	
	Input surface separation surf_num — surface number thick — surface separation
WGT surf_num iwgt	
	Set surface reflectivity weight flag surf_num — surface number iwgt — surface reflectivity weight flag

XWI surf-num xwidth	
	Input rectangular aperture width x surf_num — surface number xwidth — aperture width x
YWI surf_num ywidth	
	Input rectangular aperture height y surf_num — surface number ywidth — aperture height y
ZRA range	
	Set source distance to the first surface zrange — source distance

SAVING/RESTORING DATA

COMMAND MNEMONICS (alphabetical)

RES SAV

DATA INPUT DESCRIPTION

Save & Restore Data

Command Syntax	
Screen Prompt	Explanation
SAV [filspec]	
	Save lens in new version of filespec
RES [filspec]	
	Restore lens from filespec

RAY DATA SAVE (RSV)

RSV saves raytrace data as well as system data to a file

DATA INPUT DESCRIPTION

Command Syntax	
Screen Prompt	Explanation
RSV [filspec]	
	Save ray data as well as system data to a file

DISPLAYING DATA

LISTING OF DATA

Command Syntax	
Screen Prompt	Explanation
LIS	
	List all lens data

4.2 Performance Evaluation Commands

RANDOM RAY TRACE (WSP)

WSP traces nra successful rays randomly arranged on the first surface annulus at local Z=0. Intercepts, slopes and effective area weights are stored for the last surface for each ray.

COMMAND MNEMONICS (alphabetical)

AZM DAZ NRA

DATA INPUT DESCRIPTION

Command Syntax		
Screen Prompt	Explanation	Default
AZM azimuth middle angle		
	Set azimuth middle point	0
DAZ delta azimuth angle		
	Set azimuth range	2π
NRA number of rays		
	Set number of rays for random trace	1000

MODIFIED WHEEL SPOKE RAY TRACE (WS2)

WS2 traces modified wheel spoke rays arranged on the first surface annulus at local Z=0. Intercepts, slopes and effective area weights are stored for the last surface for each ray.

COMMAND MNEMONICS (alphabetical)

AZM DAZ NLO NAZ

DATA INPUT DESCRIPTION

Command Syntax		
Screen Prompt	Explanation	Default
AZM azimuth middle angle		
	Set azimuth middle point	0
DAZ delta azimuth angle		
	Set azimuth range	2π
NLO radial points		
	Set radial points for wheel spoke rays	100
NAZ azimuthal points		
	Set azimuthal points for wheel spoke rays	72

TRACE RAYS ON A MODIFIED GRID (GRI)

GRI traces rays on a grid with constant radial and varying azimuthal increments on the first surface annulus at local Z=0. Intercepts, slopes and effective area weights are stored for the last surface for each ray. Ray weights are set to 1.

COMMAND MNEMONICS (alphabetical)

AZM DAZ NLO NAZ

DATA INPUT DESCRIPTION

Command Syntax		
Screen Prompt	Explanation	Default
AZM azimuth middle angle		
	Set azimuth middle point	0
DAZ delta azimus angle		
	Set azimuth range	2π
NLO number of rays		
	Set number of rays along the radius	100
NAZ number of rays		
	Set maximum number of rays around the annuals	72

TRACE RAYS ON A GRID (GR2)

GR2 traces rays on a grid with constant radial and azimuthal increments on the first surface annulus at local Z=0. Intercepts, slopes and effective area weights are stored for the last surface for each ray. Ray weights are set to 1.

COMMAND MNEMONICS (alphabetical)

AZM DAZ NLO NAZ

DATA INPUT DESCRIPTION

Command Syntax		
Screen Prompt	Explanation	Default
AZM azimuth middle angle		
	Set azimuth middle point	0
DAZ delta azimuth angle		
	Set azimuth range	2π
NLO radial points		
	Set number of rays along the radius	100
NAZ azimuthal points		
	Set number of rays around the annuals	72

REFOCUS (FCS)

FCS refocuses the system.

COMMAND MNEMONICS (alphabetical)

IEN

DATA INPUT DESCRIPTION

Command Syntax		
Screen Prompt	Explanation	Default
IEN energy level		
	Cancel the default set and set desired energy level.	1

AVERAGE POSITION AND RMS **(WST)**

WST calculate average position and rms

COMMAND MNEMONICS (alphabetical)

IEN

DATA INPUT DESCRIPTION

Command Syntax		
Screen Prompt	Explanation	Default
IEN energy level		
	Cancel the default set and set desired energy level.	1

SPOT DIAGRAM (SPO)

SPO generates plots of ray interceptions with the image surface to represent image characteristics.

COMMAND MNEMONICS (alphabetical)

XCE, YCE, NRA

DATA INPUT DESCRIPTION

Command Syntax		
Screen Prompt	Explanation	Default
XCE center of X		
Center coordinate X	Override center coordinate X	Current average X
YCE center of Y		
Center coordinate Y	Center coordinate Y	Current average Y
NRA number of rays		
Number of rays for calculation	Cancel the default set and set desired ray number	1000

ENCIRCLED ENERGY (**RAD**)

RAD computes the radial energy distribution – the diameters in the image within which fixed percentages of light energy are contained.

COMMAND MNEMONICS (alphabetical)

AMA IEN NFR NRA XCE YCE

DATA INPUT DESCRIPTION

Command Syntax		
Screen Prompt	Explanation	Default
AMA ANGLE (ARC SEC)		
Maximum angle in arc sec for calculation	Cancel the default set and set desired angle	2.0
IEN energy level		
Energy level	Cancel the default set and set desired level	1
NFR number-of-fractions		
Number of fractions for radii calculation	Cancel the default set and set desired number	20
NRA number of rays		
Number of rays for calculation	Set desired ray number	500
XCE center of X		
Center coordinate X	Override center coordinate X	Current average X
YCE center of Y		
Center coordinate Y	Center coordinate Y	Current average Y

COMMAND MNEMONICS (alphabetical)

? CAN EXI GO

DATA INPUT DESCRIPTION

Command Syntax	
Screen Prompt	Explanation
?	
	? in data field entry will allow to check current value
GO	
	Execute the option using all previously entered option inputs and then return control to the command level
CAN	
	Cancel all inputs to this option and return control to the command level
EXI	
	Exit from GRAZTRACE to the operating system. When EXI is typed in, a query is issued requiring a Yes or No answer (Y or N); a Y will cancel any option you are in and complete the exit. (Default is N.)

4.3 System Parameter Commands

SYSTEM CONTROL PARAMETERS

Set system control parameter

COMMAND MNEMONICS (alphabetical)

DET MAX PRI

THE TASK — Set Control Parameters

Command Syntax	
Screen Prompt	Explanation
DET delta	
	Set ray intercept convergence criterion delta — convergence criterion
MAX kmax	
	Set maximum iteration loops for ray intercept kmax — maximum iteration loops
PRI surf_num kprint	
	Set surface ray print flag array surf_num — surface number kprint — print flag

SYSTEM RUNTIME PARAMETERS

Check system routine parameters

COMMAND MNEMONICS (alphabetical)

EFF ERR VIG PAS

THE TASK — Check routine parameters

Command Syntax	
Screen Prompt	Explanation
EFF	
	Check effective area accumulation
ERR	
	Check number of failure rays
VIG	
	Check number of vignetted rays
PAS	
	Check number of successful rays

4.4 Utility Commands

COMMAND MNEMONICS (alphabetical)

EDI SYS

DATA INPUT DESCRIPTION

Utilities

SYS ['OP_SYS_COMMAND'(250)]
Initiate a spawn out to the operating system to execute system command.
EDI
Enter the UNIX editor to edit the system prescription

APPENDIX 5

COMMAND MODE SOURCE CODE

Appendix 5 Command Mode GRAZTRACE Source Code and Sample Prescription

A5.1 cmd.f Command interpreter (FORTRAN source code)

```
cccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c       GT1 command mode graztrace
cccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      program main
      write(*,*)
      write(*,*)
      write(*,*)
      write(*,*)
      write(*,*)
      write(*,*)
      write(*,*)
      write(*,*)"                                  *****************************"
      write(*,*)"                                  *                           *"
      write(*,*)"                                  *         GrazTrace         *"
      write(*,*)"                                  *                           *"
      write(*,*)"                                  *****************************"
      write(*,*)
      write(*,*)
      write(*,*)
      write(*,*)
      write(*,*)
      write(*,*)
      write(*,*)
      write(*,*)
      write(*,*)
      write(*,*)
      write(*,*)
      call command
      end
c
        subroutine command
cccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c
c       command input for gtrace
c
cccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c
        implicit double precision (a-h, o-z)
        common /syscl/ zrange,elev,azim,foclen,source(3)
      * ,radlim(2,50),dxcirc(50),dycirc(50)
      * ,xwidth(50),ywidth(50),dxrect(50),dyrect(50),threct(50)
      * ,zlim(2,50),adata(25,50)
      * ,tilt(3,50),rmat(3,3,50)
      * ,disp(3,50),thick(50),findex(50)
      * ,sdata(25,50),delta
      * ,sp(3,50),ra(3,50),spi(3),rai(3)
      * ,energy(15),delbet(2,15,50),wgt(15,50),wgtnet(15),effa(15)
      * ,pi
      * ,imove(50),irstr(50),iwgt(50),nsurf
```

```
     * ,nnrg,kmax,kprint(51),ichief,itilt(50)
     * ,npass,nvig,nerr
     * ,iaper(50),iobs(50),itype(50),imode(50),ifdfm(50),ihead(20)
      character * 80 ihead,ifdfm
      character * 8 itype,imode,iaper,iobs
c*******************************************************************
       dimension enc(500),frac(100),rad(100),xref(15),yref(15)
       dimension work(3), tsp(3)
c*******************************************************************
c
       character*80 cmd_line, cmd_head, cmd_cont, tmp, cmd_buff
       character*8 cmd_sav, hlp_str
       character prompt*7, chr
       ibuff=0
       cmd_sav='hel'
       tmp='gt2hlp.doc'
       open (19,file=tmp,err=9801)
1000   prompt="GTRACE>"
1005   if (ibuff .eq. 1) then
       cmd_line=cmd_buff
       ibuff = 0
       go to 1010
       end if
cccccccccc    put prompt
       write(*,'(A,$)') prompt
cccccccccc    read in command line
       read (*,'(A)') cmd_line
       i=index(cmd_line,"!")
       if (i .ne. 0) then
       if (i .eq. 1 )then
       go to 1005
       else
       cmd_line=cmd_line(1:i-1)
       end if
       end if
1010   i=index(cmd_line,";")
       if (i .ne. 0) then
       tmp=cmd_line
       cmd_line=tmp(1:i-1)
       cmd_buff=tmp(i+1:80)
       ibuff = 1
       end if
       i=nindex(cmd_line," ")
       cmd_line=cmd_line(i:80)
cccccccccc    find command head and convert to all low case
       cmd_head=cmd_line(1:3)
       do i=1, 3
       n=ichar(cmd_head(i:i))
       if(n .ge. 65 .and. n .le. 90) then
       cmd_head(i:i)=char(n+32)
       end if
       end do
cccccccccc    save command head and put last command to help string
       hlp_str=cmd_sav
       cmd_sav=cmd_head
cccccccccc    search for space
       i=index( cmd_line, " ")
cccccccccc    find command content
```

```
      cmd_cont=cmd_line(i:80)
cccccccccc
c           command processing
ccccccccccccccccccccccccccccccccccccccccccccccc
c           single data field
ccccccccccccccccccccccccccccccccccccccccccccccc
c           ZRA processing
cccccccccc
c
      if (cmd_head .eq. 'zra') then
      read (cmd_cont,'(A)') tmp
      i=nindex(tmp, " ")
      if (tmp(i:i) .eq. '?') then
      write(*,*) "zrange = ",zrange
      else
      read (tmp,*,err=9101) zrange
      end if
      go to 1000
      end if
cccccccccc
c           ELE processing
cccccccccc
c
      if (cmd_head .eq. 'ele') then
      read (cmd_cont,'(A)') tmp
      i=nindex(tmp, " ")
      if (tmp(i:i) .eq. '?') then
      write(*,*) "elev = ",elev
      else
      read (tmp,*,err=9102) elev
      end if
      go to 1000
      end if
cccccccccc
c           AZI processing
cccccccccc
c
      if (cmd_head .eq. 'azi') then
      read (cmd_cont,'(A)') tmp
      i=nindex(tmp, " ")
      if (tmp(i:i) .eq. '?') then
      write(*,*) "azim = ",azim
      else
      read (tmp,*,err=9103) azim
      end if
      go to 1000
      end if
cccccccccc
c           FOC processing
cccccccccc
c
      if (cmd_head .eq. 'foc') then
      read (cmd_cont,'(A)') tmp
      i=nindex(tmp, " ")
      if (tmp(i:i) .eq. '?') then
      write(*,*) "foclen = ", foclen
      else
      read (tmp,*,err=9104) foclen
```

```
      end if
      go to 1000
      end if
cccccccccc
c              DET processing
cccccccccc
c
      if (cmd_head .eq. 'det') then
      read (cmd_cont,'(A)') tmp
      i=nindex(tmp, " ")
      if (tmp(i:i) .eq. '?') then
      write(*,*) "delta = ", delta
      else
      read (tmp,*,err=9105) delta
      end if
      go to 1000
      end if
cccccccccc
c              SUR processing
cccccccccc
c
      if (cmd_head .eq. 'sur') then
      read (cmd_cont,'(A)') tmp
      i=nindex(tmp, " ")
      if (tmp(i:i) .eq. '?') then
      write(*,*) "nsurf =", nsurf
      else
      read (tmp,*,err=9106) nsurf
      end if
      go to 1000
      end if
cccccccccc
c              NRG porcessing
cccccccccc
c
      if (cmd_head .eq. 'nrg') then
      read (cmd_cont,'(A)') tmp
      i=nindex(tmp, " ")
      if (tmp(i:i) .eq. '?') then
      write(*,*) "nnrg = ",nnrg
      else
      read (tmp,*,err=9107) nnrg
      end if
      go to 1000
      end if
cccccccccc
c              MAX processing
cccccccccc
c
      if (cmd_head .eq. 'max') then
      read (cmd_cont,'(A)') tmp
      i=nindex(tmp, " ")
      if (tmp(i:i) .eq. '?') then
      write(*,*) "kmax = ",kmax
      else
      read (tmp,*,err=9108) kmax
      end if
      go to 1000
```

```
      end if
cccccccccc
c           PAS processing
cccccccccc
c
      if (cmd_head .eq. 'pas') then
      read (cmd_cont,'(A)') tmp
      i=nindex(tmp, " ")
      if (tmp(i:i) .eq. '?') then
      write(*,*) "npass = ",npass
      else
      read (tmp,*,err=9109) npass
      end if
      go to 1000
      end if
cccccccccc
c           VIG processing
cccccccccc
c
      if (cmd_head .eq. 'vig') then
      read (cmd_cont,'(A)') tmp
      i=nindex(tmp, " ")
      if (tmp(i:i) .eq. '?') then
      write(*,*) "nvig = ",nvig
      else
      read (tmp,*,err=9110)nvig
      end if
      go to 1000
      end if
cccccccccc
c           ERR processing
cccccccccc
c
      if (cmd_head .eq. 'err') then
      read (cmd_cont,'(A)') tmp
      i=nindex(tmp, " ")
      if (tmp(i:i) .eq. '?') then
      write(*,*) "nerr = ",nerr
      else
      read (tmp,*,err=9111) nerr
      end if
      go to 1000
      end if
cccccccccc
c           AZM processing
cccccccccc
c
      if (cmd_head .eq. 'azm') then
      read (cmd_cont,'(A)') tmp
      i=nindex(tmp, " ")
      if (tmp(i:i) .eq. '?') then
      write(*,*) "azmid = ",azmid
      else
      read (tmp,*,err=9112) azmid
      end if
      go to 1005
      end if
cccccccccc
```

```
c              DAZ processing
cccccccccc
c
       if (cmd_head .eq. 'daz') then
       read (cmd_cont,'(A)') tmp
       i=nindex(tmp, " ")
       if (tmp(i:i) .eq. '?') then
       write(*,*) "delaz = ",delaz
       else
       read (tmp,*,err=9113) delaz
       end if
       go to 1005
       end if
cccccccccc
c              NRA processing
cccccccccc
c
       if (cmd_head .eq. 'nra') then
       read (cmd_cont,'(A)') tmp
       i=nindex(tmp, " ")
       if (tmp(i:i) .eq. '?') then
       write(*,*) "mspot = ",mspot
       else
       read (tmp,*,err=9114) mspot
       end if
       go to 1005
       end if
cccccccccc
c              XCE processing
cccccccccc
c
       if (cmd_head .eq. 'xce') then
       read (cmd_cont,'(A)') tmp
       i=nindex(tmp, " ")
       if (tmp(i:i) .eq. '?') then
       write(*,*) "xav = ", xav
       else
       read (tmp,*,err=9115) xav
       end if
       go to 1005
       end if
cccccccccc
c              YCE processing
cccccccccc
c
       if (cmd_head .eq. 'yce') then
       read (cmd_cont,'(A)') tmp
       i=nindex(tmp, " ")
       if (tmp(i:i) .eq. '?') then
       write(*,*) "yav = ", yav
       else
       read (tmp,*,err=9116) yav
       end if
       go to 1005
       end if
cccccccccc
c              IEN processing
cccccccccc
```

```
c
      if (cmd_head .eq. 'ien') then
      read (cmd_cont,'(A)') tmp
      i=nindex(tmp, " ")
      if (tmp(i:i) .eq. '?') then
      write(*,*) "iener = ", iener
      else
      read (tmp,*,err=9117) iener
      end if
      go to 1005
      end if
cccccccccc
c           AMA processing
cccccccccc
c
      if (cmd_head .eq. 'ama') then
      read (cmd_cont,'(A)') tmp
      i=nindex(tmp, " ")
      if (tmp(i:i) .eq. '?') then
      write(*,*) "amax = ", amax
      else
      read (tmp,*,err=9118) amax
      end if
      go to 1005
      end if
cccccccccc
c           NFR processing
cccccccccc
c
      if (cmd_head .eq. 'nfr') then
      read (cmd_cont,'(A)') tmp
      i=nindex(tmp, " ")
      if (tmp(i:i) .eq. '?') then
      write(*,*) "nf = ", nf
      else
      read (tmp,*,err=9119) nf
      end if
      go to 1005
      end if
cccccccccc
c           NLO processing
cccccccccc
c
      if (cmd_head .eq. 'nlo') then
      read (cmd_cont,'(A)') tmp
      i=nindex(tmp, " ")
      if (tmp(i:i) .eq. '?') then
      write(*,*) "nlong = ", nlong
      else
      read (tmp,*,err=9120) nlong
      end if
      go to 1005
      end if
cccccccccc
c           NLO processing
cccccccccc
c
      if (cmd_head .eq. 'naz') then
```

```
        read (cmd_cont,'(A)') tmp
        i=nindex(tmp, " ")
        if (tmp(i:i) .eq. '?') then
        write(*,*) "naz = ", naz
        else
        read (tmp,*,err=9121) naz
        end if
        go to 1005
        end if
cccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c              one dimensional array data command
cccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c              SOU processing
cccccccccc
c
        if (cmd_head .eq. "sou") then
        i=nindex(cmd_cont," ")
        if (cmd_cont(i:i).eq."?") then
         do i=1,3
         write(*,*) "source(",i,") = ",source(i)
         end do
         go to 1000
        end if
        read (cmd_cont,*,err=9201) ii
        i=nindex(cmd_cont," ")
        cmd_cont=cmd_cont(i:80)
        i=index(cmd_cont," ")
        cmd_cont=cmd_cont(i:80)
        read (cmd_cont,'(A)',err=9201) tmp
        i=nindex(tmp, " ")
        if (tmp(i:i) .eq. '?') then
        write(*,*) "source(",ii,") = ",source(ii)
        else
        read (tmp,*,err=9201) source(ii)
        end if
        go to 1000
        end if
cccccccccc
c              DXC processing
cccccccccc
c
        if (cmd_head .eq. "dxc") then
        i=nindex(cmd_cont," ")
        if (cmd_cont(i:i).eq."?") then
         do i=1,nsurf
         write(*,*) "dxcirc(",i,") = ",dxcirc(i)
         end do
         go to 1000
        end if
        read (cmd_cont,*,err=9202) ii
        i=nindex(cmd_cont," ")
        cmd_cont=cmd_cont(i:80)
        i=index(cmd_cont," ")
        cmd_cont=cmd_cont(i:80)
        read (cmd_cont,'(A)',err=9202) tmp
        i=nindex(tmp, " ")
        if (tmp(i:i) .eq. '?') then
        write(*,*) "dxcirc(",ii,") = ",dxcirc(ii)
```

```
        else
        read (tmp,*,err=9202) dxcirc(ii)
        end if
        go to 1000
        end if
cccccccccc
c             DYC processing
cccccccccc
c
        if (cmd_head .eq. "dyc") then
        i=nindex(cmd_cont," ")
        if (cmd_cont(i:i).eq."?") then
         do i=1,nsurf
         write(*,*) "dycirc(",i,") = ",dycirc(i)
         end do
         go to 1000
        end if
        read (cmd_cont,*,err=9203) ii
        i=nindex(cmd_cont," ")
        cmd_cont=cmd_cont(i:80)
        i=index(cmd_cont," ")
        cmd_cont=cmd_cont(i:80)
        read (cmd_cont,'(A)',err=9203) tmp
        i=nindex(tmp, " ")
        if (tmp(i:i) .eq. '?') then
        write(*,*) "dycirc(",ii,") = ",dycirc(ii)
        else
        read (tmp,*,err=9203) dycirc(ii)
        end if
        go to 1000
        end if
cccccccccc
c             XWI processing
cccccccccc
c
        if (cmd_head .eq. "xwi") then
        i=nindex(cmd_cont," ")
        if (cmd_cont(i:i).eq."?") then
         do i=1,nsurf
         write(*,*) "xwidth(",i,") = ",xwidth(i)
         end do
         go to 1000
        end if
        read (cmd_cont,*,err=9204) ii
        i=nindex(cmd_cont," ")
        cmd_cont=cmd_cont(i:80)
        i=index(cmd_cont," ")
        cmd_cont=cmd_cont(i:80)
        read (cmd_cont,'(A)',err=9204) tmp
        i=nindex(tmp, " ")
        if (tmp(i:i) .eq. '?') then
        write(*,*) "xwidth(",ii,") = ",xwidth(ii)
        else
        read (tmp,*,err=9204) xwidth(ii)
        end if
        go to 1000
        end if
cccccccccc
```

```
c              YWI processing
cccccccccc
c
      if (cmd_head .eq. "ywi") then
      i=nindex(cmd_cont," ")
      if (cmd_cont(i:i).eq."?") then
       do i=1,nsurf
       write(*,*) "ywidth(",i,") = ",ywidth(i)
       end do
       go to 1000
      end if
      read (cmd_cont,*,err=9205) ii
      i=nindex(cmd_cont," ")
      cmd_cont=cmd_cont(i:80)
      i=index(cmd_cont," ")
      cmd_cont=cmd_cont(i:80)
      read (cmd_cont,'(A)',err=9205) tmp
      i=nindex(tmp, " ")
      if (tmp(i:i) .eq. '?') then
      write(*,*) "ywidth(",ii,") = ",ywidth(ii)
      else
      read (tmp,*,err=9205) ywidth(ii)
      end if
      go to 1000
      end if
cccccccccc
c              DXR processing
cccccccccc
c
      if (cmd_head .eq. "dxr") then
      i=nindex(cmd_cont," ")
      if (cmd_cont(i:i).eq."?") then
       do i=1,nsurf
       write(*,*) "dxrect(",i,") = ",dxrect(i)
       end do
       go to 1000
      end if
      read (cmd_cont,*,err=9206) ii
      i=nindex(cmd_cont," ")
      cmd_cont=cmd_cont(i:80)
      i=index(cmd_cont," ")
      cmd_cont=cmd_cont(i:80)
      read (cmd_cont,'(A)',err=9206) tmp
      i=nindex(tmp, " ")
      if (tmp(i:i) .eq. '?') then
      write(*,*) "dxrect(",ii,") = ",dxrect(ii)
      else
      read (tmp,*,err=9206) dxrect(ii)
      end if
      go to 1000
      end if
cccccccccc
c              DYR processing
cccccccccc
c
      if (cmd_head .eq. "dyr") then
      i=nindex(cmd_cont," ")
      if (cmd_cont(i:i).eq."?") then
```

```
         do i=1,nsurf
         write(*,*) "dyrect(",i,") = ",dyrect(i)
         end do
         go to 1000
        end if
        read (cmd_cont,*,err=9207) ii
        i=nindex(cmd_cont," ")
        cmd_cont=cmd_cont(i:80)
        i=index(cmd_cont," ")
        cmd_cont=cmd_cont(i:80)
        read (cmd_cont,'(A)',err=9207) tmp
        i=nindex(tmp, " ")
        if (tmp(i:i) .eq. '?') then
        write(*,*) "dyrect(",ii,") = ",dyrect(ii)
        else
        read (tmp,*,err=9207) dyrect(ii)
        end if
        go to 1000
        end if
cccccccccc
c             THR processing
cccccccccc
c
        if (cmd_head .eq. "thr") then
        i=nindex(cmd_cont," ")
        if (cmd_cont(i:i).eq."?") then
         do i=1,nsurf
         write(*,*) "threct(",i,") = ",threct(i)
         end do
         go to 1000
        end if
        read (cmd_cont,*,err=9208) ii
        i=nindex(cmd_cont," ")
        cmd_cont=cmd_cont(i:80)
        i=index(cmd_cont," ")
        cmd_cont=cmd_cont(i:80)
        read (cmd_cont,'(A)',err=9208) tmp
        i=nindex(tmp, " ")
        if (tmp(i:i) .eq. '?') then
        write(*,*) "threct(",ii,") = ",threct(ii)
        else
        read (tmp,*,err=9208) threct(ii)
        end if
        go to 1000
        end if
cccccccccc
c             THI processing
cccccccccc
c
        if (cmd_head .eq. "thi") then
        i=nindex(cmd_cont," ")
        if (cmd_cont(i:i).eq."?") then
         do i=1,nsurf
         write(*,*) "thick(",i,") = ",thick(i)
         end do
         go to 1000
        end if
        read (cmd_cont,*,err=9209) ii
```

```
      i=nindex(cmd_cont," ")
      cmd_cont=cmd_cont(i:80)
      i=index(cmd_cont," ")
      cmd_cont=cmd_cont(i:80)
      read (cmd_cont,'(A)',err=9209) tmp
      i=nindex(tmp, " ")
      if (tmp(i:i) .eq. '?') then
      write(*,*) "thick (",ii,") = ",thick (ii)
      else
      read (tmp,*,err=9209)  thick(ii)
      end if
      go to 1000
      end if
cccccccccc
c            IND processing
cccccccccc
c
      if (cmd_head .eq. "ind") then
      i=nindex(cmd_cont," ")
      if (cmd_cont(i:i).eq."?") then
       do i=1,nsurf
       write(*,*) "findex(",i,") = ",findex(i)
       end do
       go to 1000
      end if
      read (cmd_cont,*,err=9210) ii
      i=nindex(cmd_cont," ")
      cmd_cont=cmd_cont(i:80)
      i=index(cmd_cont," ")
      cmd_cont=cmd_cont(i:80)
      read (cmd_cont,'(A)',err=9210) tmp
      i=nindex(tmp, " ")
      if (tmp(i:i) .eq. '?') then
      write(*,*) "findex(",ii,") = ",findex(ii)
      else
      read (tmp,*,err=9210) findex(ii)
      end if
      go to 1000
      end if
cccccccccc
c            ENE processing
cccccccccc
c
      if (cmd_head .eq. "ene") then
      i=nindex(cmd_cont," ")
      if (cmd_cont(i:i).eq."?") then
       do i=1,nnrg
       write(*,*) "energy(",i,") = ",energy(i)
       end do
       go to 1000
      end if
      read (cmd_cont,*,err=9211) ii
      i=nindex(cmd_cont," ")
      cmd_cont=cmd_cont(i:80)
      i=index(cmd_cont," ")
      cmd_cont=cmd_cont(i:80)
      read (cmd_cont,'(A)',err=9211) tmp
      i=nindex(tmp, " ")
```

```
      if (tmp(i:i) .eq. '?') then
      write(*,*) "energy(",ii,") = ",energy(ii)
      else
      read (tmp,*,err=9211) energy(ii)
      end if
      go to 1000
      end if
cccccccccc
c              EFF processing
cccccccccc
c
      if (cmd_head .eq. "eff") then
      i=nindex(cmd_cont," ")
      if (cmd_cont(i:i).eq."?") then
       do i=1,nnrg
       write(*,*) "effa(",i,") = ",effa(i)
       end do
       go to 1000
      end if
      read (cmd_cont,*,err=9212) ii
      i=nindex(cmd_cont," ")
      cmd_cont=cmd_cont(i:80)
      i=index(cmd_cont," ")
      cmd_cont=cmd_cont(i:80)
      read (cmd_cont,'(A)',err=9212) tmp
      i=nindex(tmp, " ")
      if (tmp(i:i) .eq. '?') then
      write(*,*) "  effa(",ii,") = ",  effa(ii)
      else
      read (tmp,*,err=9212)   effa(ii)
      end if
      go to 1000
      end if
cccccccccc
c              MOV processing
cccccccccc
c
      if (cmd_head .eq. "mov") then
      i=nindex(cmd_cont," ")
      if (cmd_cont(i:i).eq."?") then
       do i=1,nsurf
       write(*,*) "imove(",i,") = ",imove(i)
       end do
       go to 1000
      end if
      read (cmd_cont,*,err=9213) ii
      i=nindex(cmd_cont," ")
      cmd_cont=cmd_cont(i:80)
      i=index(cmd_cont," ")
      cmd_cont=cmd_cont(i:80)
      read (cmd_cont,'(A)',err=9213) tmp
      i=nindex(tmp, " ")
      if (tmp(i:i) .eq. '?') then
      write(*,*) " imove(",ii,") = ", imove(ii)
      else
      read (tmp,*,err=9213)  imove(ii)
      end if
      go to 1000
```

```
        end if
cccccccccc
c             RST processing
cccccccccc
c
        if (cmd_head .eq. "rst") then
        i=nindex(cmd_cont," ")
        if (cmd_cont(i:i).eq."?") then
         do i=1,nsurf
         write(*,*) "irstr(",i,") = ",irstr(i)
         end do
         go to 1000
        end if
        read (cmd_cont,*,err=9214) ii
        i=nindex(cmd_cont," ")
        cmd_cont=cmd_cont(i:80)
        i=index(cmd_cont," ")
        cmd_cont=cmd_cont(i:80)
        read (cmd_cont,'(A)',err=9214) tmp
        i=nindex(tmp, " ")
        if (tmp(i:i) .eq. '?') then
        write(*,*) " irstr(",ii,") = ", irstr(ii)
        else
        read (tmp,*,err=9214)  irstr(ii)
        end if
        go to 1000
        end if
cccccccccc
c             WGT processing
cccccccccc
c
        if (cmd_head .eq. "wgt") then
        i=nindex(cmd_cont," ")
        if (cmd_cont(i:i).eq."?") then
         do i=1,nsurf
         write(*,*) "iwgt(",i,") = ",iwgt(i)
         end do
         go to 1000
        end if
        read (cmd_cont,*,err=9215) ii
        i=nindex(cmd_cont," ")
        cmd_cont=cmd_cont(i:80)
        i=index(cmd_cont," ")
        cmd_cont=cmd_cont(i:80)
        read (cmd_cont,'(A)',err=9215) tmp
        i=nindex(tmp, " ")
        if (tmp(i:i) .eq. '?') then
        write(*,*) "  iwgt(",ii,") = ",  iwgt(ii)
        else
        read (tmp,*,err=9215)   iwgt(ii)
        end if
        go to 1000
        end if
cccccccccc
c             PRI processing
cccccccccc
c
        if (cmd_head .eq. "pri") then
```

```
      i=nindex(cmd_cont," ")
      if (cmd_cont(i:i).eq."?") then
       do i=1,nsurf
       write(*,*) "kprint(",i,") = ",kprint(i)
       end do
       go to 1000
      end if
      read (cmd_cont,*,err=9216)ii
      i=nindex(cmd_cont," ")
      cmd_cont=cmd_cont(i:80)
      i=index(cmd_cont," ")
      cmd_cont=cmd_cont(i:80)
      read (cmd_cont,'(A)',err=9216) tmp
      i=nindex(tmp, " ")
      if (tmp(i:i) .eq. '?') then
      write(*,*) "kprint(",ii,") = ",kprint(ii)
      else
      read (tmp,*,err=9216)kprint(ii)
      end if
      go to 1000
      end if
cccccccccc
c             ITI processing
cccccccccc
c
      if (cmd_head .eq. "iti") then
      i=nindex(cmd_cont," ")
      if (cmd_cont(i:i).eq."?") then
       do i=1,nsurf
       write(*,*) "itilt(",i,") = ",itilt(i)
       end do
       go to 1000
      end if
      read (cmd_cont,*,err=9217) ii
      i=nindex(cmd_cont," ")
      cmd_cont=cmd_cont(i:80)
      i=index(cmd_cont," ")
      cmd_cont=cmd_cont(i:80)
      read (cmd_cont,'(A)',err=9217) tmp
      i=nindex(tmp, " ")
      if (tmp(i:i) .eq. '?') then
      write(*,*) " itilt(",ii,") = ", itilt(ii)
      else
      read (tmp,*,err=9217) itilt(ii)
      end if
      go to 1000
      end if
ccccccccccccccccccccccccccccccccccccccccccccccccccc
c             two dimensional array data command
ccccccccccccccccccccccccccccccccccccccccccccccccccc
c             RLI processing
cccccccccc
c
      if (cmd_head .eq. "rli") then
      read (cmd_cont,*,err=9301) ii, jj
      do j=1,2
      i=nindex(cmd_cont," ")
      cmd_cont=cmd_cont(i:80)
```

```
        i=index(cmd_cont," ")
        cmd_cont=cmd_cont(i:80)
        end do
        read (cmd_cont,'(A)',err=9301) tmp
        i=nindex(tmp, " ")
        if (tmp(i:i) .eq. '?') then
        write(*,*) "radlim(",ii,",",jj,") = ",radlim(ii,jj)
        else
        read (tmp,*,err=9301) radlim(ii,jj)
        end if
        go to 1000
        end if
cccccccccc
c              ADA processing
cccccccccc
c
        if (cmd_head .eq. "ada") then
        read (cmd_cont,*,err=9302) ii, jj
        do j=1,2
        i=nindex(cmd_cont," ")
        cmd_cont=cmd_cont(i:80)
        i=index(cmd_cont," ")
        cmd_cont=cmd_cont(i:80)
        end do
        read (cmd_cont,'(A)',err=9302) tmp
        i=nindex(tmp, " ")
        if (tmp(i:i) .eq. '?') then
        write(*,*) " adata(",ii,",",jj,") = ", adata(ii,jj)
        else
        read (tmp,*,err=9302)  adata(ii,jj)
        end if
        go to 1000
        end if
cccccccccc
c              TIL processing
cccccccccc
c
        if (cmd_head .eq. "til") then
        read (cmd_cont,*,err=9303) ii, jj
        do j=1,2
        i=nindex(cmd_cont," ")
        cmd_cont=cmd_cont(i:80)
        i=index(cmd_cont," ")
        cmd_cont=cmd_cont(i:80)
        end do
        read (cmd_cont,'(A)',err=9303) tmp
        i=nindex(tmp, " ")
        if (tmp(i:i) .eq. '?') then
        write(*,*) "  tilt(",ii,",",jj,") = ",  tilt(ii,jj)
        else
        read (tmp,*,err=9303)   tilt(ii,jj)
        end if
        go to 1000
        end if
cccccccccc
c              DIS processing
cccccccccc
c
```

```
      if (cmd_head .eq. "dis") then
      read(cmd_cont,*,err=9304)ii,jj
      do j=1,2
      i=nindex(cmd_cont," ")
      cmd_cont=cmd_cont(i:80)
      i=index(cmd_cont," ")
      cmd_cont=cmd_cont(i:80)
      end do
      read (cmd_cont,'(A)',err=9304) tmp
      i=nindex(tmp, " ")
      if (tmp(i:i) .eq. '?') then
      write(*,*) "  disp(",ii,",",jj,") = ",  disp(ii,jj)
      else
      read (tmp,*,err=9304)   disp(ii,jj)
      end if
      go to 1000
      end if
cccccccccc
c             SDA processing
cccccccccc
c
      if (cmd_head .eq. "sda") then
      read (cmd_cont,*,err=9305) ii, jj
      do j=1,2
      i=nindex(cmd_cont," ")
      cmd_cont=cmd_cont(i:80)
      i=index(cmd_cont," ")
      cmd_cont=cmd_cont(i:80)
      end do
      read (cmd_cont,'(A)',err=9305) tmp
      i=nindex(tmp, " ")
      if (tmp(i:i) .eq. '?') then
      write(*,*) " sdata(",ii,",",jj,") = ", sdata(ii,jj)
      else
      read (tmp,*,err=9305)  sdata(ii,jj)
      end if
      go to 1000
      end if
cccccccccccccccccccccccccccccccccccccccccccccccccccc
c             three dimensional array data command
cccccccccccccccccccccccccccccccccccccccccccccccccccc
c             MAT processing
cccccccccc
c
      if (cmd_head .eq. "mat") then
      read (cmd_cont,*,err=9401) ii, jj, kk
      do j=1,3
      i=nindex(cmd_cont," ")
      cmd_cont=cmd_cont(i:80)
      i=index(cmd_cont," ")
      cmd_cont=cmd_cont(i:80)
      end do
      read (cmd_cont,'(A)',err=9401) tmp
      i=nindex(tmp, " ")
      if (tmp(i:i) .eq. '?') then
      write(*,*)"rmat(",ii,",",jj,",",kk,") = ",rmat(ii,jj,kk)
      else
      read (tmp,*,err=9401) rmat(ii,jj,kk)
```

```
      end if
      go to 1000
      end if
cccccccccc
c            DEB processing
cccccccccc
c
      if (cmd_head .eq. "del") then
      read (cmd_cont,*,err=9402) ii, jj, kk
      do j=1,3
      i=nindex(cmd_cont," ")
      cmd_cont=cmd_cont(i:80)
      i=index(cmd_cont," ")
      cmd_cont=cmd_cont(i:80)
      end do
      read (cmd_cont,'(A)',err=9402) tmp
      i=nindex(tmp, " ")
      if (tmp(i:i) .eq. '?') then
      write(*,*)"delbet(",ii,",",jj,",",kk,") = ",delbet(ii,jj,kk)
      else
      read (tmp,*,err=9402) delbet(ii,jj,kk)
      end if
      go to 1000
      end if
cccccccccc
c            RSI processing
cccccccccc
c
      if (cmd_head .eq. "rsi") then
      read (cmd_cont,*,err=9408) frad,ftheta,fxfld,fyfld
      do lsurf=1, nsurf
      kprint(lsurf)=1
      end do
      rr=rmin+(rmax-rmin)*frad
      theta=azmin+(azmax-azmin)*ftheta
      tsp(1)=rr*dcos(theta)
      tsp(2)=rr*dsin(theta)
      tsp(3)=0.d0
      work(1)=tsp(1)-source(1)*fxfld
      work(2)=tsp(2)-source(2)*fyfld
      work(3)=tsp(3)-source(3)
      sum=dsqrt(work(1)*work(1)+work(2)*work(2)+work(3)*work(3))
      do j=1,3
      spi(j)=tsp(j)
      rai(j)=work(j)/sum
      end do
      call wray(efact,irstat)
      do lsurf=1, nsurf
      kprint(lsurf)=0
      end do
      go to 1000
      end if
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c            One dimensional string data command
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c            APE processing
cccccccccc
c
```

```
      if (cmd_head .eq. 'ape') then
      i=nindex(cmd_cont," ")
      if (cmd_cont(i:i).eq."?") then
       do i=1,nsurf
       write(*,*) "iaper(",i,") = ",iaper(i)
       end do
       go to 1000
      end if
      read (cmd_cont,*,err=9601) ii
      i=nindex(cmd_cont," ")
      cmd_cont=cmd_cont(i:80)
      i=index(cmd_cont," ")
      cmd_cont=cmd_cont(i:80)
      read (cmd_cont,'(A)',err=9601)tmp
      i=nindex(tmp, " ")
      if (tmp(i:i) .eq. "?") then
      write(*,*)"iaper(",ii,") = ",iaper(ii)
      else
      read (tmp(i:80),'(A)',err=9601)iaper(ii)
      end if
      go to 1000
      end if
cccccccccc
c           OBS processing
cccccccccc
c
      if (cmd_head .eq. 'obs') then
      if (cmd_cont(i:i).eq."?") then
       do i=1,nsurf
       write(*,*) "iobs(",i,") = ", iobs(i)
       end do
       go to 1000
      end if
      read (cmd_cont,*,err=9602) ii
      i=nindex(cmd_cont," ")
      cmd_cont=cmd_cont(i:80)
      i=index(cmd_cont," ")
      cmd_cont=cmd_cont(i:80)
      read (cmd_cont,'(A)',err=9602) tmp
      i=nindex(tmp, " ")
      if (tmp(i:i) .eq. "?") then
      write(*,*) " iobs(",ii,") = ", iobs(ii)
      else
      read (tmp(i:80),'(A)',err=9602) iobs(ii)
      end if
      go to 1000
      end if
cccccccccc
c           TYP processing
cccccccccc
c
      if (cmd_head .eq. 'typ') then
      i=nindex(cmd_cont," ")
      if (cmd_cont(i:i).eq."?") then
       do i=1,nsurf
       write(*,*) "itype(",i,") = ",itype(i)
       end do
       go to 1000
```

```
        end if
        read (cmd_cont,*,err=9603) ii
        i=nindex(cmd_cont," ")
        cmd_cont=cmd_cont(i:80)
        i=index(cmd_cont," ")
        cmd_cont=cmd_cont(i:80)
        read (cmd_cont,'(A)',err=9603) tmp
        i=nindex(tmp, " ")
        if (tmp(i:i) .eq. "?") then
        write(*,*) "itype(",ii,") = ",itype(ii)
        else
        read (tmp(i:80),'(A)',err=9603)itype(ii)
        end if
        go to 1000
        end if
cccccccccc
c             MOD processing
cccccccccc
c
        if (cmd_head .eq. 'mod') then
        i=nindex(cmd_cont," ")
        if (cmd_cont(i:i).eq."?") then
         do i=1,nsurf
         write(*,*) "imode(",i,") = ",imode(i)
         end do
         go to 1000
        end if
        read (cmd_cont,*,err=9604) ii
        i=nindex(cmd_cont," ")
        cmd_cont=cmd_cont(i:80)
        i=index(cmd_cont," ")
        cmd_cont=cmd_cont(i:80)
        read (cmd_cont,'(A)',err=9604) tmp
        i=nindex(tmp, " ")
        if (tmp(i:i) .eq. "?") then
        write(*,*) "imode(",ii,") = ",imode(ii)
        else
        read (tmp(i:80),'(A)',err=9604)imode(ii)
        end if
        go to 1000
        end if
cccccccccc
c             FDF processing
cccccccccc
c
        if (cmd_head .eq. 'fdf') then
        i=nindex(cmd_cont," ")
        if (cmd_cont(i:i).eq."?") then
         do i=1,nsurf
         write(*,*) "ifdfm(",i,") = ",ifdfm(i)
         end do
         go to 1000
        end if
        read (cmd_cont,*,err=9605) ii
        i=nindex(cmd_cont," ")
        cmd_cont=cmd_cont(i:80)
        i=index(cmd_cont," ")
        cmd_cont=cmd_cont(i:80)
```

```
      read (cmd_cont,'(A)',err=9605) tmp
      i=nindex(tmp, " ")
      if (tmp(i:i) .eq. "?") then
      write(*,*) "ifdfm(",ii,") = ",ifdfm(ii)
      else
      read (tmp(i:80),'(A)',err=9605)ifdfm(ii)
      iurdfm=7
      call rdfm(iurdfm)
      end if
      go to 1000
      end if
cccccccccc
c           TIT processing
cccccccccc
c
      if (cmd_head .eq. 'tit') then
      i=nindex(cmd_cont," ")
      if (cmd_cont(i:i).eq."?") then
       do i=1,nsurf
       write(*,*) "ihead(",i,") = ",ihead(i)
       end do
       go to 1000
      end if
      read (cmd_cont,*,err=9606) ii
      i=nindex(cmd_cont," ")
      cmd_cont=cmd_cont(i:80)
      i=index(cmd_cont," ")
      cmd_cont=cmd_cont(i:80)
      read (cmd_cont,'(A)',err=9606) tmp
      i=nindex(tmp, " ")
      if (tmp(i:i) .eq. "?") then
      write(*,*) "ihead(",ii,") = ",ihead(ii)
      else
      read (tmp(i:80),'(A)',err=9606)ihead(ii)
      end if
      go to 1000
      end if
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c           Executable command
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c           RSV processing
cccccccccc
c
      if (cmd_head .eq. 'rsv') then
      read (cmd_cont,*,err=9700) tmp
      call wrayso(tmp)
      go to 1000
      end if
cccccccccc
c           SAV processing
cccccccccc
c
      if (cmd_head .eq. 'sav') then
      read (cmd_cont,*,err=9701) tmp
c     call system("cp presc.sxi.2 "// tmp)
      iu = 12
      istat = 1
      open (iu,file=tmp,err=9801)
```

```
        call rdout(iu,istat)
        if (istat .eq. -1) then
        write(*,*)"            SAVE ERROR"
        end if
        close(iu)
        go to 1000
        end if
cccccccccc
c              RES processing
cccccccccc
c
        if (cmd_head .eq. 'res') then
        read (cmd_cont,*,err=9702) tmp
c       call system("cp "// tmp //" presc.sxi.2 ")
        iu = 11
        istat=1
        call readin(iu, tmp, istat)
        if (istat .eq. -1) then
        write(*,*)"            RESTORE ERROR"
        end if
        call setcom(ierr)
        go to 1000
        end if
cccccccccc
c              SYS processing
cccccccccc
c
        if (cmd_head .eq. 'sys') then
        read (cmd_cont,'(A)') tmp
        if (nindex(tmp, " ") .gt. 60) then
        go to 9703
        end if
        call system(tmp)
        go to 1000
        end if
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c              no field command
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c              LEN processing
cccccccccc
        if (cmd_head .eq. "len") then
        call czero
        go to 1000
        end if
cccccccccc
c              LIS processing
cccccccccc
        if (cmd_head .eq. "lis") then
        call print01
        go to 1000
        end if
cccccccccc
c              EDI processing
cccccccccc
        if (cmd_head .eq. "edi") then
        iu = 13
        istat =1
        open (iu, file="gtracecm.tmp")
```

```
      call rdout (iu,istat)
      close(iu)
      call system("vi gtracecm.tmp")
      call readin(iu, "gtracecm.tmp", istat)
      if (istat .eq. -1) then
      write(*,*) "           EDIT ERROR"
      end if
      go to 1000
      end if
cccccccccc
c            ANA processing
cccccccccc
      if (cmd_head .eq. "ana") then
      iu = 13
      istat =1
      open (iu, file="presc.sxi.2")
      call rdout (iu,istat)
      close(iu)
      call system ("gt.exe")
      call system (" more print.gtrace")
      go to 1000
      end if
cccccccccc
c            WSP
cccccccccc
      if (cmd_head .eq. "wsp") then
      mspot=1000
      rmin=radlim(1,1)
      rmax=radlim(2,1)
      azmid=0.d0
      delaz=2.d0*pi
      prompt="   WSP>"
c     call wspot1(mspot,irand,rmin,rmax,azmin,azmax)
      go to 1005
      end if
cccccccccc
c            WS1
cccccccccc
      if (cmd_head .eq. "ws2") then
      nlong=100
      naz=72
      rmin=radlim(1,1)
      rmax=radlim(2,1)
      azmid=0.d0
      delaz=2.d0*pi
      prompt="   WS2>"
      go to 1005
      end if
cccccccccc
c            GRI
cccccccccc
      if (cmd_head .eq. "gri") then
      nlong=100
      naz=72
      rmin=radlim(1,1)*(1.d0+1.d-8)
      rmax=radlim(2,1)/(1.d0+1.d-8)
      prompt="   GRI>"
      go to 1005
```

```
       end if
cccccccccc
c            GR2
cccccccccc
       if (cmd_head .eq. "gr2") then
       nlong=100
       naz=72
       rmin=radlim(1,1)*(1.d0+1.d-8)
       rmax=radlim(2,1)/(1.d0+1.d-8)
       prompt="   GR2>"
       go to 1005
       end if
cccccccccc
c             FCS processing
cccccccccc
c
       if (cmd_head .eq. 'fcs') then
c      read (cmd_cont,'(A)') tmp
c      read (tmp,*,err=9704) iener
c      call focus(iener,xav,yav,delz)
       iener=1
       prompt="   FCS>"
       go to 1005
       end if
cccccccccc
c             WST processing
cccccccccc
c
       if (cmd_head .eq. 'wst') then
c      read (cmd_cont,'(A)') tmp
c      read (tmp,*,err=9705) iener
c      call wstat(iener,xav,yav,wav,wtot,xref(iener),yref(iener)
c    $   ,foclen,elev)
c      xref(iener)=xav
c      yref(iener)=yav
       iener=1
       prompt="   WST>"
       go to 1005
       end if
cccccccccc
c            SPO
cccccccccc
       if (cmd_head .eq. "spo") then
       mspot=0
       prompt="   SPO>"
       go to 1005
       end if
cccccccccc
c            RAD
cccccccccc
       if (cmd_head .eq. "rad") then
       mspot=500
       iener=1
       amax=2.d0
       nf=20
       prompt="   RAD>"
       go to 1005
       end if
```

```
cccccccccc
c           GO  processing
cccccccccc
      if (cmd_head .eq. "go ") then
c
      if (prompt .eq. "   WSP>") then
      azmin=azmid-delaz/2.d0
      azmax=azmid+delaz/2.d0
      irand=0
      call ranset(irand)
      call wspot1(mspot, irand,rmin,rmax,azmin,azmax)
      go to 1000
      end if
c
      if (prompt .eq. "   WS2>") then
      azmin=azmid-delaz/2.d0
      azmax=azmid+delaz/2.d0
      call wspot2(nlong, naz,rmin,rmax,azmin,azmax)
      go to 1000
      end if
c
      if (prompt .eq. "   GRI>") then
      call grid1(nlong, naz,rmin,rmax)
      go to 1000
      end if
c
      if (prompt .eq. "   GR2>") then
      call grid2(nlong, naz,rmin,rmax)
      go to 1000
      end if
c
      if (prompt .eq. "   FCS>")then
      call focus(iener,xav,yav,delz)
      go to 1000
      end if
c
      if (prompt .eq. "   WST>")then
      call wstat(iener,xav,yav,wav,wtot,xref(iener),yref(iener)
     $   ,foclen,elev)
      xref(iener)=xav
      yref(iener)=yav
      go to 1000
      end if
c
      if (prompt .eq. "   SPO>")then
      call spdiag(xav,yav,mspot)
      go to 1000
      end if
c
      if (prompt .eq. "   RAD>") then
      do i=1,nf
      frac(i)=dble(i)/dble(nf)
      end do
      frac(nf)=frac(nf)/(1.d0+1.d-8)
      call encirc(iener,xav,yav,foclen,amax,mspot,frac,rad,nf,enc
     $   ,wamax,wtot)
      go to 1000
      end if
```

```
c
      end if
cccccccccc
c           CAN processing
cccccccccc
      if (cmd_head .eq. "can" ) then
      go to 1000
      end if
cccccccccc
c           HEL processing
cccccccccc
      if (cmd_head .eq. "hel" .or. cmd_head(1:1) .eq. "?") then
      read(cmd_cont,'(A)')tmp
      if (cmd_cont .eq. '') cmd_cont=hlp_str
      call hlp(cmd_cont)
c     write(*,*)
c     write(*,*)"                                    GrazTrace Command"
c     write(*,*)
c     write(*,*)
c     write(*,*)"  ADA     ANA     AZI     APE     DEL     DET",
c    $ "     DIS     DXC     DYC     DXR"
c     write(*,*)
c     write(*,*)"  DYR     EDI     EFF     ELE     ENE     ERR",
c    $ "     EXI     FOC     FDF     IMO"
c     write(*,*)
c     write(*,*)"  IMG     IND     IRS     ITI     LEN     LIS",
c    $ "     MAX     MAT     MOD     NRG"
c     write(*,*)
c     write(*,*)"  OBS     PAS     PRI     RAD     RES     SAV",
c    $ "     SYS     THI     THR     TIT"
c     write(*,*)
c     write(*,*)"  ... ... "
c     write(*,*)
c     write(*,*)
c     write(*,*)
c     write(*,*)
c     write(*,*)
c     write(*,*)
c     write(*,*)
c     write(*,*)
c     write(*,*)
c     write(*,*)
      go to 1000
      end if
cccccccccc
c           EXI processing
cccccccccc
      if (cmd_head .eq. "exi") then
      write(*,'(A,$)') "EXITING THE PROGRAM ? (Y/N)"
      read(*,'(A)') tmp
      i=nindex(tmp," ")
      chr=tmp(i:i)
      if (chr .eq. "y" .or. chr .eq. "Y") then
      go to 9000
      end if
      go to 1000
      end if
cccccccccc
```

```
c            ENTER
cccccccccc
      if (cmd_head(1:1) .eq. "") then
      go to 1005
      end if
cccccccccc
c            UNKNOWN COMMAND
cccccccccc
      write(*,*) "Unknown command"
      go to 1005
cccccccccc
9000  stop
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c            command error
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c            command syntax prompt
cccccccccc
9101  write(*,*)"            Syntax: ZRA DATA"
      write(*,*)"                Or: ZRA ?"
      go to 1000
cccccccccc
9102  write(*,*)"            Syntax: ELE DATA"
      write(*,*)"                Or: ELE ?"
      go to 1000
cccccccccc
9103  write(*,*)"            Syntax: AZI DATA"
      write(*,*)"                Or: AZI ?"
      go to 1000
cccccccccc
9104  write(*,*)"            Syntax: FOC DATA"
      write(*,*)"                Or: FOC ?"
      go to 1000
cccccccccc
9105  write(*,*)"            Syntax: DET DATA"
      write(*,*)"                Or: DET ?"
      go to 1000
cccccccccc
9106  write(*,*)"            Syntax: SUR DATA"
      write(*,*)"                Or: SUR ?"
      go to 1000
cccccccccc
9107  write(*,*)"            Syntax: NRG DATA"
      write(*,*)"                Or: NRG ?"
      go to 1000
cccccccccc
9108  write(*,*)"            Syntax: MAX DATA"
      write(*,*)"                Or: MAX ?"
      go to 1000
cccccccccc
9109  write(*,*)"            Syntax: PAS DATA"
      write(*,*)"                Or: PAS ?"
      go to 1000
cccccccccc
9110  write(*,*)"            Syntax: VIG DATA"
      write(*,*)"                Or: VIG ?"
      go to 1000
cccccccccc
9111  write(*,*)"            Syntax: ERR DATA"
```

```
      write(*,*)"          Or: ERR ?"
      go to 1000
cccccccccc
9112  write(*,*)"      Syntax: AZM DATA"
      write(*,*)"          Or: AZM ?"
      go to 1005
cccccccccc
9113  write(*,*)"      Syntax: DAZ DATA"
      write(*,*)"          Or: DAZ ?"
      go to 1005
cccccccccc
9114  write(*,*)"      Syntax: NRA DATA"
      write(*,*)"          Or: NRA ?"
      go to 1005
cccccccccc
9115  write(*,*)"      Syntax: XCE DATA"
      write(*,*)"          Or: XCE ?"
      go to 1005
cccccccccc
9116  write(*,*)"      Syntax: YCE DATA"
      write(*,*)"          Or: YCE ?"
      go to 1005
cccccccccc
9117  write(*,*)"      Syntax: IEN DATA"
      write(*,*)"          Or: IEN ?"
      go to 1005
cccccccccc
9118  write(*,*)"      Syntax: AMA DATA"
      write(*,*)"          Or: AMA ?"
      go to 1005
cccccccccc
9119  write(*,*)"      Syntax: NFR DATA"
      write(*,*)"          Or: NFR ?"
      go to 1005
cccccccccc
9120  write(*,*)"      Syntax: NLO DATA"
      write(*,*)"          Or: NLO ?"
      go to 1005
cccccccccc
9121  write(*,*)"      Syntax: NAZ DATA"
      write(*,*)"          Or: NAZ ?"
      go to 1005
cccccccccc
9201  write(*,*)"      Syntax: SOU i DATA"
      write(*,*)"          Or: SOU i ?"
      go to 1000
cccccccccc
9202  write(*,*)"      Syntax: DXC i DATA"
      write(*,*)"          Or: DXC i ?"
      go to 1000
cccccccccc
9203  write(*,*)"      Syntax: DYC i DATA"
      write(*,*)"          Or: DYC i ?"
      go to 1000
cccccccccc
9204  write(*,*)"      Syntax: XWI i DATA"
      write(*,*)"          Or: XWI i ?"
      go to 1000
```

```
cccccccccc
9205   write(*,*)"        Syntax: YWI i DATA"
       write(*,*)"            Or: YWI i ?"
       go to 1000
cccccccccc
9206   write(*,*)"        Syntax: DXR i DATA"
       write(*,*)"            Or: DXR i ?"
       go to 1000
cccccccccc
9207   write(*,*)"        Syntax: DYR i DATA"
       write(*,*)"            Or: DYR i ?"
       go to 1000
cccccccccc
9208   write(*,*)"        Syntax: THR i DATA"
       write(*,*)"            Or: THR i ?"
       go to 1000
cccccccccc
9209   write(*,*)"        Syntax: THI i DATA"
       write(*,*)"            Or: THI i ?"
       go to 1000
cccccccccc
9210   write(*,*)"        Syntax: IND i DATA"
       write(*,*)"            Or: IND i ?"
       go to 1000
cccccccccc
9211   write(*,*)"        Syntax: ENE i DATA"
       write(*,*)"            Or: ENE i ?"
       go to 1000
cccccccccc
9212   write(*,*)"        Syntax: EFF i DATA"
       write(*,*)"            Or: EFF i ?"
       go to 1000
cccccccccc
9213   write(*,*)"        Syntax: MOV i DATA"
       write(*,*)"            Or: MOV i ?"
       go to 1000
cccccccccc
9214   write(*,*)"        Syntax: RST i DATA"
       write(*,*)"            Or: RST i ?"
       go to 1000
cccccccccc
9215   write(*,*)"        Syntax: WGT i DATA"
       write(*,*)"            Or: WGT i ?"
       go to 1000
cccccccccc
9216   write(*,*)"        Syntax: PRI i DATA"
       write(*,*)"            Or: PRI i ?"
       go to 1000
cccccccccc
9217   write(*,*)"        Syntax: ITI i DATA"
       write(*,*)"            Or: ITI i ?"
       go to 1000
cccccccccc
9301   write(*,*)"        Syntax: RLI i j DATA"
       write(*,*)"            Or: RLI i j ?"
       go to 1000
cccccccccc
9302   write(*,*)"        Syntax: ADA i j DATA"
```

```
        write(*,*)"          Or: ADA i j ?"
        go to 1000
cccccccccc
9303    write(*,*)"      Syntax: TIL i j DATA"
        write(*,*)"          Or: TIL i j ?"
        go to 1000
cccccccccc
9304    write(*,*)"      Syntax: DIS i j DATA"
        write(*,*)"          Or: DIS i j ?"
        go to 1000
cccccccccc
9305    write(*,*)"      Syntax: SDA i j DATA"
        write(*,*)"          Or: SDA i j ?"
        go to 1000
cccccccccc
9401    write(*,*)"      Syntax: MAT i j k DATA"
        write(*,*)"          Or: MAT i j k ?"
        go to 1000
cccccccccc
9402    write(*,*)"      Syntax: DEB i j k DATA"
        write(*,*)"          Or: DEB i j k ?"
        go to 1000
cccccccccc
9408    write(*,*)"      Syntax: RSI r_p theta_p x_fld y_fld"
        go to 1000
cccccccccc
9601    write(*,*)"      Syntax: APE i DESCRIPTION"
        write(*,*)"          Or: APE i ?          "
        go to 1000
cccccccccc
9602    write(*,*)"      Syntax: OBS i DESCRIPTION"
        write(*,*)"          Or: OBS i ?          "
        go to 1000
cccccccccc
9603    write(*,*)"      Syntax: TYP i DESCRIPTION"
        write(*,*)"          Or: TYP i ?          "
        go to 1000
cccccccccc
9604    write(*,*)"      Syntax: MOD i DESCRIPTION"
        write(*,*)"          Or: MOD i ?          "
        go to 1000
cccccccccc
9605    write(*,*)"      Syntax: FDF i DESCRIPTION"
        write(*,*)"          Or: FDF i ?          "
        go to 1000
cccccccccc
9606    write(*,*)"      Syntax: TIT i DESCRIPTION"
        write(*,*)"          Or: TIT i ?          "
        go to 1000
cccccccccc
9700    write(*,*)"      Syntax: RSV FILENAME"
        go to 1000
cccccccccc
9701    write(*,*)"      Syntax: SAV FILENAME"
        go to 1000
cccccccccc
9702    write(*,*)"      Syntax: RES FILENAME"
        go to 1000
```

```
cccccccccc
9703   write(*,*)"        Syntax: SYS ""SYSTEM COMMAND"" "
       go to 1000
cccccccccc
9704   write(*,*)"        Syntax: FCS energy"
       go to 1000
cccccccccc
9705   write(*,*)"        Syntax: WST energy"
       go to 1000
cccccccccc
9801   write(*,*)"        Error opening file",tmp
cccccccccc
       end
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c          function to find first nonmatching character position
c          in the string.  It is usefull to eliminate leading
c          space.
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
       function nindex(str,chr)
       character str*80, chr
       nindex=1
       do while ( index(str(nindex:80),chr) .eq. 1)
       nindex=nindex+1
       end do
       end
```

A5.2 gt2glp.f Interactive Help (FORTRAN source code)

```
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c
c          Interactive help for command mode GRAZTRACE
c
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c
c          This subroutine gives help imformation according
c          to the help word hlp_str.
c
c          hlp_str contains help word typed by user follows
c          HELp command.
c
c          If the user only type HELp with no word follows,
c          hlp_str picks up the latest command head.
c
c          If the latest command is simple <CR>, this help
c          list all the command available in GRAZTRACE.
c
c          When the hlp_str is an unknown command,  this help
c          also list all valid command in GRAZTRACE.
c
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c
      subroutine hlp(hlp_str)
      character*80 line, linep, linepp
      character*8 hlp_str
      write(*,*)
c
c   find head of help string and vonvert to all upper case
c
      i=nindex(hlp_str," ")
      hlp_str=hlp_str(i:8)
      hlp_str=hlp_str(1:3)
      do i=1,3
      n=ichar(hlp_str(i:i))
      if(n .ge. 97 .and. n .le. 122)then
      hlp_str(i:i)=char(n-32)
      end if
      end do
c
c   loop through help document
c
300   rewind(19)
      read(19,'(A)')linepp
      read(19,'(A)')line
      linep=line
      do while(line(i:i+2) .ne. 'Unk')
      read(19,'(A)')line
      i=nindex(line," ")
      if(line(i:i+2).eq.hlp_str.and.linep.eq.''.and.linepp.eq.'') then
      write(*,*)line
      do while(line(i:i+2) .ne. 'See')
      read(19,'(A)')line
```

```
      i=nindex(line," ")
      write(*,*)line
      end do
      go to 900
      else
      linepp=linep
      linep=line
      end if
      end do
c
c   unknown processing
c
      do i=1,18
      write(*,*)line
      read(19,'(A)',err=900)line
      end do
c
900   write(*,*)
      return
      end
```

A5.3 gt2.f GRAZTRACE for Command Mode (FORTRAN source code)

```
c
c******************************************************************
c
c  USER SUBROUTINE FOR SXI TELESCOPE RAY TRACE FOLLOWS
c
c*****************************************************************
c
      subroutine user
c
c  trace sxi system
c
      implicit double precision (a-h,o-z)
c*******************************************************************
      common /syscl/ zrange,elev,azim,foclen,source(3)
     * ,radlim(2,50),dxcirc(50),dycirc(50)
     * ,xwidth(50),ywidth(50),dxrect(50),dyrect(50),threct(50)
     * ,zlim(2,50),adata(25,50)
     * ,tilt(3,50),rmat(3,3,50)
     * ,disp(3,50),thick(50),findex(50)
     * ,sdata(25,50),delta
     * ,sp(3,50),ra(3,50),spi(3),rai(3)
     * ,energy(15),delbet(2,15,50),wgt(15,50),wgtnet(15),effa(15)
     * ,pi
     * ,imove(50),irstr(50),iwgt(50),nsurf
     * ,nnrg,kmax,kprint(51),ichief,itilt(50)
     * ,npass,nvig,nerr
     * ,iaper(50),iobs(50),itype(50),imode(50),ifdfm(50),ihead(20)
      character * 80 ihead,ifdfm
      character * 8 itype,imode,iaper,iobs
c********************************************************************
      dimension enc(500),frac(100),rad(100),xref(15),yref(15)
c*******************************************************************
c  output list file is default to print.gtrace
      open(6,file='print.gtrace')
c********************************************************************
c  flag for readin to open system input file
      istat=1
c*******************************************************************
c  number of systems to loop through
      nconic=1
c*******************************************************************
      do 900 iel=1,nconic
c  read in the prescription for the first element of the HRMA.
      call readin(1,'presc.sxi.2',istat)
      if(istat.ne.0) go to 900
c*******************************************************************
c  modifications
      ihead(2)=' '


c
c   parabola and hyperbola surface numbers.
      ip=5
```

```
      ih=11
c
c   modify parabola and hyperbola surface types.
c     itype(ip)='grzcon03'
c     itype(ih)='grzcon03'
c
c   reflectivity weight flags and number of energies
c     iwgt(ip)=0
c     iwgt(ih)=0
c     nnrg=1
c
      ifrom=6
      ito=3
      delbet(1,ito,ip)=delbet(1,ifrom,ip)
      delbet(2,ito,ip)=delbet(2,ifrom,ip)
      delbet(1,ito,ih)=delbet(1,ifrom,ih)
      delbet(2,ito,ih)=delbet(2,ifrom,ih)
      energy(ito)=energy(ifrom)
      nnrg=3
c   respace.
c   misc. cases
      d=0.d0
c   assume symmetric respace for the time being
c    (surface 7 is the finished end of the parabola)
c    (surface 8 is to be the position of the
c     mid point between the glass ends)
      thick(7)=thick(7)+d/2.d0
      thick(8)=thick(8)+d/2.d0
c   leave the distance between the mid point between the glass ends and
c   the nominal image plane unchanged.
c    (surface 16 is the image plane)
      thick(15)=thick(15)-d/2.d0
c
c   finite source distance to first surface
c   misc cases
c     zrange=1700.d0*12.d0*25.4d0
c   values from source to center distance and various respace errors
c   (t.casey 910129)
c     zrange=1731.d0*12.d0*25.4d0
c     n1=1
c     n2=7
c     do 600 i=n1,n2
c     zrange=zrange-thick(i)
c 600 continue
c
c  length of element
      size=zlim(2,ip)-zlim(1,ip)
c
c  elevation of source
c     elev=50.d0/3600.d0*pi/180.d0
c
c  azimuth of source
c     azim=0.d0
c     azim=pi/4.d0
      azim=pi/2.d0
c     azim=0.75d0*pi
c     azim=pi
c     azim=7.d0*pi/8.d0
```

```
c  modify distance to last surface
c     thick(nsurf-1)=thick(nsurf-1)+0.010d0
c
c  surface tilts
c     tilt(1,ih)=.15d0/3600.d0*pi/180.d0
c     tilt(2,ih)=.15d0/3600.d0*pi/180.d0
c     tilt(3,ih)=pi/4.d0
c     imove(ih)=1
c     irstr(ih)=1
c     itilt(ih)=213
c
c  hyperbola decenter and compensating tilt
c
c     decenx=0.d0
c     decenx=0.254d0
c     deceny=0.d0
c     deceny=0.254d0
c     n1=ih
c     n2=nsurf-1
c     zoff=10069.21899483571d0
c     comlen=zoff+d/2.d0
c     do 400 i=n1,n2
c     comlen=comlen+thick(i)
c 400 continue
c     comtx=-dasin(decenx/comlen)
c     comty=dasin(deceny/comlen)
c     imove(ih)=1
c     irstr(ih)=1
c     dcomtx=0.d0
c     dcomtx=.15d0/3600.d0*pi/180.d0
c     dcomty=0.d0
c     dcomty=.15d0/3600.d0*pi/180.d0
c     disp(1,ih)=decenx
c     tilt(2,ih)=comtx+dcomtx
c     disp(2,ih)=deceny
c     tilt(1,ih)=comty+dcomty
c
c  sag error
c     sdata(5,ip)=-400.d-7
c     sdata(5,ih)=-400.d-7
c  save minimum radius
      rminsv=radlim(1,1)
c  save zrange
      zrngsv=zrange
c  modify convergence criterium
c     delta=1.d-7
c
c  ray print flag
c     kprint(1)=2
c     kprint(2)=1
c     kprint(3)=ip
c     kprint(4)=ih
c     kprint(5)=nsurf
c
c  number of field angles
      nfield=2
c
      do 200 kk=1,nfield
```

```
c  adjust field angle
c
c     if(kk.eq.2) elev=1.d0/3600.d0*pi/180.d0
c     if(kk.eq.3) elev=50.d0/3600.d0*pi/180.d0
      elev=dble(kk-1)*1.d0/3600.d0*pi/180.d0
c
c  adjust tilt of first surface and zrange
c  to simulate field angle entry
c
c     if(kk.eq.2) then
c     tsv=-1.d0/3600.d0*pi/180.d0
c     tilt(2,1)=tsv
c     zrange=zrngsv/dcos(dabs(tsv))
c     imove(1)=1
c     endif
c
c     if(kk.eq.3) then
c     tsv=-50.d0/3600.d0*pi/180.d0
c     tilt(2,1)=tsv
c     zrange=zrngsv/dcos(dabs(tsv))
c     imove(1)=1
c     endif
c
c   adjust radius limits with field angle and source distance.
      radlim(1,1)=zrange/(zrange+size)*(rminsv
     *               -dtan(dabs(elev))*size)
c     radlim(1,1)=zrngsv/(zrngsv+size)*(rminsv
c    *               -dtan(dabs(tsv))*size)
c*********************************************************************
c  set up the common area.
      call setcom(ierr)
c*********************************************************************
c  print out the system common area
      call rdout(6,idum)
c*********************************************************************
c  do a weighted ray trace with random ray distribution in
c  entrance annulus
c
      ipat=1
c
      if(ipat.eq.1) then
c
      mspot=10000
      rmin=radlim(1,1)
      rmax=radlim(2,1)
      azmid=0.d0
      delaz=2.d0*pi
      azmin=azmid-delaz/2.d0
      azmax=azmid+delaz/2.d0
      irand=0
      call ranset(irand)
      call wspot1(mspot,irand,rmin,rmax,azmin,azmax)
c
      endif
c*********************************************************************
c  do a weighted ray trace with modified wheel spoke distribution in
c  entrance annulus
c
```

```
      ipat=0
c
      if(ipat.eq.1) then
c
      nlong=10000
      naz=1
      rmin=radlim(1,1)
      rmax=radlim(2,1)
      azmid=0.d0
      delaz=2.d0*pi/200.d0
c     delaz=2.d0*pi
      azmin=azmid-delaz/2.d0
      azmax=azmid+delaz/2.d0
      call wspot2(nlong,naz,rmin,rmax,azmin,azmax)
c
      endif
c**********************************************************************
c  do a ray trace with modified spoke wheel distribution.
c  (all weights set to 1)
c  (constant radial and varying azimuthal increments)
c  to compare with subroutine rfocs in vetasag.f
c     nlong=501
c     naz=72
c     rmin=radlim(1,1)*(1.d0+1.d-8)
c     rmax=radlim(2,1)/(1.d0+1.d-8)
c     call grid1(nlong,naz,rmin,rmax)
c**********************************************************************
c  do a ray trace with  spoke wheel distribution.
c  (all weights set to 1)
c  (constant radial and constant azimuthal increments)
c     naz=1
c     nlong=839
c     rmin=radlim(1,1)*(1.d0+1.d-8)
c     rmax=radlim(2,1)/(1.d0+1.d-8)
c     call grid2(nlong,naz,rmin,rmax)
c**********************************************************************
c
c  loop over energies
c
      do 300 iener=1,nnrg
c**********************************************************************
c  refocus
      call focus(iener,xav,yav,delz)
c**********************************************************************
c  calculate average position and rms
      if(kk.eq.1) then
      xref(iener)=0.d0
      yref(iener)=0.d0
      endif
      call wstat(iener,xav,yav,wav,wtot,xref(iener),yref(iener)
     * ,foclen,elev)
      if(kk.eq.1) then
c  get reference for apparent focal length calculation
      xref(iener)=xav
      yref(iener)=yav
      endif
c**********************************************************************
c  make unweighted spot diagram
```

```
      call spdiag(xav,yav,0)
c*********************************************************************
c  calculate encircled energy distribution
c     maximum angle in arc sec for calculation
      amax=2.d0
c     number of calculation points
      na=500
c     number of fractions for radii calculation
      nf=20
      do 100 i=1,nf
      frac(i)=dble(i)/dble(nf)
  100 continue
      frac(nf)=frac(nf)/(1.d0+1.d-8)
      call encirc(iener,xav,yav,foclen,amax,na,frac,rad,nf,enc
     * ,wamax,wtot)
c*********************************************************************
c end of energy loop
  300 continue
c*********************************************************************
c  write out system data and ray data to files
c      call wrayso('ring.gtray')
c*********************************************************************
c end of field angle loop
  200 continue
c*********************************************************************
c end of mirror system loop
  900 continue
c*****************************************************************
      return
      end
c
c*****************************************************************
c
c   RAY TRACE ROUTINES FOLLOW
c
c*****************************************************************
c
chen  program main
chen  implicit double precision (a-h,o-z)
chen  open(6,file='print.gtrace')
chen  call user
chen  stop
chen  end
      subroutine calwgt(lsurf)
c
c  accumulate metal reflectivity weights for applicable surface lsurf
c  and update ray effective area weight.
c
c  this is only for surfaces with iwgt(lsurf)=1
c
c  surface 1 cannot be used to calculate a reflective weight
c
      implicit double precision (a-h,o-z)
c*********************************************************************
      common /sysc1/ zrange,elev,azim,foclen,source(3)
     * ,radlim(2,50),dxcirc(50),dycirc(50)
     * ,xwidth(50),ywidth(50),dxrect(50),dyrect(50),threct(50)
     * ,zlim(2,50),adata(25,50)
```

```
     * ,tilt(3,50),rmat(3,3,50)
     * ,disp(3,50),thick(50),findex(50)
     * ,sdata(25,50),delta
     * ,sp(3,50),ra(3,50),spi(3),rai(3)
     * ,energy(15),delbet(2,15,50),wgt(15,50),wgtnet(15),effa(15)
     * ,pi
     * ,imove(50),irstr(50),iwgt(50),nsurf
     * ,nnrg,kmax,kprint(51),ichief,itilt(50)
     * ,npass,nvig,nerr
     * ,iaper(50),iobs(50),itype(50),imode(50),ifdfm(50),ihead(20)
      character * 80 ihead,ifdfm
      character * 8 itype,imode,iaper,iobs
c***********************************************************************
c
      if(lsurf.le.1) go to 3000
c  cyle through energies
      do 100 i=1,nnrg
c  weight is assumed to be 1.d0 if energy.le.0.d0
      if(energy(i).gt.0.d0) then
c  calculate angle of incidence in radians
      dot=0.d0
      do 200 j=1,3
  200 dot=dot+ra(j,lsurf-1)*ra(j,lsurf)
      anginc=(pi-dacos(dot))/2.d0
c  get reflectivity
      call metref(anginc,delbet(1,i,lsurf),delbet(2,i,lsurf),rs,rp)
c  assume no polarization
      wgt(i,lsurf)=(rs+rp)/2.d0
      else
      wgt(i,lsurf)=1.d0
      endif
      wgtnet(i)=wgt(i,lsurf)*wgtnet(i)
  100 continue
      return
c
 3000 continue
      write(6,3001) lsurf
 3001 format('0******  calwgt, ivalid surface no. ',i6)
      write(6,3002)
 3002 format('******   stop    *************')
      stop
      end
      subroutine cnvin(sp1,ra1,sp2,ra2,rmat,disp)
c
c  transform into or out of local coordinates
c
c    input position and direction cosines are sp1,ra1
c    output values are in sp2,ra2
c    rmat is transformation matrix
c    disp is displacement array
c
      implicit double precision (a-h,o-z)
      dimension sp1(3),ra1(3),sp2(3),ra2(3),disp(3),rmat(3,3)
      dimension tsp(3),tra(3)
      do 100 i=1,3
      tsp(i)=sp1(i)
  100 tra(i)=ra1(i)
      do 101 i=1,3
```

```
      sp2(i)=0.d0
      ra2(i)=0.d0
      do 101 j=1,3
      sp2(i)=sp2(i)+rmat(i,j)*(tsp(j)-disp(j))
  101 ra2(i)=ra2(i)+rmat(i,j)*tra(j)
      return
      entry cnvout(sp1,ra1,sp2,ra2,rmat,disp)
      do 200 i=1,3
      tsp(i)=sp1(i)
  200 tra(i)=ra1(i)
      do 201 i=1,3
      sp2(i)=disp(i)
      ra2(i)=0.d0
      do 201 j=1,3
      sp2(i)=sp2(i)+rmat(j,i)*tsp(j)
  201 ra2(i)=ra2(i)+rmat(j,i)*tra(j)
      return
      end
      subroutine czero
c
c  zero the common area, this routine is dependent on the
c  size, type, and order of the variables in syscl
c
      implicit double precision (a-h,o-z)
c*********************************************************************
c     common /sysc1/ zrange,elev,azim,foclen,source(3)
c    * ,radlim(2,50),dxcirc(50),dycirc(50)
c    * ,xwidth(50),ywidth(50),dxrect(50),dyrect(50),threct(50)
c    * ,zlim(2,50),adata(25,50)
c    * ,tilt(3,50),rmat(3,3,50)
c    * ,disp(3,50),thick(50),findex(50)
c    * ,sdata(25,50),delta
c    * ,sp(3,50),ra(3,50),spi(3),rai(3)
c    * ,energy(15),delbet(2,15,50),wgt(15,50),wgtnet(15),effa(15)
c    * ,pi
c    * ,imove(50),irstr(50),iwgt(50),nsurf
c    * ,nnrg,kmax,kprint(51),ichief,itilt(50)
c    * ,npass,nvig,nerr
c    * ,iaper(50),iobs(50),itype(50),imode(50),ifdfm(50),ihead(20)
c     character * 80 ihead,ifdfm
c     character * 8 itype,imode,iaper,iobs
c*********************************************************************
      common /sysc1/ zdum(6510),idum(258),cdum1(200),cdum2(70)
      character * 8 cdum1
      character * 80 cdum2
      do 100 i=1,6510
  100 zdum(i)=0.d0
      do 200 i=1,258
  200 idum(i)=0
      do 300 i=1,200
  300 cdum1(i)=' '
      do 400 i=1,70
  400 cdum2(i)=' '
      return
      end
      subroutine encirc(iener,xcen,ycen,ft,amax,na,frac,rad,nf,enc
     * ,wamax,wtot)
c********************************************************************
```

```
c
c  calculate encircled energy distribution for energy iener
c
c  input:
c
c          iener                  energy pointer.
c          xcen,ycen              assumed center of encircled energy
c                                 distribution.
c          ft                     assumed focal length.
c          amax                   maximum angle considered (arc sec)
c                                 for encircled energy distribution
c                                 calculation.
c          na                     number of radius increments for
c                                 encircled energy distribution
c                                 calculation.
c          frac                   encircled energy fractions
c                                 for radii calculations.
c          nf                     number of encircled energy fractions.
c
c  output:
c
c          rad                    radii values calculated for
c                                 nf fraction values input.
c          enc                    encircled energy distribution
c                                 (at na radius values up to amax)
c        wamax                    weight total up to radius amax
c         wtot                    total weight sum
      implicit double precision (a-h,o-z)
c*******************************************************************
      common /syscl/ zrange,elev,azim,foclen,source(3)
     * ,radlim(2,50),dxcirc(50),dycirc(50)
     * ,xwidth(50),ywidth(50),dxrect(50),dyrect(50),threct(50)
     * ,zlim(2,50),adata(25,50)
     * ,tilt(3,50),rmat(3,3,50)
     * ,disp(3,50),thick(50),findex(50)
     * ,sdata(25,50),delta
     * ,sp(3,50),ra(3,50),spi(3),rai(3)
     * ,energy(15),delbet(2,15,50),wgt(15,50),wgtnet(15),effa(15)
     * ,pi
     * ,imove(50),irstr(50),iwgt(50),nsurf
     * ,nnrg,kmax,kprint(51),ichief,itilt(50)
     * ,npass,nvig,nerr
     * ,iaper(50),iobs(50),itype(50),imode(50),ifdfm(50),ihead(20)
      character * 80 ihead,ifdfm
      character * 8 itype,imode,iaper,iobs
c*******************************************************************
      common /rsavel/ xpsv(200000),ypsv(200000),dxdzsv(200000)
     * ,dydzsv(200000),entx(200000),enty(200000),wtsv(15,200000)
     * ,zshift,nsv
c*******************************************************************
      common /worksp/ bigmat(600000)
c******************************************************************
      dimension frac(nf),rad(nf),enc(na)
      dimension work(200000)
      equivalence (work,bigmat)
c******************************************************************
      do 100 i=1,nsv
      work(i)=wtsv(iener,i)
```

```
 100 continue
     dltr=ft*dtan(amax/dble(na)/3600.d0*pi/180.d0)
     ne=na
     rmax=dltr*dble(ne)
     call red(xpsv,ypsv,xcen,ycen,work,nsv,enc,rmax,ne,frac,rad,nf
    * ,wamax,wtot)
     write(6,201) iener,energy(iener),nsv,xcen,ycen,ft,amax
    * ,amax/ne,wtot,wamax/wtot
 201 format('1encircled energy distribution for energy(',i2
    * ,')= ',e24.16/
    * ' number of rays ',i7/
    * ' assumed center: x= ',e24.16,',    y= ',e24.16/
    * '                           assumed focal length= ',e24.16/
    * '        calculation cut off radius (arc sec)= ',e24.16/
    * '             calculation interval (arc sec)= ',e24.16/
    * '                                weight sum= ',e24.16/


    * ' fraction of weight within cut off radius= ',e24.16//)
     factor=180.d0/pi*3600.d0
chen*************
c    pause
c
     write(*,*)
     write(*,*) 'Press <Enter> to continue ......'
     read(*,*)
chen*************
     do 202 i=1,nf
     write(6,203) i,frac(i),factor*datan(rad(i)/ft),rad(i)
chen 203 format(' no. ',i3,',   fraction= ',f6.4
chen     * ,',   radius(arc sec)= ',f10.4,',   radius= ',e24.16)
  203 format(' no.',i3,', frac= ',f5.3
     * ,', radius(arc sec)= ',f10.4,', radius= ',e14.6)
  202 continue
     return
     end
     subroutine focus(iener,xav,yav,delz)
c****************************************************************
c
c focus spot in storage array at energy position iener
c
c****************************************************************
     implicit double precision (a-h,o-z)
c****************************************************************
     common /syscl/ zrange,elev,azim,foclen,source(3)
    * ,radlim(2,50),dxcirc(50),dycirc(50)
    * ,xwidth(50),ywidth(50),dxrect(50),dyrect(50),threct(50)
    * ,zlim(2,50),adata(25,50)
    * ,tilt(3,50),rmat(3,3,50)
    * ,disp(3,50),thick(50),findex(50)
    * ,sdata(25,50),delta
    * ,sp(3,50),ra(3,50),spi(3),rai(3)
    * ,energy(15),delbet(2,15,50),wgt(15,50),wgtnet(15),effa(15)
    * ,pi
    * ,imove(50),irstr(50),iwgt(50),nsurf
    * ,nnrg,kmax,kprint(51),ichief,itilt(50)
    * ,npass,nvig,nerr
    * ,iaper(50),iobs(50),itype(50),imode(50),ifdfm(50),ihead(20)
```

```
      character * 80 ihead,ifdfm
      character * 8 itype,imode,iaper,iobs
c***********************************************************************
      common /rsave1/ xpsv(200000),ypsv(200000),dxdzsv(200000)
     * ,dydzsv(200000),entx(200000),enty(200000),wtsv(15,200000)
     * ,zshift,nsv
c***********************************************************************
      common /worksp/ bigmat(600000)
c***********************************************************************
      dimension work(200000)
      equivalence (work,bigmat)
c***********************************************************************
      do 100 i=1,nsv
  100 work(i)=wtsv(iener,i)
c  weighted planar focus here
      call pfocus(xpsv,ypsv,dxdzsv,dydzsv,work,nsv,xav,yav,delz)
      zshift=zshift+delz
c  report the results
      write(6,201) iener,energy(iener),nsv,delz,zshift,xav,yav
  201 format(/'  weighted planar focus: energy(',i2,')= ',e24.16/
     * ' number of rays= ',i7//
     * '      ***  stored rays modified    ***  '//
     * ' delta z  = ',e24.16,',    net zshift= ',e24.16/
     * ' new x average= ',e24.16,',    new y average= ',e24.16/)
      return
      end
      subroutine grid1(nlong,naz,rmin,rmax)
c
c    this is a routine to trace
c    rays on a grid with constant radial and
c    varying azimuthal increments on the first
c    surface between radii rmin and rmax
c
c    compare with focus routine in vetasag.f
c
c    ray weights are set to 1
c
c    nlong rays along the radius with rays butted up against
c       rmin and rmax
c    r/rmax*naz rays around the annulus.
c
      implicit double precision (a-h,o-z)
c***********************************************************************          2470
      common /sysc1/ zrange,elev,azim,foclen,source(3)
     * ,radlim(2,50),dxcirc(50),dycirc(50)
     * ,xwidth(50),ywidth(50),dxrect(50),dyrect(50),threct(50)
     * ,zlim(2,50),adata(25,50)
     * ,tilt(3,50),rmat(3,3,50)
     * ,disp(3,50),thick(50),findex(50)
     * ,sdata(25,50),delta
     * ,sp(3,50),ra(3,50),spi(3),rai(3)
     * ,energy(15),delbet(2,15,50),wgt(15,50),wgtnet(15),effa(15)
     * ,pi
     * ,imove(50),irstr(50),iwgt(50),nsurf
     * ,nnrg,kmax,kprint(51),ichief,itilt(50)
     * ,npass,nvig,nerr
     * ,iaper(50),iobs(50),itype(50),imode(50),ifdfm(50),ihead(20)
      character * 80 ihead,ifdfm
```

```
      character * 8 itype,imode,iaper,iobs
c***********************************************************************
      dimension work(3),tsp(3),tra(3),trmat(3,3),tdisp(3)
c
c  initialize ray counts for ssrt and wraysv
c
      call ssrti
      call wsvi
c
      if(nlong.gt.1) delr=(rmax-rmin)/dble(nlong-1)
c
      do 602 i=1,nnrg
  602 wgtnet(i)=1.d0
c
      do 321 m=1,nlong
c
      rr=rmin+dble(m-1)*delr
      nn=rr/rmax*dble(naz)+0.5d0
c quit if there are no points
      if(nn.le.0) go to 321
c
      dela=2.d0*pi/dble(nn)
c
      do 322 i=1,nn
c
      theta=dble(i-1)*dela
c
      tsp(1)=rr*dcos(theta)
      tsp(2)=rr*dsin(theta)
      tsp(3)=0.d0
c
c  transform to source coordinate system
      if(imove(1).ne.0) then
      do 302 j=1,3
      rai(j)=0.d0
      tdisp(j)=disp(j,1)
      do 302 k=1,3
  302 trmat(k,j)=rmat(k,j,1)
      call cnvout(tsp,rai,tsp,tra,trmat,tdisp)
      endif
c
c  set up direction
      sum=0.d0
      do 300 j=1,3
      work(j)=tsp(j)-source(j)
  300 sum=sum+work(j)*work(j)
      sum=dsqrt(sum)
      do 301 j=1,3
      spi(j)=tsp(j)
  301 rai(j)=work(j)/sum
c
c  always put ray in +z direction
      if(rai(3).lt.0.d0) then
      do 303 j=1,3
  303 rai(j)=-rai(j)
      endif
c
      ktr=2
```

```
      do 600 j=1,nsurf
      call ssrt(j,irstat)
      if(kprint(1).ne.0) then
      call rprint(j,irstat,ktr)
      endif
      if(irstat.ne.0) go to 601
  600 continue
c
c  save ray information from last surface
c
      call wraysv(ifill)
c
  601 continue
c
  322 continue
c
  321 continue
c
      write (6,350) npass,nlong,naz,rmin,rmax,elev,azim
  350 format('1',i7,' successful rays in grid1'/
     * '    :modified spoke wheel ray distribution on first surface,'/
     * '     varying azimuthal angle increments,'/
     * '     and constant radial increment, weights set to 1'/
     * ' ',i7,' radial points,    ',i7,' azimuthal points'/
     * ' rmin= ',e24.16,',    rmax= ',e24.16/
     * ' field angle (radians)= ',e24.16/
     * ' azimuth (radians)    = ',e24.16)
      write (6,465) nvig,nerr
  465 format(/22x,i7,' rays were vignetted or '
     *'obscured'/22x,i7,' rays failed in ssrt'/)
      if(nerr.ne.0) then
      write(6,351)
  351 format(///'    ***   warning, ray error(s)    ***'///)
      endif
c                                                                      2698(
      return                                                           2699
      end
      subroutine grid2(nlong,naz,rmin,rmax)
c
c    this is a routine to trace
c    rays on a grid with constant radial and
c    azimuthal increments on the first
c    surface between radii rmin and rmax
c
c    ray weights are set to 1
c
c    nlong rays along the radius with rays butted up against
c    rmin and rmax
c    naz rays around the annulus
c    (rays do not represent equal area on the first
c     surface between rmin and rmax)
c                                                                      247(
      implicit double precision (a-h,o-z)
c*********************************************************************
      common /syscl/ zrange,elev,azim,foclen,source(3)
     * ,radlim(2,50),dxcirc(50),dycirc(50)
     * ,xwidth(50),ywidth(50),dxrect(50),dyrect(50),threct(50)
     * ,zlim(2,50),adata(25,50)
```

```
     * ,tilt(3,50),rmat(3,3,50)
     * ,disp(3,50),thick(50),findex(50)
     * ,sdata(25,50),delta
     * ,sp(3,50),ra(3,50),spi(3),rai(3)
     * ,energy(15),delbet(2,15,50),wgt(15,50),wgtnet(15),effa(15)
     * ,pi
     * ,imove(50),irstr(50),iwgt(50),nsurf
     * ,nnrg,kmax,kprint(51),ichief,itilt(50)
     * ,npass,nvig,nerr
     * ,iaper(50),iobs(50),itype(50),imode(50),ifdfm(50),ihead(20)
      character * 80 ihead,ifdfm
      character * 8 itype,imode,iaper,iobs
c*********************************************************************
      dimension work(3),tsp(3),tra(3),trmat(3,3),tdisp(3)
c
c  initialize ray counts for ssrt and wraysv
c
      call ssrti
      call wsvi
c
      if(nlong.gt.1) delr=(rmax-rmin)/dble(nlong-1)
      dela=2.d0*pi/dble(naz)
c
      do 602 i=1,nnrg
  602 wgtnet(i)=1.d0
c
      do 321 m=1,nlong
c
      rr=rmin+dble(m-1)*delr
c
      if(rr.eq.0.d0) then
      nn=1
      else
      nn=naz
      endif
c
      do 322 i=1,nn
c
      theta=dble(i-1)*dela
c
      tsp(1)=rr*dcos(theta)
      tsp(2)=rr*dsin(theta)
      tsp(3)=0.d0
c
c  transform to source coordinate system
      if(imove(1).ne.0) then
      do 302 j=1,3
      rai(j)=0.d0
      tdisp(j)=disp(j,1)
      do 302 k=1,3
  302 trmat(k,j)=rmat(k,j,1)
      call cnvout(tsp,rai,tsp,tra,trmat,tdisp)
      endif
c
c  set up direction
      sum=0.d0
      do 300 j=1,3
      work(j)=tsp(j)-source(j)
```

```
  300 sum=sum+work(j)*work(j)
      sum=dsqrt(sum)
      do 301 j=1,3
      spi(j)=tsp(j)
  301 rai(j)=work(j)/sum
c
c  always put ray in +z direction
      if(rai(3).lt.0.d0) then
      do 303 j=1,3
  303 rai(j)=-rai(j)
      endif
c
      ktr=2
      do 600 j=1,nsurf
      call ssrt(j,irstat)
      if(kprint(1).ne.0) then
      call rprint(j,irstat,ktr)
      endif
      if(irstat.ne.0) go to 601
  600 continue
c
c  save ray information from last surface
c
      call wraysv(ifill)
c
  601 continue
c
  322 continue
c
  321 continue
c
      write (6,350) npass,nlong,naz,rmin,rmax,elev,azim
 350  format('1',i7,' successful rays in grid2'/
     * '    :spoke wheel ray distribution on first surface'/
     * '     annulus, constant azimuthal angle increment,'/
     * '     and constant radial increment, weights set to 1'/
     * ' ',i7,' radial points,    ',i7,' azimuthal points'/
     * ' rmin= ',e24.16,',    rmax= ',e24.16/
     * ' field angle (radians)= ',e24.16/
     * ' azimuth (radians)    = ',e24.16)
      write (6,465) nvig,nerr
 465  format(/22x,i7,' rays were vignetted or '
     *'obscured'/22x,i7,' rays failed in ssrt'/)
      if(nerr.ne.0) then
      write(6,351)
  351 format(///'    ***   warning, ray error(s)   ***'///)
      endif
c
      return
      end
      subroutine print01
c***********************************************************************
c  print out the syscl common system values
c***********************************************************************
      implicit double precision (a-h,o-z)
c*******************************************************************
      common /syscl/ zrange,elev,azim,foclen,source(3)
     * ,radlim(2,50),dxcirc(50),dycirc(50)
```

```
     * ,xwidth(50),ywidth(50),dxrect(50),dyrect(50),threct(50)
     * ,zlim(2,50),adata(25,50)
     * ,tilt(3,50),rmat(3,3,50)
     * ,disp(3,50),thick(50),findex(50)
     * ,sdata(25,50),delta
     * ,sp(3,50),ra(3,50),spi(3),rai(3)
     * ,energy(15),delbet(2,15,50),wgt(15,50),wgtnet(15),effa(15)
     * ,pi
     * ,imove(50),irstr(50),iwgt(50),nsurf
     * ,nnrg,kmax,kprint(51),ichief,itilt(50)
     * ,npass,nvig,nerr
     * ,iaper(50),iobs(50),itype(50),imode(50),ifdfm(50),ihead(20)
      character * 80 ihead,ifdfm
      character * 8 itype,imode,iaper,iobs
c*****************************************************************
      namelist /s1/ ihead,zrange,elev,azim,foclen,source,nsurf,ichief
     * ,itype,imode,sdata,delta,thick,findex,imove,irstr,itilt,tilt
     * ,rmat,disp,iaper,iobs,radlim,zlim,dxcirc,dycirc,xwidth
     * ,ywidth,dxrect,dyrect,threct,adata,iwgt,nnrg,energy,delbet,kprint
     * ,pi,ifdfm
      write(6,100)
  100 format('1print01 syscl common system values')
      write(6,s1)
      return
      end
      subroutine readin(iu,jpresc,istat)
c*****************************************************************
c
c  read in data to common area from file in jpresc using unit iu
c
c*****************************************************************
      implicit double precision (a-h,o-z)
      character * 80 jpresc
c*****************************************************************
      common /syscl/ zrange,elev,azim,foclen,source(3)
     * ,radlim(2,50),dxcirc(50),dycirc(50)
     * ,xwidth(50),ywidth(50),dxrect(50),dyrect(50),threct(50)
     * ,zlim(2,50),adata(25,50)
     * ,tilt(3,50),rmat(3,3,50)
     * ,disp(3,50),thick(50),findex(50)
     * ,sdata(25,50),delta
     * ,sp(3,50),ra(3,50),spi(3),rai(3)
     * ,energy(15),delbet(2,15,50),wgt(15,50),wgtnet(15),effa(15)
     * ,pi
     * ,imove(50),irstr(50),iwgt(50),nsurf
     * ,nnrg,kmax,kprint(51),ichief,itilt(50)
     * ,npass,nvig,nerr
     * ,iaper(50),iobs(50),itype(50),imode(50),ifdfm(50),ihead(20)
      character * 80 ihead,ifdfm
      character * 8 itype,imode,iaper,iobs
c*****************************************************************
c input namelist
      namelist /inp/ zrange,elev,azim,foclen,source
     * ,radlim,dxcirc,dycirc
     * ,xwidth,ywidth,dxrect,dyrect,threct
     * ,zlim,adata
     * ,tilt,rmat
     * ,disp,thick,findex
```

```
     * ,sdata,delta
     * ,energy,delbet,effa
     * ,imove,irstr,iwgt,nsurf
     * ,nnrg,kmax,kprint,ichief,itilt
     * ,npass,nvig,nerr
     * ,iaper,iobs,itype,imode,ifdfm,ihead
c*********************************************************************
      if(istat.ne.1.and.istat.ne.0) go to 3000
c open the file if istat.eq.1
      if(istat.eq.1) open(iu,file=jpresc)
      istat=0
c*********************************************************************
c  zero the common area ( this is dependent on array dimensions)
      call czero
c  set pi
      pi=datan(1.d0)*4.d0
c********************************************************************
c read file in jpresc using unit iu
      read(iu,inp,end=2000,err=3000)
c*********************************************************************
      return
c*********************************************************************
c end of file
 2000 continue
      close(iu)
      istat=2
      return
c*********************************************************************
c readin error
 3000 continue
      istat=-1
      return
      end
      subroutine rdout(iu,istat)
c*********************************************************************
c
c  write out syscl common area to unit iu.
c  error flag is istat ne 0
c
c*********************************************************************
      implicit double precision (a-h,o-z)
c**********************************************************************
      common /syscl/ zrange,elev,azim,foclen,source(3)
     * ,radlim(2,50),dxcirc(50),dycirc(50)
     * ,xwidth(50),ywidth(50),dxrect(50),dyrect(50),threct(50)
     * ,zlim(2,50),adata(25,50)
     * ,tilt(3,50),rmat(3,3,50)
     * ,disp(3,50),thick(50),findex(50)
     * ,sdata(25,50),delta
     * ,sp(3,50),ra(3,50),spi(3),rai(3)
     * ,energy(15),delbet(2,15,50),wgt(15,50),wgtnet(15),effa(15)
     * ,pi
     * ,imove(50),irstr(50),iwgt(50),nsurf
     * ,nnrg,kmax,kprint(51),ichief,itilt(50)
     * ,npass,nvig,nerr
     * ,iaper(50),iobs(50),itype(50),imode(50),ifdfm(50),ihead(20)
      character * 80 ihead,ifdfm
      character * 8 itype,imode,iaper,iobs
```

```
c*******************************************************************
c input namelist
      namelist /inp/ zrange,elev,azim,foclen,source
     * ,radlim,dxcirc,dycirc
     * ,xwidth,ywidth,dxrect,dyrect,threct
     * ,zlim,adata
     * ,tilt,rmat
     * ,disp,thick,findex
     * ,sdata,delta
     * ,energy,delbet,effa
     * ,imove,irstr,iwgt,nsurf
     * ,nnrg,kmax,kprint,ichief,itilt
     * ,npass,nvig,nerr
c*******************************************************************
      istat=0
      write(iu,inp,err=3000)
c*******************************************************************
c
c  character arrays must be dealt with separately
c  due to bug in sun4 fortran compiler
c
      backspace iu
      write(iu,101,err=3000) (" '"//iaper(i)//"'",i=1,nsurf)
  101 format(' iaper=',5(a))
      write(iu,102,err=3000) (" '"//iobs(i)//"'",i=1,nsurf)
  102 format(' iobs=',5(a))
      write(iu,103,err=3000) (" '"//itype(i)//"'",i=1,nsurf)
  103 format(' itype=',5(a))
      write(iu,104,err=3000) (" '"//imode(i)//"'",i=1,nsurf)
  104 format(' imode=',5(a))
      write(iu,105,err=3000) (" '"//ifdfm(i)//"'",i=1,50)
  105 format(' ifdfm=',(a))
      write(iu,106,err=3000) (" '"//ihead(i)//"'",i=1,20)
  106 format(' ihead=',(a))
      write(iu,9999,err=3000)
 9999 format(' &end')
c
c*******************************************************************
      return
c*******************************************************************
c rdout error
 3000 continue
      istat=-1
      return
      end
      subroutine rprint(lsurf,irstat,ktr)
c
c  print out ray surface information according to kprint array
c  initialize ktr to 2 before each ray is traced
c  print no rays if kprint(1)=0
c  print all rays if kprint(1)=1
c  print selected rays and failed rays if kprint(1)=2
c     selected rays must be listed in increasing order starting
c     in kprint(2) up to kprint(51)
c  otherwise print only failed rays
c
      implicit double precision (a-h,o-z)
c*******************************************************************
```

```
      common /syscl/ zrange,elev,azim,foclen,source(3)
     * ,radlim(2,50),dxcirc(50),dycirc(50)
     * ,xwidth(50),ywidth(50),dxrect(50),dyrect(50),threct(50)
     * ,zlim(2,50),adata(25,50)
     * ,tilt(3,50),rmat(3,3,50)
     * ,disp(3,50),thick(50),findex(50)
     * ,sdata(25,50),delta
     * ,sp(3,50),ra(3,50),spi(3),rai(3)
     * ,energy(15),delbet(2,15,50),wgt(15,50),wgtnet(15),effa(15)
     * ,pi
     * ,imove(50),irstr(50),iwgt(50),nsurf
     * ,nnrg,kmax,kprint(51),ichief,itilt(50)
     * ,npass,nvig,nerr
     * ,iaper(50),iobs(50),itype(50),imode(50),ifdfm(50),ihead(20)
      character * 80 ihead,ifdfm
      character * 8 itype,imode,iaper,iobs
c***********************************************************************
c
      if(irstat.eq.0) then
c
      if(kprint(1).eq.1) then
      write(6,910)lsurf,(sp(i,lsurf),i=1,3),(ra(i,lsurf),i=1,3)
c
      if(iwgt(lsurf).ne.0) then
      do 100 i=1,nnrg
      write(6,101) i,wgt(i,lsurf),wgtnet(i)
  100 continue
      endif
c
      else if(kprint(1).eq.2) then
      if(lsurf.ne.kprint(ktr)) go to 1000
      write(6,910)lsurf,(sp(i,lsurf),i=1,3),(ra(i,lsurf),i=1,3)
c
      if(iwgt(lsurf).ne.0) then
      do 102 i=1,nnrg
      write(6,101) i,wgt(i,lsurf),wgtnet(i)
  102 continue
      endif
c
      ktr=ktr+1
      endif
c
      else if(irstat.gt.0) then
      if(kprint(1).ne.0) write(6,915)lsurf,irstat
c
      else if(irstat.lt.0) then
      if(kprint(1).ne.0) write(6,921)lsurf,irstat
c
      endif
c
 1000 continue
  101 format(' energy(',i2,'),   wgt=',e24.16,',    wgtnet=',e24.16)
  910 format('     ----    ',i6,2x,3e24.15/21x,3e24.15)
  915 format('     ----    ',i6,2x,'ray vignetted,   irstat=',i6)
  921 format('     ----    ',i6,2x,'ray error,   irstat=',i6)
      return
      end
      subroutine rstart(ierr)
```

```
c
c  set up rotation matricies for surfaces with imove eq 1
c  order of rotation is x,y,z ie 1,2,3 unless reset with nonzero itilt
c        (see code below for form of itilt)
c  rotations are right handed about each axis
c  rotation matrices are determined from tilt angles in tilt (radians)
c
      implicit double precision (a-h,o-z)
c*********************************************************************
      common /sysc1/ zrange,elev,azim,foclen,source(3)
     * ,radlim(2,50),dxcirc(50),dycirc(50)
     * ,xwidth(50),ywidth(50),dxrect(50),dyrect(50),threct(50)
     * ,zlim(2,50),adata(25,50)
     * ,tilt(3,50),rmat(3,3,50)
     * ,disp(3,50),thick(50),findex(50)
     * ,sdata(25,50),delta
     * ,sp(3,50),ra(3,50),spi(3),rai(3)
     * ,energy(15),delbet(2,15,50),wgt(15,50),wgtnet(15),effa(15)
     * ,pi
     * ,imove(50),irstr(50),iwgt(50),nsurf
     * ,nnrg,kmax,kprint(51),ichief,itilt(50)
     * ,npass,nvig,nerr
     * ,iaper(50),iobs(50),itype(50),imode(50),ifdfm(50),ihead(20)
      character * 80 ihead,ifdfm
      character * 8 itype,imode,iaper,iobs
c*********************************************************************
      dimension iseq(3),rtempa(3,3),rtempb(3,3),rtempc(3,3)
c*********************************************************************
      do 100 i=1,nsurf
c cut out if surface is not transformed or if rotation is
c not specified by tilt().
      if(imove(i).ne.1) go to 100
c  get rotation sequence
      if(itilt(i).eq.0) then
      iseq(1)=1
      iseq(2)=2
      iseq(3)=3
      else
      iseq(1)=mod(itilt(i),10)
      iseq(2)=mod(itilt(i),100)/10
      iseq(3)=mod(itilt(i),1000)/100
      endif
c  initialize matrix to unit
      do 200 j=1,3
      do 200 k=1,3
      if(j.eq.k) then
      rtempb(j,k)=1.d0
      else
      rtempb(j,k)=0.d0
      endif
  200 continue
c  determine the effect of each tilt
      do 300 j=1,3
      if(iseq(j).eq.0) go to 300
      if(iseq(j).eq.1) then
      c=dcos(tilt(1,i))
      s=dsin(tilt(1,i))
      rtempa(1,1)=1.d0
```

```
      rtempa(1,2)=0.d0
      rtempa(1,3)=0.d0
      rtempa(2,1)=0.d0
      rtempa(2,2)=c
      rtempa(2,3)=s
      rtempa(3,1)=0.d0
      rtempa(3,2)=-s
      rtempa(3,3)=c
      elseif(iseq(j).eq.2) then
      c=dcos(tilt(2,i))
      s=dsin(tilt(2,i))
      rtempa(1,1)=c
      rtempa(1,2)=0.d0
      rtempa(1,3)=-s
      rtempa(2,1)=0.d0
      rtempa(2,2)=1.d0
      rtempa(2,3)=0.d0
      rtempa(3,1)=s
      rtempa(3,2)=0.d0
      rtempa(3,3)=c
      elseif(iseq(j).eq.3) then
      c=dcos(tilt(3,i))
      s=dsin(tilt(3,i))
      rtempa(1,1)=c
      rtempa(1,2)=s
      rtempa(1,3)=0.d0
      rtempa(2,1)=-s
      rtempa(2,2)=c
      rtempa(2,3)=0.d0
      rtempa(3,1)=0.d0
      rtempa(3,2)=0.d0
      rtempa(3,3)=1.d0
      else
      go to 3000
      endif
c  accumulate net rotation matrix by matrix multiplication
      call matab(rtempa,rtempb,rtempb,3,3,3,rtempc)
  300 continue
c  put the result in rmat
      do 400 j=1,3
      do 400 k=1,3
  400 rmat(j,k,i)=rtempb(j,k)
c
  100 continue
c
      ierr=0
c
      return
c
c  error return
c
 3000 continue
      ierr=1
      return
      end
      subroutine setcom(jerr)
c**********************************************************************
c
```

```
c  set up common data after readin or modification of common
c  jerr is 0 normally or 1 for error
c
c*******************************************************************
      implicit double precision (a-h,o-z)
c*******************************************************************
      common /syscl/ zrange,elev,azim,foclen,source(3)
     * ,radlim(2,50),dxcirc(50),dycirc(50)
     * ,xwidth(50),ywidth(50),dxrect(50),dyrect(50),threct(50)
     * ,zlim(2,50),adata(25,50)
     * ,tilt(3,50),rmat(3,3,50)
     * ,disp(3,50),thick(50),findex(50)
     * ,sdata(25,50),delta
     * ,sp(3,50),ra(3,50),spi(3),rai(3)
     * ,energy(15),delbet(2,15,50),wgt(15,50),wgtnet(15),effa(15)
     * ,pi
     * ,imove(50),irstr(50),iwgt(50),nsurf
     * ,nnrg,kmax,kprint(51),ichief,itilt(50)
     * ,npass,nvig,nerr
     * ,iaper(50),iobs(50),itype(50),imode(50),ifdfm(50),ihead(20)
      character * 80 ihead,ifdfm
      character * 8 itype,imode,iaper,iobs
c*******************************************************************
      jerr=0
c*******************************************************************
c set source position relative to undisplaced center of first surface
c       set zrange positive for source in front of first surface.
c       set zrange large for infinite conjugate.
c       azim is azimuthal angle in radians of source ray.
c       azim is positive from x axis toward y axis.
c       elev is angle in radians of field angle.
      source(1)=-zrange*dtan(elev)*dcos(azim)
      source(2)=-zrange*dtan(elev)*dsin(azim)
      source(3)=-zrange
c*******************************************************************
c imove ne 0 for surface means surface coordinate transformation.
c set rotation matricies from tilts for surfaces with imove eq 1.
      call rstart(ierr)
      if(ierr.ne.0) go to 3000
c*******************************************************************
c
c  check for deformed surface file input
c
      iurdfm=7
      call rdfm(iurdfm)
c
c*******************************************************************
      return
c*******************************************************************
c readin error
 3000 continue
      jerr=1
      return
      end
      subroutine spdiag(xcen,ycen,npoint)
c*******************************************************************
c
c  make up line printer spot diagram from storage array
```

```
c  use first npoint rays
c
c*********************************************************************
      implicit double precision (a-h,o-z)
c*********************************************************************
      common /syscl/ zrange,elev,azim,foclen,source(3)
     * ,radlim(2,50),dxcirc(50),dycirc(50)
     * ,xwidth(50),ywidth(50),dxrect(50),dyrect(50),threct(50)
     * ,zlim(2,50),adata(25,50)
     * ,tilt(3,50),rmat(3,3,50)
     * ,disp(3,50),thick(50),findex(50)
     * ,sdata(25,50),delta
     * ,sp(3,50),ra(3,50),spi(3),rai(3)
     * ,energy(15),delbet(2,15,50),wgt(15,50),wgtnet(15),effa(15)
     * ,pi
     * ,imove(50),irstr(50),iwgt(50),nsurf
     * ,nnrg,kmax,kprint(51),ichief,itilt(50)
     * ,npass,nvig,nerr
     * ,iaper(50),iobs(50),itype(50),imode(50),ifdfm(50),ihead(20)
      character * 80 ihead,ifdfm
      character * 8 itype,imode,iaper,iobs
c*********************************************************************
      common /rsavel/ xpsv(200000),ypsv(200000),dxdzsv(200000)
     * ,dydzsv(200000),entx(200000),enty(200000),wtsv(15,200000)
     * ,zshift,nsv
c*********************************************************************
      common /worksp/ bigmat(600000)
c*********************************************************************
      dimension workx(200000),worky(200000)
      equivalence (workx(1),bigmat(1)),(worky(1),bigmat(200001))
c*********************************************************************
      nn=nsv
      if(npoint.gt.0) then
      if(nsv.gt.npoint) nn=npoint
      endif
      do 100 i=1,nn
      workx(i)=xpsv(i)-xcen
      worky(i)=ypsv(i)-ycen
  100 continue
      write(6,201) nn,nsv,xcen,ycen
  201 format('1 spot diagram: first ',i7,' rays of',i7,' stored'/
     * ' assumed center:  x = ',e24.16,',    y = ',e24.16)
      call splot(nn,workx,worky)
      return
      end                                                          60370C
      subroutine ssrt(is,irstat)
c                                                                  60510C
c single surface ray trace to surface is
c
c  irstat=0 for succesful ray
c  irstat=1 for vignetted ray
c  irstat=-1 for ray error
c                                                                  60470C
      implicit double precision (a-h,o-z)                          60540C
c
c*********************************************************************
      common /syscl/ zrange,elev,azim,foclen,source(3)
     * ,radlim(2,50),dxcirc(50),dycirc(50)
```

```
     * ,xwidth(50),ywidth(50),dxrect(50),dyrect(50),threct(50)
     * ,zlim(2,50),adata(25,50)
     * ,tilt(3,50),rmat(3,3,50)
     * ,disp(3,50),thick(50),findex(50)
     * ,sdata(25,50),delta
     * ,sp(3,50),ra(3,50),spi(3),rai(3)
     * ,energy(15),delbet(2,15,50),wgt(15,50),wgtnet(15),effa(15)
     * ,pi
     * ,imove(50),irstr(50),iwgt(50),nsurf
     * ,nnrg,kmax,kprint(51),ichief,itilt(50)
     * ,npass,nvig,nerr
     * ,iaper(50),iobs(50),itype(50),imode(50),ifdfm(50),ihead(20)
      character * 80 ihead,ifdfm
      character * 8 itype,imode,iaper,iobs
c**********************************************************************
      irstat=0
c**********************************************************************
c  displace to z=0 at surface is, in is-1 coordinates
c
      if(is.gt.1) then
      sp(1,is)=sp(1,is-1)+ra(1,is-1)*(thick(is-1)-sp(3,is-1))/ra(3,is-1)
      sp(2,is)=sp(2,is-1)+ra(2,is-1)*(thick(is-1)-sp(3,is-1))/ra(3,is-1)
      sp(3,is)=0.d0
      do 601 i=1,3
  601 ra(i,is)=ra(i,is-1)
      else if(is.eq.1) then
      sp(1,is)=spi(1)-spi(3)*rai(1)/rai(3)
      sp(2,is)=spi(2)-spi(3)*rai(2)/rai(3)
      sp(3,is)=0.d0
      do 602 i=1,3
  602 ra(i,is)=rai(i)
      endif
c
c*********************************************************************
c  tilt and displace, then reset to z=0 in is coordinates
c
      if(imove(is).ne.0) then                                           61160000
c
      call trfin(is)
c
      sp(1,is)=sp(1,is)-ra(1,is)*sp(3,is)/ra(3,is)
      sp(2,is)=sp(2,is)-ra(2,is)*sp(3,is)/ra(3,is)
      sp(3,is)=0.d0
c
      endif
c**********************************************************************
c  trace ray                                                            61530000
c                                                                       61540000
c  grazing incidence conic or reflecting flat
c
      if(itype(is).eq.'grzcon01') go to 201
      if(itype(is).eq.'grzcon02') go to 201
      if(itype(is).eq.'grzcon03') go to 201
      if(imode(is).eq.'refl'.and.itype(is).eq.
     * 'flat') go to 201
      if(itype(is).eq.'grzcon11') go to 203
      if(itype(is).eq.'grzcon12') go to 203
      if(itype(is).eq.'grzcon13') go to 203
```

```
c
      if(itype(is).ne.'flat') go to 500
      if(imode(is).ne.'thru') go to 500
c
      go to 202
  201 continue
      call strace(isterr,is)
      if(isterr.ne.0) go to 500
      go to 202
203   continue
      call strc02(isterr,is)
      if(isterr.ne.0) go to 500
  202 continue
c*********************************************************************
c  check for vignetting in surface frame
c  skip this if ichief is 1
      if(ichief.ne.1) then
      ivig=0
      if(iobs(is).ne.' '.or.iaper(is).ne.' ')
     * call vignet(ivig,is)
      if(ivig.ne.0) go to 550
      endif
c*********************************************************************
      if(is.eq.nsurf) npass=npass+1
c*********************************************************************
c  restore coordinates if necessary and if desired
c
      if(imove(is).ne.0.and.irstr(is).ne.0) then
      call trfout(is)
      endif
c*********************************************************************
      return
c*********************************************************************
c  trace error
  500 continue
      irstat=-1
      nerr=nerr+1
      return
c*********************************************************************
c  ray vignetted
 550  continue
      irstat=1
      nvig=nvig+1
      return
c*********************************************************************
c  ray count initialization
      entry ssrti
      npass=0
      nvig=0
      nerr=0
      return
      end
      subroutine strace(isterr,is)
      implicit double precision (a-h,o-z)
c*********************************************************************
      common /sysc1/ zrange,elev,azim,foclen,source(3)
     * ,radlim(2,50),dxcirc(50),dycirc(50)
     * ,xwidth(50),ywidth(50),dxrect(50),dyrect(50),threct(50)
```

```
     * ,zlim(2,50),adata(25,50)
     * ,tilt(3,50),rmat(3,3,50)
     * ,disp(3,50),thick(50),findex(50)
     * ,sdata(25,50),delta
     * ,sp(3,50),ra(3,50),spi(3),rai(3)
     * ,energy(15),delbet(2,15,50),wgt(15,50),wgtnet(15),effa(15)
     * ,pi
     * ,imove(50),irstr(50),iwgt(50),nsurf
     * ,nnrg,kmax,kprint(51),ichief,itilt(50)
     * ,npass,nvig,nerr
     * ,iaper(50),iobs(50),itype(50),imode(50),ifdfm(50),ihead(20)
      character * 80 ihead,ifdfm
      character * 8 itype,imode,iaper,iobs
c*********************************************************************
c  common for communication with user trace
      common/userdt/x,y,z,f,fx,fy,fz,isurf,ifcalc,isferr
c*********************************************************************
c  trace to surface 'is'  (this version is for reflection or
c                          dummy surfaces only)
c
c  input is:  starting ray position sp(,is)
c             starting direction cosines ra(,is)
c
c  output is: new ray position sp(,is)
c             new direction cosines ra(,is)
c             isterr ne 0 for ray error
c
      isterr=0
c
c*********************************************************************
c
c  find intercept
c
c  set surface number for user trace
      isurf=is
c  initialize ray position and direction
      x=sp(1,is)
      y=sp(2,is)
      z=sp(3,is)
      dxds=ra(1,is)
      dyds=ra(2,is)
      dzds=ra(3,is)
c*********************************************************************
c
c  direct calculation here
      if(itype(is).eq.'grzcon01'.or.itype(is).eq.'grzcon03') then
      call utraci
      if(isferr.eq.1) go to 3000
      endif
c
c*********************************************************************
c
c  iteration here
c
      if(itype(is).eq.'grzcon02'.or.itype(is).eq.'grzcon03') then
c  initialize iteration count
      kount=0
c debug print
```

```
c     write(6,*)is,x,y,z,dxds,dyds,dzds,delta,kmax
c
c  require function value calculation in user trace
      ifcalc=1
c  iteration loop for intercept
  100 continue
      kount=kount+1
      if(kount.gt.kmax) go to 3000
      call utrace
      if(isferr.eq.1) go to 3000
      ds=-f/(fx*dxds+fy*dyds+fz*dzds)
      x=x+dxds*ds
      y=y+dyds*ds
      z=z+dzds*ds
c debug print
c     write(6,*)kount,ds,x,y,z,f,fx,fy,fz
c
      if (dabs(ds).le.delta) go to 400
      go to 100
  400 continue
c
      endif
c*********************************************************************
c  calculation for outgoing ray
c (currently covers reflection and thru  surfaces only)
c
      if(imode(is).eq.'refl') then
c dont need function value here
      ifcalc=0
      call utrace
      if(isferr.eq.1) go to 3000
      c=(dxds*fx+dyds*fy+dzds*fz)/(fx**2+fy**2+fz**2)
      ra(1,is)=dxds-2.d0*c*fx
      ra(2,is)=dyds-2.d0*c*fy
      ra(3,is)=dzds-2.d0*c*fz
c debug print
c     write(6,*)x,y,z,fx,fy,fz
c
      elseif(imode(is).eq.'thru') then
      else
c invalid surface
      go to 3000
      endif
c
      sp(1,is)=x
      sp(2,is)=y
      sp(3,is)=z
c
      return
 3000 continue
c  error return
      write(6,3001)
 3001 format('    ***    strace error, isterr set to 1    *** ')
      isterr=1
      return
      end
      subroutine trfin(is)
c
```

```
c  transform into or out of local coordinates at surface is.
c  sp(,is),ra(,is) are input.
c  sp(,is),ra(,is) are replaced
c
c trfin transforms into local coordinates
c trfout transforms out of local coordinates
c
      implicit double precision(a-h,o-z)
c**********************************************************************
      common /syscl/ zrange,elev,azim,foclen,source(3)
     * ,radlim(2,50),dxcirc(50),dycirc(50)
     * ,xwidth(50),ywidth(50),dxrect(50),dyrect(50),threct(50)
     * ,zlim(2,50),adata(25,50)
     * ,tilt(3,50),rmat(3,3,50)
     * ,disp(3,50),thick(50),findex(50)
     * ,sdata(25,50),delta
     * ,sp(3,50),ra(3,50),spi(3),rai(3)
     * ,energy(15),delbet(2,15,50),wgt(15,50),wgtnet(15),effa(15)
     * ,pi
     * ,imove(50),irstr(50),iwgt(50),nsurf
     * ,nnrg,kmax,kprint(51),ichief,itilt(50)
     * ,npass,nvig,nerr
     * ,iaper(50),iobs(50),itype(50),imode(50),ifdfm(50),ihead(20)
      character * 80 ihead,ifdfm
      character * 8 itype,imode,iaper,iobs
c**********************************************************************
      dimension tsp(3),tra(3)
      do 100 i=1,3
      tsp(i)=sp(i,is)
  100 tra(i)=ra(i,is)
      do 101 i=1,3
      sp(i,is)=0.d0
      ra(i,is)=0.d0
      do 101 j=1,3
      sp(i,is)=sp(i,is)+rmat(i,j,is)*(tsp(j)-disp(j,is))
  101 ra(i,is)=ra(i,is)+rmat(i,j,is)*tra(j)
      return
      entry trfout(is)
      do 200 i=1,3
      tsp(i)=sp(i,is)
  200 tra(i)=ra(i,is)
      do 201 i=1,3
      sp(i,is)=disp(i,is)
      ra(i,is)=0.d0
      do 201 j=1,3
      sp(i,is)=sp(i,is)+rmat(j,i,is)*tsp(j)
  201 ra(i,is)=ra(i,is)+rmat(j,i,is)*tra(j)
      return
      end
      subroutine utrace
c
c  calculate function f and gradient fx,fy,fz for surface
c
c  input
c        x,y,z          position
c        isurf or n     surface number
c        itype(n)       surface type
c        sdata(...,n)   surface parameters
```

```
c        ifcalc          calculate function value if ifcalc=1               3399
c  output                                                                   3399
c        f               function value                                     3399
c        fx,fy,fz        gradient of function                               3399
c        isferr          non zero if error occurs                           3399
c                                                                           3399
      implicit double precision (a-h,o-z)                                   3399
c*******************************************************************
      common /syscl/ zrange,elev,azim,foclen,source(3)
     * ,radlim(2,50),dxcirc(50),dycirc(50)
     * ,xwidth(50),ywidth(50),dxrect(50),dyrect(50),threct(50)
     * ,zlim(2,50),adata(25,50)
     * ,tilt(3,50),rmat(3,3,50)
     * ,disp(3,50),thick(50),findex(50)
     * ,sdata(25,50),delta
     * ,sp(3,50),ra(3,50),spi(3),rai(3)
     * ,energy(15),delbet(2,15,50),wgt(15,50),wgtnet(15),effa(15)
     * ,pi
     * ,imove(50),irstr(50),iwgt(50),nsurf
     * ,nnrg,kmax,kprint(51),ichief,itilt(50)
     * ,npass,nvig,nerr
     * ,iaper(50),iobs(50),itype(50),imode(50),ifdfm(50),ihead(20)
      character * 80 ihead,ifdfm
      character * 8 itype,imode,iaper,iobs
c*******************************************************************
c  common for communication with user trace                                32901
      common/userdt/x,y,z,f,fx,fy,fz,isurf,ifcalc,isferr                   32902
c*******************************************************************
      equivalence (n,isurf)                                                34011
c                                                                          34011
      isferr=0                                                             34012
c                                                                          34012
      if(itype(n).eq.'grzcon01'.or.itype(n).eq.'grzcon02'
     * .or.itype(n).eq.'grzcon03') then
c*******************************************************************
c  grazing conic plus sag error:
c
c  rho at z=0                 sdata(1,n)
c
c  subnormal at z=0           sdata(2,n)
c
c  1-e**2                     sdata(3,n)
c
c  full mirror length         sdata(4,n)
c
c  zero-peak sag error        sdata(5,n)
c  (mirror ends fixed)
c
c  delta-r error              sdata(6,n)
c
      rhosq=sdata(1,n)**2+2.d0*sdata(2,n)*z-sdata(3,n)*z**2                340126
      if(rhosq.le.0.d0) go to 3000                                         340127
      rho=dsqrt(rhosq)                                                     340128
      rad=dsqrt(x**2+y**2)                                                 340129
      if(rad.le.0.d0) go to 3000                                           340130
      fx=x/rad                                                             340134
      fy=y/rad                                                             340135
      fz=  z*(sdata(3,n)/rho+8.d0*sdata(5,n)/sdata(4,n)**2)                340137
```

```
     * -sdata(2,n)/rho
      if(ifcalc.eq.1) then
      f=rad-(rho
     *   -4.d0*sdata(5,n)*((z/sdata(4,n))**2-1.d0/4.d0)+sdata(6,n))
      endif
c*********************************************************************
c
      elseif(itype(n).eq.'flat') then
c*********************************************************************
c  flat surface
      fx=0.d0
      fy=0.d0
      fz=1.d0
      if(ifcalc.eq.1) f=z
c*********************************************************************
c
      else
c  invalid surface
      go to 3000
      endif
      return
 3000 continue
c  computation error
      write(6,3001)
 3001 format('    ***    utrace error, isferr set to 1    ***')
      isferr=1
      return
c*********************************************************************
c
      entry utraci
      isferr=0
      if(itype(n).eq.'grzcon01'.or.itype(n).eq.'grzcon03') then
c
c  direct calculation of intercept of ray with concave grazing conic.
c  take error return for two solutions within element
c   (in case of a convex optic this could be changed to take the first
c    solution within the element)
c  take first solution if there is no solution within the element.
c  cut out to error return for no solution
c   (in case of no solution, if desired, one could artificially set the solution
c    outside of the element to be vignetted)
c
c  look for solutions
c
        a=ra(1,n)**2+ra(2,n)**2+sdata(3,n)*ra(3,n)**2
        b=2.d0*(sp(1,n)*ra(1,n)+sp(2,n)*ra(2,n)-sdata(2,n)*ra(3,n))
        c=sp(1,n)**2+sp(2,n)**2-sdata(1,n)**2
     isol=0
c
        if(a.eq.0.d0) then
           if(b.eq.0.d0) then
c no solution
           go to 3000
           endif
           sol=-c/b
        isol=1
        elseif(c.eq.0.d0) then
        if(b.eq.0.d0) then
```

```
          sol=0.d0
          isol=1
          else
          s1=0.d0
          s2=-b/a
          endif
    else
          if(b.eq.0.d0) then
          arg=-c/a
             if(arg.lt.0.d0) then
c no solution
          go to 3000
             endif
             s1=dsqrt(arg)
             s2=-s1
          else
             arg=b**2-4.d0*a*c
             if(arg.lt.0.d0) then
c no solution
          go to 3000
             endif
             arg=dsqrt(arg)
             denom1=b+arg
             denom2=b-arg
          if(denom1.ne.0.d0) then
             if(denom2.ne.0.d0) then
                   s1=-2.d0*c/denom1
                s2=-2.d0*c/denom2
                else
                sol=-2.d0*c/denom1
                   isol=1
                endif
             elseif(denom2.ne.0.d0) then
                sol=-2.d0*c/denom2
                isol=1
             else
c no solution
                go to 3000
             endif
          endif
       endif
c
c make selection of solution if it is not unique.
c
       if(isol.eq.0) then
          test=sdata(4,n)/2.d0
          z1=dabs(s1*ra(3,n))
          z2=dabs(s2*ra(3,n))
          if(z1.lt.test) then
          if(z2.lt.test) then
c               two solutions within element
c               (there could actually be two solutions within
c                the element or this could also be caused
c                by a numerical problem above)
          go to 3000
             else
          sol=s1
             endif
```

```
          elseif(z2.lt.test) then
          sol=s2
          else
          sol=s1
          if(s2.lt.s1) sol=s2
          endif
     endif
c
c move to the solution point
c
        x=sp(1,n)+ra(1,n)*sol
        y=sp(2,n)+ra(2,n)*sol
        z=ra(3,n)*sol
c
      else
c invalid surface
        go to 3000
      endif
c*******************************************************************
      return
      end                                                             34050000
      subroutine vignet(ivig,is)
c
c  check for vignetting at surface is
c  ivig ne 0 if ray is vignetted
c
      implicit double precision (a-h,o-z)
c*******************************************************************
      common /syscl/ zrange,elev,azim,foclen,source(3)
     * ,radlim(2,50),dxcirc(50),dycirc(50)
     * ,xwidth(50),ywidth(50),dxrect(50),dyrect(50),threct(50)
     * ,zlim(2,50),adata(25,50)
     * ,tilt(3,50),rmat(3,3,50)
     * ,disp(3,50),thick(50),findex(50)
     * ,sdata(25,50),delta
     * ,sp(3,50),ra(3,50),spi(3),rai(3)
     * ,energy(15),delbet(2,15,50),wgt(15,50),wgtnet(15),effa(15)
     * ,pi
     * ,imove(50),irstr(50),iwgt(50),nsurf
     * ,nnrg,kmax,kprint(51),ichief,itilt(50)
     * ,npass,nvig,nerr
     * ,iaper(50),iobs(50),itype(50),imode(50),ifdfm(50),ihead(20)
      character * 80 ihead,ifdfm
      character * 8 itype,imode,iaper,iobs
c*******************************************************************
      if(iobs(is).eq.'circ') then
      rad=dsqrt((sp(1,is)-dxcirc(is))**2+(sp(2,is)-dycirc(is))**2)
      if(rad.gt.radlim(1,is).and.rad.lt.radlim(2,is)) go to 550
      else if(iobs(is).eq.'rect') then
      xxp=sp(1,is)-dxrect(is)
      yyp=sp(2,is)-dyrect(is)
      call rotate(xxp,yyp,-threct(is),xx,yy)
      if(dabs(xx).lt.xwidth(is)/2.d0.and.dabs(yy).lt.ywidth(is)/2.d0)
     * go to 550
      else if(iobs(is).eq.'zlim') then
      if(sp(3,is).gt.zlim(1,is).and.sp(3,is).lt.zlim(2,is)) go to 550
      endif
c
```

```
      if(iaper(is).eq.'circ') then
      rad=dsqrt((sp(1,is)-dxcirc(is))**2+(sp(2,is)-dycirc(is))**2)
      if(rad.lt.radlim(1,is).or.rad.gt.radlim(2,is)) go to 550
      else if(iaper(is).eq.'rect') then
      xxp=sp(1,is)-dxrect(is)
      yyp=sp(2,is)-dyrect(is)
      call rotate(xxp,yyp,-threct(is),xx,yy)
      if(dabs(xx).gt.xwidth(is)/2.d0.or.dabs(yy).gt.ywidth(is)/2.d0)
     * go to 550
      else if(iaper(is).eq.'zlim') then
      if(sp(3,is).lt.zlim(1,is).or.sp(3,is).gt.zlim(2,is)) go to 550
      endif
      ivig=0
      return
  550 continue
      ivig=1
      return
      end
      subroutine wray(efact,irstat)
c
c  trace ray and accumulate reflectivity weights and effective
c  area weight for ray
c
c  efact is initial effective area weight for ray
c
c  irstat is ne 0 for ray error
c
      implicit double precision (a-h,o-z)
c*********************************************************************
      common /syscl/ zrange,elev,azim,foclen,source(3)
     * ,radlim(2,50),dxcirc(50),dycirc(50)
     * ,xwidth(50),ywidth(50),dxrect(50),dyrect(50),threct(50)
     * ,zlim(2,50),adata(25,50)
     * ,tilt(3,50),rmat(3,3,50)
     * ,disp(3,50),thick(50),findex(50)
     * ,sdata(25,50),delta
     * ,sp(3,50),ra(3,50),spi(3),rai(3)
     * ,energy(15),delbet(2,15,50),wgt(15,50),wgtnet(15),effa(15)
     * ,pi
     * ,imove(50),irstr(50),iwgt(50),nsurf
     * ,nnrg,kmax,kprint(51),ichief,itilt(50)
     * ,npass,nvig,nerr
     * ,iaper(50),iobs(50),itype(50),imode(50),ifdfm(50),ihead(20)
      character * 80 ihead,ifdfm
      character * 8 itype,imode,iaper,iobs
c*********************************************************************
      dimension tsp(3),tra(3),trmat(3,3),tdisp(3)
c  initialize ray effective area weights
c  (account for initial direction of ray in local coordinates
c   of first surface. i.e. calculate effective area on first
c   surface.)
      if(imove(1).ne.0) then
      do 100 i=1,3
      tdisp(i)=disp(i,1)
      do 100 j=1,3
  100 trmat(i,j)=rmat(i,j,1)
      call cnvin(spi,rai,tsp,tra,trmat,tdisp)
      factor=dabs(tra(3))
```

```
      else
      factor=dabs(rai(3))
      endif
      do 101 j=1,nnrg
  101 wgtnet(j)=factor*efact
c
c*********************************************************************
c  initialize print surface counter
      ktr=2
c
      do 930   j  = 1,nsurf
c
      call ssrt(j,irstat)
c
c  if applicable calculate and accumulate reflectivity weights
c  and update ray effective area weight.
c
      if(iwgt(j).ne.0.and.irstat.eq.0) then
c
      call calwgt(j)
      endif
c
c  print ray?
      if(kprint(1).ne.0) then
c  print ray
      call rprint(j,irstat,ktr)
      endif
c
      if(irstat.ne.0) go to 1000
  930 continue
c*********************************************************************
 1000 continue
      return
      end
      subroutine wraysv(ifill)
c
c  save last surface ray information about ray npass
c
      implicit double precision (a-h,o-z)
c*********************************************************************
      common /sysc1/ zrange,elev,azim,foclen,source(3)
     * ,radlim(2,50),dxcirc(50),dycirc(50)
     * ,xwidth(50),ywidth(50),dxrect(50),dyrect(50),threct(50)
     * ,zlim(2,50),adata(25,50)
     * ,tilt(3,50),rmat(3,3,50)
     * ,disp(3,50),thick(50),findex(50)
     * ,sdata(25,50),delta
     * ,sp(3,50),ra(3,50),spi(3),rai(3)
     * ,energy(15),delbet(2,15,50),wgt(15,50),wgtnet(15),effa(15)
     * ,pi
     * ,imove(50),irstr(50),iwgt(50),nsurf
     * ,nnrg,kmax,kprint(51),ichief,itilt(50)
     * ,npass,nvig,nerr
     * ,iaper(50),iobs(50),itype(50),imode(50),ifdfm(50),ihead(20)
      character * 80 ihead,ifdfm
      character * 8 itype,imode,iaper,iobs
c*********************************************************************
      common /rsave1/ xpsv(200000),ypsv(200000),dxdzsv(200000)
```

```
     * ,dydzsv(200000),entx(200000),enty(200000),wtsv(15,200000)
     * ,zshift,nsv
c***********************************************************************
      dimension tra(3),tsp(3),trmat(3,3),tdisp(3)
      nsv=nsv+1
      if(nsv.le.200000) then
c
      if(nsv.eq.200000)  then
      ifill=1
      else
      ifill=0
      endif
c
c  assume flat nondisplaced nonrotated image plane
c  x value
      xpsv(nsv)=sp(1,nsurf)
c  y value
      ypsv(nsv)=sp(2,nsurf)
c  dxdz value
      dxdzsv(nsv)=ra(1,nsurf)/ra(3,nsurf)
c  dydz value
      dydzsv(nsv)=ra(2,nsurf)/ra(3,nsurf)
c  initial ray x and y values at local z=0 on surface 1
      if(imove(1).ne.0) then
      do 202 i=1,3
      tdisp(i)=disp(i,1)
      do 202 j=1,3
  202 trmat(i,j)=rmat(i,j,1)
      call cnvin(spi,rai,tsp,tra,trmat,tdisp)
      entx(nsv)=tsp(1)-tsp(3)*tra(1)/tra(3)
      enty(nsv)=tsp(2)-tsp(3)*tra(2)/tra(3)
      else
      entx(nsv)=spi(1)-spi(3)*rai(1)/rai(3)
      enty(nsv)=spi(2)-spi(3)*rai(2)/rai(3)
      endif
c
      do 100 i=1,nnrg
  100 wtsv(i,nsv)=wgtnet(i)
c
      else
      write(6,200) nsv
  200 format(///'    ***   overflow in wraysv,    nsv= ',i10)
      write(6,201)
  201 format('    ***   stop    ***'///)
      stop
      endif
c
      return
c
      entry wsvrst(factor)
c
c  reset effective area weights
c
      do 300 i=1,nsv
      do 300 j=1,nnrg
  300 wtsv(j,i)=wtsv(j,i)*factor
      return
c
```

```
c  initialize storage ray count, z shift, and effective
c  focal length
      entry wsvi
      nsv=0
      zshift=0.d0
      return
c
      end
      subroutine wrayso(fname)
c
c  output ray save data to ray file.
c
c  fname is the file prefix for the .gtray file.
c
c
      implicit double precision (a-h,o-z)
c*****************************************************************
      common /syscl/ zrange,elev,azim,foclen,source(3)
     * ,radlim(2,50),dxcirc(50),dycirc(50)
     * ,xwidth(50),ywidth(50),dxrect(50),dyrect(50),threct(50)
     * ,zlim(2,50),adata(25,50)
     * ,tilt(3,50),rmat(3,3,50)
     * ,disp(3,50),thick(50),findex(50)
     * ,sdata(25,50),delta
     * ,sp(3,50),ra(3,50),spi(3),rai(3)
     * ,energy(15),delbet(2,15,50),wgt(15,50),wgtnet(15),effa(15)
     * ,pi
     * ,imove(50),irstr(50),iwgt(50),nsurf
     * ,nnrg,kmax,kprint(51),ichief,itilt(50)
     * ,npass,nvig,nerr
     * ,iaper(50),iobs(50),itype(50),imode(50),ifdfm(50),ihead(20)
      character * 80 ihead,ifdfm
      character * 8 itype,imode,iaper,iobs
c*****************************************************************
      common /rsavel/ xpsv(200000),ypsv(200000),dxdzsv(200000)
     * ,dydzsv(200000),entx(200000),enty(200000),wtsv(15,200000)
     * ,zshift,nsv
c*****************************************************************
      character * 80 fname
c  open the ray file
      iuray=7
c     call fildf(iuray,fname,'gtray ','unformatted ')
      open(iuray,file=fname,form='unformatted')
c  write out the ray data
      nhead=20
      write(iuray)nsv,nnrg,zshift,foclen,nhead
      write(iuray)(xpsv(i),ypsv(i),dxdzsv(i),dydzsv(i),entx(i)
     * ,enty(i),(wtsv(j,i),j=1,nnrg),i=1,nsv),(energy(i),i=1,nnrg)
     * ,(ihead(i),i=1,nhead)
c
      return
c
      end
      subroutine wspot1(mspot,irand,rmin,rmax,azmin,azmax)
c
c  trace mspot successful rays randomly arranged on first surface
c      annulus at local z=0.  rays originate from source position.
c  (limit rays between radii rmin,rmax and azimuths azmin,azmax)
```

```
c    (input azimuths in radians between 0 and 2pi)
c   calculate effective area weights and effective area.
c      (effective area is calculated on first surface
c       within radius limits on first surface)
c   intercepts, slopes, and effective area weights are stored for the
c      last surface for each ray.
c   irand must be initialized by calling ranset before this routine.  irand
c   is an integer value for and reset by random number generator
c      ranf.
c
      implicit double precision (a-h,o-z)
c*********************************************************************
      common /sysc1/ zrange,elev,azim,foclen,source(3)
     * ,radlim(2,50),dxcirc(50),dycirc(50)
     * ,xwidth(50),ywidth(50),dxrect(50),dyrect(50),threct(50)
     * ,zlim(2,50),adata(25,50)
     * ,tilt(3,50),rmat(3,3,50)
     * ,disp(3,50),thick(50),findex(50)
     * ,sdata(25,50),delta
     * ,sp(3,50),ra(3,50),spi(3),rai(3)
     * ,energy(15),delbet(2,15,50),wgt(15,50),wgtnet(15),effa(15)
     * ,pi
     * ,imove(50),irstr(50),iwgt(50),nsurf
     * ,nnrg,kmax,kprint(51),ichief,itilt(50)
     * ,npass,nvig,nerr
     * ,iaper(50),iobs(50),itype(50),imode(50),ifdfm(50),ihead(20)
      character * 80 ihead,ifdfm
      character * 8 itype,imode,iaper,iobs
c*********************************************************************
      dimension work(3),tsp(3),tra(3),trmat(3,3),tdisp(3)
c
c  initialize ray counts for ssrt and wraysv
c
      call ssrti
      call wsvi
c
c  initialize effective area accumulation
c
      do 400 i=1,nnrg
  400 effa(i)=0.d0
c
c  set up for ray distribution
c
      aconst=rmax**2-rmin**2
      bconst=rmin**2
      delaz=azmax-azmin
c
c  check azimuth input
c
      if(delaz.lt.0.d0.or.delaz.gt.(2.d0*pi)) then
      write(6,4001)
 4001 format(///'   ***   skip wspot1, invalid azimuths   ***'///)
      return
      endif
c
c  initial effective area weight for ray
c
      efact=pi*(rmax**2-rmin**2)/dble(mspot)
```

```
      efact=efact*delaz/2.d0/pi
c
c  maximum number of attempted rays
      ktry=2000000
c
      do 321 i=1,ktry
c
c select ray
c
c  select ray position on first surface
c
      rr=dsqrt(aconst*ranf(irand)+bconst)
      theta=azmin+delaz*ranf(irand)
c
      tsp(1)=rr*dcos(theta)
      tsp(2)=rr*dsin(theta)
      tsp(3)=0.d0
c
c  transform to source coordinate system
      if(imove(1).ne.0) then
      do 302 j=1,3
      rai(j)=0.d0
      tdisp(j)=disp(j,1)
      do 302 k=1,3
  302 trmat(k,j)=rmat(k,j,1)
      call cnvout(tsp,rai,tsp,tra,trmat,tdisp)
      endif
c
c  set up direction
      sum=0.d0
      do 300 j=1,3
      work(j)=tsp(j)-source(j)
  300 sum=sum+work(j)*work(j)
      sum=dsqrt(sum)
      do 301 j=1,3
      spi(j)=tsp(j)
  301 rai(j)=work(j)/sum
c
c  always put ray in +z direction
      if(rai(3).lt.0.d0) then
      do 303 j=1,3
  303 rai(j)=-rai(j)
      endif
c
c  trace ray and accumulate effective area weights.
c
      call wray(efact,irstat)
      if(irstat.ne.0) go to 321
c
c  accumulate effective area
c
      do 500 j=1,nnrg
  500 effa(j)=effa(j)+wgtnet(j)
c
c  save ray information from last surface
c
      call wraysv(ifill)
c
```

```
c quit if there are enough through rays
c
      if(npass.eq.mspot) go to 627
c
  321 continue
c
  627 continue
c
c  reset effective area if necessary
c
      nfail=nvig+nerr
      if(nfail.ne.0) then
      factor=dble(npass)/dble(nfail+npass)
      do 304 i=1,nnrg
  304 effa(i)=effa(i)*factor
c  also reset stored effective area weights if necessary
      call wsvrst(factor)
      endif
c
      write (6,350) npass,rmin,rmax,azmin,azmax,elev,azim
 350  format('1',i7,' successful rays in wspot1,'/
     * ' random ray distribution on first surface annulus'/
     * ' rmin= ',e24.16,',   rmax= ',e24.16/
     * ' azmin (radians)= ',e24.16,',   azmax (radians)= ',e24.16/
     * ' field angle (radians)= ',e24.16/
     * ' azimuth (radians)    = ',e24.16/)
      write (6,465) nvig,nerr
 465  format(/22x,i7,' rays were vignetted or '
     *'obscured'/22x,i7,' rays failed in ssrt'/)
      if(nerr.ne.0) then
      write(6,351)
  351 format(///'    ***   warning, ray error(s)   ***'///)
      endif
      if(npass.ne.mspot) then
      write(6,352) mspot
  352 format(///'   ***   warning, less than ',i2
     * ,'successful rays'///)
      endif
      do 466 i=1,nnrg
      write(6,467) i,energy(i),effa(i)
  467 format(' energy(',i2,')= ',e24.16,',   effective area= '
     * ,e24.16)
  466 continue
c
      return
      end
      subroutine wspot2(nlong,naz,rmin,rmax,azmin,azmax)
c
c  trace modified wheel spoke ray arrangement on first surface
c      annulus at local z=0.  rays originate from source position.
c  (radii between rmin,rmax and azimuths between azmin,azmax)
c  ( azimuths in radians between 0 and 2pi)
c  calculate effective area weights and effective area.
c     (effective area is calculated on first surface
c      within radius limits on first surface)
c  intercepts, slopes, and effective area weights are stored for the
c     last surface for each ray.
c
```

```
      implicit double precision (a-h,o-z)
c***********************************************************************
      common /syscl/ zrange,elev,azim,foclen,source(3)
     * ,radlim(2,50),dxcirc(50),dycirc(50)
     * ,xwidth(50),ywidth(50),dxrect(50),dyrect(50),threct(50)
     * ,zlim(2,50),adata(25,50)
     * ,tilt(3,50),rmat(3,3,50)
     * ,disp(3,50),thick(50),findex(50)
     * ,sdata(25,50),delta
     * ,sp(3,50),ra(3,50),spi(3),rai(3)
     * ,energy(15),delbet(2,15,50),wgt(15,50),wgtnet(15),effa(15)
     * ,pi
     * ,imove(50),irstr(50),iwgt(50),nsurf
     * ,nnrg,kmax,kprint(51),ichief,itilt(50)
     * ,npass,nvig,nerr
     * ,iaper(50),iobs(50),itype(50),imode(50),ifdfm(50),ihead(20)
      character * 80 ihead,ifdfm
      character * 8 itype,imode,iaper,iobs
c***********************************************************************
      dimension work(3),tsp(3),tra(3),trmat(3,3),tdisp(3)
c
c  initialize ray counts for ssrt and wraysv
c
      call ssrti
      call wsvi
c
c  initialize effective area accumulation
c
      do 400 i=1,nnrg
  400 effa(i)=0.d0
c
c  set up for equal area ray distribution
c
      rmaxsq=rmax**2
      rminsq=rmin**2
      delrsq=(rmaxsq-rminsq)/dble(nlong)
      rconst=2.d0*rminsq-delrsq
      delaz=azmax-azmin
c
c  check azimuthal limits
c
      if(delaz.lt.0.d0.or.delaz.gt.(2.d0*pi)) then
      write(6,4001)
 4001 format(///'   ***   skip wspot2, invalid azimuths   ***'///)
      return
      endif
c
c  effective area weight for ray
c
      efact=pi*delrsq/dble(naz)
      efact=efact*delaz/2.d0/pi
c
      delaz=delaz/dble(naz)
      azcnst=azmin+delaz/2.d0
c

c
      do 321 i=1,nlong
```

```
c
      rr=dsqrt((rconst+delrsq*dble(2*i))/2.d0)
c
      do 321 m=1,naz
c
      theta=azcnst+delaz*dble(m-1)
c
      tsp(1)=rr*dcos(theta)
      tsp(2)=rr*dsin(theta)
      tsp(3)=0.d0
c
c  transform to source coordinate system
      if(imove(1).ne.0) then
      do 302 j=1,3
      rai(j)=0.d0
      tdisp(j)=disp(j,1)
      do 302 k=1,3
  302 trmat(k,j)=rmat(k,j,1)
      call cnvout(tsp,rai,tsp,tra,trmat,tdisp)
      endif
c
c  set up direction
      sum=0.d0
      do 300 j=1,3
      work(j)=tsp(j)-source(j)
  300 sum=sum+work(j)*work(j)
      sum=dsqrt(sum)
      do 301 j=1,3
      spi(j)=tsp(j)
  301 rai(j)=work(j)/sum
c
c  always put ray in +z direction
      if(rai(3).lt.0.d0) then
      do 303 j=1,3
  303 rai(j)=-rai(j)
      endif
c
c  trace ray and accumulate effective area weights.
c
      call wray(efact,irstat)
      if(irstat.ne.0) go to 321
c
c  accumulate effective area
c
      do 500 j=1,nnrg
  500 effa(j)=effa(j)+wgtnet(j)
c
c  save ray information from last surface
c
      call wraysv(ifill)
c
  321 continue
c
      write (6,350) npass,nlong,naz,rmin,rmax,azmin,azmax,elev,azim
  350 format('1',i7,' successful rays in wspot2,'/
     * '   :modified spoke wheel ray distribution on first surface'/
     * '    annulus, varying radial increments and constant '/
     * '    azimuthal angle increment'/
```

```
     * ' ',i7,' radial points,   ',i7,' azimuthal points'/
     * ' rmin= ',e24.16,',   rmax= ',e24.16/
     * ' azmin (radians)= ',e24.16,',   azmax (radians)= ',e24.16/
     * ' field angle (radians)= ',e24.16/
     * ' azimuth (radians)    = ',e24.16)
      write (6,465) nvig,nerr
 465  format(/22x,i7,' rays were vignetted or '
     *'obscured'/22x,i7,' rays failed in ssrt'/)
      if(nerr.ne.0) then
      write(6,351)
 351  format(///'    ***   warning, ray error(s)   ***'///)
      endif
      do 466 i=1,nnrg
      write(6,467) i,energy(i),effa(i)
 467  format(' energy(',i2,')= ',e24.16,',   effective area= '
     * ,e24.16)
 466  continue
c
      return
      end
      subroutine wstat(iener,xav,yav,wav,wtot,xref,yref,fl,ell)
c***********************************************************************
c
c  calculate average and rms of stored rays at energy iener
c
c***********************************************************************
      implicit double precision (a-h,o-z)
c***********************************************************************
      common /syscl/ zrange,elev,azim,foclen,source(3)
     * ,radlim(2,50),dxcirc(50),dycirc(50)
     * ,xwidth(50),ywidth(50),dxrect(50),dyrect(50),threct(50)
     * ,zlim(2,50),adata(25,50)
     * ,tilt(3,50),rmat(3,3,50)
     * ,disp(3,50),thick(50),findex(50)
     * ,sdata(25,50),delta
     * ,sp(3,50),ra(3,50),spi(3),rai(3)
     * ,energy(15),delbet(2,15,50),wgt(15,50),wgtnet(15),effa(15)
     * ,pi
     * ,imove(50),irstr(50),iwgt(50),nsurf
     * ,nnrg,kmax,kprint(51),ichief,itilt(50)
     * ,npass,nvig,nerr
     * ,iaper(50),iobs(50),itype(50),imode(50),ifdfm(50),ihead(20)
      character * 80 ihead,ifdfm
      character * 8 itype,imode,iaper,iobs
c***********************************************************************
      common /rsavel/ xpsv(200000),ypsv(200000),dxdzsv(200000)
     * ,dydzsv(200000),entx(200000),enty(200000),wtsv(15,200000)
     * ,zshift,nsv
c***********************************************************************
      common /worksp/ bigmat(600000)
c***********************************************************************
      dimension work(200000)
      equivalence (work,bigmat)
c***********************************************************************
      do 100 i=1,nsv
 100  work(i)=wtsv(iener,i)
      call stat(xpsv,ypsv,work,nsv,xav,yav,xrms,yrms,rms,wtot,wav,wrms
     * ,xmin,xmax,ymin,ymax,wmin,wmax)
```

```
c  report the results
      write(6,201) iener,energy(iener),nsv,elev,zshift,xav,yav,xrms
     * ,yrms,rms
  201 format(/'  length statistics for: energy(',i2,')= ',e24.16/
     * ' number of rays= ',i7,',   field angle (radians)= ',e24.16/
     * ' net zshift= ',e24.16/
     * ' x average= ',e24.16,',   y average= ',e24.16/
     * ' xrms     = ',e24.16,',   yrms     = ',e24.16/
     * '           rms= ',e24.16)
      write(6,205) xmin,xmax,ymin,ymax,wtot,wav,wrms,wmin,wmax
  205 format(
     * ' xmin= ',e24.16,',   xmax= ',e24.16/
     * ' ymin= ',e24.16,',   ymax= ',e24.16/
     * '     weight sum= ',e24.16/
     * ' weight average= ',e24.16/
     * '     weight rms= ',e24.16/
     * ' wmin= ',e24.16,',   wmax= ',e24.16//)
c  calculate angular quantities with assumed focal length fl
      if(fl.ne.0.d0) then
      factor=3600.d0*180.d0/pi
      axav=factor*datan(xav/fl)
      ayav=factor*datan(yav/fl)
      axrms=factor*datan(xrms/fl)
      ayrms=factor*datan(yrms/fl)
      arms=factor*datan(rms/fl)
      write(6,203)iener,energy(iener),fl,nsv,axav,ayav,axrms
     * ,ayrms,arms
  203 format('  arc sec statistics for: energy(',i2,')= ',e24.16/
     * ' assumed focal length= ',e24.16,',   number of rays ',i7/
     * ' x average (arc sec)= ',e24.16/
     * ' y average (arc sec)= ',e24.16/
     * ' xrms (arc sec)     = ',e24.16/
     * ' yrms (arc sec)     = ',e24.16/
     * ' rms (arc sec)      = ',e24.16/)
      endif
c  calculate apparent focal length from displacement from xref,yref
c  and from assumed field angle el1
      if(el1.ne.0.d0) then
      factor=3600.d0*180.d0/pi
      ff=dsqrt((xav-xref)**2+(yav-yref)**2)/dtan(el1)
      ael1=factor*el1
      write(6,204) xref,yref,ael1
  204 format(' xref= ',e24.16,',   yref= ',e24.16/
     * ' assumed field angle (arc sec)= ',e24.16)
      write(6,202) ff
  202 format(' apparent focal length is       ',e24.16/
     * '     :deduced from (xav-xref,yav-yref) and assumed field angle'/)
      endif
      return
      end
      subroutine dfm02
      implicit double precision (a-h,o-z)
c
c***********************************************************************
c
c  routine to compute contribution of surface errors
c  radius error and gradient of radius error
c
```

```
c  warning:  this assumes a specific form for the
c            function f to be minimized.  Specifically
c            it assumes that f is the difference between
c            the ray position radius value and
c            the surface radius value.
c
c***********************************************************************
      common /sysc1/ zrange,elev,azim,foclen,source(3)
     * ,radlim(2,50),dxcirc(50),dycirc(50)
     * ,xwidth(50),ywidth(50),dxrect(50),dyrect(50),threct(50)
     * ,zlim(2,50),adata(25,50)
     * ,tilt(3,50),rmat(3,3,50)
     * ,disp(3,50),thick(50),findex(50)
     * ,sdata(25,50),delta
     * ,sp(3,50),ra(3,50),spi(3),rai(3)
     * ,energy(15),delbet(2,15,50),wgt(15,50),wgtnet(15),effa(15)
     * ,pi
     * ,imove(50),irstr(50),iwgt(50),nsurf
     * ,nnrg,kmax,kprint(51),ichief,itilt(50)
     * ,npass,nvig,nerr
     * ,iaper(50),iobs(50),itype(50),imode(50),ifdfm(50),ihead(20)
      character * 80 ihead,ifdfm
      character * 8 itype,imode,iaper,iobs
c***********************************************************************
c  common for communication with user trace                             32901000
      common/userdt/x,y,z,f,fx,fy,fz,isurf,ifcalc,isferr                32902000
c***********************************************************************
c
c  common for surface deformation data
c
      common/deform1/tdfml(2,2),zdfml(2,2),tdfmd(2),zdfmd(2)
     * ,adfm(201201,2),ntdfm(2),nzdfm(2),idfmsf(50),nhdfm(2)
     * ,ihdfm(20,2),ifldfm(2)
      character * 80 ihdfm,ifldfm
      real adfm
c
c  interpolation coordinates are theta,z (t,z), i.e. polar coordinates.
c  bilinear approximation is given by a1 + a2*t + a3*z +a4*t*z
c  'a' values are in order of ntdfm increasing theta values for
c  each of nzdfm z values also in increasing order.  tdfmd and
c  and zdfmd are the increments in the t and z directions.  ihdfm
c  holds nhdfm comment lines about the deformation data set. tdfml and
c  zdfml are the theta and z limits of the grid, respectively.
c  idfmsf gives the storage position for each surface.  there are now 2
c  storage positions available. The input file name is stored in ifldfm.
c
c***********************************************************************
c
c  add deformation contribution to function value f and
c  gradient components fx,fy,fz
c
c  compute the grid position in the deformation array
c
      ndfm=idfmsf(isurf)
c
c  use the grid position at the edge if
c  the value is very near the edge
c
```

```
      t=datan2(y,x)
      ftplc=(t-tdfml(1,ndfm))/tdfmd(ndfm)
      itplc=idint(ftplc+1.d0)
      if(itplc.lt.1) then
        if((-ftplc).lt.1.d-14) then
             itplc=1
           else
          go to 3000
        endif
      elseif(itplc.ge.ntdfm(ndfm)) then
        if(((t-tdfml(2,ndfm))/tdfmd(ndfm)).lt.1.d-14) then
             itplc=ntdfm(ndfm)-1
           else
          go to 3000
        endif
      endif
c
      fzplc=(z-zdfml(1,ndfm))/zdfmd(ndfm)
      izplc=idint(fzplc+1.d0)
      if(izplc.lt.1) then
        if((-fzplc).lt.1.d-14) then
             izplc=1
        else
          go to 3000
        endif
      elseif(izplc.ge.nzdfm(ndfm)) then
        if(((z-zdfml(2,ndfm))/zdfmd(ndfm)).lt.1.d-14) then
             izplc=nzdfm(ndfm)-1
        else
          go to 3000
        endif
      endif
c
      n11=itplc+(izplc-1)*ntdfm(ndfm)
      n21=n11+1
      n12=n11+ntdfm(ndfm)
      n22=n12+1
      b1=dble(adfm(n11,ndfm))
      b2=dble(adfm(n21,ndfm)-adfm(n11,ndfm))/tdfmd(ndfm)
      b3=dble(adfm(n12,ndfm)-adfm(n11,ndfm))/zdfmd(ndfm)
      b4=dble(adfm(n22,ndfm)-adfm(n12,ndfm)-adfm(n21,ndfm)
     * +adfm(n11,ndfm))/tdfmd(ndfm)/zdfmd(ndfm)
      tdel=t-(tdfml(1,ndfm)+dble(itplc-1)*tdfmd(ndfm))
      zdel=z-(zdfml(1,ndfm)+dble(izplc-1)*zdfmd(ndfm))
c
c***********************************************************************
c
      if(ifcalc.eq.1.or.ifcalc.eq.3) then
c
        f=f-(b1+b2*tdel+b3*zdel+b4*tdel*zdel)
c
      endif
c
c***********************************************************************
c
      if(ifcalc.eq.2.or.ifcalc.eq.3) then
        rsq=x*x+y*y
        if(rsq.le.0.d0) go to 3000
```

```
          fac=(b2+b4*zdel)/rsq
          fx=fx-fac*(-y)
          fy=fy-fac*x
          fz=fz-(b3+b4*tdel)
        endif
c
c*********************************************************************
c
      return
c
c*********************************************************************
c
c  error return
c
3000  continue
      write(6,3001)
3001  format('0***   error in dfm02, isferr set to 1 ***')
      isferr=1
      return
      end
      subroutine prtdfm(debug,nsurf)
      implicit double precision (a-h,o-z)
c
c*********************************************************************
c
c  print out deformation storage data
c
c*********************************************************************
c
c  common for surface deformation data
c
      common/deform1/tdfml(2,2),zdfml(2,2),tdfmd(2),zdfmd(2)
     * ,adfm(201201,2),ntdfm(2),nzdfm(2),idfmsf(50),nhdfm(2)
     * ,ihdfm(20,2),ifldfm(2)
      character * 80 ihdfm,ifldfm
      real adfm
c
c  interpolation coordinates are theta,z (t,z), i.e. polar coordinates.
c  bilinear approximation is given by a1 + a2*t + a3*z +a4*t*z
c  'a' values are in order of ntdfm increasing theta values for
c  each of nzdfm z values also in increasing order.  tdfmd and
c  and zdfmd are the increments in the t and z directions.  ihdfm
c  holds nhdfm comment lines about the deformation data set. tdfml and
c  zdfml are the theta and z limits of the grid, respectively.
c  idfmsf gives the storage position for each surface.  there are now 2
c  storage positions available. The input file name is stored in ifldfm.
c
c*********************************************************************
c
      logical debug
      do 400 i=1,nsurf
      if(idfmsf(i).ne.0) then
      write(6,607) i,ifldfm(idfmsf(i)),idfmsf(i)
607   format(//' surface ',i3,' uses file:'/
     * 1x,a/
     * ' in storage area ',i1)
      endif
400   continue
```

```
      do 100 ndfm=1,2
      if(ifldfm(ndfm).eq.' ') go to 100
      write(6,601) ifldfm(ndfm),ndfm
 601  format(//' deformation surface data from file:'/
     * 1x,a/
     * ' in storage area ',i1)
      do 602 i=1,nhdfm(ndfm)
      if(ihdfm(i,ndfm).eq.' ') go to 602
      write(6,603) ihdfm(i,ndfm)
 603  format(1x,a)
 602  continue
      write(6,604) ntdfm(ndfm),nzdfm(ndfm),(tdfml(i,ndfm),i=1,2)
     * ,tdfmd(ndfm),(zdfml(i,ndfm),i=1,2),zdfmd(ndfm)
 604  format(i10,' azimuthal bins,    ',i10,' axial bins'/
     * ' azimuthal limits (radians) ',2e24.16/
     * ' azimuthal increment (radians) ',e24.16/
     * ' axial limits ',2e24.16/
     * ' axial increment ',e24.16//)
      if(debug) then
c
c  dump out the derived deformation values
c
      do 200 i=1,nzdfm(ndfm)
      zpos=zdfml(1,ndfm)+dble(i-1)*zdfmd(ndfm)
      do 200 j=1,ntdfm(ndfm)
      tpos=tdfml(1,ndfm)+dble(j-1)*tdfmd(ndfm)
      write(6,605) i,j
 605  format('t element ',i5,',    z element ',i5)
      nplace=j+(i-1)*ntdfm(ndfm)
c
c
      write(6,606) tpos,zpos,adfm(nplace,ndfm)
 606  format(' t=',e23.15,', z=',e23.15,', dr=',e15.7)
c
c
 200  continue
      endif
 100  continue
c
c*********************************************************************
c
      return
      end
      subroutine rdfm(iunit)
      implicit double precision (a-h,o-z)
c
c*********************************************************************
c
c  routine to read in deformation values to common area
c  deformation file name is in ifdfm
c
c*********************************************************************
      common /syscl/ zrange,elev,azim,foclen,source(3)
     * ,radlim(2,50),dxcirc(50),dycirc(50)
     * ,xwidth(50),ywidth(50),dxrect(50),dyrect(50),threct(50)
     * ,zlim(2,50),adata(25,50)
     * ,tilt(3,50),rmat(3,3,50)
     * ,disp(3,50),thick(50),findex(50)
```

```
     * ,sdata(25,50),delta
     * ,sp(3,50),ra(3,50),spi(3),rai(3)
     * ,energy(15),delbet(2,15,50),wgt(15,50),wgtnet(15),effa(15)
     * ,pi
     * ,imove(50),irstr(50),iwgt(50),nsurf
     * ,nnrg,kmax,kprint(51),ichief,itilt(50)
     * ,npass,nvig,nerr
     * ,iaper(50),iobs(50),itype(50),imode(50),ifdfm(50),ihead(20)
      character * 80 ihead,ifdfm
      character * 8 itype,imode,iaper,iobs
c***********************************************************************
c
c  common for surface deformation data
c
      common/deform1/tdfml(2,2),zdfml(2,2),tdfmd(2),zdfmd(2)
     * ,adfm(201201,2),ntdfm(2),nzdfm(2),idfmsf(50),nhdfm(2)
     * ,ihdfm(20,2),ifldfm(2)
      character * 80 ihdfm,ifldfm
      real adfm
c
c  interpolation coordinates are theta,z (t,z), i.e. polar coordinates.
c  bilinear approximation is given by a1 + a2*t + a3*z +a4*t*z
c  'a' values are in order of ntdfm increasing theta values for
c  each of nzdfm z values also in increasing order.  tdfmd and
c  and zdfmd are the increments in the t and z directions.  ihdfm
c  holds nhdfm comment lines about the deformation data set. tdfml and
c  zdfml are the theta and z limits of the grid, respectively.
c  idfmsf gives the storage position for each surface.  there are now 2
c  storage positions available. The input file name is stored in ifldfm.
c
c***********************************************************************
      logical debug
c
c  set number of deformation surfaces to zero
c
      ndfm=0
c
c  initialize some other values
c
      ifldfm(1)=' '
      ifldfm(2)=' '
      do 101 i=1,nsurf
101   idfmsf(i)=0
c
c***********************************************************************
c
c  loop through surfaces
c
      do 100 isurf=1,nsurf
      if(ifdfm(isurf).eq.' ') go to 100
        iread=1
c
        if(ndfm.ne.0) then
                do 103 i=1,ndfm
                if(ifldfm(i).eq.ifdfm(isurf)) then
                     idfmsf(isurf)=i
                  iread=0
                endif
```

```
          if(iread.eq.0) go to 102
103             continue
102             continue
           endif
c
           if(iread.eq.1) then
        ndfm=ndfm+1
        if(ndfm.gt.2) go to 4000
        open(iunit,file=ifdfm(isurf),form='unformatted',err=3000)
        read(iunit,end=3000,err=3000) ntdfm(ndfm),nzdfm(ndfm)
     *        ,nhdfm(ndfm)
        if(ntdfm(ndfm).lt.2.or.nzdfm(ndfm).lt.2) go to 3000
        nn=ntdfm(ndfm)*nzdfm(ndfm)
        if(nn.gt.201201) go to 3000
        if(nhdfm(ndfm).lt.1.or.nhdfm(ndfm).gt.20) go to 3000
        read(iunit,end=3000,err=3000) (adfm(j,ndfm),j=1,nn)
     *        ,(tdfml(i,ndfm),i=1,2),(zdfml(i,ndfm),i=1,2)
     *        ,(ihdfm(i,ndfm),i=1,nhdfm(ndfm))
           tdfmd(ndfm)=(tdfml(2,ndfm)-tdfml(1,ndfm))/dble(ntdfm(ndfm)-1)
           zdfmd(ndfm)=(zdfml(2,ndfm)-zdfml(1,ndfm))/dble(nzdfm(ndfm)-1)
           idfmsf(isurf)=ndfm
        ifldfm(ndfm)=ifdfm(isurf)
        close(iunit,err=3000)
        endif
c
100   continue
c
c***********************************************************************
c
c  echo the input
      debug=.false.
      call prtdfm(debug,nsurf)
c
c***********************************************************************
c
      return
c
c***********************************************************************
c
c  namelist input error
c
4000  continue
      write(6,4001)
4001  format('0namelist input error detected in rdfm, stop ')
      stop
c
c***********************************************************************
c
c  input file read error
c
3000  continue
      write(6,3001)    ifdfm(isurf)
3001  format('0input file read error or bad data in rdfm, stop '/
     * ' input file name:'/
     * 1x,a)
      stop
c
c***********************************************************************
```

```
c
      end
      subroutine strc02(isterr,is)
      implicit double precision (a-h,o-z)
c*********************************************************************
      common /syscl/ zrange,elev,azim,foclen,source(3)
     * ,radlim(2,50),dxcirc(50),dycirc(50)
     * ,xwidth(50),ywidth(50),dxrect(50),dyrect(50),threct(50)
     * ,zlim(2,50),adata(25,50)
     * ,tilt(3,50),rmat(3,3,50)
     * ,disp(3,50),thick(50),findex(50)
     * ,sdata(25,50),delta
     * ,sp(3,50),ra(3,50),spi(3),rai(3)
     * ,energy(15),delbet(2,15,50),wgt(15,50),wgtnet(15),effa(15)
     * ,pi
     * ,imove(50),irstr(50),iwgt(50),nsurf
     * ,nnrg,kmax,kprint(51),ichief,itilt(50)
     * ,npass,nvig,nerr
     * ,iaper(50),iobs(50),itype(50),imode(50),ifdfm(50),ihead(20)
      character * 80 ihead,ifdfm
      character * 8 itype,imode,iaper,iobs
c***********************************************************************
c  common for communication with user trace                            32901000
      common/userdt/x,y,z,f,fx,fy,fz,isurf,ifcalc,isferr               32902000
c***********************************************************************
c
c  this version is for deformed surfaces analagous to
c  surface types grzcon01, grzcon02, and grzcon03.
c
c  the deformed surface types are grzcon11, grzcon12, and grzcon13
c  respectively
c
c  grzcon11 is direct calculation with no deformation.
c  grzcon12 is iteration including deformation file data.
c  grzcon13 is direct calculation followed by iteration
c              including deformation file data.
c
c  The deformation file has a reqular grid of rectangles.  On each
c  rectangle there are constants for bilinear interpolation.
c  The surface errors are in terms of a radius error as
c  a function of axial position z and azimuthal position
c  theta. However the gradient is computed in the usual
c  x, y, z coordinates of the surface.  The deformation
c  file is read in when it is first used into a
c  common area.  Presently there is room for two deformed
c  surfaces.  If the process takes the ray inside the
c  aft end or outside the front end the intercept is set
c  outside of the z limits so that the ray will be
c  vignetted.  This assumes a concave surface.
c
c*******************************************************************
c  trace to surface 'is'  (this version is for reflection or           33020000
c                         dummy surfaces only)                         33030000
c                                                                      33350000
c  input is:  starting ray position sp(,is)
c             starting direction cosines ra(,is)
c                                                                      33390000
c  output is: new ray position sp(,is)
```

```
c              new direction cosines ra(,is)
c              isterr ne 0 for ray error
c
      isterr=0
c
c*****************************************************************
c
c  find intercept
c
c  set surface number for user trace
      isurf=is
c  initialize ray position and direction
      x=sp(1,is)
      y=sp(2,is)
      z=sp(3,is)
      dxds=ra(1,is)
      dyds=ra(2,is)
      dzds=ra(3,is)
c*****************************************************************
c
c  direct calculation here
      if(itype(is).eq.'grzcon11'.or.itype(is).eq.'grzcon13') then
      call utri02
      if(isferr.eq.1) go to 3000
      endif
c
c*****************************************************************
c
c  iteration here
c
      if(itype(is).eq.'grzcon12'.or.itype(is).eq.'grzcon13') then
c
c*****************************************************************
c
c  check for ray missing surface
c  if ray misses surface set z so that ray will be
c  vignetted
c  this assumes concave surface
c
      xt=x
      yt=y
      zt=z
      z=zlim(2,is)
      x=xt+dxds/dzds*(z-zt)
      y=yt+dyds/dzds*(z-zt)
c set ifcalc for function value only
      ifcalc=1
      call utrc02
      f1=f
      z=zlim(1,is)
      x=xt+dxds/dzds*(z-zt)
      y=yt+dyds/dzds*(z-zt)
      call utrc02
      f2=f
      fsign=f1*f2
      if(fsign.gt.0.d0) then
c ray misses surface region.
c This test assumes concave surface
```

```
c set x,y,z  to be vignetted and cut out of subroutine
        z=2.d0*zlim(2,is)-zlim(1,is)
        sp(1,is)=xt+dxds/dzds*(z-zt)
        sp(2,is)=yt+dyds/dzds*(z-zt)
        sp(3,is)=z
        return
      else
c
c  make sure that starting point is within the
c  element
c
        if(zt.lt.zlim(1,is)) then
          z=zlim(1,is)
          x=xt+dxds/dzds*(z-zt)
          y=yt+dyds/dzds*(z-zt)
        elseif(zt.gt.zlim(2,is)) then
          z=zlim(2,is)
          x=xt+dxds/dzds*(z-zt)
          y=yt+dyds/dzds*(z-zt)
        else
                x=xt
                y=yt
                z=zt
           endif
      endif
c
c*****************************************************************
c
c  initialize iteration count                                          33500000
      kount=0                                                          33510000
c debug print                                                          33541000
c     write(6,*)is,x,y,z,dxds,dyds,dzds,delta,kmax
c                                                                      33541200
c  require function value and gradient calculation                     33541300
      ifcalc=3                                                         33541400
c  iteration loop for intercept                                        33542000
  100 continue                                                         33550000
      kount=kount+1                                                    33560000
      if(kount.gt.kmax) go to 3000                                     33570000
      call utrc02                                                      33600000
      if(isferr.eq.1) go to 3000                                       33620000
      ds=-f/(fx*dxds+fy*dyds+fz*dzds)                                  33690000
c  dont go out of the surface region
      zt=z+dzds*ds
      icut=0
      if(zt.lt.zlim(1,is)) then
       ds=(zlim(1,is)-z)/dzds/2.d0
       icut=1
      elseif(zt.gt.zlim(2,is)) then
       ds=(zlim(2,is)-z)/dzds/2.d0
       icut=1
      endif
      x=x+dxds*ds                                                      33691000
      y=y+dyds*ds                                                      33692000
      z=z+dzds*ds                                                      33693000
c debug print
c     write(6,*)kount,ds,x,y,z,f,fx,fy,fz,icut                         33695000
c                                                                      33696000
```

```
      if (icut.eq.1) go to 100
c
      if (dabs(ds).gt.delta) go to 100
c
      endif
c*********************************************************************
c  calculation for outgoing ray
c (currently covers reflection and thru  surfaces only)
c
      if(imode(is).eq.'refl') then
c do gradient calculation only
      ifcalc=2
      call utrc02
      if(isferr.eq.1) go to 3000
      c=(dxds*fx+dyds*fy+dzds*fz)/(fx**2+fy**2+fz**2)
      ra(1,is)=dxds-2.d0*c*fx
      ra(2,is)=dyds-2.d0*c*fy
      ra(3,is)=dzds-2.d0*c*fz
c debug print
c     write(6,*)x,y,z,fx,fy,fz
c
      elseif(imode(is).eq.'thru') then
      else
c invalid surface
      go to 3000
      endif
c
      sp(1,is)=x
      sp(2,is)=y
      sp(3,is)=z
c
      return
 3000 continue
c  error return
      write(6,3001)
 3001 format('0   ***    strc02 error, isterr set to 1    *** ')
      isterr=1
      return
      end
      subroutine utrc02
c
c  calculate function f and gradient fx,fy,fz for surface
c
c  input
c        x,y,z              position
c        isurf or n         surface number
c        itype(n)           surface type
c        sdata(...,n)       surface parameters
c        ifcalc             calculate function value if ifcalc=1
c                           calculate gradient if ifcalc=2
c                           calculate both if ifcalc=3
c  output
c        f                  function value
c        fx,fy,fz           gradient of function
c        isferr             non zero if error occurs
c
      implicit double precision (a-h,o-z)
c*********************************************************************
```

```
       common /syscl/ zrange,elev,azim,foclen,source(3)
     * ,radlim(2,50),dxcirc(50),dycirc(50)
     * ,xwidth(50),ywidth(50),dxrect(50),dyrect(50),threct(50)
     * ,zlim(2,50),adata(25,50)
     * ,tilt(3,50),rmat(3,3,50)
     * ,disp(3,50),thick(50),findex(50)
     * ,sdata(25,50),delta
     * ,sp(3,50),ra(3,50),spi(3),rai(3)
     * ,energy(15),delbet(2,15,50),wgt(15,50),wgtnet(15),effa(15)
     * ,pi
     * ,imove(50),irstr(50),iwgt(50),nsurf
     * ,nnrg,kmax,kprint(51),ichief,itilt(50)
     * ,npass,nvig,nerr
     * ,iaper(50),iobs(50),itype(50),imode(50),ifdfm(50),ihead(20)
       character * 80 ihead,ifdfm
       character * 8 itype,imode,iaper,iobs
c*********************************************************************
c  common for communication with user trace
       common/userdt/x,y,z,f,fx,fy,fz,isurf,ifcalc,isferr
c********************************************************************
c
c  common for surface deformation data
c
       common/deform1/tdfml(2,2),zdfml(2,2),tdfmd(2),zdfmd(2)
     * ,adfm(201201,2),ntdfm(2),nzdfm(2),idfmsf(50),nhdfm(2)
     * ,ihdfm(20,2),ifldfm(2)
       character * 80 ihdfm,ifldfm
       real adfm
c
c  interpolation coordinates are theta,z (t,z), i.e. polar coordinates.
c  bilinear approximation is given by a1 + a2*t + a3*z +a4*t*z
c  'a' values are in order of ntdfm increasing theta values for
c  each of nzdfm z values also in increasing order.  tdfmd and
c  and zdfmd are the increments in the t and z directions.  ihdfm
c  holds nhdfm comment lines about the deformation data set. tdfml and
c  zdfml are the theta and z limits of the grid, respectively.
c  idfmsf gives the storage position for each surface.  there are now 2
c  storage positions available. The input file name is stored in ifldfm.
c
c**********************************************************************
       equivalence (n,isurf)
c
       isferr=0
c
       if(itype(n).eq.'grzcon11'.or.itype(n).eq.'grzcon12'
     * .or.itype(n).eq.'grzcon13') then
c*********************************************************************
c  grazing conic plus sag error:
c
c  rho at z=0                 sdata(1,n)
c
c  subnormal at z=0           sdata(2,n)
c
c  1-e**2                     sdata(3,n)
c
c  full mirror length         sdata(4,n)
c
c  zero-peak sag error        sdata(5,n)
```

```
c  (mirror ends fixed)
c
c  average r  error           sdata(6,n)
c
c delta r error               sdata(7,n)
c
      rhosq=sdata(1,n)**2+2.d0*sdata(2,n)*z-sdata(3,n)*z**2             340126
      if(rhosq.le.0.d0) go to 3000                                       340127
      rho=dsqrt(rhosq)                                                   340128
      rad=dsqrt(x**2+y**2)                                               340129
      if(rad.le.0.d0) go to 3000                                         340130
c
      if(ifcalc.eq.2.or.ifcalc.eq.3) then
      fx=x/rad                                                           340134
      fy=y/rad                                                           340135
      fz=  z*(sdata(3,n)/rho+8.d0*sdata(5,n)/sdata(4,n)**2)             340137
     * -sdata(2,n)/rho+sdata(7,n)/sdata(4,n)                             340138
      endif
c
      if(ifcalc.eq.1.or.ifcalc.eq.3) then
      f=rad-(rho                                                         340140
     *   -4.d0*sdata(5,n)*((z/sdata(4,n))**2-1.d0/4.d0)+sdata(6,n)      340141
     *   -sdata(7,n)*z/sdata(4,n) )
      endif                                                              340142
c
c     include deformations if present
c
      if(idfmsf(n).ne.0) then
      call dfm02
      if(isferr.ne.0) go to 3000
      endif
c
c*********************************************************************
c                                                                        340143
      elseif(itype(n).eq.'flat') then
c*********************************************************************
c  flat surface                                                          340145
      if(ifcalc.eq.2.or.ifcalc.eq.3) then
      fx=0.d0                                                            340160
      fy=0.d0                                                            340170
      fz=1.d0                                                            340180
      endif
c
      if(ifcalc.eq.1.or.ifcalc.eq.3) then
      f=z
      endif
c
c*********************************************************************
c                                                                        340190
      else                                                               340197
c  invalid surface                                                       340198
      go to 3000                                                         340199
      endif                                                              340200
      return                                                             340300
 3000 continue                                                           340310
c  computation error                                                     340311
      write(6,3001)
 3001 format('0   ***    utrc02 error, isferr set to 1   ***')
```

```
      isferr=1
      return
c***********************************************************************
c
      entry utri02
      isferr=0
      if(itype(n).eq.'grzcon11'.or.itype(n).eq.'grzcon13') then
c
c  direct calculation of intercept of ray with concave grazing conic.
c  take error return for two solutions within element
c   (in case of a convex optic this could be changed to take the first
c    solution within the element)
c  take first solution if there is no solution within the element.
c  cut out to error return for no solution
c   (in case of no solution, if desired, one could artificially set the solution
c    outside of the element to be vignetted)
c
c  look for solutions
c
        a=ra(1,n)**2+ra(2,n)**2+sdata(3,n)*ra(3,n)**2
        b=2.d0*(sp(1,n)*ra(1,n)+sp(2,n)*ra(2,n)-sdata(2,n)*ra(3,n))
        c=sp(1,n)**2+sp(2,n)**2-sdata(1,n)**2
      isol=0
c
        if(a.eq.0.d0) then
           if(b.eq.0.d0) then
c no solution
           go to 3000
           endif
           sol=-c/b
        isol=1
        elseif(c.eq.0.d0) then
        if(b.eq.0.d0) then
           sol=0.d0
           isol=1
           else
           s1=0.d0
           s2=-b/a
           endif
      else
           if(b.eq.0.d0) then
           arg=-c/a
              if(arg.lt.0.d0) then
c no solution
           go to 3000
              endif
              s1=dsqrt(arg)
              s2=-s1
           else
              arg=b**2-4.d0*a*c
              if(arg.lt.0.d0) then
c no solution
           go to 3000
              endif
              arg=dsqrt(arg)
              denom1=b+arg
              denom2=b-arg
           if(denom1.ne.0.d0) then
```

```
            if(denom2.ne.0.d0) then
                s1=-2.d0*c/denom1
              s2=-2.d0*c/denom2
              else
              sol=-2.d0*c/denom1
                isol=1
              endif
            elseif(denom2.ne.0.d0) then
              sol=-2.d0*c/denom2
              isol=1
            else
c no solution
              go to 3000
            endif
          endif
        endif
c
c make selection of solution if it is not unique.
c
        if(isol.eq.0) then
          test=sdata(4,n)/2.d0
          z1=dabs(s1*ra(3,n))
          z2=dabs(s2*ra(3,n))
          if(z1.lt.test) then
          if(z2.lt.test) then
c              two solutions within element
c              (there could actually be two solutions within
c               the element or this could also be caused
c               by a numerical problem above)
          go to 3000
            else
          sol=s1
            endif
          elseif(z2.lt.test) then
          sol=s2
          else
          sol=s1
          if(s2.lt.s1) sol=s2
          endif
      endif
c
c move to the solution point
c
        x=sp(1,n)+ra(1,n)*sol
        y=sp(2,n)+ra(2,n)*sol
        z=ra(3,n)*sol
c
      else
c invalid surface
        go to 3000
      endif
c***********************************************************************
      return                                                          34050
      end
      subroutine conic(rhoz,subn,skapa,z,rho,slope)
      implicit double precision (a-h,o-z)
      rho=dsqrt(rhoz**2+2.d0*subn*z-skapa*z**2)
      slope=(subn-skapa*z)/rho
```

```
      return
      end
      subroutine fildf(iu,name,type,mode)
c
c       routine to open unit iu to file 'name'.'type'
c
      character * 80 name,type,mode
      call nb(name,n1,n2,80)
      if(n1.le.0) go to 3000
      call nb(type,m1,m2,80)
      if(m1.le.0) go to 3000
      open(iu,file=name(n1:n2)//'.'//type(m1:m2),access='sequential',
     * form=mode,err=3000)
      return
 3000 continue
c       error return
      write(6,*) ' unable to open file, stop'
      stop
      end
      subroutine matab(a,b,c,n1,n2,n3,d)
c
c  c = a x b, matrix multiplication
c
c  actual a,b,c can be the same array
c
      implicit double precision (a-h,o-z)
      dimension a(n1,n2),b(n2,n3),c(n1,n3),d(n1,n3)
      do 100 i=1,n1
      do 100 j=1,n3
      d(i,j)=0.d0
      do 100 k=1,n2
 100  d(i,j)=d(i,j)+a(i,k)*b(k,j)
      do 200 i=1,n1
      do 200 j=1,n3
  200 c(i,j)=d(i,j)
      return
      end
      subroutine nb(array,num1,num2,nsize)                              00221030
c
c find first string of non blank characters                             00221130
c
      character * (*) array                                             00222030
      num1=0                                                            00223030
      num2=nsize
      if(nsize.le.0) go to 1000                                         00224030
      do 100 i=1,nsize                                                  00225030
      if(array(i:i).ne.' ') then
      if(num1.eq.0) num1=i
      else
      if(num1.ne.0) then
      num2=i-1
      go to 200
      endif
      endif
  100 continue                                                          00228030
  200 continue                                                          00229030
 1000 return                                                            00229130
      end                                                               00229230
```

```
      subroutine pfocus(x,y,ck,cl,w,n,xloc,yloc,zloc)
      implicit double precision (a-h,o-z)
      dimension x(n),y(n),ck(n),cl(n),w(n)
c
c               find weighted planar best focus using rays from
c               geometric ray trace
c
c               minimize weighted sum of squared differences from
c               average position in x-y plane.
c
c               input
c
c                                    x,y : positions of rays at initial
c                                    z value
c
c                                    ck,cl : dx/dz,dy/dz for each ray
c
c                                    n : number of rays
c
c                                    w : ray weights
c
c               output
c
c                                    xloc,yloc : postion of best focus in
c                                    x-y plane
c
c                                    zloc : delta z to best focus from
c                                    initial z value
c
c                                    x,y : positions of rays at new z value
c
      xloc=0.d+00
      yloc=0.d+00
      zloc=0.d+00
c
c  cut out if there are no rays
c
      if(n.lt.1) go to 1000
      xav=0.d+00
      yav=0.d+00
      ckav=0.d+00
      clav=0.d+00
      sumw=0.d0
      do 10 i=1,n
      xav=xav+x(i)*w(i)
      yav=yav+y(i)*w(i)
      ckav=ckav+ck(i)*w(i)
      clav=clav+cl(i)*w(i)
      sumw=sumw+w(i)
   10 continue
c
c  cut out if weight sum is zero
c
      if(sumw.le.0.d0) go to 1000
      xav=xav/sumw
      yav=yav/sumw
      ckav=ckav/sumw
      clav=clav/sumw
```

```
      sum1=0.d+00
      sum2=0.d+00
      do 20 i=1,n
      con1=ck(i)-ckav
      con2=cl(i)-clav
      sum1=sum1+((x(i)-xav)*con1+(y(i)-yav)*con2)*w(i)
   20 sum2=sum2+(con1*con1+con2*con2)*w(i)
c
c  cut out if there is no solution
c
      if(sum2.eq.0.d0) go to 1000
      zloc=-1.d+00*sum1/sum2
      do 30 i=1,n
      x(i)=x(i)+ck(i)*zloc
      y(i)=y(i)+cl(i)*zloc
      xloc=xloc+x(i)*w(i)
   30 yloc=yloc+y(i)*w(i)
      xloc=xloc/sumw
      yloc=yloc/sumw
 1000 return
      end
      subroutine red(x,y,xcen,ycen,w,n,enc,rmax,ne,frac,rad,nf
     * ,wrmax,wtot)
c
c  calculate radial energy distribution
c
c  input
c
c          x,y          ray intercepts
c          xcen,ycen    assumed center of radial energy distribution
c          w            ray weights
c          n            number of rays (>=1)
c          rmax         maximum radius to calculate encircled energy
c          ne           number of radii for encircled energy
c                       calculation (>=1)
c          frac         fraction values for radii calculation
c                       (must be in increasing order)
c          nf           number of fraction values (>=1)
c
c  output
c
c          enc          ne encircled energy values up to rmax radius
c          rad          radius values for input fraction values
c        wrmax          weight sum up to rmax
c         wtot          total weight sum
c
      implicit double precision (a-h,o-z)
      dimension x(n),y(n),w(n),enc(ne),frac(nf),rad(nf)
c  constant
      pi=datan(1.d0)*4.d0
c
c   interval size
      dltr=rmax/dble(ne)
c
c   zero the accumulation array
c
      do 100 i=1,ne
      enc(i)=0.d0
```

```
  100 continue
c  zero the weight sums
      wtot=0.d0
      wrmax=0.d0
c
      do 200 i=1,n
      wtot=wtot+w(i)
      gr=dsqrt((x(i)-xcen)**2+(y(i)-ycen)**2)
      nzr=idint(gr/dltr)+1
      if(nzr.gt.ne) go to 200
      wrmax=wrmax+w(i)
      enc(nzr)=enc(nzr)+w(i)
  200 continue
c
      enc(1)=enc(1)/wtot
c
      if(ne.ge.2) then
      do 300 i=2,ne
      enc(i)=enc(i-1)+enc(i)/wtot
  300 continue
      endif
c
c determine radii for fraction values
c fraction values must be in increasing order
c
      do 400 i=1,nf
  400 rad(i)=0.d0
c
      m = 1
      do 50 np = 1,nf
c
      if (frac(np).le.enc(1)) then
      rad(np)=0.d0
      elseif (frac(np).le.enc(ne)) then
   30 m = m + 1
      if(m.gt.ne) go to 51
      if (enc(m).lt.frac(np)) go to 30
      rad(np) = dble(m-1)*dltr+dltr*(frac(np)-enc(m-1))/
     *    (enc(m)-enc(m-1))
      m = m - 1
      else
      rad(np)=-dltr
      endif
c
   50 continue
c
   51 continue
c
      return
      end
      subroutine rotate(xp,yp,ang,x,y)
c
      implicit double precision (a-h,o-z)
c
      c = dcos(ang)
      s = dsin(ang)
      x = xp * c - yp * s
      y = xp * s + yp * c
```

```
      return
      end
      subroutine splot (npts,f,x)
c
      implicit double precision (a-h,o-z)
      character * 4 char,blank,dash,vline,star
c
c on-line printer plot for spot diagram
c
      character * 4 char(120),iq
      dimension x(npts), f(npts)
c
      data   blank, dash, vline, star / ' ','-','I','*'/
c
      fmin=f(1)
      fmax=f(1)
      xmin=x(1)
      xmax=x(1)
      do 3   i = 1,npts
      fmin = dmin1(fmin,f(i))
      fmax = dmax1(fmax,f(i))
      xmin = dmin1(xmin,x(i))
    3 xmax = dmax1(xmax,x(i))
c
c  watch out for equal values
c
      if(fmin.eq.fmax) then
      if(xmin.eq.xmax) then
      xmin=xmin-1.d0
      xmax=xmax+1.d0
      fmin=fmin-2.d0
      fmax=fmax+2.d0
      else
      fmin=fmin-(xmax-xmin)
      fmax=fmax+(xmax-xmin)
      endif
      endif
c
c compute vertical scale (x-axis) in nice, round numbers
c
chen4 del = (fmax-fmin)/36.d0
    4 del = (fmax-fmin)/18.d0
      k = dlog10(del)
      if (del.lt.1.d0)    k = k-1
      q = 10.d0**k
      c = del/q
      ic = c
      cp = ic
      if (cp.lt.c)   cp = cp+1.d0
      dy = cp*q
chen  kcol = 96
      kcol = 48
      col  = kcol
      dx   = (xmax-xmin)/col
c
c adjust xmin, xmax, dx if x, y should be plotted on same scale
c
      fmid = (fmin+fmax)/2.d0
```

```
      dx = .6d0*dy
      templ =  1.0005d0 + (xmax-xmin)/dx
      if(dabs(templ).lt.1.d6) then
      l=templ
      else
      l=1000000
      endif
      if (l.le.kcol)   go to 6
      fmax = fmid + 2.d0*(fmax-fmid)
      fmin = fmid + 2.d0*(fmin-fmid)
      go to 4
    6 xmid = (xmin+xmax)/2.d0
      xmax = xmid + dx*(col/2.d0)
      xmin = xmid - dx*(col/2.d0)
c
c print graph
c
chen************
c pause
c
      write (*,*)
      write (*,*) 'Press <Enter> to continue ......'
      read(*,*)
chen*************
    8 write (6,9)
    9 format('0   x-axis'/ )
      df = dy
      kzero = 1.0005d0 - xmin/dx
      if (kzero.gt.kcol)   kzero = -1
      ic = fmid/dy
cwas  if (fmid.lt.0.d0)   ic = ic-1
chen  c = ic + 18
      c = ic + 10
      y = c*dy
      yp = y+df/1000.d0
cwas  do 20   j = 1,36
chen  do 20   j = 1,37
      do 20   j = 1,19
      y = y-df
      iq = blank
      if (dabs(y).lt.(df/3.d0))  y = 0.d0
      if (dabs(y).lt.(df/3.d0))   iq = dash
      do 10   i = 1,kcol
   10 char(i) = iq
c  keep within char array!
      if (kzero.gt.0.and.kzero.lt.121)   char(kzero) = vline
      do 15   i = 1,npts
      l = 1.0005d0 + (x(i)-xmin)/dx
c  stay within  char array!
      if(l.lt.1.or.l.gt.120) go to 15
      q = f(i)
      if ((q.ge.y).and.(q.lt.yp))   char(l) = star
   15 continue
      write (6,16) y, (char(i), i=1,kcol)
   16 format(e12.3,1x,120a1)
   20 yp = y
c
      xmid = (xmin+xmax)/2.d0
```

```
      write (6,27) xmin, xmid, xmax                                     57730000
chen27format(13x,'L',46x,'M',46x,'U'/e20.6,e46.6,e42.6/ 80x,'y-axis')  57740000
   27 format(13x,'L',23x,'M',23x,'U'/'y-axis',e15.6,e23.6,e23.6)       57740000
      return                                                            57750000
      end                                                               57760000
      subroutine stat(x,y,w,n,xav,yav,xrms,yrms,rms,wtot,wav,wrms
     * ,xmin,xmax,ymin,ymax,wmin,wmax)
c
c  input
c
c           x,y           ray intercepts
c           w             ray weights
c           n             number of rays(>=1)
c
c  output
c
c           xav,yav       x,y values at centroid
c           xrms,yrms     rms values of x and y about centroid
c           rms           rms value about centroid
c           wtot          sum of weights
c           wav           average value of weights
c           wrms          rms deviation of weights
c           xmin          minimum x value
c           xmax          maximum x value
c           ymin          minimum y value
c           ymax          maximum y value
c           wmin          minimum weight value
c           wmax          maximum weight value
c
c  calculate weighted spot average and rms
c
      implicit double precision (a-h,o-z)
      dimension x(n),y(n),w(n)
      xav=0.d0
      yav=0.d0
      xrms=0.d0
      yrms=0.d0
      rms=0.d0
      wtot=0.d0
      wav=0.d0
      wrms=0.d0
      xmin=0.d0
      xmax=0.d0
      ymin=0.d0
      ymax=0.d0
      wmin=0.d0
      wmax=0.d0
      if(n.lt.1) go to 1000
      xmin=x(1)
      xmax=x(1)
      ymin=y(1)
      ymax=y(1)
      wmin=w(1)
      wmax=w(1)
      sumw=0.d0
      sumx=0.d0
      sumy=0.d0
      sumxsq=0.d0
```

```
      sumysq=0.d0
      sumwsq=0.d0
      do 100 i=1,n
      sumx=sumx+x(i)*w(i)
      sumy=sumy+y(i)*w(i)
      sumxsq=sumxsq+x(i)*x(i)*w(i)
      sumysq=sumysq+y(i)*y(i)*w(i)
      sumw=sumw+w(i)
      sumwsq=sumwsq+w(i)*w(i)
      if(x(i).lt.xmin) xmin=x(i)
      if(y(i).lt.ymin) ymin=y(i)
      if(x(i).gt.xmax) xmax=x(i)
      if(y(i).gt.ymax) ymax=y(i)
      if(w(i).lt.wmin) wmin=w(i)
      if(w(i).gt.wmax) wmax=w(i)
  100 continue
      xav=sumx/sumw
      yav=sumy/sumw
c
      xrms=sumxsq/sumw-xav*xav
      if(xrms.ge.0.d0) then
      xrms=dsqrt(xrms)
      else
      xrms=0.d0
      endif
c
      yrms=sumysq/sumw-yav*yav
      if(yrms.ge.0.d0) then
      yrms=dsqrt(yrms)
      else
      yrms=0.d0
      endif
c
      rms=dsqrt(xrms*xrms+yrms*yrms)
      wtot=sumw
      wav=sumw/dble(n)
c
      wrms=sumwsq/dble(n)-wav*wav
      if(wrms.ge.0.d0) then
      wrms=dsqrt(wrms)
      else
      wrms=0.d0
      endif
c
 1000 continue
      return
      end
      subroutine xalign(tr,tf,iener)
      implicit double precision (a-h,o-z)                              327800
c*********************************************************************
      common /rsavel/ xpsv(200000),ypsv(200000),dxdzsv(200000)
     * ,dydzsv(200000),entx(200000),enty(200000),wtsv(15,200000)
     * ,zshift,nsv
c*********************************************************************
      namelist/test/x,y,iq,ang                                         327811
      common /qxal/ qav(2,4),wsum(4),nqsum(4)
c************************************************************************
                                                                       327900
c
```

```
c  calculate x-ray alignment correction for hyperbola tilt          32800000
c  and for defocus                                                  32801000
c                                                                   32810000
c***********************************************************************
      do 10 j=1,4                                                   32820200
      nqsum(j)=0
      wsum(j)=0.d0
      do 10 i=1,2                                                   32820400
   10 qav(i,j)=0.d0                                                 32820500
      pi=datan(1.d0)*4.d0                                           32821000
c***********************************************************************
c  r is hyperbola radius hit for central ray (estimate)             32830200
c  f is axial distance to image from central ray hit on             32830300
c    hyperbola to image (estimate)                                  32830400
      r=tr                                                          32830500
      f=tf                                                          32830600
c***********************************************************************
c
c  cycle through the stored rays
c
c determine aperture quadrant                                       32831100
c (approximate with entrance aperture position)
      do 100 i=1,nsv
      x=entx(i)
      y=enty(i)
      iq=-99                                                        32833100
      if(x.eq.0.d0.and.y.eq.0.d0) go to 1000                        32833200
      ang=datan2(y,x)-pi/4.d0                                       32833300
c     write(6,test)                                                 32833400
      if(ang.lt.0.d0) ang=ang+2.d0*pi                               32833500
      iq=ang/(pi/2.d0)                                              32833600
      iq=iq-(iq/4)*4+1                                              32833700
c     write(6,test)                                                 32833800
 1000 continue                                                      32833900
c accumulate quadrant average                                       32841000
      if(iq.lt.0.or.iq.gt.4) go to 2000                             32842000
      qav(1,iq)=qav(1,iq)+xpsv(i)*wtsv(iener,i)                     32845400
      qav(2,iq)=qav(2,iq)+ypsv(i)*wtsv(iener,i)                     32845500
      nqsum(iq)=nqsum(iq)+1                                         32845600
      wsum(iq)=wsum(iq)+wtsv(iener,i)
 2000 continue                                                      32845700
c
100   continue
c***********************************************************************
c
c  calculate averages                                               32861000
      nqtot=0                                                       32861100
      wtot=0.d0
      do 21 i=1,4                                                   32862000
      if(nqsum(i).le.0) go to 3000
      nqtot=nqtot+nqsum(i)                                          32862200
      if(wsum(i).le.0.d0) go to 3000
      wtot=wtot+wsum(i)
   21 continue                                                      32862300
      do 20 j=1,4                                                   32862400
      do 20 i=1,2                                                   32862600
   20 qav(i,j)=qav(i,j)/wsum(j)                                     32862700
c***********************************************************************
```

```
c  calculate axial focus position error
      dfx=f/r*pi/4.d0/dsqrt(2.d0)*(qav(1,2)-qav(1,4))
      dfy=-f/r*pi/4.d0/dsqrt(2.d0)*(qav(2,1)-qav(2,3))
      df=(dfx+dfy)/2.d0
      diamf=2.d0*r/f*dsqrt((dfx**2+dfy**2)/2.d0)
c**********************************************************************
c  calculate tilt error estimate
c  (right handed rotation about y or x)
      thy=1.d0*pi/8.d0/f*(qav(1,2)+qav(1,4)-qav(1,1)-qav(1,3))
      thx=-1.d0*pi/8.d0/f*(qav(2,1)+qav(2,3)-qav(2,2)-qav(2,4))
      diami=2.d0*f*dsqrt(thx**2+thy**2)
      diamis=diami/f*180.d0/pi*3600.d0


      thys=thy*180.d0/pi*3600.d0
      thxs=thx*180.d0/pi*3600.d0
c**********************************************************************
c  print out the results
      write(6,31) nqtot,(i,nqsum(i),i=1,4),wtot,(i,wsum(i),i=1,4)
     *           ,dfx,dfy,df,diamf,thx,thy,diami,thxs,thys,diamis
   31 format('0 x-ray alignment,     total points ',i8/
     * ' quadrant ',i1,' points ',i8/
     * ' quadrant ',i1,' points ',i8/
     * ' quadrant ',i1,' points ',i8/
     * ' quadrant ',i1,' points ',i8/
     * '                       total weight ',e22.15/
     * ' quadrant ',i1,' weight ',e22.15/
     * ' quadrant ',i1,' weight ',e22.15/
     * ' quadrant ',i1,' weight ',e22.15/
     * ' quadrant ',i1,' weight ',e22.15/
     * ' focus error (x)      = ',e22.15/
     * ' focus error (y)      = ',e22.15/
     * ' focus error (av)     = ',e22.15/
     * ' focus diameter       = ',e22.15/
     * ' tiltx error (rad)    = ',e22.15/
     * ' tilty error (rad)    = ',e22.15/
     * ' tilt diameter        = ',e22.15/
     * ' tiltx error (sec)    = ',e22.15/
     * ' tilty error (sec)    = ',e22.15/
     * ' tilt diameter (sec)  = ',e22.15)
      write(6,32)    (j,(qav(i,j),i=1,2),j=1,4)
   32 format(' quadrant ',i1,',     xav= ',e22.15,',     yav=',e22.15)
      write(6,33) r,f
   33 format(' r value ',e22.15/
     *       ' f value ',e22.15)
      return
c**********************************************************************
 3000 continue
      write(6,34) (i,nqsum(i),i=1,4),(i,wsum(i),i=1,4)
   34 format('0 x-ray alignment error'/
     *       ' quadrant ',i1,' points ',i8/
     *       ' quadrant ',i1,' points ',i8/
     *       ' quadrant ',i1,' points ',i8/
     *       ' quadrant ',i1,' points ',i8/
     *       ' quadrant ',i1,' weight ',e22.15/
     *       ' quadrant ',i1,' weight ',e22.15/
     *       ' quadrant ',i1,' weight ',e22.15/
     *       ' quadrant ',i1,' weight ',e22.15)
```

```
      return
      end
      subroutine find(x,n,frac,imin)
      implicit double precision (a-h,o-z)
      dimension frac(n)
c************************************************************************
c
c find imin such that frac(imin) lt x lt frac(imin+1)
c nondecreasing frac array
      if(x.lt.frac(1)) then
           imin=0
      elseif(x.gt.frac(n))then
           imin=n
      else
c         keep picking half the array until
c         there is unit difference
           n1=1
           n2=n
  100      continue
        if((n2-n1).eq.1) go to 200
        nmid=(n1+n2)/2
        if(x.gt.frac(nmid)) then
             n1=nmid
        else
             n2=nmid
        endif
        go to 100
  200      continue
           imin=n1
      endif
c************************************************************************
      return
      end
czzzzzzzzz     /hdez206.utils.fort(vabs)
      function vabs(v)
      implicit double precision(a-h,o-z)
      dimension v(3)
      vabs=dsqrt(v(1)*v(1)+v(2)*v(2)+v(3)*v(3))
      return
      end
czzzzzzzzz     /hdez206.utils.fort(vcross)
      subroutine vcross(a,b,c)
      implicit double precision(a-h,o-z)
      dimension a(3),b(3),c(3),d(3)
      d(1)=a(2)*b(3)-a(3)*b(2)
      d(2)=a(3)*b(1)-a(1)*b(3)
      d(3)=a(1)*b(2)-a(2)*b(1)
      c(1)=d(1)
      c(2)=d(2)
      c(3)=d(3)
      return
      end
czzzzzzzzz     /hdez206.utils.fort(vdiff)
      subroutine vdiff(a,b,c)
      implicit double precision (a-h,o-z)
      dimension a(3),b(3),c(3)
      do 100 i=1,3
  100 c(i)=a(i)-b(i)
```

```
      return
      end
czzzzzzzzz     /hdez206.utils.fort(vdot)
      function vdot(a,b)
      implicit double precision(a-h,o-z)
      dimension a(3),b(3)
      vdot=a(1)*b(1)+a(2)*b(2)+a(3)*b(3)
      return
      end
czzzzzzzzz     /hdez206.utils.fort(vprod)
      subroutine vprod(fac,a,c)
      implicit double precision (a-h,o-z)
      dimension a(3),c(3)
      do 100 i=1,3
  100 c(i)=fac*a(i)
      return
      end
czzzzzzzzz     /hdez206.utils.fort(vsum)
      subroutine vsum(a,b,c)
      implicit double precision (a-h,o-z)
      dimension a(3),b(3),c(3)
      do 100 i=1,3
  100 c(i)=a(i)+b(i)
      return
      end
czzzzzzzzz     /hdez206.utils.fort(vunit)
      subroutine vunit(vin,vout)
      implicit double precision(a-h,o-z)
      dimension vin(3),vout(3)
      abs=dsqrt(vin(1)*vin(1)+vin(2)*vin(2)+vin(3)*vin(3))
      if(abs.le.0.) abs=1.d+00
      do 100 i=1,3
  100 vout(i)=vin(i)/abs
      return
      end
      subroutine deltbl(energy,delta,beta)
      implicit double precision (a-h,o-z)
      common/rfldat/wv(501),dlt(501),bt(501),ndt,ilabel
      character * 75 ilabel
      hc=12.399d0
c              convert energy in kev to wavelength in angstroms
      x=hc/energy
c              find closest wavelength in table
      nn=ndt-1
      do 100 i=1,nn
      if(x.lt.wv(i).or.x.gt.wv(i+1)) go to 100
      imin=i
      go to 200
  100 continue
      go to 2000
  200 continue
      diff1=dabs(x-wv(imin))
      diff2=dabs(x-wv(imin+1))
      if(diff2.lt.diff1) imin=imin+1
      if(imin.eq.1) imin=imin+1
      if(imin.eq.ndt) imin=imin-1
c
c              interpolate for delta and beta
```

```
c
c             (quadratic interpolation here)
c
      x1=wv(imin-1)
      x2=wv(imin)
      x3=wv(imin+1)
      y1=dlt(imin-1)
      y2=dlt(imin)
      y3=dlt(imin+1)
      delta=yintp(x,x1,x2,x3,y1,y2,y3)
      y1=bt(imin-1)
      y2=bt(imin)
      y3=bt(imin+1)
      beta=yintp(x,x1,x2,x3,y1,y2,y3)
      return
 2000 continue
      write(6,2001) energy,x
 2001 format('0energy value out of range in deltb1'/
     *       1x,'energy(kev)= ',e11.4,
     *       ',wavelength(angstroms)= ',e11.4)
      write(6,2002)
 2002 format(' ******* warning ******')
      return
      end
      subroutine metref(anginc,delta,beta,rs,rp)
c
      implicit double precision(a-h,o-z)
c
c     input values:
c        anginc     incident angle in radians
c        delta,beta reflectivity data
c
c     output values:
c        rs         reflectivity for parallel polarization
c        rp         reflectivity for perpendicular polarization
c
c      routine to calculate reflectivity as a function of incident
c      angle and complex index of refraction for metals.
c      reflectivity here is the ratio of reflected intensity to
c      incident intensity (not the ratio of amplitudes)
c
c      references:
c
c         1.  zombeck, m. v., advanced x-ray astrophysics facility
c         (axaf) interim report optical constants and reflectivities fo
c         nickel, gold, and platinum in the x-ray region of the spectru
c         (0.1 - 10 kev), report no. sao-axaf-83-016, smithsonian
c         astrophysical observatory, cambridge, ma., march 1983.
c
c         2.  vanspeybrock, l., optical constants, private
c         communication, april 20, 1987.
c
c         3.  born, m., wolf, e., principles of optics, (pergamon press
c         6th ed., oxford, 1980)
c
c
c             calculate reflectivities
      fr = 1.d0 - delta
```

```
      fi = -beta
c
      si = dsin(anginc)
      ci = dcos(anginc)
      ti = dtan(anginc)
c
      frsq=1.d0-2.d0*delta+delta*delta
      fisq=fi*fi
      sisq=si*si
      cisq=ci*ci
      tisq=ti*ti
c
      bbb=delta*delta-2.d0*delta-fisq+cisq
      aaa = dsqrt(bbb*bbb+4.d0*frsq*fisq)
      asq = 0.5d0*(aaa+bbb)
      a = dsqrt(asq)
      rs=2.d0*a*ci
      rs=(aaa-rs+cisq)/(aaa+rs+cisq)
      siti=si*ti
      sitisq=siti*siti
      tasiti=2.d0*a*siti
      rp=aaa+sitisq
      rp=rs*(rp-tasiti)/(rp+tasiti)
c
      return
      end
      subroutine rdref(iu,jrefr)
c
c  read reflectivity data from file in jrefr using unit iu
c
      implicit double precision (a-h,o-z)
      character * 80 ihead,jrefr
      common/rfldat/wv(501),dlt(501),bt(501),ndt,ilabel
      character * 75 ilabel
c              read complex indices of refraction
      open(unit=iu,file=jrefr)
      read(iu,101) ndt,ilabel
      if(ndt.gt.501) ndt=501
  101 format(i5,a75)
      read(iu,102) ihead
  102 format(a80)
      do 200 j=1,ndt
      read(iu,103) wv(j),dlt(j),bt(j)
  103 format(e11.5,e13.7,e13.7)
  200 continue
      close(iu)
      return
      end
      subroutine calcdb(energy,nnrg,delbet)
c
c  calculate delta,beta at nnrg energy values
c
      implicit double precision (a-h,o-z)
      dimension energy(nnrg),delbet(2,nnrg)
      common/rfldat/wv(501),dlt(501),bt(501),ndt,ilabel
      character * 75 ilabel
      do 100 i=1,nnrg
      call deltb1(energy(i),delbet(1,i),delbet(2,i))
```

```
  100 continue
      return
      end
      function yintp(x,x1,x2,x3,y1,y2,y3)
      implicit double precision (a-h,o-z)
c             quadratic interpolation
c
c        reference:
c
c        bevington, r.p., data reduction and error analysis
c        for the physical sciences, (mcgraw-hill,new york,1969),
c        p. 264.
c
      d=(x-x1)/(x2-x1)
      d2=1.d0
      d3=(x3-x1)/(x2-x1)
      a1=y1
      a2=(y2-a1)/d2
      a3=(y3-a2*d3-a1)/d3/(d3-d2)
      yintp=a1+a2*d+a3*d*(d-d2)
      return
      end
      subroutine axcir(sp,uv,asig,csig,irand)
      implicit double precision (a-h,o-z)
      dimension sp(3),uv(2)
c
c  add random bivariate gaussian distribution
c  in the x,y plane to ray intercept sp
c
c  asig and csig are the gaussian sigma values
c  in the axial and circumferential directions
c  respectively (in the x,y plane)
c
c  axial direction is along direction
c  of unit vector uv
c
c  circumferential direction is perpendicular
c  to axial direction
c
      if(asig.ne.0.d0.or.csig.ne.0.d0) then
c
      call granf(irand,w1,w2)
c
c        simulate axial slope errors first
c
      if(asig.ne.0.d0) then
          ds=w1*asig
          sp(1)=sp(1)+ds*uv(1)
          sp(2)=sp(2)+ds*uv(2)
      endif
c
c  simulate circumferential slope errors here.
c
      if(csig.ne.0.d0) then
          ds=w2*csig
          sp(1)=sp(1)-uv(2)*ds
          sp(2)=sp(2)+uv(1)*ds
      endif
```

```
c
      endif
c
      return
      end
      subroutine eescat(sp,ft,uv,n,frac,s,irand)
c**********************************************************************
c   scatter ray intercept sp() in x,y plane along line with
c   scatter direction vector uv. use integrated
c   angular scattering probability array frac() of dimension n.
c   the n corresponding angular displacements are in array s().
c   the assumed effective focal length is ft.
c
c**********************************************************************
c
c  input:
c
c            sp     input ray intercept (x,y,z)
c            ft     assumed focal length
c            uv     unit vector in scatter direction
c                   (scatter in x,y plane)
c             n     number of integrated probability fractions
c                   (assumed ge 2)
c          frac     array of integrated probability fractions
c                   (assumed ge 0 and in nondecreasing sequence)
c             s     angular displacements corresponding to fractions
c                   (radians, increasing sequence, assumed small so
c                    that s=sin(s)=tan(s) is valid)
c         irand     integer parameter for random generator
c                   ranf.  it must be initialized with ranset
c                   outside of this routine.
c                   then it is modified with each use of ranf.
c
c  output
c
c            sp     ray intercept after scattering in x,y plane
c
c**********************************************************************
      implicit double precision (a-h,o-z)
      dimension frac(n),s(n),sp(3),uv(2)
c**********************************************************************
c     namelist /out1/ x,n1,n2,imin
c     namelist /out2/ y,x1,x2,y1,y2
c**********************************************************************
c
c pick random number on (0,1)
c
  300 x=ranf(irand)
c
c**********************************************************************
c
c find imin such that frac(imin) lt x lt frac(imin+1)
c nondecreasing frac array
      if(x.lt.frac(1)) then
          imin=0
      elseif(x.gt.frac(n))then
          imin=n
      else
```

```
c        keep picking half the array until
c        there is unit difference
           n1=1
           n2=n
  100      continue
        if((n2-n1).eq.1) go to 200
        nmid=(n1+n2)/2
        if(x.gt.frac(nmid)) then
             n1=nmid
        else
             n2=nmid
        endif
        go to 100
  200      continue
        if(frac(n1).eq.frac(n2)) then
                imin=-n1
        else
                imin=n1
           endif
      endif
c*********************************************************************
c
c  debug print
c     write(6,out1)
c
c*********************************************************************
c
c watch out for imin le 0 or imin=n
c (imin is lt 0 if x equals both of the frac values, skip
c  these cases to be certain of having no points in
c  regions of zero probability)
c (imin=0 or n means x is outside of frac value range)
c
      if(imin.gt.0.and.imin.lt.n) then
           x1=frac(imin)
           x2=frac(imin+1)
           y1=s(imin)
           y2=s(imin+1)
      else
           go to 300
      endif
c
c*********************************************************************
c
c  linear interpolation of frac array
c
      y=y1+(x-x1)/(x2-x1)*(y2-y1)
      y=y*ft
      sp(1)=sp(1)+uv(1)*y
      sp(2)=sp(2)+uv(2)*y
c*********************************************************************
c
c  debug print
c     write(6,out2)
c
c*********************************************************************
      return
      end
```

```
      subroutine granf(iy,x1,x2)
c  routine to produce gaussian distribution n(0,1)
      implicit double precision (a-h,o-z)
 100  v1=2.d0*ranf(iy)-1.d0
      v2=2.d0*ranf(iy)-1.d0
      s=v1*v1+v2*v2
      if(s.gt.1.d0) go to 100
      if(s.eq.0.d0) go to 100
      x1=v1*dsqrt(-2.d0*dlog(s)/s)
      x2=v2*dsqrt(-2.d0*dlog(s)/s)
      return
      entry grset(iy)
      call ranset(iy)
      return
      end
      subroutine disc(sp,diam,irand,pi)
      implicit double precision (a-h,o-z)
      dimension sp(3)
c
c  add random errors on disc of diameter diam
c  in x,y plane to ray intercept sp
c
      if(diam.ne.0.d0) then
           radius=diam/2.d0*dsqrt(ranf(irand))
           theta=2.d0*pi*ranf(irand)
           sp(1)=sp(1)+radius*dcos(theta)
           sp(2)=sp(2)+radius*dsin(theta)
      endif
      return
      end
      function ranf(id)
c
c     uniform distribution random number generator.
c     period is approximately 2 billion numbers.
c     however the least significant bits are less random.
c
c     this routine generates random numbers in the
c     range (0.,1.).  the process can be reinitialized
c     by calling ranset(iy) where iy is an arbitrary
c     integer.  there after the function 'ranf' may
c     be used.  normally the variable iy should not be
c     altered between calls to 'ranf'.  the random
c     number seed iy may be reset when desired by
c     calling ranset(iy).
c
      implicit double precision (a-h,o-z)
c
      common /comran/ s,iy,ia,ic,mic,m2
c
c     following data is for honeywell 560 computer
c     data s,iy,ia,ic,mic,m2 /z3920000000000000,
c    $ z12b9b0a1,z3243f6ad,z1b0cb175,z64f34e8b,z40000000/
c
      iy=iy*ia
      if (iy.gt.mic) iy=(iy-m2)-m2
      iy=iy+ic
      if (iy.lt.0) iy=(iy+m2)+m2
      ranf=dble(iy)*s
```

```
      return
      entry ranset(id)
      iy=id
      m=1
10    m2=m
      m=2*m2
      if (m.gt.m2) go to 10
      halfm=m2
      ia=8*dint(halfm*datan(1.d0)/8.d0)+5
      ic=2*dint(halfm*(0.5d0-dsqrt(3.d0)/6.d0))+1
      mic=(m2-ic)+m2
      s=.5d0/halfm
      if (iy.eq.0) iy=314159265
      return
      end
      subroutine rect(sp,xlenth,ylenth,uv,irand)
      implicit double precision (a-h,o-z)
      dimension sp(3),uv(2)
c
c  add uniform random errors on rectangle of widths xlenth,ylenth
c  in x,y plane to ray intercept sp
c
c  rotate rectangle the same way that
c  x axis would be rotated ccw toward
c  unit vector uv
c
      if(xlenth.ne.0.d0.or.ylenth.ne.0.d0) then
c
c
      if(xlenth.eq.0.d0) then
          dx=0.d0
      else
          dx=(ranf(irand)-0.5d0)*xlenth
      endif
c
c
      if(ylenth.eq.0.d0) then
          dy=0.d0
      else
          dy=(ranf(irand)-0.5d0)*ylenth
      endif
c
c
      if(uv(1).ne.1.d0) call turn(dx,dy,uv)
c
      sp(1)=sp(1)+dx


      sp(2)=sp(2)+dy
c
      endif
c
      return
      end
      subroutine turn(x,y,uv)
c
c  rotate the object point in
c  the x,y plane in the same
```

```
c  manner that the x axis
c  must be rotated ccw to
c  the unit vector uv
c
      implicit double precision (a-h,o-z)
      dimension uv(2)
      xt=x
      yt=y
      x=xt*uv(1)-yt*uv(2)
      y=xt*uv(2)+yt*uv(1)
      return
      end
c
c***********************************************************************
c
c  SAMPLE PRINT OUT FOLLOWS
c
c***********************************************************************
c
```

A5.4 gt2hlp.doc Help Document (ASCII text)

```
****************************************************************
*
*     Help document for GRAZTRACE command mode
*
*    Entries begin with command mnumornic and end with 'See'
*    To prevent mismatch, command mnumornic must have two leading
*    blank lines
*    In the description, Do not let the line begin with command
*    mnumornic have such two leading blank lines.
*
*****************************************************************


    LEN

         Declares that the following entrees are for a new
      system, rather than a modification to the old.
         Initializes defaults for a new system.  ALl old system
         data are destroyed.  Len is not necessary prior to
      restoring a lens from the file.

    See also: RES.


    ADA surf_num adata_num adata

         Input surface error

         surf_num - surface number
         adata_num - surface error number
         adata - surface error

    See also: SDA.


    AZI azim

         Set source azimuth angle

    See aslo: DAZ, ELE.


    APE surf_num iaper

         Declare surface frame type

         surf_num - surface number
         iaper - character string(*80)

    See also: DXC, DYC, DXR, DYR.


    DAZ delaz
```

```
Set azimuth range

See also: AZI.


DEB delb_num iener surf_num delb_val

     Input surface reflectivity data (alpha, beta)

     delb_num - reflectivity number, 1 for alpha
                                     2 for beta
     iener - energy level
     surf_num - surface number
     delb_val - delbet value

See also: IND.


DIS dec_num surf_num dec_value

     Set displacement data

                                1 for Z dec.
     dec_num - decenter number, 2 for Y dec.
                                3 for X dec.
     surf_num - surface number
     dec_val - decenter value

See also: MOV, TIL.


DXC surf_num radius_x

     Set obscuration radius X

     surf_num - surface number
     radius_x - radius X

See also: DYC, DXR, DYR, OBS.


DYC surf_num radius_y

     Set obscuration radius Y

     surf_num - surface number
     radius_y - radius Y

See also: DXC, DXR, DYR, OBS.


DXR surf_num rect_x

     Set obscuration width X

     surf_num - surface number
     rect_x - width X
```

See also: DXC, DYC, DYR, THR, OBS.

DYR surf_num rect_y

Set obscuration height Y

surf_num - surface number
rect_y - height Y

See also: DXC, DYC, DXR, THR, OBS.

ELE elev

Set source elevation

elev - source elevation angle

See also: AZI, DAZ.

ENE iener ener_val

Set energy levels

iener - energy level number
ener_val - energy level value

See also: NRG.

FDF surf_num ifdfm

Define deformation file name

surf_num - surface number
ifdfm - deformation file name

See also: TYP.

FOC foclen

Check or overwrite focal langth

foclen - system focal length

See also: FCS.

IND surf_number findex

Input surface index

surf_num - surface number
findex - surface index

```
See also: THI.


ITI itilt

     Define tilt sequence

     surf_num - surface number
     itilt - surface tilt sequence (e.g. 123  for 1, 2, 3)

See also: TIL, MOV.


MOD imode

     Define surface ray trace mode

     imode - surface ray trace mode

See also: TYP.


MAT i j k rmat(i,j,k)

   Set surface displacement transformation matrix

   i, j, k - matrix indics
   rmat(i,j,k) - matrix values

See also: MOV.


MOV suf_num imove

     Set surface tilt flag

     surf_num - surface number
     imove - surface tilt flag, 1 for tilt
                                0 for not tilt

See also: TIL, ITI, DIS.


NRG nnrg

     Declare total energy level number

     nnrg - total energy level number

See also: ENE.


OBS surf_num iobs

     Define surface obscuration type

     surf_num - surface number
```

```
    iobs - surface obscuration type

See also: DXC, DYC, DXR, DYR.


RLI surf_num radlim_num radlim_val

     Set minimum and maximum radii of the surface

     surf_num - surface number
     radlim_num - radii number, 1 for minimum radius
                                2 for maximum radius
     radlim_val - radius value

See also: SDA.


RST surf_num irstr

  Set surface restore flag

  surface_num - surface number
  irstr - surface rstore flag, 0 for not restore
                     1 for restore

See also: MOV.


SDA surf_num sdata_num sdata

     Input surface data

     surf_num - surface number
     sdata_num - surface data number
     sdata - surface data

See also: THI, RLI.


SOU source_num source_pos

     Define source position relative to undisplaced center
     of the  first surface
                                          1 for X
     source_num - source position number, 2 for Y
                                          3 for Z
     source_pos - source position value

See also: ZRA.


SUR nsurf

     Define total numberr of the surfaces

     nsirf - total number of the surfaces.
```

See also: TYP.

TIL surf_num tilt_num tilt_val

Input surface tilt data

surf_num - surface number
tilt_num - tilt number
tilt_val - surface tilt value

See also: ITI, MOV, DIS.

TYP surf_num itype

Define surface type

surf_num - surface number
itype - surface type

See also: SUR.

TIT surface_num ihead

Set surface description

surface_num - surface number
ihead - surface head information

See also: TYP.

THR surf_num threct

Set angle of obscuration rectangle

surf_num - surface number
threct - angle of obscuration rectangle

See also: DXR, DYR, OBS.

THI surf_num thick

Input surface separation

surf_num surface number
thick - shrface separation

See also: SDA.

WGT surf_num iwgt

Set surface reflectivity weight flag

surf_num - surface number
iwgt - surface reflectivity weight flag

See also: DEB.

XWI surf_num xwidth

Input rectangular aperture width x

surf_num surrface number
xwidth - aperture width x

See also: YWI.

YWI surf_num ywidth

Input rectangular aperture height y

surf_num - surface number
ywidth - aperture heoght y

See also: XWI.

ZRA range

Set source distance to the first surface

zrange - source distance

See also: SOU.

SAV filspec

Save current system to prescription file

filspec - file name

See also: RES, LIS.

RES filspec

Restore system from prescription file

filspec - file name

See also: SAV, LIS.

LIS

List all system data

See also: RES, SAV.

WSP

Randmo ray trace

WSP traces nra successfull rays randomly arranged on the first surface annulus at location Z=0. Intercept, slopes, and effective area weights are stored for the last surface for each ray.

Options:

```
AZM azimus_middle_angle, (default is 0)
DAZ delta_azimus_angle, (default 2 pi)
NRA number_of_rays, (default 1000)
```

GO for excuting the analysis,
CAN for cancelling the analysis.

See also: WS2, GRI, GR2, RSV.

WS2

Modified wheel spoke ray trace

WS2 traces wheel spoke rays arranged on the first surface annulus at location Z=0. Intercept, slopes, and effective area weights are stored for the last surface for each ray.

Options:

```
AZM azimus_middle_angle, (default is 0)
DAZ delta_azimus_angle, (default 2 pi)
NLO radial_points, (default 100)
NAZ azimuthal_points. (default 72)
```

GO for excuting the analysis,
CAN for cancelling the analysis.

See also: WSP, GRI, GR2, RSV.

GRI

Trace rays on a modified grid

WGI traces rays on a grid with constant redial and varying azimuthal increments on the first surface annulus at location Z=0. Intercept, slopes, and effective area weights are stored for the last surface for each ray. Ray weights are set to 1.

Options:

```
    AZM azimus_middle_angle, (default is 0)
    DAZ delta_azimus_angle, (default 2 pi)
    NLO radial_points, (default 100)
    NAZ azimuthal_points. (default 72)

    GO for excuting the analysis,
    CAN for cancelling the analysis.

See also: WSP, WS2, GR2, RSV.


GR2

    Trace rays on a grid

    GR2 traces rays on a grid with constant radial and
    azimuthal increments on the first surface annulus at
    location Z=0.
    Intercept, slopes, and effective area weights are
    stored for the last surface for each ray.
    Ray weights are set to 1.

    Options:

    AZM azimus_middle_angle, (default is 0)
    DAZ delta_azimus_angle, (default 2 pi)
    NLO radial_points, (default 100)
    NAZ azimuthal_points. (default 72)

    GO  for excuting the analysis,
    CAN for cancelling the analysis.

See also: WSP, WS2, GRI, RSV.


WST

    Calculate average position and rms

  WST calculates average position and rms of stored rays at
  specified energy level.

    Options:

  IEN energy_level(default is 1)

    GO  for excuting the analysis,
    CAN for cancelling the analysis.

See also: SPO.


RSV filspec

    Save ray data as well as system data to a file

    RSV saves all the ray data as well as system data to a
    file.
```

See also: WSP, WS2, GRI, GR2.

FCS

Refocus

FCS refiocuses the system which relocate the evaluation plane to the best location.

Option:

IEN energy_level. (default is 1)

GO for excuting the refocusing
CAN for cancellin the refocusing

See also: FOC.

SPO.

Unweighted spot diagram

SPO generates plots of ray interceptions with the image surface to represent image characteristics.

Options:

XCE center_of_x, (default is current average X)
YCE center_of_y, (default is current average Y)
NRA number_of_rays, (default is 1000)

GO for execting the spot diagram plot
CAN for candelling the spot diagram.

See also: RAD.

RAD

Encircled energy

RAD computes the radial energy distribution - the diameters in the image within which fixed percentages of light energy are contained.

Options:

AMA angle_in_arc_sec, (default is 2.0)
IEN energy_level, (default is 1)
NFR number_of_fractions, (default is 20)
NRA number_of_rays, (default is 500)
XCE center_of_x, (default is current average X)
YCE center_of_y, (default is current average Y)

GO for executing the analysis,
CAN for cancelling the analysys.

See also: SPO.

AZM azimuth_middle_angle

Set azimuth middle point

See also: WSP, WS2, GRI, GR2.

NLO radial_points

Set number of rays along the radius

See also: WS2, GRI, GR2.

NAZ azimuthal_points

Set number orf rays around the annuals

See also: WS2, GRI, GR2.

IEN energy_level

Cancel the default set and set desired level for the analysis.

See also: FCS, WST, RAD.

XCE center_of_x

Override center coordinate X
Default is current average X.

See also: SPO, RAD

YCE center_of_y

Override center coordinate Y
Default is current average Y.

See also: SPO, RAD

NRA number_of_ray

Set desired ray number fro the analysis.

See also: WSP, WS2, GRI, GR2, SPO, RAD.

AMA angle

Cancel the default set and set desired angle.

See also: RAD.

NFR number_of_fractions

Cancel the default set and set desired number.

See also: RAD.

?

Help and inquiry

? only serves as help command,
? in data field entry will allow to check current value,

See also HEL.

HEL

Help

Help only will automatically provide the imformation about latest command entered before help.

Help followed by a command will provide imformation about that command.

Help followed by any unknown command will list all GRAZTRACE commands.

See also: "?".

GO

Excution option

GO excutes the analysis using all previously entered option inputs and then return control to the command level.

See also: CAN.

CAN

Cancel option

CAN cancens all inputs to the analysis and return control to the command level.

See also: GO.

EXI

```
     Exiting the program

     EXI exits the GRAZTRACE to the operation system.
     When EXI is typed in, a query is ieeued requiring a
     Yes or No answer(Y or N); a Y will cancel any
     option you are in and complete the exit.

See also: CAN.


DET delta

     Set ray intercept convergence criterion

     delta - convergence criterion.

See also: MAX.


MAX kmax

     Set maximum iteration loops for ray intercept

     kmax - maximum iteration loops.

See also: DET


PRI surf_num kprint

     Set surface ray print flag array

     surf_num - surface number
     kprint - print flag

See also: PRI.


EFF

     Check effective area accumulation

See also: ERR, VIG, PAS.


ERR

     Check number of failure rays

See also: EFF, VIG, PAS.


VIG

     Check number of vignetted rays

See also: EFF, ERR, PAS.
```

```
PAS

     Check number of successful rays

See also: EFF, ERR, VIG.


SYS op_sys_command

     Operation system shell

See also: EDI.


EDI

     UNIX editor to edit prescription.

See also: SYS.


Unknown command

                    GRAZTRACE commands list

ADA       AMA       APE       AZI       AZM       CAN       DAZ
DEB       DET       DIS       DXC       DXR       DYC       DYR
EDI       EFF       ELE       ENE       ERR       EXI       FCS
FDF       FOC       GO        GRI       GR2       HEL       IEN
IND       ITI       LEN       LIS       MAT       MAX       MOD
MOV       NAZ       NFR       NLO       NRA       NRG       OBS
PAS       PRI       RAD       RES       RLI       RST       RSV
SAV       SDA       SOU       SPO       SUR       SYS       THI
THR       TIL       TIT       TYP       VIG       WGT       WSP
WST       WS2       XCE       XWI       YCE       YWI       ZRA
?

See manual or Type HELp for further information
```

A5.5 sample.pre Sample Prescription (Text with FORTRAN name list format)

```
&inp  zrange=    1.000000000000D+50, elev=  0., azim=  0., foclen=
   656.48323128448, source=  0.  0.   -1.000000000000D+50, radlim=
   75.050255499563    76.401708618033  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.
 0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.
 0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.
 0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.
 0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.
 0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0., dxcirc=  0.  0.  0.  0.  0.
 0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.
 0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.
 0.  0.  0.  0.  0.  0.  0., dycirc=  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.
 0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.
 0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.
 0.  0., xwidth=  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.
 0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.
 0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0., ywidth=  0.
 0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.
 0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.
 0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0., dxrect=  0.  0.  0.  0.  0.  0.
 0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.
 0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.
 0.  0.  0.  0.  0.  0., dyrect=  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.
 0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.
 0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.
 0., threct=  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.
 0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.
 0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0., zlim=  0.  0.
 0.  0.  0.  0.  0.  0.   -23.750000000000    23.750000000000  0.  0.  0.  0.
 0.  0.  0.  0.  0.  0.   -23.750000000000    23.750000000000  0.  0.  0.  0.
 0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.
 0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.
 0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.
 0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0., adata=
 0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.
 0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.
 0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.
 0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.
 0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.
 0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.
 0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.
 0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.
 0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.
 0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.
 0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.
 0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.
 0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.
 0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.
 0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.
 0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.
 0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.
 0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.
 0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.
 0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.
 0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.
```

```
0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.
0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.
0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.
0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.
0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.
0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.
0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.
0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.
0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.
0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.
0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.
0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.
0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.
0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.
0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.
0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.
0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.
0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.
0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.
0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.
0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.
0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.
0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.
0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0
0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0
0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0
0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0
0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0
0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0
0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0
0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0
0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0
0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0
0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0
0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0
0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0
0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0
0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0
0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0
0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0
0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0
0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0
0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0
0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0
0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0
0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0., tilt=  0.  0.
0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0
0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0
0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0
0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0
0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0
0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0
0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0
0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0., rmat=  0.  0.
0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0
0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0
0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0
0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0
```

```
0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0., disp= 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0., thick= 0. 0. 0.
 23.750000000000  0.   23.750000000000   1.7500000000000   1.7500000000000
0.   23.750000000000  0.   23.750000000000  0.  0.   600.75000000000  0.
0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0., findex= 0. 0.
0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0. 0. 0., sdata= 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
 75.728996858220   -2.1545287284752  0.    47.500000000000  0.  0.  0.  0.
0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0. 0.    72.793822313355   -6.3087988593608
 -6.6292828670924D-03     47.500000000000  0.  0.  0.  0.  0.  0.  0.  0.  0.
0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
```

```
 0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.
 0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.
 0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.
 0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.
 0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.
 0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.
 0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.
 0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.
 0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.
 0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.
 0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.
 0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.
 0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.
 0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.
 0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.
 0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.
 0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.
 0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.
 0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.
 0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.
 0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.
 0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.
 0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.
 0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.
 0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.
 0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.
 0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.
 0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.
 0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.
 0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.
 0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.
 0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.
 0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.
 0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.
 0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.
 0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.
 0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.
 0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.
 0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.
 0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.
 0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.
 0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.
 0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.
 0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.
, delta=    1.0000000000000D-07, energy=   -1.0000000000000    0.2770000000(
  0.5728000000000     0.7050000000000     0.9297000000000     1.4867000000(
   2.9843000000000     4.9522000000000     6.4038000000000     8.0478000000(
   9.1751000000000     9.7133000000000  0.  0.  0., delbet=  0.  0.  0.  0.
 0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.
 0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.
 0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.
 0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.
 0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.
 0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.
 0.  0.  0.  0.    1.4208657313788D-02   6.8673831777586D-03
3.5204209421783D-03
   6.9893373144806D-04     2.0769290647767D-03     3.5455121337951D-04
   1.1673034750193D-03     1.1128255579277D-03     7.6680960422116D-04
```

```
     2.5360111959666D-04     1.9764095429625D-04     2.0643631808189D-05
     7.0076137264186D-05     3.1980352637323D-06     4.1075815432952D-05
     1.2218703314117D-06     2.4414542654591D-05     5.1029673751875D-07
     1.9901903499959D-05     2.4922494023518D-06     1.8132340657957D-05
     2.0365958324343D-06  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.
0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.
0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.
0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.
0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.
0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.
0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.
0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.
0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.     1.4208657313788D-02
     6.8673831777586D-03     3.5204209421783D-03     6.9893373144806D-04
     2.0769290647767D-03     3.5455121337951D-04     1.1673034750193D-03
     1.1128255579277D-03     7.6680960422116D-04     2.5360111959666D-04
     1.9764095429625D-04     2.0643631808189D-05     7.0076137264186D-05
     3.1980352637323D-06     4.1075815432952D-05     1.2218703314117D-06
     2.4414542654591D-05     5.1029673751875D-07     1.9901903499959D-05
     2.4922494023518D-06     1.8132340657957D-05     2.0365958324343D-06  0.  0.
0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.
0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.
0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.
0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.
0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.
0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.
0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.
0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.
0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.
0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.
0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.
0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.
0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.
0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.
0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.
0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.
0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.
0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.
0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.
0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.
0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.
0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.
0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.
0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.
0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.
0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.
0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.
0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.
0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.
0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.
0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.
0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.
0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.
0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.
0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.
0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.
0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.
0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.
0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.
0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.
```

```
 0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.
 0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.
 0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.
 0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.
 0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.
 0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.
 0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.
 0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.
 0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.
 0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.
 0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.
 0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.
 0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.
 0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.
 0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.
 0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.
 0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.
 0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.
 0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.
 0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.
 0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.
 0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.
 0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.
 0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0., effa=  0.  0.
 0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0., imove=  0  0  0  0  0
 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0, irstr=  0  0  0  0
 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
 1  0  0  0  0  0  1  0  0  0  0  0  0  0  0  0  0  0  0, iwgt=  0  0  0  0
 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0, nsurf=  16
, nnrg=  3, kmax=  50, kprint=  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
 0  0  0  0  0  0  0  0  0  0, ichief=  0, itilt=  0  0  0  0  0  0  0  0  0
 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
 0  0  0  0  0  0  0  0  0  0  0  0  0  0, npass=  0, nvig=  0, nerr=  0
iaper= 'circ    ' '        ' '        ' '        ' 'zlim    '
'        ' '        ' '        ' '        ' '
'zlim    ' '        ' '        ' '        ' '
'        '
iobs= '        ' '        ' '        ' '        ' '        '
'        ' '        ' '        ' '        ' '
'        ' '        ' '        ' '        ' '
'        '
itype= 'flat    ' 'flat    ' 'flat    ' 'flat    ' 'grzcon01'
'flat    ' 'flat    ' 'flat    ' 'flat    ' 'flat    '
'grzcon01' 'flat    ' 'flat    ' 'flat    ' 'flat    '
'flat    '
imode= 'thru    ' 'thru    ' 'thru    ' 'thru    ' 'refl    '
'thru    ' 'thru    ' 'thru    ' 'thru    ' 'thru    '
'refl    ' 'thru    ' 'thru    ' 'thru    ' 'thru    '
'thru    '
ifdfm= '
      '
 '
'
 '
'
```

```
 '
'
 '
'
 '
'
 '
'
 '
'
 '
'
 '
'
 '
'
 '
'
 '
'
 '
'
 '
'
 '
'
 '
'
 '
'
 '
'
 '
'
 '
'
 '
'
 '
'
 '
'
 '
'
 '
'
 '
'
 '
'
 '
'
 '
'
 '
'
 '
'
```

```
  '
 '
  '
 '
  '
 '
  '
 '
  '
 '
  '
 '
  '
 '
  '
 '
  '
 '
  '
 '
  '
 '
  '
 '
  '
 '
  '
 '
  '
 '
  '
 '
  '
 '
  '
 '
 ihead= ' SXI design #2, Henke nickel
       '
  '
 '
  '
 '
  '
 '
  '
 '
  '
 '
  '
 '
  '
 '
  '
 '
  '
 '
  '
 '
```

```
 '
'
 '
'
 '
'
 '
'
 '
'
 '
'
 '
'
 '
'
 '
'
&end
```

APPENDIX 6

STRUCTURAL ANALYSIS INTERFACE

Appendix 6. Structural Analysis Interface

A6.1 drinfea.f Structural Analysis Interface (FORTRAN source code)

```
      implicit undefined (a-z)
c***********************************************************************
c********
c
c     AXAF MIRROR DATA INTERPOLATION PROGRAM : COMPUTE DR (201 X 1001)
c     input file is COSMOS/M file
c     NAME : drincos.f
c
c     Convert deformation data file from COSMOS/M to graztrace
c
c     COSMOS/M data file is generated by using LISTLOG to open a file,
c     and NODELIST, DISLIST to dump node data and deformation data,
c     and LISTLOG to close the file.
c
c***********************************************************************
c********
c
c---------------------------
c     PARAMETER DEFINITION
c---------------------------
c
c  number of files
c
      integer*4 nbrfile
      parameter (nbrfile = 2)
c
c  number of azimuthal points interpolated in the
c  range -pi through +pi
c
      integer*4 nbthseq
      parameter (nbthseq = 1001)
c
c  number of axial points interpolated
c
      integer*4 nbzseq
      parameter (nbzseq = 201)
      integer*4 opbfsz
      parameter (opbfsz = nbthseq * nbzseq)
      integer*4 nbzseqm1
      parameter (nbzseqm1 = nbzseq - 1)
c
c  z extent of interpolation
c
      real lz
      parameter (lz = 990.6)
c
c
c  maximum number of input points
      integer*4 nbinpts
      parameter (nbinpts = 250000)
```

```
c************************************************************************
      logical*1 space, unsc, num0, num9, capa, capz, sma, smz, ch, slash
     &, period, tilde
      logical*1 msg(80, 20)
      character inpfname*132, outfname*132
      character rsp(20)*80
      character fname*133
      character * 256 fnme
      double precision thzinp(2, nbinpts)
      double precision thseq(nbthseq), zseq(nbzseq)
      double precision intdr(nbthseq, nbzseq)
      double precision dztht(nbinpts),dzzval(nbinpts)
     * ,dzdr(nbinpts),angext,t0,z0,dr0,dzts,dzzs,dzdrs
      integer*2 nbrvch, nmsgl
      integer*4 icc,ic,ndim,ierr,iu,jj
      parameter (ndim = nbinpts)
      integer*4 nn
      integer*4 i, ii, iii
      integer*4 status
      integer*4 nsize, isize, osize
      logical*1 filexists,debug
      real pi, dtht
      real thcvf
      real lzov2, lzintv, l
      real thll, thul, zll, zul
      real op(opbfsz)
      real sttime, etime, tarray(2)
c
c  this is needed by dsurf for workspace
c
      real rwksp
      common/worksp/rwksp(1698008)
c
c-------------------------------
c     PARAMETER INITIALIZATION
c-------------------------------
    1 format(1x)
    2 format(42h  NAME OF INPUT DATA FILE (1-132 CHARS. - ,
     &18hALPH/NUM/UNSC) ?  )
    3 format(42h  NAME OF OUTPUT DATA FILE (1-132 CHARS. -,
     &18hALPH/NUM/UNSC) ?  )
    4 format(a132)
    5 format(49h  INPUT DESIRED HEADER MESSAGE (MAX. OF 20 LINES/,
     &23h80 CHARS. PER LINE)  ? )
    7 format(a80)
c
c-----------------------------------------------------------------------
c---
c                          REQUEST USER TO ENTER
c     1. NAME OF INPUT DATA FILE
c     2. NAME OF OUTPUT DATA FILE
c     3. HEADER MESSAGE
c-----------------------------------------------------------------------
c---
      data space / x'20' /
      data unsc / x'5f' /
      data num0 / x'30' /
      data num9 / x'39' /
      data capa / x'41' /
```

```
      data capz / x'5a' /
      data sma / x'61' /
      data smz / x'7a' /
      data slash / x'2f' /
      data period / x'2e' /
      data tilde / x'7e' /
      data msg / 1600*x'20' /
      data debug/ .false. /
      do i = 1, nbrfile
      write(unit=*, fmt=1)
1000  if (i .eq. 1) write(unit=*, fmt=2)
      if (i .eq. 2) write(unit=*, fmt=3)
      read(unit=*, fmt=4) fname
      nsize = 0
      do ii = 133, 1, -1
      if (fname(ii:ii) .ne. ' ') then
      nsize = ii
      goto 1101
      end if
      end do
1101  continue
      if ((nsize .lt. 1) .or. (nsize .gt. 132)) then
1100  write(unit=*, fmt=*) ' FILENAME HAS INVALID LENGTH '
      goto 1000
      end if
      if (fname(1:1) .eq. '_') then
      write(unit=*, fmt=*)
     &' FILENAME CANNOT BEGIN WITH AN UNDERSCORE CHARACTER', ' : ' //
     &fname(1:nsize)
      goto 1000
      end if
      nbrvch = 0
      do ii = 1, nsize
      ch = ichar(fname(ii:ii))
      if (((((((((ch .eq. space) .or. (ch .eq. unsc)) .or. (ch .eq. slash
     &)) .or. (ch .eq. period)) .or. (ch .eq. tilde)) .or. ((ch .ge.
     &num0) .and. (ch .le. num9))) .or. ((ch .ge. capa) .and. (ch .le.
     &capz))) .or. ((ch .ge. sma) .and. (ch .le. smz))) nbrvch = nbrvch
     & + 1
      end do
      if (nbrvch .lt. nsize) then
      write(unit=*, fmt=*) ' FILE NAME CONTAINS INVALID CHARACTERS : '
     & // fname(1:nsize)
      goto 1000
      end if
      if (i .eq. 1) then
      inpfname = fname(1:nsize)
      isize = nsize
      end if
      if (i .eq. 2) then
      outfname = fname(1:nsize)
      osize = nsize
      end if
      end do
      write(unit=*, fmt=1)
      write(unit=*, fmt=5)
      write(*,*)' (END WITH ctrl-d ON A LINE BY ITSELF)'
      do i = 1, 20
      read(unit=*, fmt=7, end=1200) rsp(i)
```

```
      nmsgl = i
      end do
 1200 do i = 1, nmsgl
      do ii = 1, 80
      msg(ii,i) = ichar(rsp(i)(ii:ii))
      end do
      end do
c
c-------------------------------------------
c     CHECK FOR EXISTENCE OF INPUT DATA FILE
c              OPEN INPUT DATA FILE
c          READ/PROCESS INPUT DATA FILE
c-------------------------------------------
      inquire(file=inpfname(1:isize) , exist=filexists)
      if (.not. filexists) then
      write(unit=*, fmt=*) ' INPUT DATA FILE ', inpfname(1:isize)
     &, ' DOES NOT EXIST'
      stop
      end if
c
c  thcvf is the factor to make the spacing in the z direction
c  and theta direction comparable in magnitude.  this is required
c  by dsurf for some reason.
c
c  (ideally theta angles in radians should be scaled by the
c  radius of the cylinder so that the the interpolation is
c  done directly in surface distance coordinates on the
c  cylindrical surface)
c
c*********************************************************
      iu=10
      fnme=inpfname(1:isize)
c  coordinate convert
1260  write(*,'(a,$)')'CONVERT CARTESIEN DATA TO CYLINDICAL (Y/N)'
      read(*,'(a)')fname
      if(fname.eq.' ')go to 1260
      i=1
      do while(fname(i:i).eq.' ')
      i=i+1
      end do
      if ((fname(i:i).eq.'Y').or.(fname(i:i).eq.'y')) then
      icc = 1
      else if ((fname(i:i).eq.'N').or.(fname(i:i).eq.'n')) then
      icc = 0
      else
      go to 1260
      end if
c  cosmos/m or nastran
1300  write(*,'(a,$)') 'INPUT DATA FROM COSMOS/M / NASTRAN (C/N)'
      read(*,'(a)')fname
      if(fname.eq.' ')go to 1300
      i=1
      do while(fname(i:i).eq.' ')
      i=i+1
      end do
      if ((fname(i:i).eq.'C').or.(fname(i:i).eq.'c'))go to 1310
      if ((fname(i:i).eq.'N').or.(fname(i:i).eq.'n')) go to 1320
      go to 1300
c  read in the data from the COSMOS/M file
```

```
1310  call rcos(iu,fnme,ndim,ierr,ic,dztht,dzzval,dzdr,icc,debug)
      go to 1350
c  read in the data from the NASTRAN  file
1320  call rnas(iu,fnme,ndim,ierr,ic,dztht,dzzval,dzdr,icc,debug)
1350    if(ierr.ne.0) then
        write(*,*)' input i/o error, ierr= ',ierr
        stop
      end if
      pi = 4.0 * atan(1.0)
c  modify the coordinate system to graztrace coordinates
      t0=0
      dzts=1.0
      z0=0
      dzzs=1.0
      dr0=0
      dzdrs=1.0
1400  write(*,'(a,$)') 'NEED MODIFY COORDINATE (Y/N)'
      read(*,'(a)')fname
      if(fname.eq.' ')go to 1400
      i=1
      do while(fname(i:i).eq.' ')
      i=i+1
      end do
      if ((fname(i:i).eq.'Y').or.(fname(i:i).eq.'y'))go to 1440
      if ((fname(i:i).eq.'N').or.(fname(i:i).eq.'n')) go to 1480
      go to 1400
1440  write(*,*)'KEY IN SHIFT AND SCALE t0 ts z0 zs dr0 drs'
      read(*,*)t0,dzts,z0,dzzs,dr0,dzdrs
      write(*,*)'THETA        SHIFT AND SCALE t0=',t0,' ts=',dzts
      write(*,*)'Z            SHIFT AND SCALE z0=',z0,' zs=',dzzs
      write(*,*)'DELTA RADIUS SHIFT AND SCALE dr0=',dr0,' drs=',dzdrs
1460  write(*,'(a,$)')'CORRECT ? (Y/N)'
      read(*,'(a)')fname
      if(fname.eq.' ')go to 1460
      i=1
      do while(fname(i:i).eq.' ')
      i=i+1
      end do
      if ((fname(i:i).eq.'Y').or.(fname(i:i).eq.'y'))go to 1480
      if ((fname(i:i).eq.'N').or.(fname(i:i).eq.'n')) go to 1440
      go to 1460
1480  continue
c
c  change axial length
c
      l=lz
c
1500  write(*,'(a,$)') 'CHANGE AXIAL LENGTH(990.6)(Y/N)'
      read(*,'(a)')fname
      if(fname.eq.' ')go to 1500
      i=1
      do while(fname(i:i).eq.' ')
      i=i+1
      end do
      if ((fname(i:i).eq.'Y').or.(fname(i:i).eq.'y'))go to 1540
      if ((fname(i:i).eq.'N').or.(fname(i:i).eq.'n')) go to 1580
      go to 1500
1540  write(*,*)'KEY IN AXIAL LENGTH lz'
      read(*,*)l
```

```
      write(*,*)'AXIAL LENGTH lz=' ,l
1560  write(*,'(a,$)')'CORRECT ? (Y/N)'
      read(*,'(a)')fname
      if(fname.eq.' ')go to 1560
      i=1
      do while(fname(i:i).eq.' ')
      i=i+1
      end do
      if ((fname(i:i).eq.'Y').or.(fname(i:i).eq.'y'))go to 1580
      if ((fname(i:i).eq.'N').or.(fname(i:i).eq.'n')) go to 1540
      go to 1560
1580  continue
      write(*,*)' transform to graztrace coordinates '
c
      call modify(ndim,ic,dztht,dzzval,dzdr,t0,z0,dr0,dzts,dzzs,dzdrs)
c  extend the theta distribution
      angext=35.d0*dble(pi)/180.d0
      write(*,*)' extend distribution by ',angext,' radians'
      call extend(ndim,ierr,ic,dztht,dzzval,dzdr,angext)
      if(ierr.ne.0) then
        write(*,*)'error in extend, ierr= ',ierr
        stop
      end if
c  accumulate and scale the data for dsurf
c  dsurf does not like elongated triangle regions
c  between data points
cf    thcvf = delta_z*dble(mt)/dble(2. * pi)
cf
cf  set thcvf for COSMOS/M data
cf
      thcvf = 1
cf
cf    write(*,*) 'angle scale factor thcvf= ',thcvf
      do jj=1,ic
        thzinp(1,jj) = dztht(jj) * thcvf
        thzinp(2,jj) = dzzval(jj)
      end do
      write(unit=*, fmt=*) ' INPUT FILE HAS ', ic,
     &' SEGMENTS OF DATA'
c*********************************************************
     open(66,file='print.drinfea')
     do jj=1,ic
       write(66,*) jj,thzinp(1,jj),thzinp(2,jj),dzdr(jj)
     end do
c*********************************************************
c
c--------------------------------------------------------
c     SET-UP REGULAR GRID TO BE INTERPOLATED ACROSS AND
c                   PERFORM INTERPOLATION
c--------------------------------------------------------
      dtht = pi / real(nbthseq-1) * 2.
      lzov2 = l / 2.0
      lzintv = l / nbzseqm1
      thll = - pi
      thul = pi
      zll = - lzov2
      zul = lzov2
      do i = 1, nbthseq
      thseq(i) = (dtht * (i - 1)) - pi
```

```
      thseq(i) = thcvf * thseq(i)
      end do
      do i = 1, nbzseq
      zseq(i) = (lzintv * (i - 1)) - lzov2
      end do
c
c  etime gives elapsed job time in seconds
c  (do not use system routine dtime because imsl
c   package apparently has a routine of the same
c   name)
c
      sttime = etime(tarray)
      write(unit=*, fmt=*) ' BEGIN INTERPOLATION OF DATA'
c
c  first set up workspace area with iwkin before calling dsurf
c  (the size of the workspace needed was given by an error
c   message the first time dsurf was called without using iwkin)
c
      call iwkin(1698008)
      call dsurf(ic, thzinp, dzdr, nbthseq, nbzseq, thseq, zseq,
     &intdr, nbthseq)
      sttime = etime(tarray)-sttime
      write(unit=*, fmt=*)
     &' EXECUTION TIME FOR INTERPOLATION OF DATA (SECONDS) = ', sttime
c
c----------------------------------------------
c     OPEN OUTPUT FILE AND WRITE HEADER RECORD
c----------------------------------------------
      open(unit=11, file=outfname(1:osize) , status='NEW', form
     &='UNFORMATTED', iostat=status, err=1700)
c
c  vax format to sun format conversion required division by
c  four and byte swapping for floating point storage
c
c  movbyt was needed for e.c.richardson output format
c
c   (see version drin7215.vf)
c
c     op(1) = float(nbthseq) / 4.0
c     op(2) = thll / 4.0
c     op(3) = thul / 4.0
c     op(4) = thcvf / 4.0
c     op(5) = float(nbzseq) / 4.0
c     op(6) = zll / 4.0
c     op(7) = zul / 4.0
c
c     call bytswap(14, op)
c
c     call movbyt(1600, msg, op(8))
c
      write(unit=11, iostat=status, err=1700)nbthseq,nbzseq,20
      goto 1800
 1700 write(unit=*, fmt=*) ' I/O ERROR, STATUS = ', status
      stop
c
c----------------------------------------------
c     WRITE INTERPOLATED RESULTS TO OUTPUT FILE
c----------------------------------------------
 1800 continue
```

```
      nn=nbzseq*nbthseq
      do i = 1, nbzseq
      do ii = 1, nbthseq
      op(ii+(i-1)*nbthseq) = intdr(ii,i)
c
c  these statements were required for converting from vax format
c  to sun format for floating point storage
c
c     op(ii) = op(ii) / 4.0
c     call bytswap(2, op(ii))
c
      end do
      end do
      write(unit=11, iostat=status, err=1900)(op(iii),iii=1,nn),
     & dble(thll),dble(thul),dble(zll),dble(zul),msg
      write(unit=*, fmt=*)
     & ' INTERPOLATED VALUES HAVE BEEN WRITTEN TO OUTPUT FILE'
      goto 2000
 1900 write(unit=*, fmt=*) ' I/O ERROR, STATUS = ', status
      stop
c
c------------------------
c     TERMINATE PROGRAM
c------------------------
 2000 stop
      end
```

A6.2 rcos.f COSMOS/M Data Extraction (FORTRAN)

```
      subroutine rcos(iu,fname,ndim,ierr,ic,tht,zval,dr,icc,debug)
c**********************************************************************
c
c    Read  the data file from COSMOS/M.
c
c**********************************************************************
c
c  routine to get COSM data
c
c   input:
c
c        iu             unit to user for read
c     fname             name of input smp data file
c      ndim             size of tht,zval,and dr arrays
c     debug             .true. gives debug output
c
c   output:
c
c        ic             number of data values returned
c       tht             theta values in radians
c      zval             z values in mm
c       icc             = 1 , convert from cartesien to cylinderical
c
c      ierr             read error (i/o error or bad data)
c
c                       1   i/o error
c                       3   bad data
c
c**********************************************************************
      implicit double precision (a-h,o-z)
c**********************************************************************
      character * 256 ibuff,fname
      dimension tht(ndim),zval(ndim),dr(ndim)
      logical debug
c**********************************************************************
      nchar=256
      ierr=0
      pi=atan(1.d0)*4.d0
      factor=pi/180.d0
      ic=0
      icd=0
c**********************************************************************
c
c  open the unit
c
      open(iu,file=fname,err=3000)
c**********************************************************************
c
c  get needed parameters
c
200   continue
      read(iu,'(a)',err=3000,end=3000) ibuff
c
```

```
      if(debug) write(*,*) ibuff
c
c  ignore blank lines
      if(ibuff.eq.' ') go to 200
c  find deformation
      if (index(ibuff,'Load case').ne. 0) go to 400
c
c  find data lines begin with "0-9"
c
      i=1
      do while(ibuff(i:i).eq.' ')
      i=i+1
      end do
      if ((ibuff(i:i).ge.'0').and.(ibuff(i:i).le.'9')) go to 300
c
      go to 200
c
c  read in coordinate
c
300   read(ibuff,*,err=3000) nn,xc,yc,zc
      ic=ic+1
      if (icc .eq. 1) then
       if(xc.eq.0.0) then
           if(yc.ge.0.0) then
            tht(ic)=.5*pi
            else
            tht(ic)=1.5*pi
             end if
           else
           tht(ic)=atan(yc/xc)
           if(xc.lt.0.0) tht(ic)=tht(ic)+pi
           if((xc.gt.0.0).and.(yc.lt.0.0)) tht(ic)=tht(ic)+2*pi
          end if
      else
      tht(ic)=yc*factor
      end if
      zval(ic)=zc
      go to 200
c     read(iu,'(a)',err=3000,end=3000) ibuff
c
c     if(debug) write(*,*)ibuff
c
c     i=1
c     do while(ibuff(i:i).eq.' ')
c     i=i+1
c     end do
c     if ((ibuff(i:i).ge.'0').and.(ibuff(i:i).le.'9')) go to 310
c
c  end of coordinate data
c
400   continue
      read(iu,'(a)',err=3000,end=3000) ibuff
c
      if(debug) write(*,*) ibuff
c
c  ignore blank lines
      if(ibuff.eq.' ') go to 400
c
c
```

```
      i=1
      do while(ibuff(i:i).eq.' ')
      i=i+1
      end do
      if ((ibuff(i:i).ge.'0').and.(ibuff(i:i).le.'9')) go to 500
c
c
      go to 400
c
c
c  read in deformation
c
500   icd=0
510   read(ibuff,*,err=3000) nn,xd,yd
      icd=icd+1
      if (icc .eq. 1) then
      dr(icd)=xd*dcos(tht(icd))+yd*dsin(tht(icd))
      else
      dr(icd)=xd
      end if
      if (icd.eq.ic) go to 600
      read(iu,'(a)',err=3000,end=3000) ibuff
c
      if(debug) write(*,*)ibuff
c
      i=1
      do while(ibuff(i:i).eq.' ')
      i=i+1
      end do
      if ((ibuff(i:i).ge.'0').and.(ibuff(i:i).le.'9')) go to 510
c
c  end of deformation data
c
600   return
c
c
c*************************************************************************
3000  continue
c  flag i/o error
      write(*,*) ' i/o error in rcos, file= ',fname
      ierr=1
      return
      end
```

A6.3 rnas.f NASTRAN Data Extraction (FORTRAN source code)

```
      subroutine rnas(iu,fname,ndim,ierr,ic,tht,zval,dr,icc,debug)
c***********************************************************************
c
c    Read  the data file from NASTRAN
c
c***********************************************************************
c
c  routine to get NASTRAN data
c
c   input:
c
c         iu              unit to user for read
c      fname              name of input smp data file
c       ndim              size of tht,zval,and dr arrays
c      debug              .true. gives debug output
c
c   output:
c
c         ic              number of data values returned
c        tht              theta values in radians
c       zval              z values in mm
c        icc              = 1 , convert from cartesien to cylinderical
c
c       ierr              read error (i/o error or bad data)
c
c                         1   i/o error
c                         3   bad data
c
c***********************************************************************
      implicit double precision (a-h,o-z)
c***********************************************************************
      character * 256 ibuff,fname
      character * 4   str
      dimension tht(ndim),zval(ndim),dr(ndim)
      logical debug
c***********************************************************************
      nchar=256
      ierr=0
      ic=0
      icd=0
      pi=atan(1.d0)*4.d0
      factor=pi/180.d0
c***********************************************************************
c
c  open the unit
c
      open(iu,file=fname,err=3000)
c***********************************************************************
c
c  get needed parameters
c
200   continue
      read(iu,'(a)',err=3000,end=3000) ibuff
```

```
c
      if(debug) write(*,*) ibuff
c
c  ignore blank lines
      if(ibuff.eq.' ') go to 200
c
c  find deformation data
      if(index(ibuff,'ENDDATA                              ') .ne. 0) go to 350
c
c  find data lines begin with "GRID"
c
      i=1
      do while(ibuff(i:i).eq.' ')
      i=i+1
      end do
      if (ibuff(i:i+3).eq.'GRID') go to 300
c
      go to 200
c
c  read in coordinate
c
300   read(ibuff(55:62),*,err=3000) xc
      read(ibuff(63:70),*,err=3000) yc
      read(ibuff(71:78),*,err=3000) zc
      ic=ic+1
      if (icc .eq. 1) then
          if (xc.eq.0.0) then
              if (yc.ge.0) then
                  tht(ic)=.5*pi
                  else
                  tht(ic)=1.5*pi
              end if
              else
              tht(ic)=atan(yc/xc)
            if (xc.lt.0) tht(ic)=tht(ic)+pi
            if ((xc.gt.0).and.(yc.lt.0)) tht(ic)=tht(ic)+2*pi
        end if
      else
      tht(ic)=yc*factor
      end if
      zval(ic)=zc
c     read(iu,'(a)',err=3000,end=3000) ibuff
c
c     if(debug) write(*,*)ibuff
c
c     i=1
c     do while(ibuff(i:i).eq.' ')
c     i=i+1
c     end do
c     if ((ibuff(i:i).ge.'0').and.(ibuff(i:i).le.'9')) go to 310
c
      go to 200
c
c  end of coordinate data
c
350   continue
      read(iu,'(a)',err=3000,end=3000) ibuff
c
      if(debug) write(*,*) ibuff
```

```
c
c  ignore blank lines
      if(ibuff.eq.' ') go to 350
c
c  find deformation data
      if(index(ibuff,'D I S P L A C E M E N T') .ne. 0) go to 400
      go to 350
c
400   continue
      read(iu,'(a)',err=3000,end=3000) ibuff
c
      if(debug) write(*,*) ibuff
c
c  ignore blank lines
      if(ibuff.eq.' ') go to 400
c
c
      i=1
      do while(ibuff(i:i).eq.' ')
      i=i+1
      end do
      if ((ibuff(i:i).ge.'0').and.(ibuff(i:i).le.'9').and.(i.gt.1))
     *go to 500
c
c
      go to 400
c
c
c  read in deformation
c
500   read(ibuff,*,err=3000) nn,str,xd,yd
      icd=icd+1
      if (icc .eq. 1) then
      dr(icd)=xd*dcos(tht(icd))+yd*dsin(tht(icd))
      else
      dr(icd)=xd
      end if
      if (icd.eq.ic) go to 600
      go to 400
c     read(iu,'(a)',err=3000,end=3000) ibuff
c
c     if(debug) write(*,*)ibuff
c
c     i=i+1
c     end do
c     if ((ibuff(i:i).ge.'0').and.(ibuff(i:i).le.'9')) go to 510
c
c  end of deformation data
c
600   return
c
c
c***********************************************************************
3000  continue
c  flag i/o error
      write(*,*) ' i/o error in rnas, file= ',fname
      ierr=1
      return
      end
```

A6.4 modify.f Coordinate Modification (FORTRAN source code)

```
      subroutine modify(ndim,ic,tht,zval,dr,t0,z0,dr0,ts,zs,drs)
      implicit double precision (a-h,o-z)
      dimension tht(ndim),zval(ndim),dr(ndim)
      pi=datan(1.d0)*4.d0
      tpi=2.d0*pi
      if(ndim.lt.ic) then
     write(*,*)' ndim too small in modify, stop'
     stop
      end if
      if(ic.lt.1) then
     write(*,*)' ic is zero or less in modify, stop'
     stop
      endif
      do i=1,ic
     zval(i)=(zval(i)-z0)*zs
     dr(i)=(dr(i)-dr0)*drs
     tht(i)=(tht(i)-t0)*ts
c  make sure that theta values are within [-pi,+pi]
100     continue
     if(tht(i).ge.-pi.and.tht(i).le.pi) go to 101
     if(tht(i).lt.-pi) then
       tht(i)=tht(i)+tpi
        else
       tht(i)=tht(i)-tpi
        endif
     go to 100
101     continue
      end do
      return
      end
```

A6.5 extend.f Data Extend (FORTRAN source code)

```
      subroutine extend(ndim,ierr,ic,tht,zval,dr,angext)
c
c  extend the angular distribution above pi and below -pi
c  by angext
c
c  i.e. wrap the distribution around
c
      implicit double precision (a-h,o-z)
      dimension tht(ndim),zval(ndim),dr(ndim)
      pi=datan(1.d0)*4.d0
      ierr=0
      if(ic.gt.ndim) go to 3000
      if(ic.lt.1) go to 3000
      icc=ic
      do i=1,ic
     if((pi-tht(i)).lt.angext) then
       icc=icc+1
       if(icc.gt.ndim) go to 3000
       tht(icc)=tht(i)-2.d0*pi
       zval(icc)=zval(i)
       dr(icc)=dr(i)
     endif
     if((tht(i)+pi).lt.angext) then
       icc=icc+1
       if(icc.gt.ndim) go to 3000
       tht(icc)=tht(i)+2.d0*pi
       zval(icc)=zval(i)
       dr(icc)=dr(i)
     endif
      end do
      ic=icc
      return
3000  continue
      ierr=1
      return
      end
```

A6.6 sxccos.lis Sample COSMOS/M data format (outputed by COSMOS/M)

Coordinate system 0 (Cartesian)

Node	X-Coordinate	Y-Coordinate	Z-Coordinate
1	0.000000e+00	7.640170e+00	-4.925000e+00
2	-1.326702e+00	7.524099e+00	-4.925000e+00
3	-1.320659e+00	7.489827e+00	-3.693750e+00
4	0.000000e+00	7.605370e+00	-3.693750e+00
5	-1.314588e+00	7.455398e+00	-2.462500e+00
6	0.000000e+00	7.570410e+00	-2.462500e+00
7	-1.308489e+00	7.420812e+00	-1.231250e+00
8	0.000000e+00	7.535290e+00	-1.231250e+00
9	-1.302361e+00	7.386058e+00	0.000000e+00
10	0.000000e+00	7.500000e+00	0.000000e+00
11	-1.283880e+00	7.281245e+00	1.231250e+00
12	0.000000e+00	7.393570e+00	1.231250e+00
13	-1.265369e+00	7.176265e+00	2.462500e+00
14	0.000000e+00	7.286970e+00	2.462500e+00
15	-1.246770e+00	7.070782e+00	3.693750e+00
16	0.000000e+00	7.179860e+00	3.693750e+00
17	-1.228248e+00	6.965743e+00	4.925000e+00
18	0.000000e+00	7.073200e+00	4.925000e+00
19	1.320659e+00	7.489827e+00	-3.693750e+00
20	1.326702e+00	7.524099e+00	-4.925000e+00
21	1.314588e+00	7.455398e+00	-2.462500e+00
22	1.308489e+00	7.420812e+00	-1.231250e+00
23	1.302361e+00	7.386058e+00	0.000000e+00
24	1.283880e+00	7.281245e+00	1.231250e+00
25	1.265369e+00	7.176265e+00	2.462500e+00
26	1.246770e+00	7.070782e+00	3.693750e+00
27	1.228248e+00	6.965743e+00	4.925000e+00
28	-2.601190e+00	7.146710e+00	-3.693750e+00
29	-2.613092e+00	7.179411e+00	-4.925000e+00
30	-2.589232e+00	7.113858e+00	-2.462500e+00
31	-2.577221e+00	7.080856e+00	-1.231250e+00
32	-2.565151e+00	7.047695e+00	0.000000e+00
33	-2.528750e+00	6.947683e+00	1.231250e+00
34	-2.492290e+00	6.847512e+00	2.462500e+00
35	-2.455657e+00	6.746861e+00	3.693750e+00
36	-2.419177e+00	6.646634e+00	4.925000e+00
37	-3.802685e+00	6.586443e+00	-3.693750e+00
38	-3.820085e+00	6.616581e+00	-4.925000e+00
39	-3.785205e+00	6.556167e+00	-2.462500e+00
40	-3.767645e+00	6.525752e+00	-1.231250e+00
41	-3.750000e+00	6.495191e+00	0.000000e+00
42	-3.696785e+00	6.403020e+00	1.231250e+00
43	-3.643485e+00	6.310701e+00	2.462500e+00
44	-3.589930e+00	6.217941e+00	3.693750e+00
45	-3.536600e+00	6.125571e+00	4.925000e+00
46	2.613092e+00	7.179411e+00	-4.925000e+00
47	2.601190e+00	7.146710e+00	-3.693750e+00
48	2.589232e+00	7.113858e+00	-2.462500e+00
49	2.577221e+00	7.080856e+00	-1.231250e+00
50	2.565151e+00	7.047695e+00	0.000000e+00
51	2.528750e+00	6.947683e+00	1.231250e+00

52	2.492290e+00	6.847512e+00	2.462500e+00
53	2.455657e+00	6.746861e+00	3.693750e+00
54	2.419177e+00	6.646634e+00	4.925000e+00
55	3.802685e+00	6.586443e+00	-3.693750e+00
56	3.820085e+00	6.616581e+00	-4.925000e+00
57	3.785205e+00	6.556167e+00	-2.462500e+00
58	3.767645e+00	6.525752e+00	-1.231250e+00
59	3.750000e+00	6.495191e+00	0.000000e+00
60	3.696785e+00	6.403020e+00	1.231250e+00
61	3.643485e+00	6.310701e+00	2.462500e+00
62	3.589930e+00	6.217941e+00	3.693750e+00
63	3.536600e+00	6.125571e+00	4.925000e+00
64	-4.888638e+00	5.826051e+00	-3.693750e+00
65	-4.911006e+00	5.852710e+00	-4.925000e+00
66	-4.866165e+00	5.799271e+00	-2.462500e+00
67	-4.843591e+00	5.772367e+00	-1.231250e+00
68	-4.820907e+00	5.745333e+00	0.000000e+00
69	-4.752495e+00	5.663804e+00	1.231250e+00
70	-4.683974e+00	5.582143e+00	2.462500e+00
71	-4.615125e+00	5.500092e+00	3.693750e+00
72	-4.546566e+00	5.418386e+00	4.925000e+00
73	-5.826051e+00	4.888638e+00	-3.693750e+00
74	-5.852710e+00	4.911007e+00	-4.925000e+00
75	-5.799270e+00	4.866166e+00	-2.462500e+00
76	-5.772367e+00	4.843591e+00	-1.231250e+00
77	-5.745333e+00	4.820907e+00	0.000000e+00
78	-5.663803e+00	4.752496e+00	1.231250e+00
79	-5.582143e+00	4.683974e+00	2.462500e+00
80	-5.500092e+00	4.615125e+00	3.693750e+00
81	-5.418386e+00	4.546566e+00	4.925000e+00
82	4.911006e+00	5.852710e+00	-4.925000e+00
83	4.888638e+00	5.826051e+00	-3.693750e+00
84	4.866165e+00	5.799271e+00	-2.462500e+00
85	4.843591e+00	5.772367e+00	-1.231250e+00
86	4.820907e+00	5.745333e+00	0.000000e+00
87	4.752495e+00	5.663804e+00	1.231250e+00
88	4.683974e+00	5.582143e+00	2.462500e+00
89	4.615125e+00	5.500092e+00	3.693750e+00
90	4.546566e+00	5.418386e+00	4.925000e+00
91	5.826051e+00	4.888638e+00	-3.693750e+00
92	5.852710e+00	4.911007e+00	-4.925000e+00
93	5.799270e+00	4.866166e+00	-2.462500e+00
94	5.772367e+00	4.843591e+00	-1.231250e+00
95	5.745333e+00	4.820907e+00	0.000000e+00
96	5.663803e+00	4.752496e+00	1.231250e+00
97	5.582143e+00	4.683974e+00	2.462500e+00
98	5.500092e+00	4.615125e+00	3.693750e+00
99	5.418386e+00	4.546566e+00	4.925000e+00
100	-6.586443e+00	3.802685e+00	-3.693750e+00
101	-6.616581e+00	3.820085e+00	-4.925000e+00
102	-6.556167e+00	3.785205e+00	-2.462500e+00
103	-6.525752e+00	3.767645e+00	-1.231250e+00
104	-6.495191e+00	3.750000e+00	0.000000e+00
105	-6.403019e+00	3.696785e+00	1.231250e+00
106	-6.310701e+00	3.643485e+00	2.462500e+00
107	-6.217941e+00	3.589930e+00	3.693750e+00
108	-6.125571e+00	3.536600e+00	4.925000e+00
109	-7.146710e+00	2.601190e+00	-3.693750e+00
110	-7.179411e+00	2.613092e+00	-4.925000e+00

111	-7.113858e+00	2.589233e+00	-2.462500e+00
112	-7.080856e+00	2.577221e+00	-1.231250e+00
113	-7.047695e+00	2.565151e+00	0.000000e+00
114	-6.947683e+00	2.528750e+00	1.231250e+00
115	-6.847512e+00	2.492291e+00	2.462500e+00
116	-6.746861e+00	2.455657e+00	3.693750e+00
117	-6.646634e+00	2.419177e+00	4.925000e+00
118	6.616581e+00	3.820085e+00	-4.925000e+00
119	6.586443e+00	3.802685e+00	-3.693750e+00
120	6.556167e+00	3.785205e+00	-2.462500e+00
121	6.525752e+00	3.767645e+00	-1.231250e+00
122	6.495191e+00	3.750000e+00	0.000000e+00
123	6.403019e+00	3.696785e+00	1.231250e+00
124	6.310701e+00	3.643485e+00	2.462500e+00
125	6.217941e+00	3.589930e+00	3.693750e+00
126	6.125571e+00	3.536600e+00	4.925000e+00
127	7.146710e+00	2.601190e+00	-3.693750e+00
128	7.179411e+00	2.613092e+00	-4.925000e+00
129	7.113858e+00	2.589233e+00	-2.462500e+00
130	7.080856e+00	2.577221e+00	-1.231250e+00
131	7.047695e+00	2.565151e+00	0.000000e+00
132	6.947683e+00	2.528750e+00	1.231250e+00
133	6.847512e+00	2.492291e+00	2.462500e+00
134	6.746861e+00	2.455657e+00	3.693750e+00
135	6.646634e+00	2.419177e+00	4.925000e+00
136	-7.489827e+00	1.320659e+00	-3.693750e+00
137	-7.524098e+00	1.326702e+00	-4.925000e+00
138	-7.455398e+00	1.314589e+00	-2.462500e+00
139	-7.420812e+00	1.308490e+00	-1.231250e+00
140	-7.386058e+00	1.302361e+00	0.000000e+00
141	-7.281245e+00	1.283881e+00	1.231250e+00
142	-7.176265e+00	1.265369e+00	2.462500e+00
143	-7.070782e+00	1.246770e+00	3.693750e+00
144	-6.965743e+00	1.228249e+00	4.925000e+00
145	-7.605370e+00	2.353971e-07	-3.693750e+00
146	-7.640170e+00	4.245420e-07	-4.925000e+00
147	-7.570410e+00	3.377823e-07	-2.462500e+00
148	-7.535289e+00	2.226650e-07	-1.231250e+00
149	-7.500000e+00	1.980977e-07	0.000000e+00
150	-7.393570e+00	5.344057e-07	1.231250e+00
151	-7.286970e+00	3.303264e-07	2.462500e+00
152	-7.179860e+00	4.802638e-07	3.693750e+00
153	-7.073200e+00	3.140545e-07	4.925000e+00
154	7.524098e+00	1.326702e+00	-4.925000e+00
155	7.489827e+00	1.320659e+00	-3.693750e+00
156	7.455398e+00	1.314589e+00	-2.462500e+00
157	7.420812e+00	1.308490e+00	-1.231250e+00
158	7.386058e+00	1.302361e+00	0.000000e+00
159	7.281245e+00	1.283881e+00	1.231250e+00
160	7.176265e+00	1.265369e+00	2.462500e+00
161	7.070782e+00	1.246770e+00	3.693750e+00
162	6.965743e+00	1.228249e+00	4.925000e+00
163	7.605370e+00	2.353971e-07	-3.693750e+00
164	7.640170e+00	4.245420e-07	-4.925000e+00
165	7.570410e+00	3.377823e-07	-2.462500e+00
166	7.535289e+00	2.226650e-07	-1.231250e+00
167	7.500000e+00	1.980977e-07	0.000000e+00
168	7.393570e+00	5.344057e-07	1.231250e+00
169	7.286970e+00	3.303264e-07	2.462500e+00

170	7.179860e+00	4.802638e-07	3.693750e+00
171	7.073200e+00	3.140545e-07	4.925000e+00
172	0.000000e+00	-7.605370e+00	-3.693750e+00
173	0.000000e+00	-7.640170e+00	-4.925000e+00
174	-1.320659e+00	-7.489827e+00	-3.693750e+00
175	-1.326702e+00	-7.524099e+00	-4.925000e+00
176	0.000000e+00	-7.570410e+00	-2.462500e+00
177	-1.314588e+00	-7.455398e+00	-2.462500e+00
178	0.000000e+00	-7.535290e+00	-1.231250e+00
179	-1.308489e+00	-7.420812e+00	-1.231250e+00
180	0.000000e+00	-7.500000e+00	0.000000e+00
181	-1.302361e+00	-7.386058e+00	0.000000e+00
182	0.000000e+00	-7.393570e+00	1.231250e+00
183	-1.283880e+00	-7.281245e+00	1.231250e+00
184	0.000000e+00	-7.286970e+00	2.462500e+00
185	-1.265369e+00	-7.176265e+00	2.462500e+00
186	0.000000e+00	-7.179860e+00	3.693750e+00
187	-1.246770e+00	-7.070782e+00	3.693750e+00
188	0.000000e+00	-7.073200e+00	4.925000e+00
189	-1.228248e+00	-6.965743e+00	4.925000e+00
190	1.320659e+00	-7.489827e+00	-3.693750e+00
191	1.326702e+00	-7.524099e+00	-4.925000e+00
192	1.314588e+00	-7.455398e+00	-2.462500e+00
193	1.308489e+00	-7.420812e+00	-1.231250e+00
194	1.302361e+00	-7.386058e+00	0.000000e+00
195	1.283880e+00	-7.281245e+00	1.231250e+00
196	1.265369e+00	-7.176265e+00	2.462500e+00
197	1.246770e+00	-7.070782e+00	3.693750e+00
198	1.228248e+00	-6.965743e+00	4.925000e+00
199	-2.613092e+00	-7.179411e+00	-4.925000e+00
200	-2.601190e+00	-7.146710e+00	-3.693750e+00
201	-2.589232e+00	-7.113858e+00	-2.462500e+00
202	-2.577221e+00	-7.080856e+00	-1.231250e+00
203	-2.565151e+00	-7.047695e+00	0.000000e+00
204	-2.528750e+00	-6.947683e+00	1.231250e+00
205	-2.492290e+00	-6.847512e+00	2.462500e+00
206	-2.455657e+00	-6.746861e+00	3.693750e+00
207	-2.419177e+00	-6.646634e+00	4.925000e+00
208	-3.802685e+00	-6.586443e+00	-3.693750e+00
209	-3.820085e+00	-6.616581e+00	-4.925000e+00
210	-3.785205e+00	-6.556167e+00	-2.462500e+00
211	-3.767645e+00	-6.525752e+00	-1.231250e+00
212	-3.750000e+00	-6.495191e+00	0.000000e+00
213	-3.696785e+00	-6.403020e+00	1.231250e+00
214	-3.643485e+00	-6.310701e+00	2.462500e+00
215	-3.589930e+00	-6.217941e+00	3.693750e+00
216	-3.536600e+00	-6.125571e+00	4.925000e+00
217	2.601190e+00	-7.146710e+00	-3.693750e+00
218	2.613092e+00	-7.179411e+00	-4.925000e+00
219	2.589232e+00	-7.113858e+00	-2.462500e+00
220	2.577221e+00	-7.080856e+00	-1.231250e+00
221	2.565151e+00	-7.047695e+00	0.000000e+00
222	2.528750e+00	-6.947683e+00	1.231250e+00
223	2.492290e+00	-6.847512e+00	2.462500e+00
224	2.455657e+00	-6.746861e+00	3.693750e+00
225	2.419177e+00	-6.646634e+00	4.925000e+00
226	3.802685e+00	-6.586443e+00	-3.693750e+00
227	3.820085e+00	-6.616581e+00	-4.925000e+00
228	3.785205e+00	-6.556167e+00	-2.462500e+00

229	3.767645e+00	-6.525752e+00	-1.231250e+00
230	3.750000e+00	-6.495191e+00	0.000000e+00
231	3.696785e+00	-6.403020e+00	1.231250e+00
232	3.643485e+00	-6.310701e+00	2.462500e+00
233	3.589930e+00	-6.217941e+00	3.693750e+00
234	3.536600e+00	-6.125571e+00	4.925000e+00
235	-4.911006e+00	-5.852710e+00	-4.925000e+00
236	-4.888638e+00	-5.826051e+00	-3.693750e+00
237	-4.866165e+00	-5.799271e+00	-2.462500e+00
238	-4.843591e+00	-5.772367e+00	-1.231250e+00
239	-4.820907e+00	-5.745333e+00	0.000000e+00
240	-4.752495e+00	-5.663804e+00	1.231250e+00
241	-4.683974e+00	-5.582143e+00	2.462500e+00
242	-4.615125e+00	-5.500092e+00	3.693750e+00
243	-4.546566e+00	-5.418386e+00	4.925000e+00
244	-5.826051e+00	-4.888638e+00	-3.693750e+00
245	-5.852710e+00	-4.911007e+00	-4.925000e+00
246	-5.799270e+00	-4.866166e+00	-2.462500e+00
247	-5.772367e+00	-4.843591e+00	-1.231250e+00
248	-5.745333e+00	-4.820907e+00	0.000000e+00
249	-5.663803e+00	-4.752496e+00	1.231250e+00
250	-5.582143e+00	-4.683974e+00	2.462500e+00
251	-5.500092e+00	-4.615125e+00	3.693750e+00
252	-5.418386e+00	-4.546566e+00	4.925000e+00
253	4.888638e+00	-5.826051e+00	-3.693750e+00
254	4.911006e+00	-5.852710e+00	-4.925000e+00
255	4.866165e+00	-5.799271e+00	-2.462500e+00
256	4.843591e+00	-5.772367e+00	-1.231250e+00
257	4.820907e+00	-5.745333e+00	0.000000e+00
258	4.752495e+00	-5.663804e+00	1.231250e+00
259	4.683974e+00	-5.582143e+00	2.462500e+00
260	4.615125e+00	-5.500092e+00	3.693750e+00
261	4.546566e+00	-5.418386e+00	4.925000e+00
262	5.826051e+00	-4.888638e+00	-3.693750e+00
263	5.852710e+00	-4.911007e+00	-4.925000e+00
264	5.799270e+00	-4.866166e+00	-2.462500e+00
265	5.772367e+00	-4.843591e+00	-1.231250e+00
266	5.745333e+00	-4.820907e+00	0.000000e+00
267	5.663803e+00	-4.752496e+00	1.231250e+00
268	5.582143e+00	-4.683974e+00	2.462500e+00
269	5.500092e+00	-4.615125e+00	3.693750e+00
270	5.418386e+00	-4.546566e+00	4.925000e+00
271	-6.616581e+00	-3.820085e+00	-4.925000e+00
272	-6.586443e+00	-3.802685e+00	-3.693750e+00
273	-6.556167e+00	-3.785205e+00	-2.462500e+00
274	-6.525752e+00	-3.767645e+00	-1.231250e+00
275	-6.495191e+00	-3.750000e+00	0.000000e+00
276	-6.403019e+00	-3.696785e+00	1.231250e+00
277	-6.310701e+00	-3.643485e+00	2.462500e+00
278	-6.217941e+00	-3.589930e+00	3.693750e+00
279	-6.125571e+00	-3.536600e+00	4.925000e+00
280	-7.146710e+00	-2.601190e+00	-3.693750e+00
281	-7.179411e+00	-2.613092e+00	-4.925000e+00
282	-7.113858e+00	-2.589233e+00	-2.462500e+00
283	-7.080856e+00	-2.577221e+00	-1.231250e+00
284	-7.047695e+00	-2.565151e+00	0.000000e+00
285	-6.947683e+00	-2.528750e+00	1.231250e+00
286	-6.847512e+00	-2.492291e+00	2.462500e+00
287	-6.746861e+00	-2.455657e+00	3.693750e+00

288	-6.646634e+00	-2.419177e+00	4.925000e+00
289	6.586443e+00	-3.802685e+00	-3.693750e+00
290	6.616581e+00	-3.820085e+00	-4.925000e+00
291	6.556167e+00	-3.785205e+00	-2.462500e+00
292	6.525752e+00	-3.767645e+00	-1.231250e+00
293	6.495191e+00	-3.750000e+00	0.000000e+00
294	6.403019e+00	-3.696785e+00	1.231250e+00
295	6.310701e+00	-3.643485e+00	2.462500e+00
296	6.217941e+00	-3.589930e+00	3.693750e+00
297	6.125571e+00	-3.536600e+00	4.925000e+00
298	7.146710e+00	-2.601190e+00	-3.693750e+00
299	7.179411e+00	-2.613092e+00	-4.925000e+00
300	7.113858e+00	-2.589233e+00	-2.462500e+00
301	7.080856e+00	-2.577221e+00	-1.231250e+00
302	7.047695e+00	-2.565151e+00	0.000000e+00
303	6.947683e+00	-2.528750e+00	1.231250e+00
304	6.847512e+00	-2.492291e+00	2.462500e+00
305	6.746861e+00	-2.455657e+00	3.693750e+00
306	6.646634e+00	-2.419177e+00	4.925000e+00
307	-7.524098e+00	-1.326702e+00	-4.925000e+00
308	-7.489827e+00	-1.320659e+00	-3.693750e+00
309	-7.455398e+00	-1.314589e+00	-2.462500e+00
310	-7.420812e+00	-1.308490e+00	-1.231250e+00
311	-7.386058e+00	-1.302361e+00	0.000000e+00
312	-7.281245e+00	-1.283881e+00	1.231250e+00
313	-7.176265e+00	-1.265369e+00	2.462500e+00
314	-7.070782e+00	-1.246770e+00	3.693750e+00
315	-6.965743e+00	-1.228249e+00	4.925000e+00
316	7.489827e+00	-1.320659e+00	-3.693750e+00
317	7.524098e+00	-1.326702e+00	-4.925000e+00
318	7.455398e+00	-1.314589e+00	-2.462500e+00
319	7.420812e+00	-1.308490e+00	-1.231250e+00
320	7.386058e+00	-1.302361e+00	0.000000e+00
321	7.281245e+00	-1.283881e+00	1.231250e+00
322	7.176265e+00	-1.265369e+00	2.462500e+00
323	7.070782e+00	-1.246770e+00	3.693750e+00
324	6.965743e+00	-1.228249e+00	4.925000e+00

Load case 1

Node	UX	UY	UZ	RX	RY	RZ
1	2.144e-20	-1.836e-06	-1.486e-07	-3.109e-07	-3.555e-21	2.159e-2
2	2.673e-08	-1.602e-06	-1.256e-07	-2.251e-07	-1.233e-08	-2.576e-0
3	2.793e-08	-1.479e-06	-1.050e-07	-5.408e-08	1.558e-08	-1.512e-0
4	1.614e-20	-1.568e-06	-1.264e-07	-1.868e-07	-3.325e-21	1.572e-2
5	4.712e-08	-1.400e-06	-9.085e-08	-7.225e-08	7.884e-09	2.534e-0
6	1.070e-20	-1.316e-06	-1.038e-07	-2.916e-07	-2.888e-21	1.116e-2
7	7.190e-08	-1.269e-06	-7.287e-08	-9.367e-08	-1.247e-08	3.468e-0
8	5.202e-21	-8.148e-07	-6.738e-08	-6.364e-07	-1.932e-21	4.839e-2
9	7.475e-08	-1.102e-06	-5.644e-08	-6.073e-08	-6.046e-08	7.692e-0
10	0.000e+00	0.000e+00	0.000e+00	0.000e+00	0.000e+00	0.000e+0
11	6.703e-08	-1.181e-06	-5.859e-08	1.287e-07	-2.908e-08	4.466e-0
12	-6.375e-21	-7.374e-07	-9.247e-09	5.525e-07	-1.691e-21	-6.339e-2
13	4.639e-08	-1.297e-06	-5.338e-08	4.168e-08	-2.345e-08	1.095e-0
14	-1.257e-20	-1.208e-06	-3.176e-08	2.341e-07	-1.858e-21	-1.337e-2
15	2.772e-08	-1.343e-06	-4.188e-08	3.732e-08	1.141e-09	-9.636e-0
16	-1.857e-20	-1.399e-06	-3.717e-08	1.033e-07	-1.445e-21	-2.116e-2
17	3.183e-08	-1.463e-06	-3.701e-08	1.868e-07	4.198e-08	-2.284e-0
18	-2.439e-20	-1.569e-06	-3.654e-08	2.613e-07	-9.135e-22	-2.766e-2
19	-2.793e-08	-1.479e-06	-1.050e-07	-5.408e-08	-1.558e-08	1.512e-0
20	-2.673e-08	-1.602e-06	-1.256e-07	-2.251e-07	1.233e-08	2.576e-0

21	-4.712e-08	-1.400e-06	-9.085e-08	-7.225e-08	-7.884e-09	-2.534e-08
22	-7.190e-08	-1.269e-06	-7.287e-08	-9.367e-08	1.247e-08	-3.468e-07
23	-7.475e-08	-1.102e-06	-5.644e-08	-6.073e-08	6.046e-08	-7.692e-07
24	-6.703e-08	-1.181e-06	-5.859e-08	1.287e-07	2.908e-08	-4.466e-07
25	-4.639e-08	-1.297e-06	-5.338e-08	4.168e-08	2.345e-08	-1.095e-07
26	-2.772e-08	-1.343e-06	-4.188e-08	3.732e-08	-1.141e-09	9.636e-08
27	-3.183e-08	-1.463e-06	-3.701e-08	1.868e-07	-4.198e-08	2.284e-07
28	-2.741e-08	-1.146e-06	-7.108e-08	4.182e-08	4.672e-08	-3.970e-07
29	-4.154e-08	-1.163e-06	-8.641e-08	-1.202e-07	-8.801e-09	-4.887e-07
30	3.889e-08	-1.249e-06	-6.240e-08	8.268e-08	5.832e-08	-3.447e-07
31	1.142e-07	-1.350e-06	-6.246e-08	3.122e-08	3.484e-08	-3.197e-07
32	1.557e-07	-1.367e-06	-7.168e-08	-1.856e-08	2.024e-08	-2.584e-07
33	1.222e-07	-1.319e-06	-7.310e-08	-1.033e-07	-8.441e-09	-2.442e-07
34	4.629e-08	-1.186e-06	-5.588e-08	-1.481e-07	-1.817e-08	-3.102e-07
35	-2.143e-08	-1.049e-06	-3.347e-08	-1.224e-07	-2.366e-09	-3.648e-07
36	-4.791e-08	-9.973e-07	-1.343e-08	3.105e-08	5.740e-08	-4.005e-07
37	-2.855e-07	-5.205e-07	-4.728e-08	4.853e-08	4.852e-08	-5.151e-07
38	-3.111e-07	-4.765e-07	-6.029e-08	-2.236e-08	5.233e-09	-4.244e-07
39	-2.386e-07	-5.872e-07	-3.872e-08	4.448e-08	5.097e-08	-6.333e-07
40	-1.926e-07	-6.369e-07	-2.947e-08	2.067e-08	5.104e-08	-7.476e-07
41	-1.515e-07	-6.497e-07	-2.010e-08	-2.427e-08	5.883e-08	-7.778e-07
42	-1.637e-07	-6.543e-07	-8.924e-09	-7.552e-08	6.333e-08	-7.509e-07
43	-1.974e-07	-6.022e-07	6.972e-09	-1.012e-07	4.763e-08	-6.158e-07
44	-2.295e-07	-5.389e-07	2.587e-08	-7.587e-08	4.651e-08	-4.784e-07
45	-2.131e-07	-5.456e-07	3.868e-08	3.071e-08	9.975e-08	-4.210e-07
46	4.154e-08	-1.163e-06	-8.641e-08	-1.202e-07	8.801e-09	4.887e-07
47	2.741e-08	-1.146e-06	-7.108e-08	4.182e-08	-4.672e-08	3.970e-07
48	-3.889e-08	-1.249e-06	-6.240e-08	8.268e-08	-5.832e-08	3.447e-07
49	-1.142e-07	-1.350e-06	-6.246e-08	3.122e-08	-3.484e-08	3.197e-07
50	-1.557e-07	-1.367e-06	-7.168e-08	-1.856e-08	-2.024e-08	2.584e-07
51	-1.222e-07	-1.319e-06	-7.310e-08	-1.033e-07	8.441e-09	2.442e-07
52	-4.629e-08	-1.186e-06	-5.588e-08	-1.481e-07	1.817e-08	3.102e-07
53	2.143e-08	-1.049e-06	-3.347e-08	-1.224e-07	2.366e-09	3.648e-07
54	4.791e-08	-9.973e-07	-1.343e-08	3.105e-08	-5.740e-08	4.005e-07
55	2.855e-07	-5.205e-07	-4.728e-08	4.853e-08	-4.852e-08	5.151e-07
56	3.111e-07	-4.765e-07	-6.029e-08	-2.236e-08	-5.233e-09	4.244e-07
57	2.386e-07	-5.872e-07	-3.872e-08	4.448e-08	-5.097e-08	6.333e-07
58	1.926e-07	-6.369e-07	-2.947e-08	2.067e-08	-5.104e-08	7.476e-07
59	1.515e-07	-6.497e-07	-2.010e-08	-2.427e-08	-5.883e-08	7.778e-07
60	1.637e-07	-6.543e-07	-8.924e-09	-7.552e-08	-6.333e-08	7.509e-07
61	1.974e-07	-6.022e-07	6.972e-09	-1.012e-07	-4.763e-08	6.158e-07
62	2.295e-07	-5.389e-07	2.587e-08	-7.587e-08	-4.651e-08	4.784e-07
63	2.131e-07	-5.456e-07	3.868e-08	3.071e-08	-9.975e-08	4.210e-07
64	-5.450e-07	-9.527e-08	-5.196e-08	-8.438e-08	-6.007e-08	-1.622e-07
65	-4.599e-07	-1.968e-07	-6.692e-08	-9.951e-08	-7.527e-08	-6.553e-08
66	-6.192e-07	1.339e-08	-3.287e-08	-9.089e-08	-6.468e-08	-3.111e-07
67	-6.831e-07	1.255e-07	-2.708e-09	-5.839e-08	-3.161e-08	-4.768e-07
68	-6.816e-07	1.763e-07	4.558e-08	-1.566e-08	4.757e-08	-5.419e-07
69	-6.494e-07	9.676e-08	7.475e-08	1.673e-08	1.106e-07	-4.798e-07
70	-5.529e-07	-3.953e-08	8.542e-08	5.923e-08	1.312e-07	-2.853e-07
71	-4.626e-07	-1.550e-07	8.672e-08	4.062e-08	9.817e-08	-1.308e-07
72	-3.868e-07	-2.345e-07	8.863e-08	5.831e-08	1.011e-07	-1.706e-08
73	-3.616e-07	-2.351e-07	-6.127e-08	-1.186e-07	-1.156e-07	5.090e-07
74	-2.476e-07	-3.642e-07	-7.585e-08	-1.128e-07	-1.124e-07	4.860e-07
75	-5.156e-07	-3.996e-08	-4.269e-08	-1.668e-07	-1.688e-07	5.035e-07
76	-6.987e-07	2.068e-07	-1.065e-08	-1.575e-07	-1.597e-07	4.067e-07
77	-7.667e-07	3.576e-07	3.722e-08	-6.326e-09	-4.142e-08	4.236e-07
78	-6.784e-07	1.897e-07	9.895e-08	1.670e-07	1.881e-07	4.997e-07
79	-4.764e-07	-6.824e-08	9.527e-08	1.554e-07	1.532e-07	4.352e-07

80	-3.004e-07	-2.780e-07	8.389e-08	1.399e-07	1.256e-07	4.568e-0
81	-1.379e-07	-4.453e-07	7.228e-08	1.400e-07	1.249e-07	4.839e-0
82	4.599e-07	-1.968e-07	-6.692e-08	-9.951e-08	7.527e-08	6.553e-0
83	5.450e-07	-9.527e-08	-5.196e-08	-8.438e-08	6.007e-08	1.622e-0
84	6.192e-07	1.339e-08	-3.287e-08	-9.089e-08	6.468e-08	3.111e-0
85	6.831e-07	1.255e-07	-2.708e-09	-5.839e-08	3.161e-08	4.768e-0
86	6.816e-07	1.763e-07	4.558e-08	-1.566e-08	-4.757e-08	5.419e-0
87	6.494e-07	9.676e-08	7.475e-08	1.673e-08	-1.106e-07	4.798e-0
88	5.529e-07	-3.953e-08	8.542e-08	5.923e-08	-1.312e-07	2.853e-0
89	4.626e-07	-1.550e-07	8.672e-08	4.062e-08	-9.817e-08	1.308e-0
90	3.868e-07	-2.345e-07	8.863e-08	5.831e-08	-1.011e-07	1.706e-0
91	3.616e-07	-2.351e-07	-6.127e-08	-1.186e-07	1.156e-07	-5.090e-0
92	2.476e-07	-3.642e-07	-7.585e-08	-1.128e-07	1.124e-07	-4.860e-0
93	5.156e-07	-3.996e-08	-4.269e-08	-1.668e-07	1.688e-07	-5.035e-0
94	6.987e-07	2.068e-07	-1.065e-08	-1.575e-07	1.597e-07	-4.067e-0
95	7.667e-07	3.576e-07	3.722e-08	-6.326e-09	4.142e-08	-4.236e-0
96	6.784e-07	1.897e-07	9.895e-08	1.670e-07	-1.881e-07	-4.997e-0
97	4.764e-07	-6.824e-08	9.527e-08	1.554e-07	-1.532e-07	-4.352e-0
98	3.004e-07	-2.780e-07	8.389e-08	1.399e-07	-1.256e-07	-4.568e-0
99	1.379e-07	-4.453e-07	7.228e-08	1.400e-07	-1.249e-07	-4.839e-0
100	4.559e-07	-7.784e-07	-6.318e-08	-8.407e-08	-5.934e-08	8.543e-0
101	5.459e-07	-8.905e-07	-7.431e-08	-1.104e-07	-1.158e-07	7.966e-0
102	3.900e-07	-6.406e-07	-5.189e-08	-1.210e-07	-9.851e-08	1.042e-0
103	2.360e-07	-4.061e-07	-3.369e-08	-2.331e-07	-2.329e-07	1.250e-0
104	0.000e+00	0.000e+00	0.000e+00	0.000e+00	0.000e+00	0.000e+0
105	2.031e-07	-3.855e-07	-4.624e-09	2.269e-07	1.880e-07	1.191e-0
106	3.448e-07	-6.109e-07	-1.588e-08	1.348e-07	5.731e-08	9.458e-0
107	3.852e-07	-7.322e-07	-1.858e-08	8.473e-08	1.072e-08	7.318e-0
108	4.333e-07	-8.182e-07	-1.827e-08	1.064e-07	8.944e-08	5.512e-0
109	1.272e-06	-1.146e-06	-4.369e-08	2.197e-10	1.215e-07	3.579e-0
110	1.209e-06	-1.174e-06	-4.974e-08	-6.090e-08	-4.723e-08	2.283e-0
111	1.459e-06	-1.126e-06	-4.816e-08	1.987e-08	1.702e-07	5.289e-0
112	1.663e-06	-1.074e-06	-6.222e-08	2.356e-08	1.289e-07	7.535e-0
113	1.685e-06	-9.717e-07	-9.367e-08	5.470e-08	-5.663e-08	1.193e-0
114	1.560e-06	-1.025e-06	-1.575e-07	1.010e-08	-1.418e-07	9.463e-0
115	1.325e-06	-1.055e-06	-1.487e-07	-1.384e-08	-1.869e-07	5.447e-0
116	1.086e-06	-1.047e-06	-1.258e-07	-2.111e-08	-1.510e-07	3.605e-0
117	9.661e-07	-1.046e-06	-1.093e-07	1.887e-08	-9.339e-10	2.555e-0
118	-5.459e-07	-8.905e-07	-7.431e-08	-1.104e-07	1.158e-07	-7.966e-0
119	-4.559e-07	-7.784e-07	-6.318e-08	-8.407e-08	5.934e-08	-8.543e-0
120	-3.900e-07	-6.406e-07	-5.189e-08	-1.210e-07	9.851e-08	-1.042e-0
121	-2.360e-07	-4.061e-07	-3.369e-08	-2.331e-07	2.329e-07	-1.250e-0
122	0.000e+00	0.000e+00	0.000e+00	0.000e+00	0.000e+00	0.000e+0
123	-2.031e-07	-3.855e-07	-4.624e-09	2.269e-07	-1.880e-07	-1.191e-0
124	-3.448e-07	-6.109e-07	-1.588e-08	1.348e-07	-5.731e-08	-9.458e-0
125	-3.852e-07	-7.322e-07	-1.858e-08	8.473e-08	-1.072e-08	-7.318e-0
126	-4.333e-07	-8.182e-07	-1.827e-08	1.064e-07	-8.944e-08	-5.512e-0
127	-1.272e-06	-1.146e-06	-4.369e-08	2.197e-10	-1.215e-07	-3.579e-0
128	-1.209e-06	-1.174e-06	-4.974e-08	-6.090e-08	4.723e-08	-2.283e-0
129	-1.459e-06	-1.126e-06	-4.816e-08	1.987e-08	-1.702e-07	-5.289e-0
130	-1.663e-06	-1.074e-06	-6.222e-08	2.356e-08	-1.289e-07	-7.535e-0
131	-1.685e-06	-9.717e-07	-9.367e-08	5.470e-08	5.663e-08	-1.193e-0
132	-1.560e-06	-1.025e-06	-1.575e-07	1.010e-08	1.418e-07	-9.463e-0
133	-1.325e-06	-1.055e-06	-1.487e-07	-1.384e-08	1.869e-07	-5.447e-0
134	-1.086e-06	-1.047e-06	-1.258e-07	-2.111e-08	1.510e-07	-3.605e-0
135	-9.661e-07	-1.046e-06	-1.093e-07	1.887e-08	9.339e-10	-2.555e-0
136	1.169e-06	-1.117e-06	-1.912e-08	-9.715e-09	1.627e-07	-5.592e-0
137	1.046e-06	-1.114e-06	-1.949e-08	-3.702e-08	1.535e-08	-5.542e-0
138	1.422e-06	-1.120e-06	-2.952e-08	1.398e-09	2.118e-07	-6.558e-0

139	1.677e-06	-1.105e-06	-5.975e-08	-1.638e-08	1.108e-07	-7.966e-07
140	1.755e-06	-1.051e-06	-1.173e-07	-7.583e-08	-1.618e-08	-8.003e-07
141	1.612e-06	-1.068e-06	-1.478e-07	-1.317e-07	-1.634e-07	-7.240e-07
142	1.292e-06	-1.051e-06	-1.413e-07	-1.423e-07	-2.542e-07	-5.955e-07
143	9.951e-07	-1.021e-06	-1.202e-07	-1.239e-07	-1.915e-07	-4.955e-07
144	8.301e-07	-9.924e-07	-1.021e-07	-9.263e-08	-4.548e-08	-4.176e-07
145	2.867e-13	-1.015e-06	-9.003e-15	-3.552e-08	3.983e-14	-1.030e-06
146	4.334e-13	-1.015e-06	-1.040e-14	-3.142e-08	1.553e-14	-8.488e-07
147	4.581e-13	-1.000e-06	-1.411e-14	-4.381e-08	5.357e-14	-1.267e-06
148	3.471e-13	-9.706e-07	-1.816e-14	-6.773e-08	2.191e-14	-1.495e-06
149	3.158e-13	-9.122e-07	-2.747e-14	-1.262e-07	1.064e-14	-1.556e-06
150	7.730e-13	-9.379e-07	-7.750e-14	-1.852e-07	-6.940e-14	-1.502e-06
151	3.894e-13	-9.440e-07	-4.995e-14	-1.837e-07	-6.662e-14	-1.232e-06
152	4.299e-13	-9.364e-07	-6.034e-14	-1.564e-07	-1.117e-13	-9.567e-07
153	2.008e-13	-9.147e-07	-3.393e-14	-1.421e-07	-5.540e-14	-8.420e-07
154	-1.046e-06	-1.114e-06	-1.949e-08	-3.702e-08	-1.535e-08	5.542e-07
155	-1.169e-06	-1.117e-06	-1.912e-08	-9.715e-09	-1.627e-07	5.592e-07
156	-1.422e-06	-1.120e-06	-2.952e-08	1.398e-09	-2.118e-07	6.558e-07
157	-1.677e-06	-1.105e-06	-5.975e-08	-1.638e-08	-1.108e-07	7.966e-07
158	-1.755e-06	-1.051e-06	-1.173e-07	-7.583e-08	1.618e-08	8.003e-07
159	-1.612e-06	-1.068e-06	-1.478e-07	-1.317e-07	1.634e-07	7.240e-07
160	-1.292e-06	-1.051e-06	-1.413e-07	-1.423e-07	2.542e-07	5.955e-07
161	-9.951e-07	-1.021e-06	-1.202e-07	-1.239e-07	1.915e-07	4.955e-07
162	-8.301e-07	-9.924e-07	-1.021e-07	-9.263e-08	4.548e-08	4.176e-07
163	-2.867e-13	-1.015e-06	-9.003e-15	-3.552e-08	-3.983e-14	1.030e-06
164	-4.334e-13	-1.015e-06	-1.040e-14	-3.142e-08	-1.553e-14	8.488e-07
165	-4.581e-13	-1.000e-06	-1.411e-14	-4.381e-08	-5.357e-14	1.267e-06
166	-3.471e-13	-9.706e-07	-1.816e-14	-6.773e-08	-2.191e-14	1.495e-06
167	-3.158e-13	-9.122e-07	-2.747e-14	-1.262e-07	-1.064e-14	1.556e-06
168	-7.730e-13	-9.379e-07	-7.750e-14	-1.852e-07	6.940e-14	1.502e-06
169	-3.894e-13	-9.440e-07	-4.995e-14	-1.837e-07	6.662e-14	1.232e-06
170	-4.299e-13	-9.364e-07	-6.034e-14	-1.564e-07	1.117e-13	9.567e-07
171	-2.008e-13	-9.147e-07	-3.393e-14	-1.421e-07	5.540e-14	8.420e-07
172	1.229e-20	-1.568e-06	1.264e-07	-1.868e-07	-3.407e-21	-1.240e-20
173	1.643e-20	-1.836e-06	1.486e-07	-3.109e-07	-3.578e-21	-1.640e-20
174	-2.793e-08	-1.479e-06	1.050e-07	-5.408e-08	-1.558e-08	-1.512e-07
175	-2.673e-08	-1.602e-06	1.256e-07	-2.251e-07	1.233e-08	-2.576e-07
176	8.094e-21	-1.316e-06	1.038e-07	-2.916e-07	-2.896e-21	-8.657e-21
177	-4.712e-08	-1.400e-06	9.085e-08	-7.225e-08	-7.884e-09	2.534e-08
178	3.841e-21	-8.148e-07	6.738e-08	-6.364e-07	-1.929e-21	-4.495e-21
179	-7.190e-08	-1.269e-06	7.287e-08	-9.367e-08	1.247e-08	3.468e-07
180	0.000e+00	0.000e+00	0.000e+00	0.000e+00	0.000e+00	0.000e+00
181	-7.475e-08	-1.102e-06	5.644e-08	-6.073e-08	6.046e-08	7.692e-07
182	-4.115e-21	-7.374e-07	9.247e-09	5.525e-07	-1.660e-21	3.978e-22
183	-6.703e-08	-1.181e-06	5.859e-08	1.287e-07	2.908e-08	4.466e-07
184	-8.255e-21	-1.208e-06	3.176e-08	2.341e-07	-1.822e-21	5.080e-21
185	-4.639e-08	-1.297e-06	5.338e-08	4.168e-08	2.345e-08	1.095e-07
186	-1.206e-20	-1.399e-06	3.717e-08	1.033e-07	-1.626e-21	8.540e-21
187	-2.772e-08	-1.343e-06	4.188e-08	3.732e-08	-1.141e-09	-9.636e-08
188	-1.599e-20	-1.569e-06	3.654e-08	2.613e-07	-1.337e-21	1.290e-20
189	-3.183e-08	-1.463e-06	3.701e-08	1.868e-07	-4.198e-08	-2.284e-07
190	2.793e-08	-1.479e-06	1.050e-07	-5.408e-08	1.558e-08	1.512e-07
191	2.673e-08	-1.602e-06	1.256e-07	-2.251e-07	-1.233e-08	2.576e-07
192	4.712e-08	-1.400e-06	9.085e-08	-7.225e-08	7.884e-09	-2.534e-08
193	7.190e-08	-1.269e-06	7.287e-08	-9.367e-08	-1.247e-08	-3.468e-07
194	7.475e-08	-1.102e-06	5.644e-08	-6.073e-08	-6.046e-08	-7.692e-07
195	6.703e-08	-1.181e-06	5.859e-08	1.287e-07	-2.908e-08	-4.466e-07
196	4.639e-08	-1.297e-06	5.338e-08	4.168e-08	-2.345e-08	-1.095e-07
197	2.772e-08	-1.343e-06	4.188e-08	3.732e-08	1.141e-09	9.636e-08

198	3.183e-08	-1.463e-06	3.701e-08	1.868e-07	4.198e-08	2.284e-0
199	4.154e-08	-1.163e-06	8.641e-08	-1.202e-07	8.801e-09	-4.887e-0
200	2.741e-08	-1.146e-06	7.108e-08	4.182e-08	-4.672e-08	-3.970e-0
201	-3.889e-08	-1.249e-06	6.240e-08	8.268e-08	-5.832e-08	-3.447e-0
202	-1.142e-07	-1.350e-06	6.246e-08	3.122e-08	-3.484e-08	-3.197e-0
203	-1.557e-07	-1.367e-06	7.168e-08	-1.856e-08	-2.024e-08	-2.584e-0
204	-1.222e-07	-1.319e-06	7.310e-08	-1.033e-07	8.441e-09	-2.442e-0
205	-4.629e-08	-1.186e-06	5.588e-08	-1.481e-07	1.817e-08	-3.102e-0
206	2.143e-08	-1.049e-06	3.347e-08	-1.224e-07	2.366e-09	-3.648e-0
207	4.791e-08	-9.973e-07	1.343e-08	3.105e-08	-5.740e-08	-4.005e-0
208	2.855e-07	-5.205e-07	4.728e-08	4.853e-08	-4.852e-08	-5.151e-0
209	3.111e-07	-4.765e-07	6.029e-08	-2.236e-08	-5.233e-09	-4.244e-0
210	2.386e-07	-5.872e-07	3.872e-08	4.448e-08	-5.097e-08	-6.333e-0
211	1.926e-07	-6.369e-07	2.947e-08	2.067e-08	-5.104e-08	-7.476e-0
212	1.515e-07	-6.497e-07	2.010e-08	-2.427e-08	-5.883e-08	-7.778e-0
213	1.637e-07	-6.543e-07	8.924e-09	-7.552e-08	-6.333e-08	-7.509e-0
214	1.974e-07	-6.022e-07	-6.972e-09	-1.012e-07	-4.763e-08	-6.158e-0
215	2.295e-07	-5.389e-07	-2.587e-08	-7.587e-08	-4.651e-08	-4.784e-0
216	2.131e-07	-5.456e-07	-3.868e-08	3.071e-08	-9.975e-08	-4.210e-0
217	-2.741e-08	-1.146e-06	7.108e-08	4.182e-08	4.672e-08	3.970e-0
218	-4.154e-08	-1.163e-06	8.641e-08	-1.202e-07	-8.801e-09	4.887e-0
219	3.889e-08	-1.249e-06	6.240e-08	8.268e-08	5.832e-08	3.447e-0
220	1.142e-07	-1.350e-06	6.246e-08	3.122e-08	3.484e-08	3.197e-0
221	1.557e-07	-1.367e-06	7.168e-08	-1.856e-08	2.024e-08	2.584e-0
222	1.222e-07	-1.319e-06	7.310e-08	-1.033e-07	-8.441e-09	2.442e-0
223	4.629e-08	-1.186e-06	5.588e-08	-1.481e-07	-1.817e-08	3.102e-0
224	-2.143e-08	-1.049e-06	3.347e-08	-1.224e-07	-2.366e-09	3.648e-0
225	-4.791e-08	-9.973e-07	1.343e-08	3.105e-08	5.740e-08	4.005e-0
226	-2.855e-07	-5.205e-07	4.728e-08	4.853e-08	4.852e-08	5.151e-0
227	-3.111e-07	-4.765e-07	6.029e-08	-2.236e-08	5.233e-09	4.244e-0
228	-2.386e-07	-5.872e-07	3.872e-08	4.448e-08	5.097e-08	6.333e-0
229	-1.926e-07	-6.369e-07	2.947e-08	2.067e-08	5.104e-08	7.476e-0
230	-1.515e-07	-6.497e-07	2.010e-08	-2.427e-08	5.883e-08	7.778e-0
231	-1.637e-07	-6.543e-07	8.924e-09	-7.552e-08	6.333e-08	7.509e-0
232	-1.974e-07	-6.022e-07	-6.972e-09	-1.012e-07	4.763e-08	6.158e-0
233	-2.295e-07	-5.389e-07	-2.587e-08	-7.587e-08	4.651e-08	4.784e-0
234	-2.131e-07	-5.456e-07	-3.868e-08	3.071e-08	9.975e-08	4.210e-0
235	4.599e-07	-1.968e-07	6.692e-08	-9.951e-08	7.527e-08	-6.553e-0
236	5.450e-07	-9.527e-08	5.196e-08	-8.438e-08	6.007e-08	-1.622e-0
237	6.192e-07	1.339e-08	3.287e-08	-9.089e-08	6.468e-08	-3.111e-0
238	6.831e-07	1.255e-07	2.708e-09	-5.839e-08	3.161e-08	-4.768e-0
239	6.816e-07	1.763e-07	-4.558e-08	-1.566e-08	-4.757e-08	-5.419e-0
240	6.494e-07	9.676e-08	-7.475e-08	1.673e-08	-1.106e-07	-4.798e-0
241	5.529e-07	-3.953e-08	-8.542e-08	5.923e-08	-1.312e-07	-2.853e-0
242	4.626e-07	-1.550e-07	-8.672e-08	4.062e-08	-9.817e-08	-1.308e-0
243	3.868e-07	-2.345e-07	-8.863e-08	5.831e-08	-1.011e-07	-1.706e-0
244	3.616e-07	-2.351e-07	6.127e-08	-1.186e-07	1.156e-07	5.090e-0
245	2.476e-07	-3.642e-07	7.585e-08	-1.128e-07	1.124e-07	4.860e-0
246	5.156e-07	-3.996e-08	4.269e-08	-1.668e-07	1.688e-07	5.035e-0
247	6.987e-07	2.068e-07	1.065e-08	-1.575e-07	1.597e-07	4.067e-0
248	7.667e-07	3.576e-07	-3.722e-08	-6.326e-09	4.142e-08	4.236e-0
249	6.784e-07	1.897e-07	-9.895e-08	1.670e-07	-1.881e-07	4.997e-0
250	4.764e-07	-6.824e-08	-9.527e-08	1.554e-07	-1.532e-07	4.352e-0
251	3.004e-07	-2.780e-07	-8.389e-08	1.399e-07	-1.256e-07	4.568e-0
252	1.379e-07	-4.453e-07	-7.228e-08	1.400e-07	-1.249e-07	4.839e-0
253	-5.450e-07	-9.527e-08	5.196e-08	-8.438e-08	-6.007e-08	1.622e-0
254	-4.599e-07	-1.968e-07	6.692e-08	-9.951e-08	-7.527e-08	6.553e-0
255	-6.192e-07	1.339e-08	3.287e-08	-9.089e-08	-6.468e-08	3.111e-0
256	-6.831e-07	1.255e-07	2.708e-09	-5.839e-08	-3.161e-08	4.768e-0

257	-6.816e-07	1.763e-07	-4.558e-08	-1.566e-08	4.757e-08	5.419e-07
258	-6.494e-07	9.676e-08	-7.475e-08	1.673e-08	1.106e-07	4.798e-07
259	-5.529e-07	-3.953e-08	-8.542e-08	5.923e-08	1.312e-07	2.853e-07
260	-4.626e-07	-1.550e-07	-8.672e-08	4.062e-08	9.817e-08	1.308e-07
261	-3.868e-07	-2.345e-07	-8.863e-08	5.831e-08	1.011e-07	1.706e-08
262	-3.616e-07	-2.351e-07	6.127e-08	-1.186e-07	-1.156e-07	-5.090e-07
263	-2.476e-07	-3.642e-07	7.585e-08	-1.128e-07	-1.124e-07	-4.860e-07
264	-5.156e-07	-3.996e-08	4.269e-08	-1.668e-07	-1.688e-07	-5.035e-07
265	-6.987e-07	2.068e-07	1.065e-08	-1.575e-07	-1.597e-07	-4.067e-07
266	-7.667e-07	3.576e-07	-3.722e-08	-6.326e-09	-4.142e-08	-4.236e-07
267	-6.784e-07	1.897e-07	-9.895e-08	1.670e-07	1.881e-07	-4.997e-07
268	-4.764e-07	-6.824e-08	-9.527e-08	1.554e-07	1.532e-07	-4.352e-07
269	-3.004e-07	-2.780e-07	-8.389e-08	1.399e-07	1.256e-07	-4.568e-07
270	-1.379e-07	-4.453e-07	-7.228e-08	1.400e-07	1.249e-07	-4.839e-07
271	-5.459e-07	-8.905e-07	7.431e-08	-1.104e-07	1.158e-07	7.966e-07
272	-4.559e-07	-7.784e-07	6.318e-08	-8.407e-08	5.934e-08	8.543e-07
273	-3.900e-07	-6.406e-07	5.189e-08	-1.210e-07	9.851e-08	1.042e-06
274	-2.360e-07	-4.061e-07	3.369e-08	-2.331e-07	2.329e-07	1.250e-06
275	0.000e+00	0.000e+00	0.000e+00	0.000e+00	0.000e+00	0.000e+00
276	-2.031e-07	-3.855e-07	4.624e-09	2.269e-07	-1.880e-07	1.191e-06
277	-3.448e-07	-6.109e-07	1.588e-08	1.348e-07	-5.731e-08	9.458e-07
278	-3.852e-07	-7.322e-07	1.858e-08	8.473e-08	-1.072e-08	7.318e-07
279	-4.333e-07	-8.182e-07	1.827e-08	1.064e-07	-8.944e-08	5.512e-07
280	-1.272e-06	-1.146e-06	4.369e-08	2.197e-10	-1.215e-07	3.579e-07
281	-1.209e-06	-1.174e-06	4.974e-08	-6.090e-08	4.723e-08	2.283e-07
282	-1.459e-06	-1.126e-06	4.816e-08	1.987e-08	-1.702e-07	5.289e-07
283	-1.663e-06	-1.074e-06	6.222e-08	2.356e-08	-1.289e-07	7.535e-07
284	-1.685e-06	-9.717e-07	9.367e-08	5.470e-08	5.663e-08	1.193e-06
285	-1.560e-06	-1.025e-06	1.575e-07	1.010e-08	1.418e-07	9.463e-07
286	-1.325e-06	-1.055e-06	1.487e-07	-1.384e-08	1.869e-07	5.447e-07
287	-1.086e-06	-1.047e-06	1.258e-07	-2.111e-08	1.510e-07	3.605e-07
288	-9.661e-07	-1.046e-06	1.093e-07	1.887e-08	9.339e-10	2.555e-07
289	4.559e-07	-7.784e-07	6.318e-08	-8.407e-08	-5.934e-08	-8.543e-07
290	5.459e-07	-8.905e-07	7.431e-08	-1.104e-07	-1.158e-07	-7.966e-07
291	3.900e-07	-6.406e-07	5.189e-08	-1.210e-07	-9.851e-08	-1.042e-06
292	2.360e-07	-4.061e-07	3.369e-08	-2.331e-07	-2.329e-07	-1.250e-06
293	0.000e+00	0.000e+00	0.000e+00	0.000e+00	0.000e+00	0.000e+00
294	2.031e-07	-3.855e-07	4.624e-09	2.269e-07	1.880e-07	-1.191e-06
295	3.448e-07	-6.109e-07	1.588e-08	1.348e-07	5.731e-08	-9.458e-07
296	3.852e-07	-7.322e-07	1.858e-08	8.473e-08	1.072e-08	-7.318e-07
297	4.333e-07	-8.182e-07	1.827e-08	1.064e-07	8.944e-08	-5.512e-07
298	1.272e-06	-1.146e-06	4.369e-08	2.197e-10	1.215e-07	-3.579e-07
299	1.209e-06	-1.174e-06	4.974e-08	-6.090e-08	-4.723e-08	-2.283e-07
300	1.459e-06	-1.126e-06	4.816e-08	1.987e-08	1.702e-07	-5.289e-07
301	1.663e-06	-1.074e-06	6.222e-08	2.356e-08	1.289e-07	-7.535e-07
302	1.685e-06	-9.717e-07	9.367e-08	5.470e-08	-5.663e-08	-1.193e-06
303	1.560e-06	-1.025e-06	1.575e-07	1.010e-08	-1.418e-07	-9.463e-07
304	1.325e-06	-1.055e-06	1.487e-07	-1.384e-08	-1.869e-07	-5.447e-07
305	1.086e-06	-1.047e-06	1.258e-07	-2.111e-08	-1.510e-07	-3.605e-07
306	9.661e-07	-1.046e-06	1.093e-07	1.887e-08	-9.339e-10	-2.555e-07
307	-1.046e-06	-1.114e-06	1.949e-08	-3.702e-08	-1.535e-08	-5.542e-07
308	-1.169e-06	-1.117e-06	1.912e-08	-9.715e-09	-1.627e-07	-5.592e-07
309	-1.422e-06	-1.120e-06	2.952e-08	1.398e-09	-2.118e-07	-6.558e-07
310	-1.677e-06	-1.105e-06	5.975e-08	-1.638e-08	-1.108e-07	-7.966e-07
311	-1.755e-06	-1.051e-06	1.173e-07	-7.583e-08	1.618e-08	-8.003e-07
312	-1.612e-06	-1.068e-06	1.478e-07	-1.317e-07	1.634e-07	-7.240e-07
313	-1.292e-06	-1.051e-06	1.413e-07	-1.423e-07	2.542e-07	-5.955e-07
314	-9.951e-07	-1.021e-06	1.202e-07	-1.239e-07	1.915e-07	-4.955e-07
315	-8.301e-07	-9.924e-07	1.021e-07	-9.263e-08	4.548e-08	-4.176e-07

316	1.169e-06	-1.117e-06	1.912e-08	-9.715e-09	1.627e-07	5.592e-0
317	1.046e-06	-1.114e-06	1.949e-08	-3.702e-08	1.535e-08	5.542e-0
318	1.422e-06	-1.120e-06	2.952e-08	1.398e-09	2.118e-07	6.558e-0
319	1.677e-06	-1.105e-06	5.975e-08	-1.638e-08	1.108e-07	7.966e-0
320	1.755e-06	-1.051e-06	1.173e-07	-7.583e-08	-1.618e-08	8.003e-0
321	1.612e-06	-1.068e-06	1.478e-07	-1.317e-07	-1.634e-07	7.240e-0
322	1.292e-06	-1.051e-06	1.413e-07	-1.423e-07	-2.542e-07	5.955e-0
323	9.951e-07	-1.021e-06	1.202e-07	-1.239e-07	-1.915e-07	4.955e-0
324	8.301e-07	-9.924e-07	1.021e-07	-9.263e-08	-4.548e-08	4.176e-0

A6.7 print.drinfea Sample deformation data (printed by the interface)

```
1    1.5707963267949   -49.250000000000   -1.8360000000000D-05
2    1.7453292959202   -49.250000000000   -1.5823036250514D-05
3    1.7453293062579   -36.937500000000   -1.4613806478636D-05
4    1.5707963267949   -36.937500000000   -1.5680000000000D-05
5    1.7453292754984   -24.625000000000   -1.3869131517254D-05
6    1.5707963267949   -24.625000000000   -1.3160000000000D-05
7    1.7453292030675   -12.312500000000   -1.2622063498638D-05
8    1.5707963267949   -12.312500000000   -8.1480000000000D-06
9    1.7453292117515  0.   -1.0982383498385D-05
10    1.5707963267949  0.  0.
11    1.7453292591006   12.312500000000   -1.1746975926682D-05
12    1.5707963267949   12.312500000000   -7.3740000000000D-06
13    1.7453292330449   24.625000000000   -1.2853511980209D-05
14    1.5707963267949   24.625000000000   -1.2080000000000D-05
15    1.7453293011935   36.937500000000   -1.3274103296497D-05
16    1.5707963267949   36.937500000000   -1.3990000000000D-05
17    1.7453291919049   49.250000000000   -1.4463009775340D-05
18    1.5707963267949   49.250000000000   -1.5690000000000D-05
19    1.3962633473319   -36.937500000000   -1.4613806478636D-05
20    1.3962633576696   -49.250000000000   -1.5823036250514D-05
21    1.3962633780914   -24.625000000000   -1.3869131517254D-05
22    1.3962634505223   -12.312500000000   -1.2622063498638D-05
23    1.3962634418383  0.   -1.0982383498385D-05
24    1.3962633944892   12.312500000000   -1.1746975926682D-05
25    1.3962634205449   24.625000000000   -1.2853511980209D-05
26    1.3962633523963   36.937500000000   -1.3274103296497D-05
27    1.3962634616849   49.250000000000   -1.4463009775340D-05
28    1.9198622126628   -36.937500000000   -1.0675129564763D-05
29    1.9198621890549   -49.250000000000   -1.0786549960393D-05
30    1.9198621073355   -24.625000000000   -1.1869772740249D-05
31    1.9198621999993   -12.312500000000   -1.3076437303462D-05
32    1.9198621521119  0.   -1.3378123569872D-05
33    1.9198622014542   12.312500000000   -1.2812494201724D-05
34    1.9198621066836   24.625000000000   -1.1303075862210D-05
35    1.9198622323289   36.937500000000   -9.7840804664130D-06
36    1.9198621859538   49.250000000000   -9.2076926226064D-06
37    2.0943951434944   -36.937500000000   -3.0801620181092D-06
38    2.0943951226304   -49.250000000000   -2.5711109462944D-06
39    2.0943951272970   -24.625000000000   -3.8923010464451D-06
40    2.0943951398757   -12.312500000000   -4.5527156148206D-06
41    2.0943950709521  0.   -4.8690671917756D-06
42    2.0943950648375   12.312500000000   -4.8479043930672D-06
43    2.0943951117671   24.625000000000   -4.2282049373399D-06
44    2.0943951132302   36.937500000000   -3.5195108502554D-06
45    2.0943950943380   49.250000000000   -3.6595346398883D-06
46    1.2217304645349   -49.250000000000   -1.0786549960393D-05
47    1.2217304409270   -36.937500000000   -1.0675129564763D-05
48    1.2217305462543   -24.625000000000   -1.1869772740249D-05
49    1.2217304535905   -12.312500000000   -1.3076437303462D-05
50    1.2217305014779  0.   -1.3378123569872D-05
51    1.2217304521356   12.312500000000   -1.2812494201724D-05
52    1.2217305469062   24.625000000000   -1.1303075862210D-05
53    1.2217304212608   36.937500000000   -9.7840804664130D-06
54    1.2217304676360   49.250000000000   -9.2076926226064D-06
```

55	1.0471975100954	-36.937500000000	-3.0801620181092D-06
56	1.0471975309594	-49.250000000000	-2.5711109462944D-06
57	1.0471975262928	-24.625000000000	-3.8923010464451D-06
58	1.0471975137141	-12.312500000000	-4.5527156148206D-06
59	1.0471975826377	0.	-4.8690671917756D-06
60	1.0471975887523	12.312500000000	-4.8479043930672D-06
61	1.0471975418227	24.625000000000	-4.2282049373399D-06
62	1.0471975403596	36.937500000000	-3.5195108502554D-06
63	1.0471975592518	49.250000000000	-3.6595346398883D-06
64	2.2689281036071	-36.937500000000	2.7733822957384D-06
65	2.2689279471409	-49.250000000000	1.4486043676840D-06
66	2.2689279104967	-24.625000000000	4.0827136847642D-06
67	2.2689280254862	-12.312500000000	5.3522679285598D-06
68	2.2689280478887	0.	5.7317767838149D-06
69	2.2689279397876	12.312500000000	4.9154869582748D-06
70	2.2689280136658	24.625000000000	3.2511552630665D-06
71	2.2689280095997	36.937500000000	1.7861665138869D-06
72	2.2689280607141	49.250000000000	6.8992840321956D-07
73	2.4434608767776	-36.937500000000	1.2588227223633D-06
74	2.4434609329784	-49.250000000000	-4.4430652012895D-07
75	2.4434608837908	-24.625000000000	3.6928669700838D-06
76	2.4434609548985	-12.312500000000	6.6816373070273D-06
77	2.4434609324960	0.	8.1718711932067D-06
78	2.4434608500488	12.312500000000	6.4162132989699D-06
79	2.4434609667189	24.625000000000	3.2107975120964D-06
80	2.4434609707850	36.937500000000	5.1424802526172D-07
81	2.4434609196706	49.250000000000	-1.8059580812158D-06
82	0.87266470644885	-49.250000000000	1.4486043676840D-06
83	0.87266454998266	-36.937500000000	2.7733822957384D-06
84	0.87266474309312	-24.625000000000	4.0827136847642D-06
85	0.87266462810357	-12.312500000000	5.3522679285598D-06
86	0.87266460570114	0.	5.7317767838149D-06
87	0.87266471380215	12.312500000000	4.9154869582748D-06
88	0.87266463992403	24.625000000000	3.2511552630665D-06
89	0.87266464399008	36.937500000000	1.7861665138869D-06
90	0.87266459287571	49.250000000000	6.8992840321956D-07
91	0.69813177681223	-36.937500000000	1.2588227223633D-06
92	0.69813172061141	-49.250000000000	-4.4430652012895D-07
93	0.69813176979897	-24.625000000000	3.6928669700838D-06
94	0.69813169869133	-12.312500000000	6.6816373070273D-06
95	0.69813172109376	0.	8.1718711932067D-06
96	0.69813180354102	12.312500000000	6.4162132989699D-06
97	0.69813168687086	24.625000000000	3.2107975120964D-06
98	0.69813168280482	36.937500000000	5.1424802526172D-07
99	0.69813173391918	49.250000000000	-1.8059580812158D-06
100	2.6179938368903	-36.937500000000	-7.8402099992322D-06
101	2.6179938577543	-49.250000000000	-9.1801327800904D-06
102	2.6179938530877	-24.625000000000	-6.5804991643571D-06
103	2.6179938405090	-12.312500000000	-4.0743200405252D-06
104	2.6179939094326	0.	0.
105	2.6179938479209	12.312500000000	-3.6863976649411D-06
106	2.6179938686176	24.625000000000	-6.0405556256813D-06
107	2.6179938671545	36.937500000000	-6.9969299032233D-06
108	2.6179938860467	49.250000000000	-7.8434880349719D-06
109	2.7925267677219	-36.937500000000	-1.5872441206563D-05
110	2.7925267913298	-49.250000000000	-1.5376200349751D-05
111	2.7925267489221	-24.625000000000	-1.7561262454523D-05
112	2.7925267803854	-12.312500000000	-1.9300384723434D-05
113	2.7925268282728	0.	-1.9157230308464D-05

114	2.7925267789305	12.312500000000	-1.8164911457579D-05
115	2.7925267447459	24.625000000000	-1.6059240052049D-05
116	2.7925267480557	36.937500000000	-1.3786013100016D-05
117	2.7925267944309	49.250000000000	-1.2655901165758D-05
118	0.52359879583553	-49.250000000000	-9.1801327800904D-06
119	0.52359881669953	-36.937500000000	-7.8402099992322D-06
120	0.52359880050209	-24.625000000000	-6.5804991643571D-06
121	0.52359881308079	-12.312500000000	-4.0743200405252D-06
122	0.52359874415719	0.	0.
123	0.52359880566892	12.312500000000	-3.6863976649411D-06
124	0.52359878497223	24.625000000000	-6.0405556256813D-06
125	0.52359878643526	36.937500000000	-6.9969299032233D-06
126	0.52359876754311	49.250000000000	-7.8434880349719D-06
127	0.34906588586790	-36.937500000000	-1.5872441206563D-05
128	0.34906586225996	-49.250000000000	-1.5376200349751D-05
129	0.34906590466765	-24.625000000000	-1.7561262454523D-05
130	0.34906587320437	-12.312500000000	-1.9300384723434D-05
131	0.34906582531700	0.	-1.9157230308464D-05
132	0.34906587465928	12.312500000000	-1.8164911457579D-05
133	0.34906590884386	24.625000000000	-1.6059240052049D-05
134	0.34906590553405	36.937500000000	-1.3786013100016D-05
135	0.34906585915894	49.250000000000	-1.2655901165758D-05
136	2.9670596741268	-36.937500000000	-1.3452053264016D-05
137	2.9670596617362	-49.250000000000	-1.2235530405896D-05
138	2.9670595747999	-24.625000000000	-1.5948827152525D-05
139	2.9670596466244	-12.312500000000	-1.8434039032913D-05
140	2.9670597686332	0.	-1.9108418118758D-05
141	2.9670595880862	12.312500000000	-1.7729664598982D-05
142	2.9670597473398	24.625000000000	-1.4548758362578D-05
143	2.9670596791912	36.937500000000	-1.1572770253881D-05
144	2.9670596492489	49.250000000000	-9.8981743323099D-06
145	3.1415926226384	-36.937500000000	-3.1811570459104D-12
146	3.1415925980227	-49.250000000000	-4.8980059471230D-12
147	3.1415926089710	-24.625000000000	-5.0271875903228D-12
148	3.1415926240402	-12.312500000000	-3.7578087085528D-12
149	3.1415926271768	0.	-3.3989396322871D-12
150	3.1415925813100	12.312500000000	-8.4079121696551D-12
151	3.1415926082587	24.625000000000	-4.3219256290310D-12
152	3.1415925866994	36.937500000000	-4.9253618262717D-12
153	3.1415926091892	49.250000000000	-2.4141325175787D-12
154	0.17453299185364	-49.250000000000	-1.2235530405896D-05
155	0.17453297946304	-36.937500000000	-1.3452053264016D-05
156	0.17453307878992	-24.625000000000	-1.5948827152525D-05
157	0.17453300696537	-12.312500000000	-1.8434039032913D-05
158	0.17453288495656	0.	-1.9108418118758D-05
159	0.17453306550359	12.312500000000	-1.7729664598982D-05
160	0.17453290625004	24.625000000000	-1.4548758362578D-05
161	0.17453297439857	36.937500000000	-1.1572770253881D-05
162	0.17453300434087	49.250000000000	-9.8981743323099D-06
163	3.0951433000630D-08	-36.937500000000	-3.1811570449564D-12
164	5.5567088166886D-08	-49.250000000000	-4.8980059448939D-12
165	4.4618759089666D-08	-24.625000000000	-5.0271875908967D-12
166	2.9549629748773D-08	-12.312500000000	-3.7578087063416D-12
167	2.6413026666667D-08	0.	-3.3989396292533D-12
168	7.2279791765006D-08	12.312500000000	-8.4079121669640D-12
169	4.5331104697837D-08	24.625000000000	-4.3219256283476D-12
170	6.6890412904987D-08	36.937500000000	-4.9253618264423D-12
171	4.4400624893966D-08	49.250000000000	-2.4141325159051D-12
172	-1.5707963267949	-36.937500000000	1.5680000000000D-05

173	-1.5707963267949	-49.250000000000	1.8360000000000D-05
174	-1.7453293062579	-36.937500000000	1.4613806478636D-05
175	-1.7453292959202	-49.250000000000	1.5823036250514D-05
176	-1.5707963267949	-24.625000000000	1.3160000000000D-05
177	-1.7453292754984	-24.625000000000	1.3869131517254D-05
178	-1.5707963267949	-12.312500000000	8.1480000000000D-06
179	-1.7453292030675	-12.312500000000	1.2622063498638D-05
180	-1.5707963267949	0.	0.
181	-1.7453292117515	0.	1.0982383498385D-05
182	-1.5707963267949	12.312500000000	7.3740000000000D-06
183	-1.7453292591006	12.312500000000	1.1746975926682D-05
184	-1.5707963267949	24.625000000000	1.2080000000000D-05
185	-1.7453292330449	24.625000000000	1.2853511980209D-05
186	-1.5707963267949	36.937500000000	1.3990000000000D-05
187	-1.7453293011935	36.937500000000	1.3274103296497D-05
188	-1.5707963267949	49.250000000000	1.5690000000000D-05
189	-1.7453291919049	49.250000000000	1.4463009775340D-05
190	-1.3962633473319	-36.937500000000	1.4613806478636D-05
191	-1.3962633576696	-49.250000000000	1.5823036250514D-05
192	-1.3962633780914	-24.625000000000	1.3869131517254D-05
193	-1.3962634505223	-12.312500000000	1.2622063498638D-05
194	-1.3962634418383	0.	1.0982383498385D-05
195	-1.3962633944892	12.312500000000	1.1746975926682D-05
196	-1.3962634205449	24.625000000000	1.2853511980209D-05
197	-1.3962633523963	36.937500000000	1.3274103296497D-05
198	-1.3962634616849	49.250000000000	1.4463009775340D-05
199	-1.9198621890549	-49.250000000000	1.0786549960393D-05
200	-1.9198622126628	-36.937500000000	1.0675129564763D-05
201	-1.9198621073355	-24.625000000000	1.1869772740249D-05
202	-1.9198621999993	-12.312500000000	1.3076437303462D-05
203	-1.9198621521119	0.	1.3378123569872D-05
204	-1.9198622014542	12.312500000000	1.2812494201724D-05
205	-1.9198621066836	24.625000000000	1.1303075862210D-05
206	-1.9198622323289	36.937500000000	9.7840804664130D-06
207	-1.9198621859538	49.250000000000	9.2076926226064D-06
208	-2.0943951434944	-36.937500000000	3.0801620181092D-06
209	-2.0943951226304	-49.250000000000	2.5711109462944D-06
210	-2.0943951272970	-24.625000000000	3.8923010464451D-06
211	-2.0943951398757	-12.312500000000	4.5527156148206D-06
212	-2.0943950709521	0.	4.8690671917756D-06
213	-2.0943950648375	12.312500000000	4.8479043930672D-06
214	-2.0943951117671	24.625000000000	4.2282049373399D-06
215	-2.0943951132302	36.937500000000	3.5195108502554D-06
216	-2.0943950943380	49.250000000000	3.6595346398883D-06
217	-1.2217304409270	-36.937500000000	1.0675129564763D-05
218	-1.2217304645349	-49.250000000000	1.0786549960393D-05
219	-1.2217305462543	-24.625000000000	1.1869772740249D-05
220	-1.2217304535905	-12.312500000000	1.3076437303462D-05
221	-1.2217305014779	0.	1.3378123569872D-05
222	-1.2217304521356	12.312500000000	1.2812494201724D-05
223	-1.2217305469062	24.625000000000	1.1303075862210D-05
224	-1.2217304212608	36.937500000000	9.7840804664130D-06
225	-1.2217304676360	49.250000000000	9.2076926226064D-06
226	-1.0471975100954	-36.937500000000	3.0801620181092D-06
227	-1.0471975309594	-49.250000000000	2.5711109462944D-06
228	-1.0471975262928	-24.625000000000	3.8923010464451D-06
229	-1.0471975137141	-12.312500000000	4.5527156148206D-06
230	-1.0471975826377	0.	4.8690671917756D-06
231	-1.0471975887523	12.312500000000	4.8479043930672D-06

232	-1.0471975418227	24.625000000000	4.2282049373399D-06
233	-1.0471975403596	36.937500000000	3.5195108502554D-06
234	-1.0471975592518	49.250000000000	3.6595346398883D-06
235	-2.2689279471409	-49.250000000000	-1.4486043676840D-06
236	-2.2689281036071	-36.937500000000	-2.7733822957384D-06
237	-2.2689279104967	-24.625000000000	-4.0827136847642D-06
238	-2.2689280254862	-12.312500000000	-5.3522679285598D-06
239	-2.2689280478887	0.	-5.7317767838149D-06
240	-2.2689279397876	12.312500000000	-4.9154869582748D-06
241	-2.2689280136658	24.625000000000	-3.2511552630665D-06
242	-2.2689280095997	36.937500000000	-1.7861665138869D-06
243	-2.2689280607141	49.250000000000	-6.8992840321956D-07
244	-2.4434608767776	-36.937500000000	-1.2588227223633D-06
245	-2.4434609329784	-49.250000000000	4.4430652012895D-07
246	-2.4434608837908	-24.625000000000	-3.6928669700838D-06
247	-2.4434609548985	-12.312500000000	-6.6816373070273D-06
248	-2.4434609324960	0.	-8.1718711932067D-06
249	-2.4434608500488	12.312500000000	-6.4162132989699D-06
250	-2.4434609667189	24.625000000000	-3.2107975120964D-06
251	-2.4434609707850	36.937500000000	-5.1424802526172D-07
252	-2.4434609196706	49.250000000000	1.8059580812158D-06
253	-0.87266454998266	-36.937500000000	-2.7733822957384D-06
254	-0.87266470644885	-49.250000000000	-1.4486043676840D-06
255	-0.87266474309312	-24.625000000000	-4.0827136847642D-06
256	-0.87266462810357	-12.312500000000	-5.3522679285598D-06
257	-0.87266460570114	0.	-5.7317767838149D-06
258	-0.87266471380215	12.312500000000	-4.9154869582748D-06
259	-0.87266463992403	24.625000000000	-3.2511552630665D-06
260	-0.87266464399008	36.937500000000	-1.7861665138869D-06
261	-0.87266459287571	49.250000000000	-6.8992840321956D-07
262	-0.69813177681223	-36.937500000000	-1.2588227223633D-06
263	-0.69813172061141	-49.250000000000	4.4430652012895D-07
264	-0.69813176979897	-24.625000000000	-3.6928669700838D-06
265	-0.69813169869133	-12.312500000000	-6.6816373070273D-06
266	-0.69813172109376	0.	-8.1718711932067D-06
267	-0.69813180354102	12.312500000000	-6.4162132989699D-06
268	-0.69813168687086	24.625000000000	-3.2107975120964D-06
269	-0.69813168280482	36.937500000000	-5.1424802526172D-07
270	-0.69813173391918	49.250000000000	1.8059580812158D-06
271	-2.6179938577543	-49.250000000000	9.1801327800904D-06
272	-2.6179938368903	-36.937500000000	7.8402099992322D-06
273	-2.6179938530877	-24.625000000000	6.5804991643571D-06
274	-2.6179938405090	-12.312500000000	4.0743200405252D-06
275	-2.6179939094326	0.	0.
276	-2.6179938479209	12.312500000000	3.6863976649411D-06
277	-2.6179938686176	24.625000000000	6.0405556256813D-06
278	-2.6179938671545	36.937500000000	6.9969299032233D-06
279	-2.6179938860467	49.250000000000	7.8434880349719D-06
280	-2.7925267677219	-36.937500000000	1.5872441206563D-05
281	-2.7925267913298	-49.250000000000	1.5376200349751D-05
282	-2.7925267489221	-24.625000000000	1.7561262454523D-05
283	-2.7925267803854	-12.312500000000	1.9300384723434D-05
284	-2.7925268282728	0.	1.9157230308464D-05
285	-2.7925267789305	12.312500000000	1.8164911457579D-05
286	-2.7925267447459	24.625000000000	1.6059240052049D-05
287	-2.7925267480557	36.937500000000	1.3786013100016D-05
288	-2.7925267944309	49.250000000000	1.2655901165758D-05
289	-0.52359881669953	-36.937500000000	7.8402099992322D-06
290	-0.52359879583553	-49.250000000000	9.1801327800904D-06

291	-0.52359880050209	-24.625000000000	6.5804991643571D-06
292	-0.52359881308079	-12.312500000000	4.0743200405252D-06
293	-0.52359874415719	0.	0.
294	-0.52359880566892	12.312500000000	3.6863976649411D-06
295	-0.52359878497223	24.625000000000	6.0405556256813D-06
296	-0.52359878643526	36.937500000000	6.9969299032233D-06
297	-0.52359876754311	49.250000000000	7.8434880349719D-06
298	-0.34906588586790	-36.937500000000	1.5872441206563D-05
299	-0.34906586225997	-49.250000000000	1.5376200349751D-05
300	-0.34906590466765	-24.625000000000	1.7561262454523D-05
301	-0.34906587320437	-12.312500000000	1.9300384723434D-05
302	-0.34906582531700	0.	1.9157230308464D-05
303	-0.34906587465928	12.312500000000	1.8164911457579D-05
304	-0.34906590884385	24.625000000000	1.6059240052049D-05
305	-0.34906590553405	36.937500000000	1.3786013100016D-05
306	-0.34906585915894	49.250000000000	1.2655901165758D-05
307	-2.9670596617362	-49.250000000000	1.2235530405896D-05
308	-2.9670596741268	-36.937500000000	1.3452053264016D-05
309	-2.9670595747999	-24.625000000000	1.5948827152525D-05
310	-2.9670596466244	-12.312500000000	1.8434039032913D-05
311	-2.9670597686332	0.	1.9108418118758D-05
312	-2.9670595880862	12.312500000000	1.7729664598982D-05
313	-2.9670597473398	24.625000000000	1.4548758362578D-05
314	-2.9670596791912	36.937500000000	1.1572770253881D-05
315	-2.9670596492489	49.250000000000	9.8981743323099D-06
316	-0.17453297946304	-36.937500000000	1.3452053264016D-05
317	-0.17453299185364	-49.250000000000	1.2235530405896D-05
318	-0.17453307878992	-24.625000000000	1.5948827152525D-05
319	-0.17453300696537	-12.312500000000	1.8434039032913D-05
320	-0.17453288495656	0.	1.9108418118758D-05
321	-0.17453306550359	12.312500000000	1.7729664598982D-05
322	-0.17453290625004	24.625000000000	1.4548758362578D-05
323	-0.17453297439857	36.937500000000	1.1572770253881D-05
324	-0.17453300434087	49.250000000000	9.8981743323099D-06
325	-3.6651914702893	-36.937500000000	-7.8402099992322D-06
326	-3.6651914494253	-49.250000000000	-9.1801327800904D-06
327	-3.6651914540919	-24.625000000000	-6.5804991643571D-06
328	-3.6651914666706	-12.312500000000	-4.0743200405252D-06
329	-3.6651913977470	0.	0.
330	-3.6651914592587	12.312500000000	-3.6863976649411D-06
331	-3.6651914385620	24.625000000000	-6.0405556256813D-06
332	-3.6651914400251	36.937500000000	-6.9969299032233D-06
333	-3.6651914211329	49.250000000000	-7.8434880349719D-06
334	-3.4906585394577	-36.937500000000	-1.5872441206563D-05
335	-3.4906585158498	-49.250000000000	-1.5376200349751D-05
336	-3.4906585582574	-24.625000000000	-1.7561262454523D-05
337	-3.4906585267942	-12.312500000000	-1.9300384723434D-05
338	-3.4906584789068	0.	-1.9157230308464D-05
339	-3.4906585282491	12.312500000000	-1.8164911457579D-05
340	-3.4906585624336	24.625000000000	-1.6059240052049D-05
341	-3.4906585591238	36.937500000000	-1.3786013100016D-05
342	-3.4906585127487	49.250000000000	-1.2655901165758D-05
343	-3.3161256330528	-36.937500000000	-1.3452053264016D-05
344	-3.3161256454434	-49.250000000000	-1.2235530405896D-05
345	-3.3161257323797	-24.625000000000	-1.5948827152525D-05
346	-3.3161256605552	-12.312500000000	-1.8434039032913D-05
347	-3.3161255385464	0.	-1.9108418118758D-05
348	-3.3161257190934	12.312500000000	-1.7729664598982D-05
349	-3.3161255598398	24.625000000000	-1.4548758362578D-05

350	-3.3161256279884	36.937500000000	-1.1572770253881D-05
351	-3.3161256579307	49.250000000000	-9.8981743323099D-06
352	-3.1415926845412	-36.937500000000	-3.1811570459104D-12
353	-3.1415927091569	-49.250000000000	-4.8980059471230D-12
354	-3.1415926982086	-24.625000000000	-5.0271875903228D-12
355	-3.1415926831394	-12.312500000000	-3.7578087085528D-12
356	-3.1415926800028	0.	-3.3989396322871D-12
357	-3.1415927258696	12.312500000000	-8.4079121696551D-12
358	-3.1415926989209	24.625000000000	-4.3219256290310D-12
359	-3.1415927204802	36.937500000000	-4.9253618262717D-12
360	-3.1415926979904	49.250000000000	-2.4141325175787D-12
361	3.6651914494253	-49.250000000000	9.1801327800904D-06
362	3.6651914702893	-36.937500000000	7.8402099992322D-06
363	3.6651914540919	-24.625000000000	6.5804991643571D-06
364	3.6651914666706	-12.312500000000	4.0743200405252D-06
365	3.6651913977470	0.	0.
366	3.6651914592587	12.312500000000	3.6863976649411D-06
367	3.6651914385620	24.625000000000	6.0405556256813D-06
368	3.6651914400251	36.937500000000	6.9969299032233D-06
369	3.6651914211329	49.250000000000	7.8434880349719D-06
370	3.4906585394577	-36.937500000000	1.5872441206563D-05
371	3.4906585158498	-49.250000000000	1.5376200349751D-05
372	3.4906585582574	-24.625000000000	1.7561262454523D-05
373	3.4906585267942	-12.312500000000	1.9300384723434D-05
374	3.4906584789068	0.	1.9157230308464D-05
375	3.4906585282491	12.312500000000	1.8164911457579D-05
376	3.4906585624336	24.625000000000	1.6059240052049D-05
377	3.4906585591238	36.937500000000	1.3786013100016D-05
378	3.4906585127487	49.250000000000	1.2655901165758D-05
379	3.3161256454434	-49.250000000000	1.2235530405896D-05
380	3.3161256330528	-36.937500000000	1.3452053264016D-05
381	3.3161257323797	-24.625000000000	1.5948827152525D-05
382	3.3161256605552	-12.312500000000	1.8434039032913D-05
383	3.3161255385464	0.	1.9108418118758D-05
384	3.3161257190934	12.312500000000	1.7729664598982D-05
385	3.3161255598398	24.625000000000	1.4548758362578D-05
386	3.3161256279884	36.937500000000	1.1572770253881D-05
387	3.3161256579307	49.250000000000	9.8981743323099D-06

SOFTWARE TO MODEL AXAF IMAGE QUALITY

APPENDIX 7

SAMPLE SESSION

Appendix 7 Sample Sessions

A7.1 Sample session for command mode GRAZTRACE

```
zeus{chen}63> gt2




                          ****************************
                          *                          *
                          *         GrazTrace        *
                          *                          *
                          ****************************





GTRACE>res sample ! restore from file "sample"
GTRACE>wsp ! random ray trace option
   WSP>go ! execute the option
1   1000 successful rays in wspot1,
 random ray distribution on first surface annulus
 rmin=   0.7505025549956299E+02,   rmax=   0.7640170861803300E+02
 azmin (radians)=  -0.3141592653589793E+01,   azmax (radians)=
0.3141592653589793E+01
 field angle (radians)=   0.0000000000000000E+00
 azimuth (radians)    =   0.0000000000000000E+00


                           0 rays were vignetted or obscured
                           0 rays failed in ssrt

 energy( 1)=  -0.1000000000000000E+01,   effective area=   0.6430219044059306E+03
 energy( 2)=   0.2770000000000000E+00,   effective area=   0.4770593063472806E+03
 energy( 3)=   0.5728000000000000E+00,   effective area=   0.4832782607539592E+03
GTRACE>fcs ! refocus option
   FCS>go ! execute the option

  weighted planar focus: energy( 1)=  -0.1000000000000000E+01
 number of rays=    1000
```

```
   ***  stored rays modified   ***

 delta z  =  -0.5466777139285604E-11,   net zshift=  -0.5466777139285604E-11
 new x average=  -0.8423862330255359E-16,   new y average=
0.8400834138655648E-15

GTRACE>wst ! average position and rms option
   WST>go ! execute the option

  length statistics for: energy( 1)=  -0.1000000000000000E+01
 number of rays=    1000,   field angle (radians)=   0.0000000000000000E+00
 net zshift=  -0.5466777139285604E-11
 x average=  -0.8423862330255359E-16,   y average=   0.8400834138655648E-15
 xrms     =   0.2011815720717621E-13,   yrms     =   0.1974284384504887E-13
           rms=   0.2818723350211297E-13
 xmin=  -0.7117447022322689E-13,   xmax=   0.7377058711339266E-13
 ymin=  -0.7081536115041014E-13,   ymax=   0.6300754746166225E-13
    weight sum=   0.6430219044059306E+03
 weight average=   0.6430219044059307E+00
    weight rms=   0.0000000000000000E+00
 wmin=   0.6430219044059191E+00,   wmax=   0.6430219044059191E+00


  arc sec statistics for: energy( 1)=  -0.1000000000000000E+01
 assumed focal length=   0.6564832312844800E+03,   number of rays    1000
 x average (arc sec)=  -0.2646748993119960E-13
 y average (arc sec)=   0.2639513613368916E-12
 xrms (arc sec)     =   0.6321056807905900E-11
 yrms (arc sec)     =   0.6203134621577281E-11
 rms (arc sec)      =   0.8856333231207158E-11

GTRACE>spo ! spot diagram option
   SPO>go ! execute the option
1 spot diagram: first   1000 rays of   1000 stored
 assumed center:  x =  -0.8423862330255359E-16,   y =   0.8400834138655648E-1!

Press <Enter> to continue ......
```

```
0   x-axis

   0.810E-13                          I
   0.720E-13                          *
   0.630E-13                        * I    *
   0.540E-13                       *  ***
   0.450E-13                        * * ** *  *
   0.360E-13                    ** **********
   0.270E-13                  ** ************** **
   0.180E-13                 * ************** ***
   0.900E-14                  *********************
   0.000E+00 ---------------*********************-------------
  -0.900E-14             *   *******************
  -0.180E-13                * ********************
  -0.270E-13                  ** ***************
  -0.360E-13                  * ****************
  -0.450E-13               *    *  ********* **
  -0.540E-13                  *    ***I* ** *  *
  -0.630E-13                        * *
  -0.720E-13                          I* **
  -0.810E-13                          I
              L                       M                       U
y-axis  -0.134344E-12           -0.474399E-14           0.124856E-12
```

GTRACE>**zra** ? !check z range
zrange =   1.0000000000000D+50
GTRACE>**zra 10000** ! try to change z range
GTRACE>**zra ?** ! check it again
zrange =   10000.0000000000
GTRACE>**foc ?** ! check focal length
foclen =   656.48323128448
GTRACE>**exi** ! exit the program
EXITING THE PROGRAM ? (Y/N)**y**
zeus{chen}63>

A7.2 Sample Session for Command Mode with Interactive Help

```
zeus{chen}63>gt2



                     ****************************
                     *                          *
                     *        GrazTrace         *
                     *                          *
                     ****************************



GTRACE>help

    HEL

        Help

        Help only will automatically provide the information
        about latest command entered before help.

        Help followed by a command will provide information
        about that command.

        Help followed by any unknown command will list all
        GRAZTRACE commands.

    See also: "?".

GTRACE>
GTRACE>help

    Unknown command

                GRAZTRACE commands list

    ADA     AMA     APE     AZI     AZM     CAN     DAZ
    DEB     DET     DIS     DXC     DXR     DYC     DYR
    EDI     EFF     ELE     ENE     ERR     EXI     FCS
    FDF     FOC     GO      GRI     GR2     HEL     IEN
    IND     ITI     LEN     LIS     MAT     MAX     MOD
```

```
     MOV     NAZ     NFR     NLO     NRA     NRG     OBS
     PAS     PRI     RAD     RES     RLI     RST     RSV
     SAV     SDA     SOU     SPO     SUR     SYS     THI
     THR     TIL     TIT     TYP     VIG     WGT     WSP
     WST     WS2     XCE     XWI     YCE     YWI     ZRA
     ?

     See manual or Type HELp for further information


GTRACE>hel ?

     ?

          Help and inquiry

     ? only serves as help command,
     ? in data field entry will allow to check current value,

     See also HEL.

GTRACE>res sample
GTRACE>hel

     RES filspec

          Restore system from prescription file

          filspec - file name

     See also: SAV, LIS.

GTRACE>wsp
   WSP>hel

     WSP

          Random ray trace

          WSP traces nra successful rays randomly arranged on
          the first surface annulus at location Z=0.
          Intercept, slopes, and effective area weights are
          stored for the last surface for each ray.

          Options:

          AZM azimus_middle_angle, (default is 0)
          DAZ delta_azimus_angle, (default 2 pi)
          NRA number_of_rays, (default 1000)

          GO for executing the analysis,
          CAN for cancelling the analysis.

     See also: WS2, GRI, GR2, RSV.

GTRACE>wsp
   WSP>go
1   1000 successful rays in wspot1,
```

```
 random ray distribution on first surface annulus
 rmin=   0.7505025549956299E+02,   rmax=   0.7640170861803300E+02
 azmin (radians)=  -0.3141592653589793E+01,   azmax (radians)=
0.3141592653589793E+01
 field angle (radians)=   0.0000000000000000E+00
 azimuth (radians)    =   0.0000000000000000E+00


                              0 rays were vignetted or obscured
                              0 rays failed in ssrt

 energy( 1)=  -0.1000000000000000E+01,   effective area=   0.6430219044059306E+
 energy( 2)=   0.2770000000000000E+00,   effective area=   0.4770593063472806E+
 energy( 3)=   0.5728000000000000E+00,   effective area=   0.4832782607539592E+
GTRACE>hel

     GO

          Execution option

          GO executes the analysis using all previously
            entered option inputs and then return control to
          the command level.

     See also: CAN.

GTRACE>hel can

     CAN

          Cancel option

          CAN cancels all inputs to the analysis and
          return control to the command level.

     See also: GO.

GTRACE>exi
EXITING THE PROGRAM ? (Y/N)y
zeus{chen}60>
```

A7.3 Sample Session for Command Mode with Deformation

A7.3.1 Deformation data from COSMOS/M

```
zeus{chen}60>gt2




                        ****************************
                        *                          *
                        *         GrazTrace        *
                        *                          *
                        ****************************





GTRACE>res sample
GTRACE>wsp
   WSP>go
1   1000 successful rays in wspot1,
 random ray distribution on first surface annulus
 rmin=   0.7505025549956299E+02,   rmax=   0.7640170861803300E+02
 azmin (radians)=  -0.3141592653589793E+01,   azmax (radians)=
0.3141592653589793E+01
 field angle (radians)=   0.0000000000000000E+00
 azimuth (radians)    =   0.0000000000000000E+00


                            0 rays were vignetted or obscured
                            0 rays failed in ssrt

 energy( 1)=  -0.1000000000000000E+01,   effective area=   0.6430219044059306E+03
 energy( 2)=   0.2770000000000000E+00,   effective area=   0.4770593063472806E+03
 energy( 3)=   0.5728000000000000E+00,   effective area=   0.4832782607539592E+03
GTRACE>spo
   SPO>go
1 spot diagram: first    1000 rays of   1000 stored
 assumed center:  x =   0.0000000000000000E+00,   y =   0.0000000000000000E+00

Press <Enter> to continue ......
```

```
 0   x-axis

   0.720E-12                          I
   0.640E-12                      ** *****
   0.560E-12                   ****************
   0.480E-12                ******    I    ******
   0.400E-12               ****       I       ****
   0.320E-12             ***          I        ****
   0.240E-12            ***           I         ****
   0.160E-12           ***            I          ****
   0.800E-13           ***            I          ****
   0.000E+00 ---------***-------------I----------****---------
  -0.800E-13          ****            I           ***
  -0.160E-12           ***            I           **
  -0.240E-12           ***            I          ***
  -0.320E-12            ***           I         ****
  -0.400E-12             ***          I        ***
  -0.480E-12             ******       I       ****
  -0.560E-12                ******    I   *******
  -0.640E-12                  ****************
  -0.720E-12                      ******
             L                        M                       U
y-axis  -0.115911E-11          -0.710543E-14           0.114489E-11
GTRACE>fdf 5 sxicos.dfm

 surface    5 uses file:
 sxicos.dfm
 in storage area 1


 deformation surface data from file:
 sxicos.dfm
 in storage area 1
 sxi from cosmos/m
      1001 azimuthal bins,          201 axial bins
 azimuthal limits (radians)  -0.3141592741012573E+01  0.3141592741012573E+01
 azimuthal increment (radians)   0.6283185482025147E-02
 axial limits  -0.4950000000000000E+02  0.4950000000000000E+02
 axial increment   0.4950000000000000E+00


GTRACE>fdf 11 sxicos.dfm


 surface    5 uses file:
 sxicos.dfm
 in storage area 1


 surface   11 uses file:
 sxicos.dfm
 in storage area 1


 deformation surface data from file:
 sxicos.dfm
 in storage area 1
```

```
 sxi from cosmos/m
      1001 azimuthal bins,            201 axial bins
 azimuthal limits (radians)  -0.3141592741012573E+01  0.3141592741012573E+01
 azimuthal increment (radians)   0.6283185482025147E-02
 axial limits  -0.4950000000000000E+02  0.4950000000000000E+02
 axial increment   0.4950000000000000E+00


GTRACE>typ ?
itype(  1) = flat
itype(  2) = flat
itype(  3) = flat
itype(  4) = flat
itype(  5) = grzcon01
itype(  6) = flat
itype(  7) = flat
itype(  8) = flat
itype(  9) = flat
itype(  10) = flat
itype(  11) = grzcon01
itype(  12) = flat
itype(  13) = flat
itype(  14) = flat
itype(  15) = flat
itype(  16) = flat
GTRACE>typ 5 grazcon13
GTRACE>typ 11 grzcon13
GTRACE>typ ?
itype(  1) = flat
itype(  2) = flat
itype(  3) = flat
itype(  4) = flat
itype(  5) = grzcon13
itype(  6) = flat
itype(  7) = flat
itype(  8) = flat
itype(  9) = flat
itype(  10) = flat
itype(  11) = grzcon13
itype(  12) = flat
itype(  13) = flat
itype(  14) = flat
itype(  15) = flat
itype(  16) = flat
GTRACE>wsp
   WSP>go
1   1000 successful rays in wspot1,
 random ray distribution on first surface annulus
 rmin=   0.7505025549956299E+02,   rmax=   0.7640170861803300E+02
 azmin (radians)=  -0.3141592653589793E+01,   azmax (radians)=
0.3141592653589793E+01
 field angle (radians)=   0.0000000000000000E+00
 azimuth (radians)    =   0.0000000000000000E+00


                                  0 rays were vignetted or obscured
                                  0 rays failed in ssrt
```

```
 energy( 1)=  -0.1000000000000000E+01,   effective area=   0.6430219044059306E+
 energy( 2)=   0.2770000000000000E+00,   effective area=   0.4770593401842925E+
 energy( 3)=   0.5728000000000000E+00,   effective area=   0.4832782976740266E+
GTRACE>spo
   SPO>go
1 spot diagram: first    1000 rays of   1000 stored
 assumed center:  x =   0.0000000000000000E+00,    y =   0.0000000000000000E+0(

Press <Enter> to continue ......
```

```
0   x-axis

  0.360E-02                        I
  0.320E-02                        I
  0.280E-02                       *I*
  0.240E-02                   *    I*
  0.200E-02                    *  *I     **
  0.160E-02                    ** ***    **
  0.120E-02                    ******   *
  0.800E-03                     **********
  0.400E-03                   ********* * *
  0.000E+00 -----------**-*************-****--***-*---------
 -0.400E-03          *  **  ****************  *
 -0.800E-03                  * *********
 -0.120E-02                     ********
 -0.160E-02                     ****   *
 -0.200E-02                      ***   ***
 -0.240E-02                 *   * *I     *
 -0.280E-02                   *    I     *
 -0.320E-02                  *     I
 -0.360E-02                        I      *
           L                        M                        U
y-axis  -0.569277E-02          0.672324E-04          0.582723E-02
GTRACE>exi
EXITING THE PROGRAM ? (Y/N)y
```

A7.3.2 Deformation data from NASTRAN

```
zeus{chen}57> gt2
```

```
                         ****************************
                         *                          *
                         *        GrazTrace         *
                         *                          *
                         ****************************
```

```
GTRACE>res sample
GTRACE>wsp
   WSP>go
1   1000 successful rays in wspot1,
 random ray distribution on first surface annulus
 rmin=   0.7505025549956299E+02,   rmax=   0.7640170861803300E+02
 azmin (radians)=  -0.3141592653589793E+01,   azmax (radians)=
0.3141592653589793E+01
 field angle (radians)=   0.0000000000000000E+00
 azimuth (radians)    =   0.0000000000000000E+00


                         0 rays were vignetted or obscured
                         0 rays failed in ssrt

 energy( 1)=  -0.1000000000000000E+01,   effective area=   0.6430219044059306E+
 energy( 2)=   0.2770000000000000E+00,   effective area=   0.4770593063472806E+
 energy( 3)=   0.5728000000000000E+00,   effective area=   0.4832782607539592E+
GTRACE>spo
   SPO>go
1 spot diagram: first    1000 rays of   1000 stored
 assumed center:  x =   0.0000000000000000E+00,   y =   0.0000000000000000E+0(

Press <Enter> to continue ......
```

```
0   x-axis

   0.720E-12                            I
   0.640E-12                        ** *****
   0.560E-12                     ****************
   0.480E-12                  ******    I     ******
   0.400E-12                 ****       I        ****
   0.320E-12                ***         I         ****
   0.240E-12               ***          I          ****
   0.160E-12              ***           I           ****
   0.800E-13              ***           I           ****
   0.000E+00 ---------***------------I----------****---------
  -0.800E-13             ****           I            ***
  -0.160E-12              ***           I            **
  -0.240E-12              ***           I           ***
  -0.320E-12               ***          I          ****
  -0.400E-12                ***         I         ***
  -0.480E-12                ******      I        ****
  -0.560E-12                  ******    I    *******
  -0.640E-12                    ****************
  -0.720E-12                         ******
                L                       M                       U
y-axis  -0.115911E-11         -0.710543E-14           0.114489E-11
GTRACE>fdf 5 sxinas.dfm


 surface   5 uses file:
 sxinas.dfm
 in storage area 1


 deformation surface data from file:
 sxinas.dfm
 in storage area 1
 sxi from nastran
      1001 azimuthal bins,          201 axial bins
 azimuthal limits (radians)  -0.3141592741012573E+01  0.3141592741012573E+01
 azimuthal increment (radians)   0.6283185482025147E-02
 axial limits  -0.4950000000000000E+02  0.4950000000000000E+02
 axial increment   0.4950000000000000E+00


GTRACE>fdf 11 sxinas.dfm


 surface   5 uses file:
 sxinas.dfm
 in storage area 1


 surface  11 uses file:
 sxinas.dfm
 in storage area 1


 deformation surface data from file:
 sxinas.dfm
```

```
 in storage area 1
 sxi from nastran
      1001 azimuthal bins,          201 axial bins
 azimuthal limits (radians)  -0.3141592741012573E+01  0.3141592741012573E+01
 azimuthal increment (radians)   0.6283185482025147E-02
 axial limits  -0.4950000000000000E+02  0.4950000000000000E+02
 axial increment   0.4950000000000000E+00


GTRACE>typ 5 grzcon13
GTRACE>typ 11 grzcon13
GTRACE>typ ?
itype(  1) = flat
itype(  2) = flat
itype(  3) = flat
itype(  4) = flat
itype(  5) = grzcon13
itype(  6) = flat
itype(  7) = flat
itype(  8) = flat
itype(  9) = flat
itype(  10) = flat
itype(  11) = grzcon13
itype(  12) = flat
itype(  13) = flat
itype(  14) = flat
itype(  15) = flat
itype(  16) = flat
GTRACE>wsp
   WSP>go
1   1000 successful rays in wspot1,
 random ray distribution on first surface annulus
 rmin=   0.7505025549956299E+02,   rmax=   0.7640170861803300E+02
 azmin (radians)=  -0.3141592653589793E+01,   azmax (radians)=
0.3141592653589793E+01
 field angle (radians)=   0.0000000000000000E+00
 azimuth (radians)    =   0.0000000000000000E+00


                              0 rays were vignetted or obscured
                              0 rays failed in ssrt

 energy( 1)=  -0.1000000000000000E+01,   effective area=   0.6430219044059306E+
 energy( 2)=   0.2770000000000000E+00,   effective area=   0.4770593398247984E+
 energy( 3)=   0.5728000000000000E+00,   effective area=   0.4832782973241684E+
GTRACE>spo
   SPO>go
1 spot diagram: first    1000 rays of   1000 stored
 assumed center:  x =   0.0000000000000000E+00,   y =   0.0000000000000000E+0(

Press <Enter> to continue ......
```

```
0   x-axis

   0.540E-02                                  I
   0.480E-02                                 *I
   0.420E-02                                  I
   0.360E-02                             *    I
   0.300E-02                                 *I*
   0.240E-02                              ** *I*
   0.180E-02                               ****   **
   0.120E-02                               **** ***
   0.600E-03                              * ***** *
   0.000E+00 --------------************_******--------------
  -0.600E-03                          ** ************    *
  -0.120E-02                              * *******
  -0.180E-02                                *** **
  -0.240E-02                              * ***   **
  -0.300E-02                             *    I    *
  -0.360E-02                                  I
  -0.420E-02                             *   *I     *
  -0.480E-02                                  I
  -0.540E-02                                  I
             L                                 M                        U
y-axis  -0.845256E-02               0.187438E-03             0.882744E-02
GTRACE>exi
EXITING THE PROGRAM ? (Y/N)y
zeus{chen}58>
```

A7.4 Sample Running Structure Analysis Interface

A7.4.1 Convert data from COSMOS/M list file

```
zeus{chen}58>drinfea

  NAME OF INPUT DATA FILE (1-132 CHARS. - ALPH/NUM/UNSC) ?
sxccos.lis

  NAME OF OUTPUT DATA FILE (1-132 CHARS. -ALPH/NUM/UNSC) ?
sxicos.dfm

  INPUT DESIRED HEADER MESSAGE (MAX. OF 20 LINES/80 CHARS. PER LINE)  ?
 (END WITH ctrl-d ON A LINE BY ITSELF)
sxi from cosmos/m
CONVERT CARTESIEN DATA TO CYLINDICAL (Y/N)y
INPUT DATA FROM COSMOS/M / NASTRAN (C/N)c
NEED MODIFY COORDINATE (Y/N)y
KEY IN SHIFT AND SCALE t0 ts z0 zs dr0 drs
0 1 0 10 0 10
THETA        SHIFT AND SCALE t0=  0. ts=    1.0000000000000
Z            SHIFT AND SCALE z0=  0. zs=   10.0000000000000
DELTA RADIUS SHIFT AND SCALE dr0=  0. drs=   10.0000000000000
CORRECT ? (Y/N)y
CHANGE AXIAL LENGTH(990.6)(Y/N)y
KEY IN AXIAL LENGTH lz
99
AXIAL LENGTH lz=    99.0000
CORRECT ? (Y/N)y
 transform to graztrace coordinates
 extend distribution by    0.61086525519689 radians
 INPUT FILE HAS   387 SEGMENTS OF DATA
 BEGIN INTERPOLATION OF DATA
 EXECUTION TIME FOR INTERPOLATION OF DATA (SECONDS) =     25.2500
 INTERPOLATED VALUES HAVE BEEN WRITTEN TO OUTPUT FILE
zeus{chen}47>
```

A7.4.2 Convert data from NASTRAN standard output

```
zeus{chen}47> drinfea

  NAME OF INPUT DATA FILE (1-132 CHARS. - ALPH/NUM/UNSC) ?
sxinas.out

  NAME OF OUTPUT DATA FILE (1-132 CHARS. -ALPH/NUM/UNSC) ?
sxinas.dfm

  INPUT DESIRED HEADER MESSAGE (MAX. OF 20 LINES/80 CHARS. PER LINE)  ?
 (END WITH ctrl-d ON A LINE BY ITSELF)
sxi from nastran
CONVERT CARTESIEN DATA TO CYLINDICAL (Y/N)y
INPUT DATA FROM COSMOS/M / NASTRAN (C/N)n
NEED MODIFY COORDINATE (Y/N)y
KEY IN SHIFT AND SCALE t0 ts z0 zs dr0 drs
0 1 0 10 0 10
THETA        SHIFT AND SCALE t0=  0. ts=    1.0000000000000
Z            SHIFT AND SCALE z0=  0. zs=   10.0000000000000
DELTA RADIUS SHIFT AND SCALE dr0=  0. drs=   10.0000000000000
CORRECT ? (Y/N)y
CHANGE AXIAL LENGTH(990.6)(Y/N)y
KEY IN AXIAL LENGTH lz
99
AXIAL LENGTH lz=    99.0000
CORRECT ? (Y/N)y
 transform to graztrace coordinates
 extend distribution by    0.61086525519689 radians
 INPUT FILE HAS   387 SEGMENTS OF DATA
 BEGIN INTERPOLATION OF DATA
 EXECUTION TIME FOR INTERPOLATION OF DATA (SECONDS) =     25.5900
 INTERPOLATED VALUES HAVE BEEN WRITTEN TO OUTPUT FILE
zeus{chen}50>
```

APPENDIX 8

DATA BASE FOR AXAF TECHNICAL DOCUMENTATION USER MANUAL

TO LOG ON TO DATA BASE

ID:	wanda
PASSWORD:	jazzy10
	xwin
	wpp
	Ctrl F5 FILE%

Use word perfect commands to go to the end of file to input records.

OR

AFTER TYPING IN PASSWORD DO THE FOLLOWING
vi FILE%

Use the getting started with SunOS Quick Reference
to move around in text editor files.

TO LOGOFF USING WORD PERFECT

Ctrl F5 and follow steps to save as unix text.

TO LOCATE A FILE BY NAME NOT SENDING TO PRINTER

at the venus prompt, type: grep NAME FILE%

TO LOCATE A FILE AND SORT THE FILE

at the venus prompt, type: grep NAME FILE%>> OUT

go into word perfect and sort, then send to the printer

Getting Started with SunOS
Quick Reference

This quick reference lists the commands presented in this manual concisely by function. Each listing includes a syntax diagram, and a brief description of the command.

1. Work Session

1.1. Log In

Type username to system login prompt.
Type password to password prompt.

1.2. Change Password

Type `passwd`, followed by old password, and repeat new password.

1.3. Log Out

Type `logout` or CTRL-D depending upon system setup.

2. File System

2.1. Create File

Type `cat >`*filename*, then text ending with CTRL-D, or see **Editing Files.**

2.2. Make (or Create) Directory

Type `mkdir` *directory-name*.

2.3. Look at File

Type `cat` *filename*
or `more` *filename*.

2.4. Print File

Type `lpr` *filename*.

2.5. List Files and Directories

Type

`ls` for listing of current directory

`ls` *directory-name*
for listing of another directory

`ls` *filename*
for listing of a single file

`ls -t` or

`ls -t` *filename* or

`ls -t` *directory-name*
to get a listing reverse sorted by time of last modification

`ls -F` or

`ls -F` *directory-name*
to get a listing that marks directory names by appending a / character to them.

2.6. Move (or Rename) Files and Directories

Type

`mv` *source-filename destination-filename*
to rename a file

`mv` *source-filename destination-directory*
to move a file into another directory

`mv` *source-directory-name destination-directory-name* to rename a directory, or move it into another directory.

2.7. Copy Files

Type

`cp` *source-filename destination-filename*
to copy a file into another filename

`cp` *source-filename destination-directory*
to copy a file into another directory

2.8. Remove (or Delete) File

Type

`rm` *filename*
to remove a file

`rmdir` *directory-name*
to remove an empty directory

`rm -r` *directory-name*
to remove a directory and its contents.

2.9. Change Working Directory

Type

`cd` to change directories to your home directory

`cd` *directory-name*
to change directories to another directory.

2.10. Find Name of Current Directory

Type `pwd`.

2.11. Pathnames

simple:	One filename or directory name to access local file or directory.
absolute:	List of directory names from root directory (first /) to desired filename or directory name, each name separated by /.
relative:	List of directory names from current position to desired filename or directory name, each name separated by /.

2.12. Directory Abbreviation

~	Home directory.
~*username*	Another user's home directory.
.	Working directory.
..	Parent of working directory.

3. Commands

3.1. Date and Time

Type `date`. For universal time (Greenwich Mean Time) type `date -u`

3.2. Calendar

Type

`cal` *year*
for yearly calendar

`cal` *month-number year*
for monthly calendar

3.3. Wild Cards

- `?` Single character wild card.
- `*` Arbitrary number of characters.

3.4. Redirecting Output

System types output of command to file rather than screen, replacing current contents of file, if any. Type *command-name* > *filename*.

System types output of command to file rather than screen, appending to current contents of file, if any. Type *command-name* >> *filename*.

3.5. Basic Calculator

Type `bc` to enter interactive program. Type arithmetic expressions, using +, -, *, and / symbols, followed by RETURN. To change number of decimal places, type `scale =` *number*.

4. Editing Files

Type `vi` to enter text editor, then any of following commands (in command mode, unless preceded by a `:`):

- `a` to add text
- `cc` to substitute a line with a string (enters insert mode)
- `cw` to substitute, or change, a word with a string (enters insert mode)
- `dd` to delete the entire line the cursor is on
- `dw` to delete the word, or portion of word, under and after the cursor
- `h` to move left, or "west," one character
- `i` to insert text under the cursor (enters insert mode)
- `j` to move down, or "south," one line
- `k` to move up, or "north," one line
- `l` to move right, or "east," one character
- `o` to insert text on a new blank line after the current line (enters insert mode)
- `O` to insert text on a new blank line before the current line (enters insert mode)
- `s` to substitute a character with a string (enters insert mode)
- `x` to delete the character under the cursor
- `:q` to quit `vi`
- `:q!` to quit `vi`, without writing changes
- `:w` to save, or write a file

5. Formatting Files

Construct source file to run through `nroff` formatter, including any of the following commands:

- `.LP` to *left-justify* a paragraph
- `.IP` to created an *itemized* paragraph (like this one)
- `.ce` to center text on the page
- `.ul` to underline portions of text
- `.sp` to create a blank line space
- `.br` to force the end of a line, a line break

To format the source file, type `nroff -ms` *source-filename*. You will probably want to redirect the output of `nroff` into a *destination-filename*, so you can print it out afterward.

6. Search Files

Type

`grep` *search-string filename*
to type out lines containing the string in a specific file

`grep` *search-string filename(s)*
to type out lines containing the string in more than one file

`grep -v` *search-string filename(s)*
to type out lines that **don't** contain the string

7. Timesavers

7.1. Aliases

To "alias," or abbreviate a command string with an alias string, type `alias` *alias-string command-string*.

8. History: Command Repetition

- `!!` Repeat the entire last command line at any point in the current command line.
- `!$` Repeat the last word of the last command line at any point in the current command line.

9. Run Command in Background: Job Control

To run a command in the background, as opposed to the more common method of running commands in the foreground, type a `&` after the command line. Then, you can type more commands to the command prompt, or even run more commands in the background for simultaneous command execution.

10. Online Documentation

To see online Man Pages, type `man` *command-name*.