



NRL Memorandum Report 6709

DISPLAY3D

A Graphics Preprocessor for CHIEF

C. M. Siders

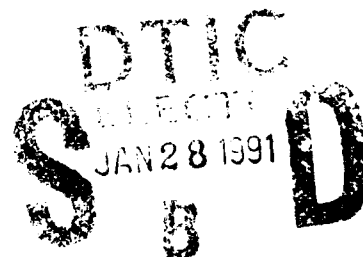
*Transducer Branch
Underwater Sound Reference Detachment
Naval Research Laboratory
P.O. Box 568337
Orlando, FL 32856-8337*

and

D. M. Bolling, Jr.

*Texas Research Institute, Inc.
9063 Bee Caves Rd.
Austin, TX 78733-6201*

27 December 1990



AD-A231 345

REPORT DOCUMENTATION PAGE			Form Approved OMB No. 0704-0188	
Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188), Washington, DC 20503.				
1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE 27 December 1990	3. REPORT TYPE AND DATES COVERED Final		
4. TITLE AND SUBTITLE DISPLAY3D - A Graphics Preprocessor for CHIEF		5. FUNDING NUMBERS PE #62314N WU #59-0593-0-0 WU Assn #DN880-326		
6. AUTHOR(S) C. M. Siders, Code 5976 and D. M. Bolling, Jr., Texas Research Institute, Inc.				
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Standards Branch, Underwater Sound Reference Detachment, Naval Research Laboratory PO Box 568337 Orlando, FL 32856-8337		8. PERFORMING ORGANIZATION REPORT NUMBER NRL Memorandum Report 6709		
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) Office of Naval Research (ONT 231) 800 N. Quincy St. Arlington, VA 22217		10. SPONSORING/MONITORING AGENCY REPORT NUMBER		
11. SUPPLEMENTARY NOTES None				
12a. DISTRIBUTION/AVAILABILITY STATEMENT Approved for public release; distribution unlimited.		12b. DISTRIBUTION CODE		
13. ABSTRACT (Maximum 200 words) The computer program CHIEF [G.W. Benthien, D. Barach, and D. Gillette, "CHIEF Users Manual," Naval Ocean Systems Center Report 920, Sep 88] is designed to obtain approximate solutions to exterior steady-state acoustic radiation problems for surfaces of arbitrary shapes vibrating with a prescribed normal velocity. Being unable to view the shapes defined by CHIEF can lead to many difficult debugging problems as well as incorrect solutions. DISPLAY3D provides points as though the data were locations on a three-dimensional surface or solid and a description of how these points are connected. DISPLAY3D has several options that makes it a versatile package. The code permits the user to change the orientation from which the data are viewed. DISPLAY3D also provides the option of viewing the direction of the normal velocity vector as defined by CHIEF. These options, as well as others, will be explained in detail.				
14. SUBJECT TERMS CHIEF CID graphics		15. NUMBER OF PAGES 91		
		16. PRICE CODE		
17. SECURITY CLASSIFICATION OF REPORT Unclassified	18. SECURITY CLASSIFICATION OF THIS PAGE Unclassified	19. SECURITY CLASSIFICATION OF ABSTRACT Unclassified	20. LIMITATION OF ABSTRACT SAR	

(Blank Page)

CONTENTS

INTRODUCTION	1
CODE DESCRIPTION	1
PLOTCID (First Option)	3
PLOTCHIEF	4
CODE OPERATION	5
CODE INSTALLATION	9
ACKNOWLEDGMENT	10
REFERENCES	10
APPENDIX A - A CHIEF Driver of a Free-Flooded Cylinder	11
APPENDIX B - TEST2.CRD and TEST2.CON	15
TEST2.CRD, The Coordinate File	15
TEST2.CON, The Connectivity File	20
APPENDIX C - Two Examples of Graphics Drivers	25
APPENDIX D - Listing of Programs and Subroutines in DISPLAY3D	31
PLOTCID	31
PLOTCHIEF	37
LOAD_CID_DATA()	41
PUT_POINT()	46
RECTANGULAR_PLANE()	48
CIRCULAR_PLANE()	50
ELLIPTICAL_PLANE()	52
CIRCULAR_CYLINDER()	54
ELLIPTICAL_CYLINDER()	56
SPHEROID()	58
PROLATE_OBLATE_SPHEROID()	61

For	☒
	☐
	☐
by	
ty Codes	
and/or	
Special	

A-1

TOROID()	64
QUADRILATERAL()	67
AXISYMMETRIC()	69
TRIANGLE()	72
CONE()	74
CONNECT_ARROWS()	76
ROT3D()	77
DR_ARROW()	84

DISPLAY3D

A Graphics Preprocessor for CHIEF

INTRODUCTION

DISPLAY3D processes CHIEF defined geometries into a format that can be easily displayed. The computer program CHIEF [1] is designed to obtain approximate solutions to exterior steady-state acoustic radiation problems for surfaces of arbitrary shapes vibrating with a prescribed normal velocity. Being unable to view the shapes defined by CHIEF can lead to many difficult debugging problems as well as incorrect solutions. DISPLAY3D provides points as though the data were locations on a three-dimensional surface or solid and a description of how these points are connected. A graphics program commonly called a graphics driver, is required to display these data points. Since there are so many graphics standards (protocols) and graphics devices available it would be virtually impossible to develop a generalized graphics program; however two examples of graphics drivers are provided. If these drivers cannot be used on available graphics devices, the user may write a graphics program that can read DISPLAY3D output files, or use one of the commercial plotting packages available. The format of the data and an example of a graphics driver can be found later in this report. Because most research laboratories have graphics capabilities, the task of taking the results of DISPLAY3D and plotting it on an available graphics device usually requires little effort.

DISPLAY3D has several options that makes it a versatile package. The code permits the user to change the orientation from which the data is viewed. DISPLAY3D also provides the option of viewing the direction of the normal velocity vector as defined by CHIEF. These options, as well as others, will be explained in detail.

CODE DESCRIPTION

The DISPLAY3D program has two options for reading the CHIEF geometry data. The first option uses the main program PLOTCHIEF to read the data from the output file CID (Chief Interactive Driver) [2] which is a computer program that interactively generates the control routine for CHIEF. While creating a CHIEF control program, CID captures the information from the user's interactive session and saves it in a separate data file which is read the next time CID is executed. This data file is also used for option 1 in the DISPLAY3D package. The second option uses the subroutine PLOTCHIEF which is placed in the CHIEF driver program after the last LDSURR call. PLOTCHIEF shares the same variables as the CHIEF driver program through the use of a

FORTTRAN COMMON statement. Both of these options generate two output files which include all translations, rotations, and reflections as prescribed in the CHIEF program.

The hierarchy of the two options is given in Fig. 1. The program names in quotes indicate the programs from which the CHIEF surface data are obtained for each option. The advantage of option 1 is that the CHIEF driver does not have to be compiled, linked, and executed in order to create the graphing data files. However, if for some reason CID was not executed, the second option is provided.

DISPLAY3D

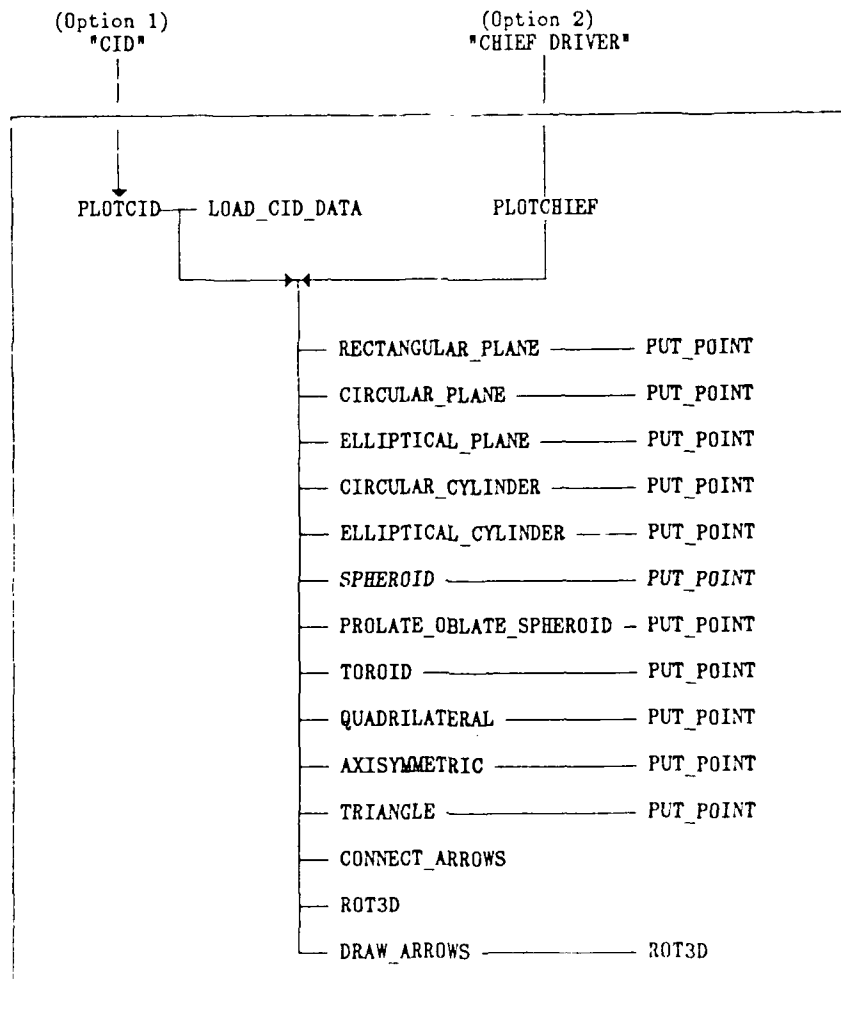


Fig. 1 - The program hierarchy of DISPLAY3D.

PLOTCID (First Option)

PLOTCID is the main program of the computer code and is designed as the control program for ten different geometry subroutines. This program determines the necessary rotations or reflections for each surface in a block. This program generates two files that are used for displaying the three-dimensional wire mesh of the CHIEF surfaces. The names and descriptions of the subroutines are given below.

LOAD_CID_DATA	Reads necessary data from a CID save file into variables used by PLOTCID.
PUT_POINT	Translates a 3D point from a local coordinate system to the global coordinate system. Depending upon the symmetry condition chosen in CHIEF, this subroutine reflects or rotates the point. PUT_POINT stores these points in an array which is rotated by subroutine ROT3D.
RECTANGULAR_PLANE	Generates points and lines for drawing a rectangular planar region.
CIRCULAR_PLANE	Generates points and lines for drawing a circular planar region or part of one.
ELLIPTICAL_PLANE	Generates points and lines for drawing an elliptical planar region or part of one.
CIRCULAR_CYLINDER	Generates points and lines for drawing a cylinder or section of a cylinder.
ELLIPTICAL_CYLINDER	Generates points and lines for drawing an elliptical cylinder or section of cylinder.
SPHEROID	Generates points and lines for drawing a spheroid or section of a spheroid.

PROLATE_OBLATE_SPHEROID	Generates points and lines for drawing a prolate spheroid or section of a prolate spheroid. This same routine is also used for generating points and lines for an oblate spheroid.
TOROID	Generates points and lines for drawing a toroid or part of a toroid.
QUADRILATERAL	Generates points and lines for drawing these finite-element-type inputs using linear or quadratic interpolation over a quadrilateral.
AXISYMMETRIC	Generates points and lines for drawing these finite-element-type inputs using linear or quadratic axisymmetric interpolation.
TRIANGLE	Generates points and lines for drawing these finite-element-type inputs using linear or quadratic interpolation over a triangle.
ROT3D	Rotates coordinates by a user specified angle for different viewing perspectives and stores these points in a data file.
CONNECT_ARROWS	Generates points used to describe the connectivity of the reference axis and associated labels.
DRAW_ARROWS	Generates the coordinates of the points needed for the reference axes and associated labels.
PLOTCHIEF	This is a subroutine that is placed in the CHIEF control program after the last call to LDSURR. PLOTCHIEF and PLOTCID are identical after the initial variables are determined.

CODE OPERATION

Depending upon which option of DISPLAY3D is used (see section entitled CODE DESCRIPTION) the user must access the appropriate graphics code. As a first example, option 2 will be exercised on a free-flooded cylinder given in sample run 2 of the CHIEF [1] users manual. Appendix A contains the CHIEF driver program with the subroutine call to PLOTCHIEF as follows:

```
CALL PLOTCHIEF(RUNID,NBLKS,SYMTYP,SUBDIV,AX,AY,AZ)..
```

This subroutine call is highlighted with an arrow in the left-hand margin. The variables RUNID, NBLKS, and SYMTYP are all local to the CHIEF driver; however, the user has the ability to vary RUNID and NBLKS independently of the CHIEF driver. For example, the user might want to set NBLKS to 1 in the PLOTCHIEF call in order to display only the user-defined section regardless of symmetry type. If NBLKS is set to a positive value, the direction of the normal velocities (SIGN(IZAX)) will be identified by dashed and solid-lines. But if NBLKS is negative, only solid-line drawings will be produced. The variable SUBDIV is a user-specified flag. When the flag is set (SUBDIV=1), PLOTCHIEF will honor the subdivisions (NSU and NSV) used in the CHIEF driver. If the flag is cleared (SUBDIV=0), PLOTCHIEF uses default values for the number of subdivisions which should be optimal for viewing. The user might set the SUBDIV flag equal to one in order to determine visually whether the proper number of subdivisions was chosen. The variables AX, AY, and AZ are the angles in degrees used to rotate the figure about its X, Y, and Z axis, respectively. See Figure 2.

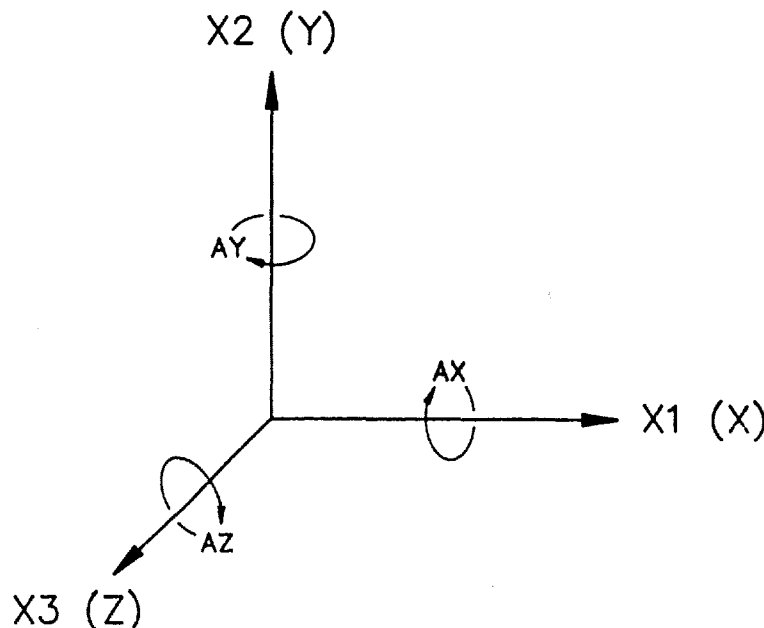


Figure 2 illustrates the direction of the rotation angles.

Since rotations are not associative, the user must know the order in which the rotations are performed. The first rotation is around the object's X axis, next around its Y axis and then about its Z axis. It is important to note that after a rotation around the X axis has occurred the object's Y and Z axes are no longer the global Y and Z axes.

This CHIEF driver must be compiled, linked, and executed and once executed, two data files are created for graphing purposes. The extensions of these files are CRD and CON, coordinate and connectivity data, respectively. For this example, the files created are TEST2.CON and TEST2.CRD. The format of the coordinate data is 3(E15.0), and the format of the connectivity of the points is 3(I12). The connectivity file contains an array of three numbers. The first two entries determine the start and stop of a line segment. The third entry is the color. The sign of the color entry determines if the line segment will be solid (positive) or dashed (negative). The value of this entry determines the color of the line segment depending on the characteristics of the graphics output device. These colors can be changed by changing the parameters COLOR_USER_DEFINED and COLOR_TRANSFORMED at the top of the PLOTCHIEF and PLOTCHIEF programs. These parameters represent the user defined section of the drawing and the transformed portion. An example of both of these data files are listed in Appendix B.

These data files can be plotted using almost any available plotting package as long as the protocol has "move to a point" and "draw a line from one point to another" commands. Two graphics drivers are included in Appendix C for the user's convenience. These drivers provide the basic graphing capability. The first driver, DRAW240, can be used on any VT240 or compatible terminal. This program reads in the coordinate and connectivity data, scales it, and displays the drawing on the screen. The user can modify this basic driver by adding zoom, rotation, scaling, or more advanced color capability. Also, an option to save the data to file rather than displaying the drawing on the screen could easily be added. The second driver provided is written in PV-WAVE [4] and uses PV-WAVE's powerful graphing and data manipulation functions. This driver creates a ReGIS (remote graphics instruction set) display file for viewing, or a QMS (Quality MicroSystems) file for dumping to a Talaris or QMS printer.

Figure 3 is the display of the example described above with a 60,10,0 viewing perspective with the negative normals illustrated by dashed lines and positive normals with solid lines. Figure 3 was plotted using the PV-WAVE driver on a Talaris 800 printer. The reference axes in the lower left corner of Fig. 3 are rotated in the same manner as the CHIEF display.

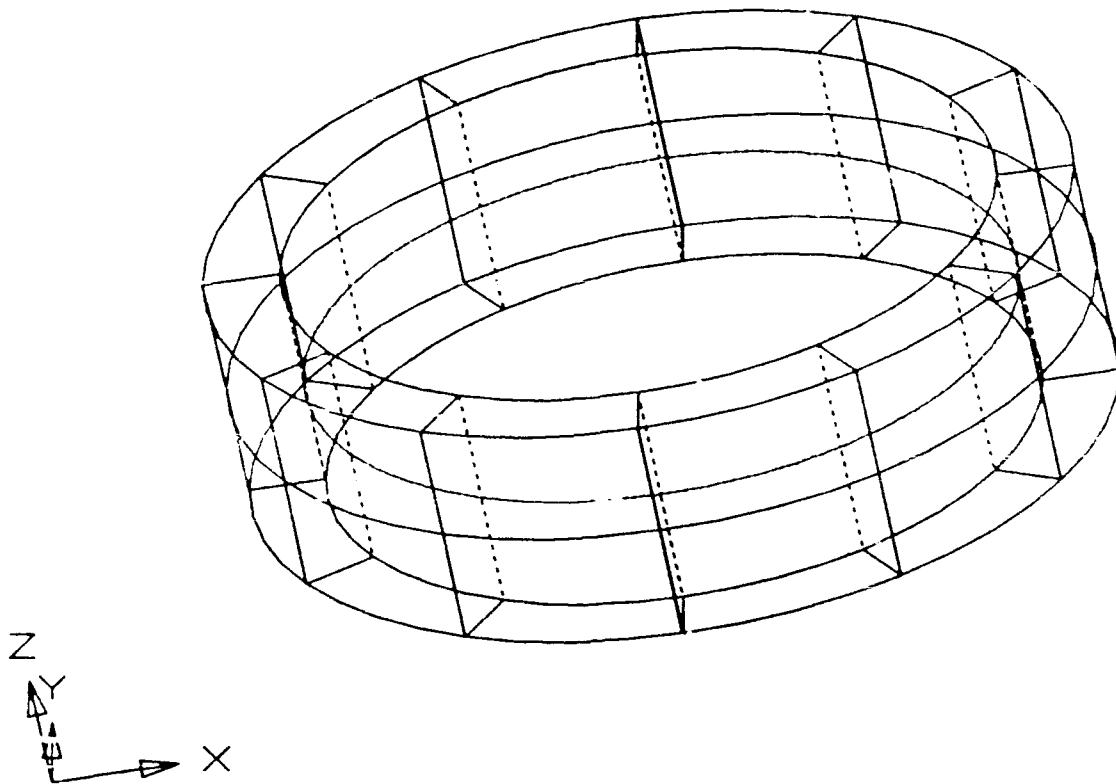


Fig. 3 - A free flooded cylinder using the
DISPLAY3D plotting routine.

If the CID software is used to create the CHIEF control program, the user can access DISPLAY3D by implementing option 1 by the command RUN PLOT CID. The program prompts for the name of the file that was created by CID and the rotation angle (see Fig. 4). The output of PLOT CID is identical to files created by PLOT CHIEF provided that the CHIEF surfaces are identical for both program runs.

Siders and Bolling

Enter the filename used in the CID run: TEST2

DISPLAY:

- 1.) Only the user defined surfaces.
- 2.) User defined surfaces with selected transformations.

Enter 1 or 2 (default 2): 2

THE NUMBER OF SUBDIVISIONS PLOTTED:

- 1.) The optimal number for visualization.
- 2.) The number defined in the CHIEF driver.

Enter 1 or 2 (default 1): 1

DISTINGUISH BETWEEN POSITIVE AND NEGATIVE NORMAL VELOCITY:

- 1.) Yes. (solid lines for positive, dashed for negative)
- 2.) No. (only solid lines)

Enter 1 or 2 (default 2): 1

Enter the angles (in degrees) for rotating the model
about the X, Y, and Z axes. (default 0.0, 0.0, 0.0)

(XROT, YROT, ZROT) : 60,10,0

Coordinate data has been stored in TEST2.CRD

Connection data has been stored in TEST2.CON

Fig. 4 - An interactive run from DOTCID program.

A second example is an array of hemispheres generated using the CHIEF program. The display is shown in Fig. 5. DISPLAY3D can handle all of the CHIEF surfaces including the finite-element-type inputs; however, the number of subdivisions in DISPLAY3D is chosen to best view the three-dimensional surface.



Fig. 5 - An array of hemispheres drawn using DISPLAY3D program.

CODE INSTALLATION

The authors have been using the DISPLAY3D program on the VAX and MICROVAX computers running under the VMS 5.1 operating system. The source code is device independent, but any graphics driver is device dependent. A listing of the source code is given in Appendix D.

To receive more information on DISPLAY3D or the Fortran computer code, contact Tina Siders at P.O. Box 568337, Orlando, FL 32856-8337.

ACKNOWLEDGMENT

This work was performed in the Transducer Technology Project and supported by the Office of Naval Technology.

REFERENCES

1. G.W. Benthien, D. Barach and D. Gillette, "CHIEF Users Manual," Naval Ocean Systems Center Report 970, Sep 1988.
2. C.M. Siders and R.A. Raymond, "CHIEF Preprocessor," NRL Report 6536, Sep 1989.
3. David F. Rogers and J. Alan Adams, *Mathematical Elements for Computer Graphics*, (McGraw-Hill, Inc., New York, NY, 1976).
4. Precision Visuals, Inc, *PV-WAVE User's Guide*, (Boulder, CO, 1988).

APPENDIX A - A CHIEF Driver of a Free-Flooded Cylinder

This driver appears in the CHIEF user's manual as example 2.

```

C      PROGRAM TEST2
C      Book Examples

C      VARIABLE DECLARATION

C      ***** CONTROL 88 *****

C      PROGRAM CHIEF88 DRIVER
C      MXSREG - MAXIMUM NUMBER OF SURFACE REGIONS
C      MXIPS  - MAXIMUM NUMBER OF INTERIOR POINTS
C      MXARS  - MAXIMUM NUMBER OF SURFACE SUBDIVISIONS/SYM BLK
C      MXGAUS - MAXIMUM ORDER OF GAUSSIAN QUADRATURE
C      MXQPTS - MAXIMUM NUMBER OF QUADRATURE POINTS/SUBDIVISION
C      MXBLKS - MAXIMUM NUMBER OF SYMMETRY BLOCKS
C      MXFFP  - MAXIMUM NUMBER OF FAR-FIELD POINTS
C      MXNFP  - MAXIMUM NUMBER OF NEAR-FIELD POINTS
C      MXPTSC - MAXIMUM NUMBER OF POINT SOURCES
C      MAXCOR - MAXIMUM NUMBER OF FINITE ELEMENT NODES
C      MXFPS  -  $\frac{1}{2} \times (MXARS + MXIPS + MXFFP + MXNFP)$ 
C      PARAMETER (MXSREG=500)
C      PARAMETER (MAXCOR=1000)
C      PARAMETER (MXIPS=20)
C      PARAMETER (MXARS=500)
C      PARAMETER (MXGAUS=64)
C      PARAMETER (MXQPTS=512)
C      PARAMETER (MXBLKS=100)
C      PARAMETER (MXFFP=361)
C      PARAMETER (MXNFP=361)
C      PARAMETER (MXPTSC=20)
C      PARAMETER (MXFPS=520)
C      PARAMETER (MXFPS=MAX0(MXARS+MXIPS,MXFFP,MXNFP))
C      PARAMETER (NWDVEC=2*MXARS)

C      INPUT COMMONS

C      COMMON/CONST/RHO,C
C      COMMON/PRTCOM/NUNPRT,NUNERR
C      COMMON/PRTSD/RUNID,DATE
C      CHARACTER*4 RUNID
C      CHARACTER*8 DATE
C      COMMON/NDASG/NDQPTS,NDPMXS,NDVMXS,NDDECM,NDVELS,NDSPS,
1      NDPMXF,NDVMXF,NDPMXN,NDVMXN,NDPSSP,NDEXPR,NDCOMV,
1      NDEMP,NDZROB,NDPATB
C      COMMON/SVALS/NSREG,NSEQNS(MXSREG),SUL(MXSREG),SUU(MXSREG),
1      SVL(MXSREG),SVU(MXSREG),NSU(MXSREG),NSV(MXSREG),
1      CCS(10,MXSREG),TRNSS(3,MXSREG),IZAX(MXSREG),
1      IORDU(MXSREG),IORDV(MXSREG),NCCEQS
C      COMMON/COORDS/MAXCOR,3)
C      COMMON/IPTS/NUMIPS,IPXS(3,MXIPS)
C      REAL IPXS
C      COMMON/PTSINP/NUMPTS,PTSRC(4,MXPTSC),PTSWT(MXPTSC),
1      IOPTSC(MXPTSC)
C      COMPLEX PTSWT
C      COMMON/PLWINP/AINC,THINC,PHIINC,ISCATR
C      COMMON/BAFFLE/INBFAF
C      COMMON/FFINP/NUMTHP,THTPHI(2,MXFFP)
C      COMMON/NFINP/NUMFPN,NFPXS(3,MXNFP)
C      REAL NFPXS

C      OUTPUT COMMONS

C      COMMON/TAPREC/RECRD(10),IRECRD(30)
C      COMMON/TAPRC1/ARECRD(10)
C      CHARACTER*4 ARECRD
C      COMMON/PROVLS/NDIMPV,NUMARS,NUMSEF,NUMFFP,NUMNFP,NWDVEC
C      COMMON/SURARS/AREAS(MXARS)
C      COMMON/ODSVEC/IVECT(MXARS),B(MXARS),IPIVTR(MXFPS)

```

Siders and Bolling

```

COMPLEX TVECT
COMMON/VELSPS/VEL(MXARS),SP(MXARS)
COMPLEX VEL,SP
COMMON/PDISL/POWER,DIRIND,SRCLVL
COMMON/FFVALS/FFP(MXFFP),PNRMFF(MXFFP),IFFNRM,RMFNRM
COMPLEX FFP
COMMON/TSCOM/TGTSTH(MXFFP)
COMMON/NFVALS/NFP(MXNFP),PNRMNF(MXNFP),INFNRM,RMNRM
COMPLEX NFP
COMMON/PTSCOM/PTSSP(MXARS)
COMPLEX PTSSP
COMMON/EXTCOM/EXTPRS(MXFPS),IEXTFG
COMPLEX EXTPRS
COMMON/NBPRTC/IRHSPT,NARSPT,NPTBLK,FRQPT
COMMON/NBPRTS/SYMTPT
CHARACTER*3 SYMTPT

DIMENSION CC(10), TRNS(3), IELTS(8,300)
real x1(1000),y1(1000)

CHARACTER*3 SYMTYP
CHARACTER*4 FLDTYP,TAPEID,PRTTYP
INTEGER      XIZAX,XNSEQNS,XIRG,XNSU,XNSV,XIORDU,XIORDV
INTEGER      ICOOR,IELEM

COMPLEX PMATX(342 ), VMATX(342 )
MDSIZE = 342
RUNID = 'TES'
DATE='14-APR-89'
CALL INITCM
CALL OPNSFL
RHO = 1000
C = 1500

OPEN(UNIT=NUNPRT,FILE=RUNID//'.OUT',STATUS='NEW',
1  FORM='FORMATTED')

C      FREQUENCY AND SYMMETRY INPUTS

      PI=ACOS(-1.0)
      FREQ=238.7

      SYMTYP='REF'
      NBLKS=2 ** 3
      IRHSYM = 1

      CONVERT=1.0

C      SURFACE REGION INPUTS

      ROTLIM=PI/NBLKS

      NSREG= 3

      DO 1 I=1,10
1      CC(I)=0.0
      DO 2 I = 1,3
2      TRNS(I)=0.0

C      TOP
      XIRG= 1
      XNSEQNS= 2
      CC( 1)= .6 * CONVERT
      TRNS(1)=0.00E+00* CONVERT
      TRNS(2)=0.00E+00* CONVERT
      TRNS(3)=0.00E+00* CONVERT
      XIZAX=-.3
      XSUL=1.68 * CONVERT
      XSUJ=2.04 * CONVERT
      XSVL=0.000000
      XSVU=1.57

```

NRL Memorandum Report 6709

```

XNSU=2
XNSV=3
XIORDU=4
XIORDV=16
CALL LDSURR(XIRG,XNSEQNS,CC,TRNS,XIZAX,
1          XSUL,XSUU,XSVL,XSVU,XNSU,XNSV,XIORDU,XIORDV)

C  OUTSIDE
XIRG= 2
XNSEQNS= 4
CC( 1)=2.04      * CONVERT
TRNS(1)=0.00E+00* CONVERT
TRNS(2)=0.00E+00* CONVERT
TRNS(3)=0.00E+00* CONVERT
XIZAX=+3
XSUL=0.0      * CONVERT
XSUU= 6      * CONVERT
XSVL=0.0
XSVU=1.57
XNSU=2
XNSV=3
XIORDU=4
XIORDV=16
CALL LDSURR(XIRG,XNSEQNS,CC,TRNS,XIZAX,
1          XSUL,XSUU,XSVL,XSVU,XNSU,XNSV,XIORDU,XIORDV)

C  INSIDE
XIRG= 3
XNSEQNS= 4
CC( 1)=1.68      * CONVERT
TRNS(1)=0.00E+00* CONVERT
TRNS(2)=0.00E+00* CONVERT
TRNS(3)=0.00E+00* CONVERT
XIZAX=-3
XSUL=0      * CONVERT
XSUU= 6      * CONVERT
XSVL=0
XSVU=1.57
XNSU=2
XNSV=3
XIORDU=4
XIORDV=16
CALL LDSURR(XIRG,XNSEQNS,CC,TRNS,XIZAX,
1          XSUL,XSUU,XSVL,XSVU,XNSU,XNSV,XIORDU,XIORDV)

C
C  PLOTCHIEF parameters are:
C
C  RUNID - name for graphics files created.
C
C  NBLKS  1 - displays only user-defined surfaces.
C          NBLKS - (defined by CHIEF driver) displays the surfaces
C              including symmetries.
C
C  SYMTYP - defined in CHIEF.
C
C  ISUBDIV  1 - displays surfaces using CHIEF defined
C              parameters NSU and NSV.
C              0 - displays surfaces using optimal subdivisions
C                  for viewing.
C
C  AX - (REAL) angle in degrees to rotate view about X axis
C
C  AY - (REAL) angle in degrees to rotate view about Y axis
C
C  AZ - (REAL) angle in degrees to rotate view about Z axis
C
C
CALL PLOTCHIEF('TEST2', NBLKS, SYMTYP, 0, 60 0, 10 0, 0 0)

```

Siders and Bolling

```

CALL PRTOUT('GEOM',1)

C   GENERATE SURFACE P AND V MATRICES
CALL SURMAT(FREQ,SYMTYP,NBLKS,PMATX,VMATX,MDSIZE)
CALL PRTOUT('AREA',1)

C   DECOMPOSE MATRICIES
CALL DECOMM(SYMTYP,NBLKS,IRHSYM,PMATX,VMATX,MDSIZE)
CALL IOSUB(NDVELS,10,VEL,0)

C   BLOCK # 1
      VEL( 1) = (0.0000,0.0000)
      VEL( 2) = (0.0000,0.0000)
      VEL( 3) = (0.0000,0.0000)
      VEL( 4) = (0.0000,0.0000)
      VEL( 5) = (0.0000,0.0000)
      VEL( 6) = (0.0000,0.0000)
      VEL( 7) = (0.0000,0.0000)
      VEL( 8) = (0.0000,0.0000)
      VEL( 9) = (0.0000,0.0000)
      VEL(10) = (0.0000,0.0000)
      VEL(11) = (0.0000,0.0000)
      VEL(12) = (0.0000,0.0000)
      VEL(13) = (0.0000,0.0000)
      VEL(14) = (0.0000,0.0000)
      VEL(15) = (0.0000,0.0000)
      VEL(16) = (0.0000,0.0000)
      VEL(17) = (0.0000,0.0000)
      VEL(18) = (0.0000,0.0000)
CALL IOSUB(NDVELS,1,VEL,NWDVEC)

C   GENERATE SURFACE PRESSURES

CALL SURPRS(FREQ,SYMTYP,NBLKS,IRHSYM,PMATX,VMATX,MDSIZE)

CALL PRTOUT('SP ',1)

      FLDTYP = 'FAR'
      NUMTHP = 19
      DO 25 I = 1,NUMTHP
      THTPHI(1,I) = (I-1)*5
      THTPHI(2,I) = 0.0
25  CONTINUE

C   CALCULATE FAR FIELD MATRICIES
CALL FLDMAT(FREQ,SYMTYP,FLDTYP,NBLKS,PMATX,VMATX,MDSIZE)

      IFFNRM = 19

C   CALCULATE FAR-FIELD PRESSURES
CALL FLDPRS(FREQ,FLDTYP,NBLKS,IRHSYM,PMATX,VMATX,MDSIZE)

CALL PRTOUT(FLDTYP,0)

STOP
END

```

NRL Memorandum Report 6709

APPENDIX B - TEST2.CRD and TEST2.CON

TEST2.CRD is the coordinate file.

X	Y	Z	X	Y	Z
-1.550288	-0.7643659	-0.4413067	-1.550288	-0.7643659	-0.4413067
-1.536148	-0.8717934	-0.2502503	-1.536148	-0.8717934	-0.2502503
-1.493970	-0.9730662	-5.9966244E-02	-1.493970	-0.9730662	-5.9966244E-02
-1.424474	-1.066453	0.1262928	-1.424474	-1.066453	0.1262928
-1.328849	-1.150358	0.3053430	-1.328849	-1.150358	0.3053430
-1.208730	-1.223347	0.4741242	-1.208730	-1.223347	0.4741242
-1.066168	-1.284172	0.6297510	-1.066168	-1.284172	0.6297510
-0.9036023	-1.331794	0.7695637	-0.9036023	-1.331794	0.7695637
-0.7238098	-1.365398	0.8911722	-0.7238098	-1.365398	0.8911722
-0.5298645	-1.384410	0.9924980	-0.5298645	-1.384410	0.9924980
-0.3250813	-1.388506	1.071809	-0.3250813	-1.388506	1.071809
-0.1129605	-1.377614	1.127750	-0.1129605	-1.377614	1.127750
0.1026719	-1.351922	1.159364	0.1026719	-1.351922	1.159364
-1.904818	-0.8185040	-0.4725634	1.758665	-0.2590785	-0.1495778
-1.887648	-0.9489513	-0.2405663	1.744528	-0.3708224	3.8985357E-02
-1.836432	-1.071926	-0.5071597E-03	1.702347	-0.4849768	0.2218322
-1.752045	-1.135325	0.2166646	1.632852	-0.5995883	0.3958373
-1.635929	-1.287209	0.4340828	1.537227	-0.7126979	0.5580264
-1.490070	-1.375838	0.6390312	1.417108	-0.8223723	0.7056271
-1.316959	-1.449697	0.8280068	1.274548	-0.9267365	0.8361166
-1.119557	-1.507524	0.9977794	1.111980	-1.024007	0.9472645
-0.9012382	-1.548329	0.145447	0.9321876	-1.112521	1.037171
-0.6657331	-1.571415	1.268485	0.7382423	-1.190766	1.104299
-0.4170677	-1.576388	1.364792	0.5334591	-1.257403	1.147501
-0.1594926	-1.563163	1.432720	0.3213384	-1.311295	1.166039
0.1025897	-1.531965	1.471108	0.1055059	-1.351520	1.159596
-2.009007	-0.3067828	-0.1771211	2.113196	-0.2049384	-0.1183212
-1.991837	-0.4372306	5.4876041E-02	2.096028	-0.3406296	-0.1106484
-1.940621	-0.5602046	0.2859351	2.044810	-0.4792459	0.3326767
-1.856234	-0.6736033	0.5121069	1.960423	-0.6184171	0.5439687
-1.740118	-0.7754080	0.7295250	1.844307	-0.7557644	0.7409126
-1.594258	-0.8641173	0.9344736	1.698447	-0.8889402	0.9201420
-1.421148	-0.9379763	1.123449	1.525337	-1.015668	1.078594
-1.223748	-0.9958025	1.293227	1.327935	-1.133783	1.213559
-1.005427	-1.036608	1.4408.9	1.109616	-1.241264	1.322731
-0.7699220	-1.059694	1.563928	0.8741109	-1.338275	1.404243
-0.5212568	-1.064667	1.660234	0.6254455	-1.417192	1.456703
-0.2636815	-1.051442	1.728162	0.3678704	-1.482633	1.479214
-1.5992371E-03	-1.020244	1.766550	0.1057881	-1.531477	1.471390
-1.904818	-0.8185040	-0.4725634	2.009007	0.3067828	0.1771211
-1.887648	-0.9489518	-0.2405663	1.991837	0.1710913	0.4060907
-1.836432	-1.071926	-0.5071597E-03	1.940621	3.2475308E-02	0.6281190
-1.752045	-1.185325	0.2166646	1.856234	-0.1066958	0.8394110
-1.635929	-1.287209	0.4340828	1.740118	-0.2440431	1.036355
-1.490070	-1.375838	0.6390312	1.594258	-0.3772191	1.215584
-1.316959	-1.449697	0.8280068	1.421148	-0.5039472	1.374036
-1.119557	-1.507524	0.9977794	1.223748	-0.6220614	1.509001
-0.9012382	-1.548329	1.145447	1.005427	-0.7295428	1.618173
-0.6657331	-1.571415	1.268485	0.7699220	-0.8245541	1.699686
-0.4170677	-1.576388	1.364792	0.5212568	-0.9054713	1.752146
-0.1594926	-1.563163	1.432720	0.2636815	-0.9709113	1.774658
0.1025897	-1.531965	1.471108	1.5992371E-03	-1.019755	1.768832
-1.854477	-0.2526447	-0.1458644	2.113196	-0.2049384	-0.1183212
-1.640337	-0.3600722	4.5192011E-02	2.096028	-0.3406299	0.1106484
-1.598158	-0.4613450	0.2354760	2.044810	-0.4792459	0.3326767
-1.528663	-0.5547321	0.4217350	1.960423	-0.6184171	0.5439687
-1.433038	-0.6386372	0.6007853	1.844307	-0.7557644	0.7409126
-1.312919	-0.7116260	0.7695664	1.698447	-0.8889402	0.9201420
-1.170357	-0.7724510	0.9251933	1.525337	-1.015668	1.078594
-1.007791	-0.8200727	1.065003	1.327935	-1.133783	1.213559
-0.8279988	-0.8536767	1.186815	1.109616	-1.241264	1.322731
-0.6340534	-0.8726891	1.287940	0.8741109	-1.338275	1.404243
-0.4292701	-0.8767845	1.367252	0.6254455	-1.417192	1.456703
-0.2171495	-0.8858929	1.423192	0.3678704	-1.482633	1.479214
-1.3170240E-03	-0.8402009	1.454806	0.1057881	-1.531477	1.471390

Siders and Bolling

1.654477	0.2526447	0.1458644	-1.119557	0.1103402	-1.804443
1.640337	0.1408987	0.3344277	-0.9012381	0.2178216	-1.913615
1.598158	2.6744332E-02	0.5172745	-0.6657332	0.3128329	-1.995128
1.528663	-8.7867148E-02	0.6912796	-0.4170677	0.3937501	-2.047588
1.433038	-0.2009767	0.8534638	-0.1594926	0.4591901	-2.070098
1.312919	-0.3106510	1.001069	0.1025897	0.5080343	-2.062275
1.170357	-0.4150153	1.131559	-1.654477	-0.2526447	-0.1458644
1.007791	-0.5122858	1.242707	-1.640337	-0.1408987	-0.3344277
0.8279988	-0.6007999	1.332613	-1.598158	-2.6744332E-02	0.5172745
0.6340534	-0.6790445	1.399741	-1.528663	8.7867148E-02	0.6912796
0.4292702	-0.7456822	1.442943	-1.433038	0.2009767	-0.8534638
0.2171495	-0.7995740	1.461481	-1.312919	0.3106510	-1.001069
1.3170240E-03	-0.8397987	1.455039	-1.170357	0.4150153	-1.131559
1.758665	-0.2590765	-0.1495778	-1.007791	0.5122858	-1.242707
1.744528	-0.3708224	3.8985357E-02	-0.8279988	0.6007999	-1.332613
1.702347	-0.4849768	0.2218322	-0.6340534	0.6790445	-1.399741
1.632852	-0.5995883	0.3958373	-0.4292702	0.7456822	-1.442943
1.537227	-0.7126979	0.5580264	-0.2171495	0.7995740	-1.461481
1.417108	-0.8223723	0.7056271	-1.3170240E-03	0.8397987	-1.455039
1.274546	-0.9267365	0.8361166	-1.550288	-0.7643659	-0.4413067
1.111980	-1.024007	0.9472645	-1.536148	-0.6526198	-0.6298699
0.9321876	-1.112521	1.037171	-1.493970	-0.5384656	-0.8127167
0.7382423	-1.190766	1.104299	-1.424474	-0.4238540	-0.9867219
0.5334591	-1.257403	1.147501	-1.328849	-0.3107444	-1.148911
0.3213364	-1.311295	1.166039	-1.208730	-0.2010702	-1.296512
0.1055059	-1.351520	1.159596	-1.066168	-9.6705869E-02	-1.427001
-1.550288	-0.7643659	-0.4413067	-0.9036024	5.6463428E-04	-1.538149
-1.536148	-0.6526198	-0.6298699	-0.7238098	8.9078687E-02	-1.628055
-1.493970	-0.5384656	-0.8127167	-0.5298645	0.1673233	-1.695183
-1.424474	-0.4238540	-0.9867219	-0.3250813	0.2339610	-1.738386
-1.328849	-0.3107444	-1.148911	-0.1129606	0.2878528	-1.756924
-1.208730	-0.2010702	-1.296512	0.1028719	0.3280776	-1.750481
-1.066168	-9.6705869E-02	-1.427001	1.758665	-0.2590765	-0.1495778
-0.9036024	5.6463428E-04	-1.538149	1.744528	-0.1516490	-0.3406343
-0.7238098	8.9078687E-02	-1.628055	1.702347	-5.0376166E-02	-0.5309183
-0.5298645	0.1673233	-1.695183	1.632852	4.3010946E-02	-0.7171774
-0.3250813	0.2339610	-1.738386	1.537227	0.1269160	-0.8962277
-0.1129606	0.2878528	-1.756924	1.417108	0.1999048	-1.065009
0.1028719	0.3280776	-1.750481	1.274546	0.2607298	-1.220636
-1.904818	-0.8185040	-0.4725634	1.111980	0.3083515	-1.360448
-1.887648	-0.6828125	-[0.7015330	0.9321876	0.3419556	-1.482057
-1.836432	-0.5441964	-0.9235613	0.7382423	0.3609679	-1.583383
-1.752045	-0.4050253	-1.134853	0.5334591	0.3650633	-1.662694
-1.635929	-0.2676781	-1.331797	0.3213384	0.3541718	-1.718634
-1.490070	-0.1345021	-1.511027	0.1055059	0.3284798	-1.750249
-1.316959	-7.7739796E-03	-1.669478	2.113196	-0.2049384	-0.1183212
-1.119557	0.1103402	-1.804443	2.096026	-7.4490622E-02	-0.3503184
-0.9012381	0.2178216	-1.913615	2.044810	4.8483498E-02	-0.5813774
-0.6657332	0.3128329	-1.995128	1.960423	0.1618821	-0.8075492
-0.4170677	0.3937501	-2.047588	1.844307	0.2637668	-1.024967
-0.1594926	0.4591901	-2.070098	1.698447	0.3523961	-1.229918
0.1025897	0.5080343	-2.062275	1.525337	0.4262551	-1.418891
-2.009007	-0.3067828	-0.1771211	1.327935	0.4840814	-1.588664
-1.991837	-0.1710913	-0.4060907	1.109616	0.5248864	-1.736331
-1.940621	-3.2475308E-02	-0.6281190	0.8741109	0.5479728	-1.859370
-1.858234	0.1066958	-0.8394110	0.6254455	0.5529457	-1.955676
-1.740118	0.2440431	-1.036355	0.3678704	0.5397204	-2.023604
-1.594258	0.3772191	-1.215584	0.1057881	0.5085227	-2.061993
-1.421148	0.5039472	-1.374036	2.009007	0.3067828	0.1771211
-1.223746	0.6220614	-1.509001	1.991837	0.4372306	-5.4876041E-02
-1.005427	0.7295428	-1.618173	1.940621	0.5602046	-0.2859351
-0.7699220	0.8245541	-1.699686	1.856234	0.6736033	-0.5121069
-0.5212566	0.9054713	-1.752146	1.740118	0.7754880	-0.7295250
-0.2636815	0.9709113	-1.774656	1.594258	0.8641173	-0.9344736
-1.5992371E-03	1.019755	-1.768832	1.421148	0.9379763	-1.123449
-1.904818	-0.8185040	-0.4725634	1.223746	0.9958025	-1.293222
-1.887648	-0.6828125	-0.7015330	1.005427	1.036608	-1.440889
-1.836432	-0.5441964	-0.9235613	0.7699220	1.059694	-1.563928
-1.752045	-0.4050253	-1.134853	0.5212566	1.064667	-1.660234
-1.635929	-0.2676781	-1.331797	0.2636815	1.051442	-1.728162
-1.490070	-0.1345021	-1.511027	1.5992371E-03	1.020244	-1.766550
-1.316959	-7.7739796E-03	-1.669478	2.113196	-0.2049384	-0.1183212

NRL Memorandum Report 6709

2.096028	-7.4490622E-02	-0.3503184	-1.005427	-1.036608	1.440889
2.044810	4.8483498E-02	-0.5813774	-0.7699220	-1.059694	1.563928
1.960423	0.1618821	-0.8075492	-0.5212566	-1.064667	1.660234
1.844307	0.2637668	-1.024967	-0.2636815	-1.051442	1.728162
1.698447	0.3523961	-1.229916	-1.5992371E-03	-1.020244	1.766550
1.525337	0.4262551	-1.418891	-2.113196	0.2049384	0.1183212
1.327935	0.4840814	-1.588664	-2.096026	7.4490622E-02	0.3503184
1.109616	0.5248864	-1.736331	-2.044810	-4.8483498E-02	0.5813774
0.8741109	0.5479728	-1.859370	-1.960423	-0.1618821	0.8075492
0.6254455	0.5529457	-1.955676	-1.844307	-0.2637668	1.024967
0.3678704	0.5397204	-2.023604	-1.698447	-0.3523961	1.229916
0.1057881	0.5085227	-2.061993	-1.525337	-0.4262551	1.418891
1.054477	0.2526447	0.1458644	-1.327935	-0.4840814	1.588664
1.540337	0.3600722	-4.5192011E-02	-1.109616	-0.5248864	1.736331
1.598153	0.4613450	-0.2354760	-0.8741109	-0.5479728	1.859370
1.528663	0.5547321	-0.4217350	-0.6254455	-0.5529457	1.955676
1.433038	0.6386372	-0.6007853	-0.3678704	-0.5397204	2.023604
1.312919	0.7116260	-0.7695664	-0.1057881	-0.5085227	2.061993
1.170357	0.7724510	-0.9251933	-1.054477	-0.2526447	-0.1458644
1.007791	0.8200727	-1.065006	-1.540337	-0.3600722	4.5192011E-02
0.827998	0.8536767	-1.186615	-1.598153	-0.4613450	0.2354760
0.6340534	0.8726491	-1.287940	-1.528663	-0.5547321	0.4217350
0.4292701	0.8767845	-1.367252	-1.433038	-0.6386372	0.6007853
0.2171495	0.8655929	-1.423192	-1.312919	-0.7116260	0.7695664
1.3170240E-03	0.8402009	-1.454806	-1.170357	-0.7724510	0.9251933
1.758565	-0.2590765	-0.1495778	-1.007791	-0.8200727	1.065006
1.744526	-0.1516490	-0.3406343	-0.827998	-0.8536767	1.186615
1.702347	-5.0376166E-02	-0.5309183	-0.6340534	-0.8726491	1.287940
1.632852	4.3010946E-02	-0.7171774	-0.4292701	-0.8767845	1.367252
1.537227	0.1269160	-0.8962277	-0.2171495	-0.8655929	1.423192
1.417108	0.1990048	-1.065009	-1.3170240E-03	-0.8402009	1.454806
1.274546	0.2607298	-1.220636	-1.758565	0.2590765	0.1495778
1.111980	0.3083515	-1.360448	-1.744526	0.1516490	0.3406343
0.9321878	0.3419556	-1.482057	-1.702347	5.0376166E-02	0.5309183
0.7382423	0.3609679	-1.583383	-1.632852	-4.3010946E-02	0.7171774
0.5334591	0.3650633	-1.662694	-1.537227	-0.1269160	0.8962277
0.3213384	0.3541718	-1.718634	-1.417108	-0.1990048	1.065009
0.1055059	0.3284798	-1.750249	-1.274546	-0.2607298	1.220636
-1.758665	0.2590765	0.1495778	-1.111980	-0.3083515	1.360448
-1.744526	0.1516490	0.3406343	-0.9321878	-0.3419556	1.482057
-1.702347	5.0376166E-02	0.5309183	-0.7382423	-0.3609679	1.583383
-1.632852	-4.3010946E-02	0.7171774	-0.5334591	-0.3650633	1.662694
-1.537227	-0.1269160	0.8962277	-0.3213384	-0.3541718	1.718634
-1.417108	-0.1990048	1.065009	-0.1055059	-0.3284798	1.750249
-1.274546	-0.2607298	1.220636	1.550288	0.7643659	0.4413067
-1.111980	-0.3083515	1.360448	1.536148	0.6526198	0.6298699
-0.9321878	-0.3419556	1.482057	1.493970	0.5384656	0.8127167
-0.7382423	-0.3609679	1.583383	1.424474	0.4238540	0.9867219
-0.5334591	-0.3650633	1.662694	1.328849	0.3107444	1.148911
-0.3213384	-0.3541718	1.718634	1.208730	0.2010702	1.296512
-0.1055059	-0.3284798	1.750249	1.066168	9.6705869E-02	1.427001
-2.113196	0.2049384	0.1183212	0.9036024	-5.6483428E-04	1.538149
-2.096026	7.4490622E-02	0.3503184	0.7238098	-0.9078687E-02	1.628055
-2.044810	-4.8483498E-02	0.5813774	0.5298645	-0.1673233	1.695183
-1.960423	-0.1618821	0.8075492	0.3250813	-0.2339610	1.738386
-1.844307	-0.2637668	1.024967	0.1129606	-0.2878528	1.756924
-1.698447	-0.3523961	1.229916	-0.1028719	-0.3280776	1.750481
-1.525337	-0.4262551	1.418891	1.904818	0.8185040	0.4725634
-1.327935	-0.4840814	1.588664	1.887648	0.6828125	0.7015330
-1.109616	-0.5248864	1.736331	1.836432	0.5441964	0.9235613
-0.8741109	-0.5479728	1.859370	1.752045	0.4050253	1.134853
-0.6254455	-0.5529457	1.955676	1.635929	0.2676781	1.331797
-0.3678704	-0.5397204	2.023604	1.490070	0.1345021	1.511027
-0.1057881	-0.5085227	2.061993	1.316959	7.7739796E-03	1.669478
-2.009007	-0.3067828	-0.1771211	1.119557	-0.1103492	1.804443
-1.9491837	-0.4372306	5.4076041E-02	0.9012381	-0.2178218	1.913615
-1.940021	-0.5602048	0.2859351	0.6657332	-0.3128329	1.995128
-1.856234	-0.6736033	0.5121069	0.4170677	0.3937501	2.047508
-1.740118	-0.7754880	0.7295250	0.1594926	-0.4591901	2.070098
-1.594258	-0.8841173	0.9344736	-0.1025897	-0.5080343	2.062275
-1.421148	-0.9379763	1.123449	2.009007	0.3067628	0.1771211
-1.223746	-0.9958025	1.293222	1.991837	0.1710913	0.4060907

Siders and Bolling

1.940621	3.2475308E-02	0.6281190	-0.8741109	1.336275	-1.404243
1.856234	-0.1066958	0.8394110	-0.6254455	1.417192	-1.456703
1.740118	-0.2440431	1.036355	-0.3678704	1.482633	-1.479214
1.594258	-0.3772191	1.215584	-0.1057881	1.531477	-1.471390
1.421148	-0.5039472	1.374036	-2.009007	-0.3067828	-0.1771211
1.223746	-0.6220514	1.509001	1.991837	-0.1710313	-0.4060907
1.005427	-0.7295428	1.618173	-1.940621	-3.2475308E-02	-0.6281190
0.7699220	-0.8245541	1.699686	-1.856234	0.1066958	-0.8394110
0.5212566	-0.9054713	1.752146	-1.740118	0.2440431	-1.036355
0.2636815	-0.9709113	1.774656	-1.594258	0.3772191	-1.215584
1.5992371E-03	-1.019755	1.766832	-1.421148	0.5039472	-1.374036
1.904818	0.8185040	0.4725634	-1.223746	0.6220514	-1.509001
1.887648	0.6028125	0.7015330	-1.005427	0.7295428	-1.618173
1.836432	0.5441964	0.9235613	-0.7699220	0.8245541	-1.699686
1.752045	0.4050253	1.134853	-0.5212566	0.9054713	-1.752146
1.635929	0.2676781	1.331797	-0.2636815	0.9709113	-1.774656
1.490070	0.1345021	1.511027	-1.5992371E-03	1.019755	-1.766832
1.316959	7.7739796E-03	1.669478	-2.113196	0.2049384	0.1183212
1.119557	-0.1103402	1.804443	-2.096026	0.3406299	-0.1106484
0.9012381	-0.2178216	1.913615	-2.044810	0.4792459	-0.3326767
0.6657332	-0.3128329	1.995128	-1.960423	0.6184171	-0.5439687
0.4170677	-0.3937501	2.047588	-1.844307	0.7557644	-0.7409126
0.1594926	-0.4591901	2.070098	-1.698447	0.8889402	-0.9201420
-0.1025897	-0.5080343	2.062275	-1.525337	1.015668	-1.078594
1.654477	0.2526447	0.1458644	-1.327935	1.133783	-1.213559
1.640337	0.1408987	0.3344277	-1.109616	1.241264	-1.322731
1.598158	2.6744332E-02	0.5172745	-0.8741109	1.336275	-1.404243
1.528663	-8.7867148E-02	0.6912796	-0.6254455	1.417192	-1.456703
1.433038	-0.2009767	0.8534686	-0.3678704	1.482633	-1.479214
1.312919	-0.3106510	1.001069	-0.1057881	1.531477	-1.471390
1.170357	-0.4150153	1.131559	-1.654477	-0.2526447	-0.1458644
1.007791	-0.5122858	1.242707	-1.640337	-0.1408987	-0.3344277
0.8279988	-0.6007999	1.332613	-1.598158	-2.6744332E-02	-0.5172745
0.6340534	-0.6790445	1.399741	-1.528663	8.7867148E-02	-0.6912796
0.4292702	-0.7456822	1.442943	-1.433038	0.2009767	-0.8534686
0.2171495	-0.7995740	1.461481	-1.312919	0.3106510	-1.001069
1.3170240E-03	-0.8397987	1.455039	-1.170357	0.4150153	-1.131559
1.550288	0.7843659	0.4413067	-1.007791	0.5122858	-1.242707
1.536148	0.6526198	0.6298699	-0.8279988	0.6007999	-1.332613
1.493970	0.5384656	0.8127167	-0.6340534	0.6790445	-1.399741
1.424474	0.4238540	0.9867219	-0.4292702	0.7456822	-1.442943
1.328849	0.3107444	1.148911	-0.2171495	0.7995740	-1.461481
1.208730	0.2010702	1.296512	-1.3170240E-03	0.8397987	-1.455039
1.066168	9.6705869E-02	1.427001	-1.758665	0.2590765	0.1495778
0.9036024	-5.6463428E-04	1.538149	-1.744526	0.3708224	-3.8985357E-02
0.7238098	-8.9078687E-02	1.628055	-1.702347	0.4849768	-0.2218322
0.5298645	-0.1673233	1.695183	-1.632852	0.5995883	-0.3958373
0.3250813	-0.2339610	1.738386	-1.537227	0.7126979	-0.5580264
0.1129606	-0.2878528	1.756924	-1.417108	0.8223723	-0.7056271
-0.1028719	-0.3280776	1.750481	-1.274546	0.9267365	-0.8361166
-1.758665	0.2590765	0.1495778	-1.111980	1.024007	-0.9472645
-1.744526	0.3708224	-3.8985357E-02	-0.9321876	1.112521	-1.037171
-1.702347	0.4849768	-0.2218322	-0.7382423	1.190768	-1.104299
-1.632852	0.5995883	-0.3958373	-0.5334591	1.257403	-1.147501
-1.537227	0.7126979	-0.5580264	-0.3213384	1.311295	-1.166039
-1.417108	0.8223723	-0.7056271	-0.1055059	1.351520	-1.159596
-1.274546	0.9267365	-0.8361166	1.550288	0.7643659	0.4413067
-1.111980	1.024007	-0.9472645	1.536148	0.8717934	0.2502503
-0.9321876	1.112521	-1.037171	1.493970	0.9730662	5.9966244E-02
-0.7382423	1.190768	-1.104299	1.424474	1.066453	-0.1262928
-0.5334591	1.257403	-1.147501	1.328849	1.150358	-0.3053430
-0.3213384	1.311295	-1.166039	1.208730	1.223347	-0.4741242
-0.1055059	1.351520	-1.159596	1.066168	1.284172	-0.6297510
-2.113196	0.2049384	0.1183212	0.9038023	1.331794	-0.7695637
-2.096026	0.3406299	-0.1106484	0.7236098	1.365398	-0.8911722
-2.044810	0.4792459	-0.3326767	0.5298645	1.384410	-0.9924980
-1.960423	0.6184171	-0.5439687	0.3250813	1.388506	-1.071809
-1.844307	0.7557644	-0.7409126	0.1129605	1.377614	-1.127750
-1.698447	0.8889402	-0.9201420	-0.1028719	1.351922	-1.159364
-1.525337	1.015668	-1.078594	1.904018	0.8185040	0.4725634
-1.327935	1.133783	-1.213559	1.887648	0.9489518	0.2405863
-1.109616	1.241264	-1.322731	1.836432	1.071926	9.5071597E-03

NRL Memorandum Report 6709

1.752045	1.185325	-0.216646	-1.995171	-2.105464	-2.722512
1.635929	1.287209	-0.4340828	-2.112570	-2.222863	-2.722512
1.490070	1.375838	-0.6390312	-2.112570	-2.105464	-2.722512
1.316959	1.449697	-0.8280068	-1.995171	-2.222863	-2.722512
1.119557	1.507524	-0.9977794	-2.876295	-1.746618	-3.293020
0.9012382	1.548329	-1.145447	-2.817595	-1.805318	-3.293020
0.6657331	1.571415	-1.268485	-2.758895	-1.746618	-3.293020
0.4170677	1.576388	-1.364792	-2.817595	-1.864018	-3.293020
0.1594928	1.563163	-1.432720	-3.015398	-1.538882	-2.479676
-0.1025897	1.531965	-1.471108	-2.897999	-1.538882	-2.479676
2.009007	0.3067828	0.1771211	-3.015398	-1.656282	-2.479676
1.991837	0.4372308	-5.4876041E-02	-2.897999	-1.656282	-2.479676
1.940621	0.5602046	-0.2859351			
1.856234	0.6736033	-0.5121069			
1.740118	0.7754880	-0.7295250			
1.594258	0.8641173	-0.9344736			
1.421148	0.9379763	-1.123449			
1.223746	0.9958025	-1.293222			
1.005427	1.036608	-1.440689			
0.7699220	1.059694	-1.563928			
0.5212568	1.064667	-1.660234			
0.2636815	1.051442	-1.728162			
1.5992371E-03	1.020244	-1.766550			
1.904818	0.8185040	0.4725634			
1.887648	0.9489518	0.2405663			
1.836432	1.071926	9.5071597E-03			
1.752045	1.185325	-0.216646			
1.635929	1.287209	-0.4340828			
1.490070	1.375838	-0.6390312			
1.316959	1.449697	-0.8280068			
1.119557	1.507524	-0.9977794			
0.9012382	1.548329	-1.145447			
0.6657331	1.571415	-1.268485			
0.4170677	1.576388	-1.364792			
0.1594928	1.563163	-1.432720			
-0.1025897	1.531965	-1.471108			
1.654477	0.2526447	0.1458644			
1.640337	0.3600722	-4.5192011E-02			
1.598158	0.4613450	-0.2354760			
1.528663	0.5547321	-0.4217350			
1.433038	0.6386372	-0.60075			
1.312919	0.7116260	-0.769566			
1.170357	0.7724510	-0.9251933			
1.007791	0.8200727	-1.065006			
0.8279988	0.8536767	-1.186615			
0.6340534	0.8726891	-1.287940			
0.4292701	0.8767845	-1.367252			
0.2171495	0.8658929	-1.423192			
1.3170240E-03	0.8402009	-1.454806			
1.550209	0.7643659	0.4413067			
1.536148	0.6717334	0.2502503			
1.493970	0.9730862	0.9966244E-02			
1.424474	1.066453	-0.1262928			
1.328849	1.150358	-0.3053430			
1.208730	1.223347	-0.4741242			
1.066168	1.284172	-0.6297510			
0.9036023	1.331794	-0.7695637			
0.7238098	1.365398	-0.8911722			
0.5298645	1.384410	-0.9924980			
0.3250813	1.388500	-1.071809			
0.1129605	1.377614	-1.127750			
-0.1028719	1.351922	-1.159364			
-2.817595	-2.280787	-2.774497			
-2.227952	-2.190746	-2.722512			
-2.817595	-1.981417	-3.293020			
-2.921565	-1.770141	-2.479676			
-2.395388	-2.260850	-2.722512			
-2.408680	-2.173809	-2.722512			
-2.773570	-2.157517	-3.293020			
-2.801620	-2.157517	-3.293020			
-2.843291	-1.933917	-2.479676			
-2.929571	-1.951464	-2.479676			

Siders and Bolling

TEST2.CON is the connectivity file.

START	STOP	COLOR	START	STOP	COLOR
1	2	2	64	65	-2
2	3	2	66	67	-2
3	4	2	67	68	-2
4	5	2	68	69	-2
5	6	2	69	70	-2
6	7	2	70	71	-2
7	8	2	71	72	-2
8	9	2	72	73	-2
9	10	2	73	74	-2
10	11	2	74	75	-2
11	12	2	75	76	-2
12	13	2	76	77	-2
13	14	2	77	78	-2
14	15	2	53	66	-2
15	16	2	57	70	-2
16	17	2	61	74	-2
17	18	2	65	78	-2
18	19	2	79	80	2
19	20	2	80	81	2
20	21	2	81	82	2
21	22	2	82	83	2
22	23	2	83	84	2
23	24	2	84	85	2
24	25	2	85	86	2
25	26	2	86	87	2
1	14	2	87	88	2
5	18	2	88	89	2
9	22	2	89	90	2
13	26	2	90	91	2
27	28	2	92	93	2
28	29	2	93	94	2
29	30	2	94	95	2
30	31	2	95	96	2
31	32	2	96	97	2
32	33	2	97	98	2
33	34	2	98	99	2
34	35	2	99	100	2
35	36	2	100	101	2
36	37	2	101	102	2
37	38	2	102	103	2
38	39	2	103	104	2
40	41	2	79	92	2
41	42	2	83	96	2
42	43	2	87	100	2
43	44	2	91	104	2
44	45	2	105	106	2
45	46	2	106	107	2
46	47	2	107	108	2
47	48	2	108	109	2
48	49	2	109	110	2
49	50	2	110	111	2
50	51	2	111	112	2
51	52	2	112	113	2
27	40	2	113	114	2
31	44	2	114	115	2
35	48	2	115	116	2
39	52	2	116	117	2
53	54	-2	118	119	2
54	55	-2	119	120	2
55	56	-2	120	121	2
56	57	-2	121	122	2
57	58	-2	122	123	2
58	59	-2	123	124	2
59	60	-2	124	125	2
60	61	-2	125	126	2
61	62	-2	126	127	2
62	63	-2	127	128	2
63	64	-2	128	129	2

NRL Memorandum Report 6709

129	130	?	194	195	2
105	118	2	196	197	2
109	122	2	197	198	2
113	126	2	198	199	2
117	130	2	199	200	2
131	132	-2	200	201	2
132	133	-2	201	202	2
133	134	-2	202	203	2
134	135	-2	203	204	2
135	136	-2	204	205	2
136	137	-2	205	206	2
137	138	-2	206	207	2
138	139	-2	207	208	2
139	140	-2	183	196	2
140	141	-2	187	200	2
141	142	-2	191	204	2
142	143	-2	195	208	2
144	145	-2	209	210	-2
145	146	-2	210	211	-2
146	147	-2	211	212	-2
147	148	-2	212	213	-2
148	149	-2	213	214	-2
149	150	-2	214	215	-2
150	151	-2	215	216	-2
151	152	-2	216	217	-2
152	153	-2	217	218	-2
153	154	-2	218	219	-2
154	155	-2	219	220	-2
155	156	-2	220	221	-2
131	144	-2	222	223	-2
135	148	-2	223	224	-2
139	152	-2	224	225	-2
143	156	-2	225	226	-2
157	158	2	226	227	-2
158	159	2	227	228	-2
159	160	2	228	229	-2
160	161	2	229	230	-2
161	162	2	230	231	-2
162	163	2	231	232	-2
163	164	2	232	233	-2
164	165	2	233	234	-2
165	166	2	209	222	-2
166	167	2	213	226	-2
167	168	2	217	230	-2
168	169	2	221	234	-2
170	171	2	235	236	2
171	172	2	236	237	2
172	173	2	237	238	2
173	174	2	238	239	2
174	175	2	239	240	2
175	176	2	240	241	2
176	177	2	241	242	2
177	178	2	242	243	2
178	179	2	243	244	2
179	180	2	244	245	2
180	181	2	245	246	2
181	182	2	246	247	2
157	170	2	248	249	2
161	174	2	249	250	2
165	178	2	250	251	2
169	182	2	251	252	2
183	184	2	252	253	2
184	185	2	253	254	2
185	186	2	254	255	2
186	187	2	255	256	2
187	188	2	256	257	2
188	189	2	257	258	2
189	190	2	258	259	2
190	191	2	259	260	2
191	192	2	235	248	2
192	193	2	239	252	2
193	194	2	243	256	2

Siders and Bolling

247	260	2	329	330	2
261	262	2	330	331	2
262	263	2	331	332	2
263	264	2	332	333	2
264	265	2	333	334	2
265	266	2	334	335	2
266	267	2	335	336	2
267	268	2	336	337	2
268	269	2	337	338	2
269	270	2	313	326	2
270	271	2	317	330	2
271	272	2	321	334	2
272	273	2	325	338	2
274	275	2	339	340	2
275	276	2	340	341	2
276	277	2	341	342	2
277	278	2	342	343	2
278	279	2	343	344	2
279	280	2	344	345	2
280	281	2	345	346	2
281	282	2	346	347	2
282	283	2	347	348	2
283	284	2	348	349	2
284	285	2	349	350	2
285	286	2	350	351	2
261	274	2	352	353	2
265	278	2	353	354	2
269	282	2	354	355	2
273	286	2	355	356	2
287	288	-2	356	357	2
288	289	-2	357	358	2
289	290	-2	358	359	2
290	291	-2	359	360	2
291	292	-2	360	361	2
292	293	-2	361	362	2
293	294	-2	362	363	2
294	295	-2	363	364	2
295	296	-2	339	352	2
296	297	-2	343	356	2
297	298	-2	347	360	2
298	299	-2	351	364	2
300	301	-2	365	366	-2
301	302	-2	366	367	-2
302	303	-2	367	368	-2
303	304	-2	368	369	-2
304	305	-2	369	370	-2
305	306	-2	370	371	-2
306	307	-2	371	372	-2
307	308	-2	372	373	-2
308	309	-2	373	374	-2
309	310	-2	374	375	-2
310	311	-2	375	376	-2
311	312	-2	376	377	-2
287	300	-2	378	379	-2
291	304	-2	379	380	-2
295	308	-2	380	381	-2
299	312	-2	381	382	-2
313	314	2	382	383	-2
314	315	2	383	384	-2
315	316	2	384	385	-2
316	317	2	385	386	-2
317	318	2	386	387	-2
318	319	2	387	388	-2
319	320	2	388	389	-2
320	321	2	389	390	-2
321	322	2	365	378	-2
322	323	2	369	382	-2
323	324	2	373	386	-2
324	325	2	377	390	-2
228	327	2	391	392	2
327	328	2	392	393	2
328	329	2	393	394	2

NRL Memorandum Report 6709

394	395	2	463	464	-2
395	396	2	464	465	-2
398	397	2	465	466	-2
397	398	2	466	467	-2
398	399	2	467	468	-2
399	400	2	443	456	-2
400	401	2	447	460	-2
401	402	2	451	464	-
402	403	2	455	468	-2
404	405	2	469	470	2
405	406	2	470	471	2
406	407	2	471	472	2
407	408	2	472	473	2
408	409	2	473	474	2
409	410	2	474	475	2
410	411	2	475	476	2
411	412	2	475	477	2
412	413	2	477	478	2
413	414	2	478	479	2
414	415	2	479	480	2
415	416	2	480	481	2
391	434	2	482	483	2
395	438	2	483	484	2
399	412	2	484	485	2
403	416	2	485	486	2
417	418	2	486	487	2
418	419	2	487	488	2
419	420	2	488	489	2
420	421	2	489	490	2
421	422	2	490	491	2
422	423	2	491	492	2
423	424	2	492	493	2
424	425	2	493	494	2
425	426	2	469	482	2
426	427	2	473	486	2
427	428	2	477	490	2
428	429	2	481	494	2
430	431	2	495	496	2
431	432	2	496	497	2
432	433	2	497	498	2
433	434	2	498	499	2
434	435	2	499	500	2
435	436	2	500	501	2
436	437	2	501	502	2
437	438	2	502	503	2
438	439	2	503	504	2
439	440	2	504	505	2
440	441	2	505	506	2
441	442	2	506	507	2
417	430	2	508	509	2
421	434	2	509	510	2
425	438	2	510	511	2
429	442	2	511	512	2
443	444	-2	512	513	2
444	445	-2	513	514	2
445	446	-2	514	515	2
446	447	-2	515	516	2
447	448	-2	516	517	2
448	449	-2	517	518	2
449	450	-2	518	519	2
450	451	-2	519	520	2
451	452	-2	495	508	2
452	453	-2	499	512	2
453	454	-2	503	516	2
454	455	-2	507	520	2
456	457	-2	521	522	-2
457	458	-2	522	523	-2
458	459	-2	523	524	-2
459	460	-2	524	525	-2
460	461	-2	525	526	-2
461	462	-2	526	527	-2
462	463	-2	527	528	-2

Siders and Bolling

528	529	-2	597	598	255
529	530	-2	573	586	255
530	531	-2	577	590	255
531	532	-2	581	594	255
532	533	-2	585	598	255
534	535	-2	599	600	-255
535	536	-2	600	601	-255
536	537	-2	601	602	-255
537	538	-2	602	603	-255
538	539	-2	603	604	-255
539	540	-2	604	605	-255
540	541	-2	605	606	-255
541	542	-2	606	607	-255
542	543	-2	607	608	-255
543	544	-2	608	609	-255
544	545	-2	609	610	-255
545	546	-2	610	611	-255
521	534	-2	612	613	-255
525	538	-2	613	614	-255
529	542	-2	614	615	-255
533	546	-2	615	616	-255
547	548	255	616	617	-255
548	549	255	617	618	-255
549	550	255	618	619	-255
550	551	255	619	620	-255
551	552	255	620	621	-255
552	553	255	621	622	-255
553	554	255	622	623	-255
554	555	255	623	624	-255
555	556	255	599	612	-255
556	557	255	603	616	-255
557	558	255	607	620	-255
558	559	255	611	624	-255
560	561	255	625	626	5
561	562	255	625	627	5
562	563	255	625	628	5
563	564	255	629	628	5
564	565	255	626	630	5
565	566	255	630	629	5
566	567	255	631	627	5
567	568	255	627	632	5
568	569	255	632	631	5
569	570	255	633	628	5
570	571	255	628	634	5
571	572	255	634	633	5
547	580	255	635	636	5
551	564	255	637	638	5
555	568	255	639	640	5
559	572	255	640	641	5
573	574	255	640	642	5
574	575	255	643	644	5
575	576	255	644	645	5
576	577	255	645	646	5
577	578	255			
578	579	255			
579	580	255			
580	581	255			
581	582	255			
582	583	255			
583	584	255			
584	585	255			
586	587	255			
587	588	255			
588	589	255			
589	590	255			
590	591	255			
591	592	255			
592	593	255			
593	594	255			
594	595	255			
595	596	255			
596	597	255			

APPENDIX C - Two Examples of Graphics Drivers

```

C-----
C
C DRAW240 - written by D. Mitchell Bolling Jr.
C           TRI/TESSCO, Inc.
C           August 1989
C
C Purpose: This program is an elementary graphics driver. DRAW240
C loads in the coordinate and connectivity files that result from
C PLOTCID or PLOTCHIEF and displays the figure on the screen in REGIS
C format. It may be used on any VT240 or compatible terminal or any
C terminal/computer using a VT240 emulator program.
C
C Subroutines used: None.
C
C Modification Log:
C-----

PARAMETER PLT_UNIT = 8      ! lun number of output (8 = screen) default
PARAMETER CRD_UNIT = 8      ! lun number of coordinate file
PARAMETER CON_UNIT = 9      ! lun number of connectivity file
PARAMETER MAXCOORDS = 10000 ! maximum number of coordinates for plotting

C-----
C The following variables are used:
C-----
CHARACTER*50 TEMP ! a temporary string variable
CHARACTER*20 RUNID ! driver name used in CID or CHIEF run

REAL COORDS(3, MAXCOORDS) ! coordinates read in from coordinate file
REAL NCOORDS(3, MAXCOORDS) ! coordinates modified by scaling
REAL MAXX      ! largest x coordinate
REAL MAXY      ! largest y coordinate
REAL MINX      ! smallest x coordinate
REAL MINY      ! smallest y coordinate
REAL DISX      ! largest distance between 2 x values
REAL DISY      ! largest distance between 2 y values
REAL DIS       ! maximum distance
REAL OFFSETX   ! offset for centering horizontally
REAL OFFSETY   ! offset for centering vertically

INTEGER CONNECTS(3, MAXCOORDS) ! connectivity data read in from file
INTEGER MAXP      ! number of points
INTEGER MAXC      ! number of lines
INTEGER I         ! scratch counter
INTEGER H         ! number of pixels horizontally (0-799)
INTEGER V         ! number of pixels vertically (0-479)
INTEGER COLOR     ! number associated with different colors

C-----
C Prompt the user for filename.
C-----
      WRITE(6,10) 'Enter filename used in CID or CHIEF run: '
10    FORMAT('$',A)
      READ(5,20) LEN, TEMP
20    FORMAT(Q,A)
      TYPE *, ' '

C-----
C Strip off the file extension.
C-----
      L = INDEX(TEMP, '.')
      IF (L .GT. 0) THEN
        LEN = L - 1
        RUNID = TEMP(1: LEN)
      ELSE
        RUNID = TEMP

```

Siders and Bolling

```

END IF

C-----
C Read in the coordinate data.
C-----
      TEMP = RUNID(1:LEN) // '.CRD'
      OPEN (UNIT=CRD_UNIT, FILE=TEMP,TYPE='OLD')
      MAXP = 1
100  READ (CRD_UNIT,*,ERR=125) COORDS(1, MAXP), COORDS(2, MAXP),
      *                               COORDS(3, MAXP)
      COORDS(2, MAXP) = -COORDS(2, MAXP)
      MAXP = MAXP + 1
      IF (MAXP .LE. MAXCOORDS) GOTO 100
125  MAXP = MAXP - 1
      CLOSE (CRD_UNIT)

C-----
C Read in the connectivity and color data.
C-----
      TEMP = RUNID(1:LEN) // '.CON'
      OPEN (UNIT=CON_UNIT, FILE=TEMP,TYPE='OLD')
      MAXC = 1
200  READ (CON_UNIT,*,ERR=225) CONNECTS(1, MAXC),
      *                               CONNECTS(2, MAXC), CONNECTS(3, MAXC)
      MAXC = MAXC + 1
      IF (MAXC .LE. MAXCOORDS) GOTO 200
225  MAXC = MAXC - 1
      CLOSE (CON_UNIT)

C-----
C Copy the coordinates into a working array  A possible enhancement
C might be to pass this working array to ROT3D(), allowing the user
C to interactively rotate the view.
C-----
      DO I = 1, MAXP
        DO J = 1, 3
          NCOORDS(J,I) = COORDS(J,I)
        END DO
      END DO

300  MINX = 9999999999.0
      MINY = MINX
      MAXX = -MINX
      MAXY = -MINX

C-----
C Find the maximum and minimum values of the figure in the x and y
C directions.
C-----
      DO I = 1, MAXP
        IF (NCOORDS(1,I) .GT. MAXX) MAXX = NCOORDS(1,I)
        IF (NCOORDS(1,I) .LT. MINX) MINX = NCOORDS(1,I)
        IF (NCOORDS(2,I) .GT. MAXY) MAXY = NCOORDS(2,I)
        IF (NCOORDS(2,I) .LT. MINY) MINY = NCOORDS(2,I)
      END DO

C-----
C Find the maximum width and height of the figure.
C-----
      DISX = MAXX - MINX
      DISY = MAXY - MINY

C-----
C Find out which is greater, the maximum width or maximum height.
C-----
      IF (DISX .GT. DISY) THEN
        DIS = DISX
      ELSE
        DIS = DISY
      END IF

      OFFSETX = 400 - (521 * DISX) / (2 * DIS)
      OFFSETY = 240 - (431 * DISY) / (2 * DIS)

```

NRL Memorandum Report 6709

```

C-----
C Scale the coordinates so that the drawing will fit on the screen.
C-----
      DO I = 1, MAXP
        NCOORDS(1,I) = 521 * (NCOORDS(1,I) - MINX) / DIS + OFFSETX
        NCOORDS(2,I) = 431 * (NCOORDS(2,I) - MINY) / DIS + OFFSETY
      END DO

C-----
C Set Regis graphics mode and clear the screen.
C-----
      WRITE(PLT_UNIT,*) CHAR(27) // 'P0p'
      WRITE(PLT_UNIT,*) 'S(E)'
      COLOR = 0

C-----
C Create the picture.
C-----
      DO I = 1, MAXC
C-----
C If there is a color change, then change the color.
C-----
        IF (CONNECTS(3,I) .NE. COLOR) THEN
          COLOR = CONNECTS(3,I)
          WRITE(PLT_UNIT,900) ABS(COLOR)
900      FORMAT('$(I',I3,')')
          IF (COLOR .LT. 0) THEN
            WRITE(PLT_UNIT,*) 'W(P2)'
          ELSE
            WRITE(PLT_UNIT,*) 'W(P1)'
          END IF
        END IF

C-----
C Move the cursor to first coordinates.
C-----
        H = NCOORDS(1,CONNECTS(1,I))
        V = NCOORDS(2,CONNECTS(1,I))
        WRITE(PLT_UNIT,1000) H, V
1000     FORMAT('$(I',I4,')$(I',I4,')')

C-----
C Draw a line to the second coordinates.
C-----
        H = NCOORDS(1,CONNECTS(2,I))
        V = NCOORDS(2,CONNECTS(2,I))
        WRITE(PLT_UNIT,1100) H, V
1100     FORMAT(' V$(I',I4,')$(I',I4,')')
      END DO

C-----
C Exit ReGIS mode and end.
C-----
      WRITE(PLT_UNIT,*) CHAR(27) // '\ '
      END

```

Siders and Bolling

DRAW3D.PRO - written by D. Mitchell Bolling Jr.
TRI/TESSCO, Inc.
August 1989

Purpose: This program is an elementary graphics driver. DRAW3D.PRO is a PVWAVE program that loads in the coordinate and connectivity files that result from the PLOTCID or PLOTCHIEF programs and displays the figure on the screen using the graphics protocol defined in the PVWAVE software. PVWAVE is a data visualization and color graphics package from Precision Visuals.

Subroutines used: None.

FILENAME = ''
READ, 'Enter filename of plot data: ', FILENAME

; Read in the coordinate data.

GET_LUN, F
OPENR, F, FILENAME+'.CRD'
PI = DOUBLE(3.14159265358979)
I = 0
RECORD = DBLARR(3)
WHILE NOT EOF(F) DO BEGIN
 READF, F, RECORD
 IF (I EQ 0) THEN BEGIN
 COORDS = RECORD
 I = 1
 ENDIF ELSE BEGIN
 COORDS = [[COORDS], [RECORD]]
 ENDELSE
ENDWHILE
CLOSE, F

; Read in the connectivity data.

OPENR, F, FILENAME+'.CON'
I = 0
RECORD = INTARR(3)
WHILE NOT EOF(F) DO BEGIN
 READF, F, RECORD
 IF (I EQ 0) THEN BEGIN
 CONNECTS = RECORD
 I = 1
 ENDIF ELSE BEGIN
 CONNECTS = [[CONNECTS], [RECORD]]
 ENDELSE
ENDWHILE
CLOSE, F
FREE_LUN, F

; Adjust connectivity data due to PVWAVE's indexing starting at 0.

CONNECTS(0,0) = CONNECTS(0:1,*) - 1

; If the color is negative, then lines should be dashed to represent
inward normal. Outward normal is shown by solid lines.

LINEYPES = CONNECTS(2,*) LT 0
CONNECTS(2,0) = ABS(CONNECTS(2,*))
NCOORDS = COORDS

; Determine the scaling for the plot.

NRL Memorandum Report 6709

```

-----
!TYPE = 4 + 8 + 32 + 64          ; inhibit x and y axis rounding
MINX = MIN(NCOORDS(0,*))
MAXX = MAX(NCOORDS(0,*))
MIDX = (MINX + MAXX) / 2
DISX = MAXX - MINX

```

```

MINY = MIN(NCOORDS(1,*))
MAXY = MAX(NCOORDS(1,*))
MIDY = (MINY + MAXY) / 2
DISY = MAXY - MINY

```

```

DIS = .55 * MAX([DISX,DISY])

```

```

-----
; Use the whole screen for plotting.
-----

```

```

!SC1 = 0          ; Left side of screen
!SC2 = 800        ; Right side of screen
!SC3 = 0          ; Bottom of screen
!SC4 = 480        ; Top of screen

```

```

-----
; Create the plot array. XCOORDS will contain the starting and ending
; X coordinates for all the line segments to be drawn. YCOORDS will
; contain the starting and ending Y coordinates for all line segments
; to be drawn.
-----

```

```

XCOORDS = NCOORDS(0,*)
YCOORDS = NCOORDS(1,*)
XCOORDS = XCOORDS(CONNECTS(0:1,*))
YCOORDS = YCOORDS(CONNECTS(0:1,*))

```

```

-----
; Draw the object
-----

```

```

SET_XY, (MIDX-1.56*DIS), (MIDX+1.56*DIS), MIDY-DIS, MIDY+DIS
!C = 0
LINEPLOTXY, XCOORDS, YCOORDS, CONNECTS(2,*), LINETYPES
END

```

Siders and Bolling

(Blank page)

APPENDIX D - Listing of Programs and Subroutines in DISPLAY3D

```

C-----
C
C      PLOTCID      - written by D. Mitchell Bolling Jr.
C                   TRI/TESSCO, Inc.
C                   August 1989
C
C Purpose: The PLOTCID program is a graphics routine used to display
C three dimensional surfaces defined by the CHIEF program.
C CHIEF (Combined Helmholtz Integral Equation Formulation), developed
C at NOSC, computes the acoustic radiation from arbitrary-shaped bodies.
C PLOTCID creates a plot file which can be viewed and rotated before
C compiling, linking and executing the CHIEF driver. PLOTCID reads in
C the surface information from the CID (Chief Interactive Driver) program.
C
C Subroutines used:  LOAD_CID_DATA()      RECTANGULAR_PLANE()
C                   CIRCULAR_PLANE()     ELLIPTICAL_PLANE()
C                   CIRCULAR_CYLINDER()  ELLIPTICAL_CYLINDER()
C                   SPHEROID()           PROLATE_OBLATE_SPHEROID()
C                   TOROID()             QUADRILATERAL()
C                   AXISYMMETRIC()       TRIANGLE()
C                   CONE()               ROT3D()
C                   CONNECT_ARROWS()    DRAW_ARROWS()
C
C Modification Log:
C
C   AUG 89 - Added QUADRILATERAL(), AXISYMMETRIC() and TRIANGLE().
C   Also modified program by adding ROT3D so that the user can rotate
C   the points from within PLOTCHIEF. This requires that the points
C   are stored in an array, then passed to ROT3D, which writes them out
C   to a file. Before, the points were sent straight to a file.
C
C   JAN 90 - Added CONNECT_ARROWS() and DRAW_ARROWS() in order to
C   better show the axis orientation during rotations.
C
C   JUL 90 - Added CONE() geometry to PLOTCID.
C-----
C
C   PARAMETER PI = 3.1415926536
C   PARAMETER COLOR_USER_DEFINED = 255 ! color to use for user defined part
C   PARAMETER COLOR_TRANSFORMED = 2   ! color for transformed part
C   PARAMETER CRD_UNIT = 8             ! lun number of coordinate file
C   PARAMETER CON_UNIT = 9             ! lun number of connectivity file
C   PARAMETER MXSREG = 500             ! maximum number of surface regions
C   PARAMETER MAXCOORDS = 100000       ! maximum number of coordinates for plotting
C-----
C List of global variables:
C-----
C   INTEGER      NSREG                ! number of CHIEF surfaces
C   INTEGER      NSEQNS(MXSREG)       ! types of CHIEF surfaces
C   REAL         SUL(MXSREG)           ! lower limits of U
C   REAL         SUU(MXSREG)           ! upper limits of U
C   REAL         SVL(MXSREG)           ! lower limits of V
C   REAL         SVU(MXSREG)           ! upper limits of V
C   REAL         CCS(10,MXSREG)        ! constants needed for CHIEF equations
C   REAL         TRNSS(3,MXSREG)       ! translation from local to global origin
C   INTEGER      IZAX(MXSREG)          ! global axis that corresponds to local
C                                       ! Z axis
C
C   COMMON/SVALS/NSREG,NSEQNS(MXSREG),SUL(MXSREG),SUU(MXSREG),
C   *           SVL(MXSREG),SVU(MXSREG),NSU(MXSREG),NSV(MXSREG),
C   *           CCS(10,MXSREG),TRNSS(3,MXSREG),IZAX(MXSREG),
C   *           IORDU(MXSREG),IORDV(MXSREG),NCCEQS
C
C   INTEGER      SYMTYP                ! type of symmetry: reflective (1),
C                                       ! rotational (2), none (0)
C
C   INTEGER      NBLKS                ! number of symmetry blocks
C                                       ! for reflective symmetry:

```

Siders and Belling

```

REAL      SYMX      ! if -1, then reflects about YZ plane
REAL      SYMY      ! if -1, then reflects about XZ plane
REAL      SYMZ      ! if -1, then reflects about XY plane

! for rotational symmetry:
REAL      SYMANG     ! radians to rotate object

COMMON / PLOT_INFO / SYMTYP, NBLKS, SYMX, SYMY, SYMZ, SYMANG

REAL      COORD3D     ! array of 3d points
REAL      ROTATED     ! 3d points after rotations

COMMON / ROT_INFO / COORD3D(3, MAXCOORDS), ROTATED(3,MAXCOORDS)

C -----
C List of local variables:
C -----
CHARACTER*50 TEMP     ! a temporary string variable
CHARACTER*20 RUNID    ! driver name used in CID run

INTEGER LEN, L        ! holds length of string RUNID
INTEGER ISUBDIV       ! set when NSV and NSU are to be used
INTEGER COLOR        ! number associated with different colors
INTEGER POINTCNT      ! keeps track of next point to be created
INTEGER SHOW_NORMALS ! when set, uses dotted lines for inward normals
INTEGER REGION        ! counter used for looping through each region
INTEGER BLOCK         ! counter used for looping through each symmetry
                   ! block (whether reflective or rotational)

REAL AX      ! number of degrees to rotate view about x axis
REAL AY      ! number of degrees to rotate view about y axis
REAL AZ      ! number of degrees to rotate view about z axis

C -----
C Some or all of these constants are multiplied by each data point
C -----
REAL REFLECTX(8) ! array of 8 constants for reflecting about yz plane
REAL REFLECTY(8) ! array of 8 constants for reflecting about xz plane
REAL REFLECTZ(8) ! array of 8 constants for reflecting about xy plane

C -----
C Below is a table that indicates the number of constants needed for
C each reflective symmetry option. Thus, the dashed horizontal line
C is over the constants needed for each option.
C -----
C 3 plane symmetry |<----->|
C 2 plane symmetry |<----->|
C 1 plane symmetry |<---->|
C
DATA REFLECTX / 1, -1, 1, -1, 1, -1, 1, -1 /
DATA REFLECTY / 1, 1, -1, -1, 1, 1, -1, -1 /
DATA REFLECTZ / 1, 1, 1, 1, -1, -1, -1, -1 /

C -----
C Get CID run name from user.
C -----
WRITE(6,100) 'Enter the filename used in the CID run: '
100 FORMAT('S',A)
READ(5,125) LEN, TEMP
125 FORMAT(Q,A)

L = INDEX(TEMP, '.')
IF (L .GT. 0) THEN
    LEN = L - 1
    RUNID = TEMP(1 : LEN)
ELSE
    RUNID = TEMP
END IF

C -----
C Read global data from the CID output file.

```

NRL Memorandum Report 6709

```

C-----
TEMP = RUNID(1:LEN) // '.DAT'
CALL LOAD_CID_DATA(TEMP)
C-----
C If the model is using reflective symmetry, then NBLKS needs to be
C converted from the number of planes of symmetry to the total number
C of blocks.
C-----
      IF (SYMTYP .EQ. 1) NBLKS = 2 ** NBLKS

C-----
C Ask user if rotations or reflections should be drawn.
C-----
      IF (SYMTYP .NE. 0 .AND. NBLKS .GT. 1) THEN
        WRITE(6,200) ' '
        WRITE(6,200) ' '
        WRITE(6,200) ' DISPLAY:'
        WRITE(6,200) ' 1.) Only the user defined surfaces.'
        WRITE(6,200) ' 2.) User defined surfaces with selected' //
          ' transformations.'
        +
        WRITE(6,200) ' '
        WRITE(6,100) 'Enter 1 or 2 (default 2): '
        READ(5,150) L
150      FORMAT(BN10)
        IF (L .EQ. 1) NBLKS = 1
      END IF

C-----
C Find out if NSU and NSV subdivisions should be used in plot.
C-----
      ISUBDIV = 0
      WRITE(6,200) ' '
      WRITE(6,200) ' '
      WRITE(6,200) ' THE NUMBER OF SUBDIVISIONS PLOTTED:'
      WRITE(6,200) ' 1.) The optimal number for visualization.'
      WRITE(6,200) ' 2.) The number defined in the CHIEF driver.'
      WRITE(6,200) ' '
      WRITE(6,100) 'Enter 1 or 2 (default 1): '
      READ(5,150) L
      IF (L .EQ. 2) ISUBDIV = 1

C-----
C Find out if the user wishes to use dotted lines to represent
C negative normals.
C-----
      WRITE(6,200) ' '
      WRITE(6,200) ' '
      WRITE(6,200) ' DISTINGUISH BETWEEN POSITIVE AND NEGATIVE' //
        ' NORMAL VELOCITY:'
        +
      WRITE(6,200) ' 1.) Yes. (solid lines for positive,' //
        ' dashed for negative)'
        +
      WRITE(6,200) ' 2.) No. (only solid lines)'
      WRITE(6,200) ' '
      WRITE(6,100) 'Enter 1 or 2 (default 2): '
      READ(5,150) L

      IF (L .EQ. 1) THEN
        SHOW_NORMALS = 1
      ELSE
        SHOW_NORMALS = 0
      END IF

C-----
C Get angles of rotation (in degrees) about the 3 axes from the user.
C-----
      WRITE(6,200) ' '
      WRITE(6,200) ' '
      WRITE(6,200) ' Enter the angles (in degrees) for rotating ' //
        ' the model'
        +
200      FORMAT(A)
      WRITE(6,100) 'about the X, Y, and Z axes. (default 0.0, 0.0, 0.0)'
      WRITE(6,200) ' '

```

Siders and Bolling

```

WRITE(6,100) '(XROT, YROT, ZROT) : '
READ(5,225) AX, AY, AZ
225 FORMAT(3(E20.0))

C-----
C Initialize variables
C-----
POINTCNT = 0
SYMX = 1
SYMY = 1
SYMZ = 1

C-----
C Open the connectivity file. PLOTCID.CON will contain the line
C drawing data in the format: STARTPOINT ENDPOINT LINECOLOR
C Note: The actual coordinate data will be written to PLOTCID.CRD
C in the ROT3D() subroutine located near the end of this program,
C since the coordinates must be rotated before they can be written to
C a file.
C-----
TEMP = RUNID(1:LEN) // '.CON'
OPEN (UNIT=CON_UNIT, FILE=TEMP, TYPE='NEW')

C-----
C Generate points in each symmetry block.
C-----
DO BLOCK = NBLKS, 1, -1

    IF (SYMTYP .EQ. 1) THEN
C-----
C For current block, determines which planes of reflection
C-----
        SYMX = REFLECTX(BLOCK)
        SYMY = REFLECTY(BLOCK)
        SYMZ = REFLECTZ(BLOCK)
    ELSE IF (SYMTYP .EQ. 2) THEN
C-----
C For current block, determines angle to rotate
C-----
        SYMANG = (2 * PI * (BLOCK - 1) / NBLKS)
    END IF

C-----
C Loop through all the types of surfaces defined and call the proper
C shape drawing subroutine.
C-----
    DO REGION = 1, NSREG

C-----
C User defined block will be a different color from the computer
C generated block(s). If IZAX is negative (inward normal) then make
C color negative. The graphics driver should draw a dotted line when
C colors are negative.
C-----
        IF (BLOCK .EQ. 1) THEN
            IF (SHOW_NORMALS .EQ. 1) THEN
                COLOR = ISIGN(COLOR_USER_DEFINED, IZAX(REGION))
            ELSE
                COLOR = COLOR_USER_DEFINED
            END IF
        ELSE
            IF (SHOW_NORMALS .EQ. 1) THEN
                COLOR = ISIGN(COLOR_TRANSFORMED, IZAX(REGION))
            ELSE
                COLOR = COLOR_TRANSFORMED
            END IF
        END IF

        IF (NSEQNS(REGION) .EQ. 1) THEN
            CALL RECTANGULAR_PLANE(REGION, POINTCNT,
                COLOR, CON_UNIT, ISUBDIV)
        
```

NRL Memorandum Report 6709

```

      ELSE IF (NSEQNS(REGION) .EQ. 2) THEN
        CALL CIRCULAR_PLANE(REGION, POINTCNT,
          COLOR, CON_UNIT, ISUBDIV)
      *
      ELSE IF (NSEQNS(REGION) .EQ. 3) THEN
        CALL ELLIPTICAL_PLANE(REGION, POINTCNT,
          COLOR, CON_UNIT, ISUBDIV)
      *
      ELSE IF (NSEQNS(REGION) .EQ. 4) THEN
        CALL CIRCULAR_CYLINDER(REGION, POINTCNT,
          COLOR, CON_UNIT, ISUBDIV)
      *
      ELSE IF (NSEQNS(REGION) .EQ. 5) THEN
        CALL ELLIPTICAL_CYLINDER(REGION, POINTCNT,
          COLOR, CON_UNIT, ISUBDIV)
      *
      ELSE IF (NSEQNS(REGION) .EQ. 6) THEN
        CALL SPHEROID(REGION, POINTCNT, COLOR,
          CON_UNIT, ISUBDIV)
      *
      ELSE IF (NSEQNS(REGION) .EQ. 7) THEN
        CALL PROLATE_OBLATE_SPHEROID(REGION, POINTCNT, COLOR,
          CON_UNIT, ISUBDIV)
      *
      ELSE IF (NSEQNS(REGION) .EQ. 8) THEN
        CALL PROLATE_OBLATE_SPHEROID(REGION, POINTCNT, COLOR,
          CON_UNIT, ISUBDIV)
      *
      ELSE IF (NSEQNS(REGION) .EQ. 9) THEN
        CALL TOROID(REGION, POINTCNT, COLOR,
          CON_UNIT, ISUBDIV)
      *
      ELSE IF (NSEQNS(REGION) .EQ. 10) THEN
        CALL QUADRILATERAL(REGION, POINTCNT, COLOR,
          CON_UNIT, ISUBDIV)
      *
      ELSE IF (NSEQNS(REGION) .EQ. 11) THEN
        CALL AXISYMMETRIC(REGION, POINTCNT, COLOR,
          CON_UNIT, ISUBDIV)
      *
      ELSE IF (NSEQNS(REGION) .EQ. 12) THEN
        CALL TRIANGLE(REGION, POINTCNT, COLOR,
          CON_UNIT, ISUBDIV)
      *
      ELSE IF (NSEQNS(REGION) .EQ. 13) THEN
        CALL CONE(REGION, POINTCNT, COLOR,
          CON_UNIT, ISUBDIV)
      *
      END IF
    END DO
  END DO

  CALL CONNECT_ARROWS(CON_UNIT, POINTCNT)

  CLOSE(CON_UNIT)

```

 C Completed creating the shapes. Rotated coordinates about the X, Y,
 C and Z axes.
 C-----

```
CALL ROT3D(POINTCNT, AX, AY, AZ)
```

 C Write the rotated coordinates to coordinate file.
 C-----

```

TEMP = RUNID(1:LEN) // ' CRD'
OPEN (UNIT=CRD_UNIT, FILE=TEMP, TYPE='NEW')
DO I = 1, POINTCNT
  WRITE(CRD_UNIT,*) ROTATED(1,I), ROTATED(2,I), ROTATED(3,I)
END DO

CALL DRAW_ARROWS(CRD_UNIT, POINTCNT, AX, AY, AZ)

```

Siders and Bolling

```
CLOSE(CRD_UNIT)

WRITE(6,200) ' '
WRITE(6,200) ' '
TYPE *, 'Coordinate data has been stored in ' // TEMP
TEMP = RUNID(1:LEN) // '.CON'
TYPE *, 'Connection data has been stored in ' // TEMP
TYPE *, ' '
END
```

NRL Memorandum Report 6709

```

C-----
C
C   PLOTCHIEF() - written by D. Mitchell Bolling Jr.
C                   TRI/TESSCO, Inc.
C                   August 1989
C
C Purpose: The PLOTCHIEF subroutine is a graphics routine used to display
C three dimensional surfaces defined by the CHIEF program.
C CHIEF (Combined Helmholtz Integral Equation Formulation), developed
C at NOSC, computes the acoustic radiation from arbitrary-shaped bodies.
C PLOTCHIEF creates a plot file which can be viewed and rotated before
C compiling, linking and executing the CHIEF driver. PLOTCHIEF should
C be called after all calls to LDSURR are made.
C
C Subroutines used:  RECTANGULAR_PLANE()  CIRCULAR_PLANE()
C                   ELLIPTICAL_PLANE()   CIRCULAR_CYLINDER()
C                   ELLIPTICAL_CYLINDER() SPHEROID()
C                   TOROID()             PROLATE_OBLATE_SPHEROID()
C                   QUADRILATERAL()      AXISYMMETRIC()
C                   TRIANGLE()          QUADRILATERAL()
C                   AXISYMMETRIC()      TRIANGLE()
C                   CONE()              ROT3D()
C                   CONNECT_ARROWS()    DRAW_ARROWS()
C
C Modification Log:
C
C   AUG 89 - Added QUADRILATERAL(), AXISYMMETRIC() and TRIANGLE().
C   Also modified program by adding ROT3D so that the user can rotate
C   the points from within PLOTCHIEF. This requires that the points
C   are stored in an array, then passed to ROT3D, which writes them out
C   to a file. Before, the points were sent straight to a file.
C
C   JAN 90 - Added CONNECT_ARROWS() and DRAW_ARROWS() in order to
C   better show the axis orientation during rotations.
C
C   JUL 90 - Added CONE() geometry to PLOTCHIEF.
C-----
C
C   SUBROUTINE PLOTCHIEF(RUNID, NUMBLKS, CSYMTYPE, ISUBDIV,
C   *                AX, AY, AZ)
C
C   PARAMETER PI = 3.1415926536
C   PARAMETER COLOR_USER_DEFINED = 255 ! color to use for user defined part
C   PARAMETER COLOR_TRANSFORMED = 2    ! color for transformed part
C   PARAMETER CRD_UNIT = 8             ! run number of coordinate file
C   PARAMETER CON_UNIT = 9             ! run number of connectivity file
C   PARAMETER MXSREG = 500             ! Maximum number of surface regions
C   PARAMETER MAXCOORDS = 10000 ! maximum number of coordinates to plot
C-----
C Parameters used:
C-----
C   CHARACTER*(*) RUNID    ! title of CHIEF run
C   INTEGER NUMBLKS    ! number of symmetry blocks
C   CHARACTER*3 CSYMTYPE ! symmetry type from CHIEF
C   INTEGER ISUBDIV    ! if 1, then NSV and NSU are to be used
C   REAL AX            ! rotation angle in degrees about x axis
C   REAL AY            ! rotation angle in degrees about y axis
C   REAL AZ            ! rotation angle in degrees about z axis
C-----
C List of global variables:
C-----
C   INTEGER NSREG      ! number of CHIEF surfaces
C   INTEGER NSEQNS(MXSREG) ! types of CHIEF surfaces
C
C   COMMON/SVALS/NSREG,NSEQNS(MXSREG),SUL(MXSREG),SUJ(MXSREG),
C   *          SVL(MXSREG),SVU(MXSREG),NSU(MXSREG),NSV(MXSREG),
C   *          CCS(10,MXSREG),TRNSS(3,MXSREG),IZAX(MXSREG),
C   *          IORDU(MXSREG),IORDV(MXSREG),NCCEQS
C
C   INTEGER SYMTYP      ! type of symmetry: reflective (1),
C   *                  ! rotational (2), none (0)

```

Siders and Bolling

```

INTEGER      NBLKS      ! number of symmetry blocks

REAL         SYMX        ! for reflective symmetry:
REAL         SYMY        ! if -1, then reflects about YZ plane
REAL         SYMZ        ! if -1, then reflects about XZ plane
REAL         SYMZ        ! if -1, then reflects about XY plane

REAL         SYMANG      ! for rotational symmetry:
REAL         SYMANG      ! radians to rotate object

COMMON / PLOT_INFO / SYMTYP, NBLKS, SYMX, SYMY, SYMZ, SYMANG

REAL         COORD3D     ! array of 3d points
REAL         ROTATED     ! 3d points after rotations

COMMON / ROT_INFO / COORD3D(3, MAXCOORDS), ROTATED(3,MAXCOORDS)

C-----
C List of local variables:
C-----
CHARACTER*50 TEMP      ! a temporary string variable

INTEGER COLOR          ! number associated with different colors
INTEGER POINTCNT       ! keeps track of next point to be created
INTEGER SHOW_NORMALS   ! when set, uses dotted lines for inward normals
INTEGER REGION         ! counter used for looping through each region
INTEGER BLOCK          ! counter used for looping through each symmetry
                     ! block (whether reflective or rotational)

C-----
C Some or all of these constants are multiplied by each data point
C-----
REAL REFLECTX(8) ! array of 8 constants for reflecting about yz plane
REAL REFLECTY(8) ! array of 8 constants for reflecting about xz plane
REAL REFLECTZ(8) ! array of 8 constants for reflecting about xy plane

C-----
C Below is a table that indicates the number of constants needed for
C each reflective symmetry option. Thus, the dashed horizontal line
C is over the constants needed for each option.
C-----
C 3 plane symmetry |<----->|
C 2 plane symmetry |<----->|
C 1 plane symmetry |<---->|
C                  |<---->|
DATA REFLECTX / 1, -1, 1, -1, 1, -1, 1, -1 /
DATA REFLECTY / 1, 1, -1, -1, 1, 1, -1, -1 /
DATA REFLECTZ / 1, 1, 1, 1, -1, -1, -1, -1 /

C-----
C Initialize variables
C-----
POINTCNT = 0
SYMX = 1
SYMY = 1
SYMZ = 1

C-----
C If NUMBLKS is positive, then display of normals is differentiated
C by line type. Solid for positive normals, dashed for negative normal.
C-----
IF (NUMBLKS .LT. 0) THEN
  NBLKS = ABS(NUMBLKS)
  SHOW_NORMALS = 0
ELSE
  NBLKS = NUMBLKS
  SHOW_NORMALS = 1
END IF

IF (CSYMTYPE .EQ. 'REF') THEN
  SYMTYP = 1

```

NRL Memorandum Report 6709

```

ELSE
  SYMTYP = 2
END IF

C-----
C Open the connectivity file. <RUNID>.CON will contain the line
C drawing data in the format:  STARTPOINT  ENDPOINT  LINECOLOR
C Note: The actual coordinate data will be written to <RUNID>.CRD
C in the ROT3D() subroutine located near the end of this program,
C since the coordinates must be rotated before they can be written to
C a file.
C-----
TEMP = RUNID // '.CON'
OPEN (UNIT=CON_UNIT,FILE=TEMP,TYPE='NEW')

C-----
C Generate points in each symmetry block.
C-----
DO BLOCK = NBLKS, 1, -1

  IF (SYMTYP .EQ. 1) THEN
C-----
C For current block, determines which planes of reflection
C-----
    SYMX = REFLECTX(BLOCK)
    SYMY = REFLECTY(BLOCK)
    SYMZ = REFLECTZ(BLOCK)
  ELSE IF (SYMTYP .EQ. 2) THEN
C-----
C For current block, determines angle to rotate
C-----
    SYMANG = (2 * PI * (BLOCK - 1) / NBLKS)
  END IF

C-----
C Loop through all the types of surfaces defined and call the proper
C shape drawing subroutine.
C-----
  DO REGION = 1, NSREG
C-----
C User defined block will be a different color from the computer
C generated block(s). If IZAX is negative (inward normal) then make
C color negative. The graphics driver should draw a dotted line when
C colors are negative.
C-----
    IF (BLOCK .EQ. 1) THEN
      IF (SHOW_NORMALS .EQ. 1) THEN
        COLOR = ISIGN(COLOR_USER_DEFINED, IZAX(REGION))
      ELSE
        COLOR = COLOR_USER_DEFINED
      END IF
    ELSE
      IF (SHOW_NORMALS .EQ. 1) THEN
        COLOR = ISIGN(COLOR_TRANSFORMED, IZAX(REGION))
      ELSE
        COLOR = COLOR_TRANSFORMED
      END IF
    END IF

    IF (NSEQNS(REGION) .EQ. 1) THEN
      CALL RECTANGULAR_PLANE(REGION, POINTCNT,
        COLOR, CON_UNIT, ISUBDIV)
    *
    ELSE IF (NSEQNS(REGION) .EQ. 2) THEN
      CALL CIRCULAR_PLANE(REGION, POINTCNT,
        COLOR, CON_UNIT, ISUBDIV)
    *
    ELSE IF (NSEQNS(REGION) .EQ. 3) THEN
      CALL ELLIPTICAL_PLANE(REGION, POINTCNT,
        COLOR, CON_UNIT, ISUBDIV)
    *
    ELSE IF (NSEQNS(REGION) .EQ. 4) THEN
      CALL CIRCULAR_CYLINDER(REGION, POINTCNT,

```

Siders and Bolling

```

      COLOR, CON_UNIT, ISUBDIV)

      ELSE IF (NSEQNS(REGION) .EQ. 5) THEN
        CALL ELLIPTICAL_CYLINDER(REGION, POINTCNT,
      COLOR, CON_UNIT, ISUBDIV)

      ELSE IF (NSEQNS(REGION) .EQ. 6) THEN
        CALL SPHEROID(REGION, POINTCNT, COLOR,
      CON_UNIT, ISUBDIV)

      ELSE IF (NSEQNS(REGION) .EQ. 7) THEN
        CALL PROLATE_OBLATE_SPHEROID(REGION, POINTCNT,
      COLOR, CON_UNIT, ISUBDIV)

      ELSE IF (NSEQNS(REGION) .EQ. 8) THEN
        CALL PROLATE_OBLATE_SPHEROID(REGION, POINTCNT,
      COLOR, CON_UNIT, ISUBDIV)

      ELSE IF (NSEQNS(REGION) .EQ. 9) THEN
        CALL TOROID(REGION, POINTCNT, COLOR, CON_UNIT, ISUBDIV)

      ELSE IF (NSEQNS(REGION) .EQ. 10) THEN
        CALL QUADRILATERAL(REGION, POINTCNT,
      COLOR, CON_UNIT, ISUBDIV)

      ELSE IF (NSEQNS(REGION) .EQ. 11) THEN
        CALL AXISYMMETRIC(REGION, POINTCNT,
      COLOR, CON_UNIT, ISUBDIV)

      ELSE IF (NSEQNS(REGION) .EQ. 12) THEN
        CALL TRIANGLE(REGION, POINTCNT,
      COLOR, CON_UNIT, ISUBDIV)

      ELSE IF (NSEQNS(REGION) .EQ. 13) THEN
        CALL CONE(REGION, POINTCNT,
      COLOR, CON_UNIT, ISUBDIV)
      END IF
    END DO
  END DO

  CALL CONNECT_ARROWS(CON_UNIT, POINTCNT)

  CLOSE(CON_UNIT)

C-----
C Completed creating the shapes. Rotate coordinates about the X, Y,
C and Z axes.
C-----
  CALL ROT3D(POINTCNT, AX, AY, AZ)

C-----
C Write the rotated coordinates to coordinate file.
C-----
  TEMP = RUNID // '.CRD'
  OPEN (UNIT=CRD_UNIT, FILE=TEMP, TYPE='NEW')
  DO I = 1, POINTCNT
    WRITE(CRD_UNIT,*) ROTATED(1,I), ROTATED(2,I), ROTATED(3,I)
  END DO

  CALL DRAW_ARROWS(CRD_UNIT, POINTCNT, AX, AY, AZ)

  CLOSE(CRD_UNIT)

  WRITE (6,*) ' '
  WRITE (6,*) 'Coordinate data has been stored in ' // TEMP
  TEMP = RUNID // '.CON'
  WRITE (6,*) 'Connection data has been stored in ' // TEMP
  WRITE (6,*) ' '
  RETURN
END

```

NRL Memorandum Report 6709

```

C-----
C
C  LOAD_CID_DATA() - written by D. Mitchell Bolling Jr.
C                   TRI/TESSCO, Inc.
C                   August 1989
C
C  Purpose: This subroutine was designed to read necessary data from a
C  CID save file into global variables used by PLOT_CID.
C
C  Subroutines used: None
C
C  Modification Log:
C-----

      SUBROUTINE LOAD_CID_DATA(FILENAME)

      PARAMETER MXSREG = 500      ! Maximum number of surface regions
      PARAMETER MAXCOR = 1000     ! Maximum number of finite element nodes
      PARAMETER PI = 3.1415926536

C-----
C  List of parameters:
C-----
      CHARACTER*(*) FILENAME      ! name of CID data file to load
C-----
C  List of global variables:
C-----
      INTEGER      NSREG           ! number of surfaces
      INTEGER      NSEQNS(MXSREG) ! types of surfaces
      REAL         SUL(MXSREG)     ! lower limits of U
      REAL         SUU(MXSREG)     ! upper limits of U
      REAL         SVL(MXSREG)     ! lower limits of V
      REAL         SVU(MXSREG)     ! upper limits of V
      REAL         CCS(10,MXSREG) ! constants needed for CHIEF equations
      REAL         TRNSS(3,MXSREG) ! translation from local to global origin
      INTEGER      IZAX(MXSREG)    ! global axis that corresponds to local
                                   ! Z axis

      COMMON/SVALS/NSREG,NSEQNS(MXSREG),SUL(MXSREG),SUU(MXSREG),
      *           SVL(MXSREG),SVU(MXSREG),NSU(MXSREG),NSV(MXSREG),
      *           CCS(10,MXSREG),TRNSS(3,MXSREG),IZAX(MXSREG),
      *           IORDU(MXSREG),IORDV(MXSREG),NCCEQS

      INTEGER      SYMTYP          ! type of symmetry
      INTEGER      NBLKS           ! number of symmetry blocks
                                   ! for reflective symmetry:
      REAL         SYMX            ! if -1, then reflects across YZ plane
      REAL         SYMY            ! if -1, then reflects across XZ plane
      REAL         SYMZ            ! if -1, then reflects across XY plane

                                   ! for rotational symmetry:
      REAL         SYMANG          ! radians to rotate object

      COMMON / PLOT_INFO / SYMTYP, NBLKS, SYMX, SYMY, SYMZ, SYMANG

      COMMON / CORD / COORDS(MAXCOR, 3)

C-----
C  List of local variables.
C-----
      CHARACTER*8  COOR_FILE
      CHARACTER*3  COOR_EXT
      CHARACTER*55 COOR_FORMAT
      CHARACTER*8  ELEM_FILE
      CHARACTER*3  ELEM_EXT
      CHARACTER*55 ELEM_FORMAT

      INTEGER      CTEMP(10)

C-----
C  The following variables are dummy variables that are used to
C  read past unwanted data. Some of them are also used to temporarily
C  hold string data so that it may be converted into numeric data.

```

Siders and Bolling

```

C-----
      INTEGER      IDUM
      INTEGER      TDUM50(50)

      CHARACTER*3  CDUM3
      CHARACTER*4  CDUM4
      CHARACTER*7  CDUM7
      CHARACTER*8  CDUM8
      CHARACTER*9  CDUM9
      CHARACTER*58 CDUM58
      CHARACTER*1  CDUM1_50(50)
      CHARACTER*2  CDUM2_50(50)
      CHARACTER*8  CDUM8_100_3(100,3)
      CHARACTER*8  CDUM8_50_3(100,3)
      CHARACTER*10 CDUM10_50(50)
      CHARACTER*10 CDUM10_50_10(50,10)
      CHARACTER*20 CDUM20_50(50)

C-----
C  Open the CID interactive driver's save file.
C-----
      OPEN(UNIT=20,FILE=FILENAME,STATUS='OLD',
      *      FORM='UNFORMATTED')

      READ(20) CDUM8
      READ(20) CDUM9
      READ(20) CDUM58
      READ(20) CDUM7
      READ(20) CDUM7
      READ(20) CDUM7
      READ(20) CDUM7
      READ(20) CDUM4
      READ(20) CDUM4

C-----
C  Depending on the value of CDUM3, SYMTYP is set for either reflective
C  or rotational symmetry.
C-----
      READ(20) CDUM3
      IF (CDUM3 .EQ. 'REF') THEN
        SYMTYP = 1
      ELSE IF (CDUM3 .EQ. 'ROT') THEN
        SYMTYP = 2
      ELSE
        SYMTYP = 0
      END IF

C-----
C  Read and convert the string CDUM3 into a number and store in NBLKS.
C  NBLKS will be either 1, 2 or 3 for reflective symmetry, representing
C  the number of planes, or number of blocks for rotational symmetry.
C-----
      READ(20) CDUM3
      READ(UNIT=CDUM3(1:3),FMT='(BNI3)') NBLKS

      READ(20) IDUM
      READ(20) IDUM
      READ(20) ((CDUM8_100_3(X,XX),X=1,100),XX=1,3)
      READ(20) IDUM

C-----
C  Reading in the number of CHIEF regions defined
C-----
      READ(20) NSREG

      READ(20) (CDUM20_50(X),X=1,50)

C-----
C  Read in the surface type for each CHIEF region
C-----
      READ(20) (NSEQNS(X),X=1,50)
C-----

```

NRL Memorandum Report 6709

C Reading in the local-to-global translations. There are 3 of them
C for each region.

```

C-----
      READ(20) ((CDUM8_50_3(X,XX),X=1,50),XY=1,3)
      DO X = 1, 50
        DO XX = 1, 3
          READ(UNIT=CDUM8_50_3(X,XX)(1:8),
            *          FMT='(BNE8.0)') TRNSS(X,X)
        END DO
      END DO

```

C-----
C Read in IZAX for each region. IZAX describes the global orientation
C of each region.

```

C-----
      READ(20) (CDUM2_50(X),X=1,50)
      READ(20) (CDUM1_50(X),X=1,50)
      DO X = 1, 50
        READ(UNIT=CDUM2_50(X)(2:2),FMT='(BN11)') IZAX(X)
        IF (CDUM1_50(X)(1:1).EQ. '-') IZAX(X) = -IZAX(X)
      END DO

      READ(20) (CDUM2_50(X),X=1,50)
      READ(20) (CDUM2_50(X),X=1,50)

```

C-----
C Read in the lower limit of U for all the regions

```

C-----
      READ(20) (CDUM10_50(X),X=1,50)
      DO X = 1, 50
        READ(UNIT=CDUM10_50(X)(1:10),
            *          FMT='(BNE10.0)') SUL(X)
      END DO

```

C-----
C Read in the lower limit of V for all the regions

```

C-----
      READ(20) (CDUM10_50(X),X=1,50)
      DO X = 1, 50
        READ(UNIT=CDUM10_50(X)(1:10),
            *          FMT='(BNE10.0)') SVL(X)
      END DO

```

C-----
C Read in the upper limit of U for all the regions

```

C-----
      READ(20) (CDUM10_50(X),X=1,50)
      DO X = 1, 50
        READ(UNIT=CDUM10_50(X)(1:10),
            *          FMT='(BNE10.0)') SUU(X)
      END DO

```

C-----
C Read in the upper limit of V for all the regions

```

C-----
      READ(20) (CDUM10_50(X),X=1,50)
      DO X = 1, 50
        READ(UNIT=CDUM10_50(X)(1:10),
            *          FMT='(BNE10.0)') SVU(X)
      END DO

```

C-----
C Read in the constants for all the regions used in CHIEF subroutine CCUMND

```

C-----
      READ(20) ((CDUM10_50_10(X,XX),X=1,50),XX=1,10)
      DO X = 1, 50
        DO XX = 1, 10
          READ(UNIT=CDUM10_50_10(X,XX)(1:10),
            *          FMT='(BNE10.0)',ERR=100) CCS(X,X)
        END DO
      END DO
100

```

Siders and Bolling

```

C-----
C Read in values for NSU, the number of subdivisions in the U direction.
C-----
      READ(20) (CDUM2_50(X),X=1,50)
      DO X = 1, 50
        READ(UNIT=CDUM2_50(X)(1:2),FMT='(BN12)') NSU(X)
      END DO

C-----
C Read in values for NSV, the number of subdivisions in the V direction.
C-----
      READ(20) (CDUM2_50(X),X=1,50)
      DO X = 1, 50
        READ(UNIT=CDUM2_50(X)(1:2),FMT='(BN12)') NSV(X)
      END DO

      READ(20) (CDUM1_50(X),X=1,50)
      READ(20) (CDUM10_50(X),X=1,50)
      READ(20) (CDUM10_50(X),X=1,50)

C-----
C The coordinate and element files are used with shapes 10, 11, and 12.
C-----
      READ(20) COOR_FILE
      READ(20) COOR_EXT
      READ(20) COOR_FORMAT

      READ(20) ELEM_FILE
      READ(20) ELEM_EXT
      READ(20) ELEM_FORMAT

C-----
C Close the file.
C-----
      CLOSE(20)

      IF ((NSREG .EQ. 1) .AND. (NSEQNS(1) .EQ. 10)) THEN
C-----
C If there is only one surface, and the surface equation is
C linear or quadratic interpolation over a quadrilateral, then
C read the nodes into the COORDS array.
C-----
        CDUM58 = COOR_FILE // '.' // COOR_EXT
        OPEN(UNIT=20,FILE=CDUM58,TYPE='OLD')
        IDUM = 1
        CDUM58 = '(' // COOR_FORMAT // ')'
200      READ(20,FMT=CDUM58,END=225) (COORDS(IDUM,X),X=1,3)
        IDUM = IDUM + 1
        GO TO 200
225      CLOSE(20)

C-----
C Read in the element data into CCS(1) through CCS(8).
C-----
        CDUM58 = ELEM_FILE // '.' // ELEM_EXT
        OPEN(UNIT=20,FILE=CDUM58,TYPE='OLD')
        NSREG = 1
        CDUM58 = '(' // ELEM_FORMAT // ')'
250      READ(20,FMT=CDUM58,END=275) (CCTEMP(X),X=1,8)
        DO I = 1, 8
          CCS(I,NSREG) = CCTEMP(I)
        END DO
        NSEQNS(NSREG) = 10
        NSREG = NSREG + 1
        GO TO 250
275      CLOSE(20)
        NSREG = NSREG - 1
      ELSE IF ((NSREG .EQ. 1) .AND. (NSEQNS(1) .EQ. 11)) THEN
C-----
C If there is only one surface, and the surface equation is
C linear or quadratic axisymmetric interpolation, then

```

NRL Memorandum Report 6709

```

C read the nodes into the COORDS array.
C-----
      CDUM58 = COOR FILE // ' ' // COOR EXT
      OPEN(UNIT=20,FILE=CDUM58,TYPE='OLD')
      IDUM = 1
      CDUM58 = '(' // COOR FORMAT // ')'
302  READ(20,FMT=CDUM58,END=325) (COORDS(IDUM,X),X=1,2)
      IDUM = IDUM + 1
      GOTO 302
325  CLOSE(20)

C-----
C Read in the element data into CCS(1) through CCS(3).
C-----
      CDUM58 = ELEM FILE // ' ' // ELEM EXT
      OPEN(UNIT=20,FILE=CDUM58,TYPE='OLD')
      NSREG = 1
      CDUM58 = '(' // ELEM FORMAT // ')'
350  READ(20,FMT=CDUM58,END=375) (CCTEMP(X),X=1,3)
      DO I = 1, 3
         CCS(I,NSREG) = CCTEMP(I)
      END DO
      NSEQNS(NSREG) = 11
      SVL(NSREG) = -PI / NBLKS
      SVU(NSREG) = PI / NBLKS
      NSREG = NSREG + 1
      GOTO 350
375  CLOSE(20)
      NSREG = NSREG - 1
      ELSE IF ((NSREG.EQ. 1) .AND. (NSEQNS(1).EQ. 12)) THEN
C-----
C If there is only one surface, and the surface equation is
C linear or quadratic interpolation over a triangle, then
C read the nodes into the COORDS array.
C-----
      CDUM58 = COOR FILE // ' ' // COOR EXT
      OPEN(UNIT=20,FILE=CDUM58,TYPE='OLD')
      IDUM = 1
      CDUM58 = '(' // COOR FORMAT // ')'
400  READ(20,FMT=CDUM58,END=425) (COORDS(IDUM,X),X=1,3)
      IDUM = IDUM + 1
      GOTO 400
425  CLOSE(20)

C-----
C Read in the element data into CCS(1) through CCS(6).
C-----
      CDUM58 = ELEM FILE // ' ' // ELEM EXT
      OPEN(UNIT=20,FILE=CDUM58,TYPE='OLD')
      NSREG = 1
      CDUM58 = '(' // ELEM FORMAT // ')'
450  READ(20,FMT=CDUM58,END=475) (CCTEMP(X),X=1,6)
      DO I = 1, 6
         CCS(I,NSREG) = CCTEMP(I)
      END DO
      NSEQNS(NSREG) = 12
      NSREG = NSREG + 1
      GOTO 450
475  CLOSE(20)
      NSREG = NSREG - 1
      END IF

      RETURN
      END

```

Siders and Bolling

```

C-----
C
C   PUT_POINT()   - written by D. Mitchell Bolling Jr.
C                   TRI/TESSCO, Inc.
C                   August 1989
C
C Purpose: This subprogram takes a 3D point and translates it from
C the local coordinate system to the global coordinate system. It
C then reflects or rotates the point, if necessary and writes it to
C a file.
C
C Subroutines used: none
C
C Modification Log:
C-----

      SUBROUTINE PUT_POINT(IRC, N, X, Y, Z)

      PARAMETER MXSREG = 500      ! Maximum number of surface regions
      PARAMETER MAXCOORDS = 10000

C-----
C List of parameters used:
C-----
      INTEGER IRC      ! CHIEF surface region identification number
      INTEGER N        ! current coordinate number
      REAL      X, Y, Z ! Local coordinates

C-----
C List of global variables used:
C-----
      REAL      TRNSS(3,MXSREG) ! translation from local to global origin
      INTEGER   IZAX(MXSREG)    ! global axis that corresponds to local
                                ! Z axis

      COMMON/SVALS/NSREG, NSEQNS(MXSREG), SUL(MXSREG), SUU(MXSREG),
      *          SVL(MXSREG), SVU(MXSREG), NSU(MXSREG), NSV(MXSREG),
      *          CCS(10,MXSREG), TRNSS(3,MXSREG), IZAX(MXSREG),
      *          IORDU(MXSREG), IORDV(MXSREG), NCCEQS

      INTEGER   SYMTYP      ! type of symmetry
      INTEGER   NBLKS       ! number of symmetry blocks
                                ! for reflective symmetry:
      REAL      SYMX        ! if -1, then reflects across YZ plane
      REAL      SYMY        ! if -1, then reflects across XZ plane
      REAL      SYMZ        ! if -1, then reflects across XY plane

                                ! for rotational symmetry:
      REAL      SYMANG       ! radians to rotate object

      COMMON / PLOT_INFO / SYMTYP, NBLKS, SYMX, SYMY, SYMZ, SYMANG

      REAL      COORD3D      ! array of 3d points
      REAL      ROTATED      ! 3d points after rotations

      COMMON / ROT_INFO / COORD3D(3, MAXCOORDS), ROTATED(3, MAXCOORDS)

C-----
C Local variables used.
C-----

      REAL      PX, PY, PZ   ! holds global coordinates of point

C-----
C Translate from local to global coordinates according to IZAX.
C-----
      IF (IZAX(IRC) .EQ. 1) THEN
        PX = Z + TRNSS(1, IRC)
        PY = X + TRNSS(2, IRC)
        PZ = Y + TRNSS(3, IRC)
      ELSE IF (IZAX(IRC) .EQ. 2) THEN

```

NRL Memorandum Report 6709

```

PX = Y + TRNSS(1, IRG)
PY = Z + TRNSS(2, IRG)
PZ = X + TRNSS(3, IRG)
ELSE IF (IZAX(IRG) .EQ. 3) THEN
PX = X + TRNSS(1, IRG)
PY = Y + TRNSS(2, IRG)
PZ = Z + TRNSS(3, IRG)
ELSE IF (IZAX(IRG) .EQ. -1) THEN
PX = -Z + TRNSS(1, IRG)
PY = X + TRNSS(2, IRG)
PZ = -Y + TRNSS(3, IRG)
ELSE IF (IZAX(IRG) .EQ. -2) THEN
PX = -Y + TRNSS(1, IRG)
PY = -Z + TRNSS(2, IRG)
PZ = X + TRNSS(3, IRG)
ELSE IF (IZAX(IRG) .EQ. -3) THEN
PX = X + TRNSS(1, IRG)
PY = -Y + TRNSS(2, IRG)
PZ = -Z + TRNSS(3, IRG)
END IF

```

C-----
C Do reflective or rotational symmetry if needed on the GLOBAL coordinates.
C-----

```

IF (SYMTYP .EQ. 1) THEN
PX = PX * SYMX
PY = PY * SYMY
PZ = PZ * SYMZ
ELSE
R = SQRT(PX*PX + PY*PY)
IF (R .NE. 0.0) THEN
A = ATAN2(PY, PX)
A = A + SYMANG
PX = R * COS(A)
PY = R * SIN(A)
END IF
END IF

```

C-----
C Output the point to the coordinate file for plotting.
C-----

```

N = N + 1
COORD3D(1, N) = PX
COORD3D(2, N) = PY
COORD3D(3, N) = PZ
RETURN
END

```

Siders and Bolling

```

C-----
C
C  RECTANGULAR_PLANE() - written by D. Mitchell Bolling Jr.
C                        TRI/TESSCO, Inc.
C                        August 1989
C
C  Purpose:  RECTANGULAR_PLANE is called by PLOTCID or PLOTCHIEF. It
C            generates the points and lines required for drawing a cylinder or
C            section of a cylinder.
C
C  Subroutines used:  PUT_POINT()
C
C  Modification Log:
C-----

      SUBROUTINE RECTANGULAR_PLANE(IRG, N, COLOR, CON_UNIT, SUBDIV)

      PARAMETER MXSREG = 500      ! Maximum number of surface regions

C-----
C  List of subroutine parameters:
C-----
      INTEGER IRG      ! surface region id number (1 < IRG < NSREG)
      INTEGER N        ! current coordinate number
      INTEGER COLOR    ! number representing the color to draw plane
      INTEGER CON_UNIT ! lun number for connections file
      INTEGER SUBDIV   ! if set, then use NSU and NSV for subdivisions

C-----
C  List of global variables:
C-----
      REAL      SUL(MXSREG)      ! lower limits of U
      REAL      SUU(MXSREG)      ! upper limits of U
      REAL      SVL(MXSREG)      ! lower limits of V
      REAL      SVU(MXSREG)      ! upper limits of V
      REAL      CCS(10,MXSREG)   ! constants needed for CHIEF equations

      COMMON/SVALS/NSREG,NSEQNS(MXSREG),SUL(MXSREG),SUU(MXSREG),
      *          SVL(MXSREG),SVU(MXSREG),NSU(MXSREG),NSV(MXSREG),
      *          CCS(10,MXSREG),TRNSS(3,MXSREG),IZAX(MXSREG),
      *          IORDU(MXSREG),IORDV(MXSREG),NCCEQS

C-----
C  List of local variables:
C-----
      INTEGER ICOUNT      ! counter used to draw segmenting
      INTEGER JCOUNT     ! counter used to draw segmenting
      INTEGER BEGINPOINT  ! used for connecting segments
      INTEGER ENDPOINT    ! used for connecting segments
      INTEGER NUMVPOINTS  ! number of points used when drawing U lines
      INTEGER NUMUPOINTS  ! number of points used when drawing V lines

      REAL I, J           ! scratch variables
      REAL X, Y, Z        ! current calculated x, y, z values (local)
      REAL USTEP          ! distance between each U segment drawn
      REAL VSTEP          ! distance between each V segment drawn

C-----
C  If SUBDIV is set, then use NSV and NSU to determine segmenting,
C  otherwise, segmenting defaults to 4 for U and V direction.
C-----
      IF (SUBDIV.EQ.1) THEN
        NUMVPOINTS = NSV(IRG)
        NUMUPOINTS = NSU(IRG)
      ELSE
        NUMVPOINTS = 4
        NUMUPOINTS = 4
      END IF

      USTEP = (SUU(IRG) - SUL(IRG)) / NUMUPOINTS
      VSTEP = (SVU(IRG) - SVL(IRG)) / NUMVPOINTS

```

NRL Memorandum Report 6709

```
C-----
C Draw parallel lines in V direction.
C-----
      Z = CCS(1, IRG)
      X = SUL(IRG)
      DO JCOUNT = 0, NUMVPOINTS
        Y = SVL(IRG)
        CALL PUT_POINT(IRG, N, X, Y, Z)
        Y = SVU(IRG)
        CALL PUT_POINT(IRG, N, X, Y, Z)
        WRITE(CON UNIT,*) N - 1, N, COLOR
        X = X + USTEP
      END DO

C-----
C Draw lines parallel to U direction.
C-----
      Y = SVL(IRG)
      DO ICOUNT = 0, NUMVPOINTS
        X = SUL(IRG)
        CALL PUT_POINT(IRG, N, X, Y, Z)
        X = SUU(IRG)
        CALL PUT_POINT(IRG, N, X, Y, Z)
        WRITE(CON UNIT,*) N - 1, N, COLOR
        Y = Y + VSTEP
      END DO

      RETURN
      END
```

Siders and Bolling

```

C-----
C
C CIRCULAR_PLANE() - written by D. Mitchell Bolling Jr.
C TRI/TESSCO, Inc.
C August 1989
C
C Purpose: CIRCULAR_PLANE is called by PLOTCHIEF. It
C generates the points and lines required for drawing a circular
C planar region or section of a circular planar region.
C
C Subroutines used: PUT_POINT()
C
C Modification Log:
C-----

      SUBROUTINE CIRCULAR_PLANE(IRG, N, COLOR, CON_UNIT, SUBDIV)

      PARAMETER PI = 3.1415926536
      PARAMETER MXSREG = 500      ! Maximum number of surface regions
C-----
C List of subroutine parameters:
C-----
      INTEGER IRG      ! CHIEF surface region id number (1 < IRG < NSREG)
      INTEGER N        ! current coordinate number
      INTEGER COLOR    ! number representing color to draw the plane.
      INTEGER CON_UNIT ! lun number for connections file
      INTEGER SUBDIV   ! if set, then use NSU and NSV for subdivisions
C-----
C List of global variables:
C-----
      REAL      SUL(MXSREG)      ! lower limits of U
      REAL      SUU(MXSREG)      ! upper limits of U
      REAL      SVL(MXSREG)      ! lower limits of V
      REAL      SVU(MXSREG)      ! upper limits of V
      REAL      CCS(10,MXSREG)   ! constants needed for CHIEF equations

      COMMON/SVALS/NSREG, NSEQNS(MXSREG), SUL(MXSREG), SUU(MXSREG),
      * SVL(MXSREG), SVU(MXSREG), NSU(MXSREG), NSV(MXSREG),
      * CCS(10,MXSREG), TRNSS(3,MXSREG), IZAX(MXSREG),
      * IORCU(MXSREG), IORDV(MXSREG), NCCEQS
C-----
C List of local variables:
C-----
      INTEGER ICOUNT      ! counter used to draw segmenting
      INTEGER JCOUNT    ! counter used to draw segmenting
      INTEGER BEGINPOINT ! first point of circular plane
      INTEGER ENDPOINT   ! last point of circular plane
      INTEGER NUMVPOINTS ! number of points to use when drawing V arcs
      INTEGER NUMUPOINTS ! number of points to use when drawing U lines

      REAL I, J          ! scratch variable
      REAL X, Y, Z       ! current calculated x, y, z values (local)
      REAL VSTEP         ! angle between each V arc point drawn (in radians)
      REAL USTEP         ! distance between each U segment drawn
C-----
C Determine the number of points to plot for arc and the step angle
C between each point. For a full 2 * PI (360 degree) arc, 48 points
C will be used. The minimum number of points used is 4.
C-----
      IF (SUBDIV .EQ. 1) THEN
        NUMVPOINTS = 4 * NSV(IRG)
        NUMUPOINTS = NSU(IRG)
      ELSE
        I = 24.0 * (SVU(IRG) - SVL(IRG)) / PI
        NUMVPOINTS = 4 * INT(I / 4.0 + .99)
        NUMUPOINTS = 1
      END IF

```

NRL Memorandum Report 6709

```
VSTEP = (SVU(IRC) - SVL(IRC)) / NUMVPOINTS
USTEP = (SUU(IRC) - SUL(IRC)) / NUMUPOINTS
```

```
J = SUL(IRC)
BEGINPOINT = N + 1
DO ICOUNT = 0, NUMUPOINTS
```

```
C-----
C Plot the first point of the outer arc.
```

```
C-----
      X = J * COS(SVL(IRC))
      Y = J * SIN(SVL(IRC))
      Z = CCS(1,IRC)
      CALL PUT POINT(IRC, N, X, Y, Z)
      ENDPOINT = N
```

```
C-----
C Plot the outer arc.
```

```
C-----
      I = SVL(IRC) + VSTEP
      DO ICOUNT = 1, NUMVPOINTS
        X = J * COS(I)
        Y = J * SIN(I)
        CALL PUT POINT(IRC, N, X, Y, Z)
        WRITE(CON UNIT,*) N - 1, N, COLOR
        I = I + VSTEP
      END DO
      J = J + USTEP
    END DO
```

```
C-----
C Subdivide it into sections to show solidity and exit.
```

```
C-----
      DO ICOUNT = BEGINPOINT, BEGINPOINT + NUMVPOINTS, 4
        WRITE(CON UNIT,*) ICOUNT, ENDPOINT, COLOR
        ENDPOINT = ENDPOINT + 4
      END DO

      RETURN
    END
```

Siders and Bolling

```

C-----
C
C ELLIPTICAL_PLANE() - written by D. Mitchell Bolling Jr.
C                      TRI/TESSCO, Inc.
C                      August 1989
C
C Purpose: ELLIPTICAL_PLANE is called by PLOTCHIEF. It
C generates the points and lines required for drawing a elliptical
C planar region or section of an elliptical planar region.
C
C Subroutines used: PUT_POINT()
C
C Modification Log:
C-----

      SUBROUTINE ELLIPTICAL_PLANE(IRG, N, COLOR, CON_UNIT, SUBDIV)

      PARAMETER PI = 3.1415926536
      PARAMETER MXSREG = 500      ! Maximum number of surface regions
C-----
C List of subroutine parameters:
C-----
      INTEGER IRG      ! CHIEF surface region id number (1 < IRG < NSREG)
      INTEGER N        ! current coordinate number
      INTEGER COLOR    ! number representing color to draw the plane
      INTEGER CON_UNIT ! lun number for connections file
      INTEGER SUBDIV   ! if set, then use NSU and NSV for subdivisions
C-----
C List of global variables:
C-----
      REAL      SUL(MXSREG)      ! lower limits of U
      REAL      SUU(MXSREG)      ! upper limits of U
      REAL      SVL(MXSREG)      ! lower limits of V
      REAL      SVU(MXSREG)      ! upper limits of V
      REAL      CCS(10,MXSREG)   ! constants needed for CHIEF equations

      COMMON/SVALS/NSREG,NSEQNS(MXSREG),SUL(MXSREG),SUU(MXSREG),
      *          SVL(MXSREG),SVU(MXSREG),NSU(MXSREG),NSV(MXSREG),
      *          CCS(10,MXSREG),TRNSS(3,MXSREG),IZAX(MXSREG),
      *          IORDU(MXSREG),IORDV(MXSREG),NCCEQS
C-----
C List of local variables:
C-----
      INTEGER ICOUNT      ! counter used to draw segmenting
      INTEGER JCOUNT    ! counter used to draw segmenting
      INTEGER BEGINPOINT ! first point of circular plane
      INTEGER ENDPOINT   ! last point of circular plane
      INTEGER NUMVPOINTS ! number of points to use when drawing V arc
      INTEGER NUMUPOINTS ! number of points to use when drawing U lines

      REAL I, J          ! scratch variable
      REAL X, Y, Z       ! current calculated x, y, z values (local)
      REAL VSTEP         ! angle between each V arc point drawn (in radians)
      REAL USTEP         ! distance between each U segment drawn
C-----
C Determine the number of points to plot for arc and the step angle
C between each point. For a full 2 * PI (360 degree) arc, 48 points
C will be used. The minimum number of points used is 4.
C-----
      IF (SUBDIV .EQ. 1) THEN
        NUMVPOINTS = 4 * NSV(IRG)
        NUMUPOINTS = NSU(IRG)
      ELSE
        I = 24.0 * (SVU(IRG) - SVL(IRG)) / PI
        NUMVPOINTS = 4 * INT(I / 4.0 + .99)
        NUMUPOINTS = 1
      END IF

```

NRL Memorandum Report 6709

```

VSTEP = (SVU(IRG) - SVL(IRG)) / NUMVPOINTS
USTEP = (SUU(IRG) - SUL(IRG)) / NUMUPOINTS

J = SUL(IRG)
BEGINPOINT = N + 1
DO JCOUNT = 0, NUMUPOINTS
C-----
C Calculate the first point of outer ellipse
C-----
      X = CCS(1,IRG) * J * COS(SVL(IRG))
      Y = CCS(1,IRG) * Sqrt(J * J - 1.0) * SIN(SVL(IRG))
      Z = CCS(2,IRG)
      CALL PUT_POINT(IRG, N, X, Y, Z)
      ENDPOINT = N

C-----
C Plot the outer ellipse
C-----
      I = SVL(IRG) + VSTEP
      DO ICOUNT = 1, NUMVPOINTS
        X = CCS(1,IRG) * J * COS(I)
        Y = CCS(1,IRG) * Sqrt(J * J - 1.0) * SIN(I)
        CALL PUT_POINT(IRG, N, X, Y, Z)
        WRITE(CON UNIT,*) N - 1, N, COLOR
        I = I + VSTEP
      END DO
      J = J + USTEP
    END DO

C-----
C Subdivide the surface into sections to show solidity and exit.
C-----
      DO ICOUNT = BEGINPOINT, BEGINPOINT + NUMVPOINTS, 4
        WRITE(CON UNIT,*) ICOUNT, ENDPOINT, COLOR
        ENDPOINT = ENDPOINT + 4
      END DO

      RETURN
      END

```

Siders and Bolling

```

C-----
C
C CIRCULAR_CYLINDER() - written by D. Mitchell Bolling Jr.
C                      TRI/TESSCO, Inc.
C                      August 1989
C
C Purpose: CIRCULAR_CYLINDER is called by PLOTCID or PLOTCHIEF.
C It generates the points and lines required for drawing a cylinder
C or section of a cylinder.
C
C Subroutines used: PUT_POINT()
C
C Modification Log:
C-----

      SUBROUTINE CIRCULAR_CYLINDER(IRG, N, COLOR, CON_UNIT, SUBDIV)

      PARAMETER PI = 3.1415926536
      PARAMETER MXSREG = 500      ! Maximum number of surface regions

C-----
C List of subroutine parameters:
C-----
      INTEGER IRG      ! CHIEF surface region id number (1 < IRG < NSREG)
      INTEGER N        ! coordinate number of last point plotted
      INTEGER COLOR    ! number representing color to draw this shape
      INTEGER CON_UNIT ! lun number for connections file
      INTEGER SUBDIV   ! if set, then use NSU and NSV for subdivisions

C-----
C List of global variables:
C-----
      REAL      SUL(MXSREG)      ! lower limits of U
      REAL      SUU(MXSREG)      ! upper limits of U
      REAL      SVL(MXSREG)      ! lower limits of V
      REAL      SVU(MXSREG)      ! upper limits of V
      REAL      CCS(10,MXSREG)   ! constants needed for CHIEF equations

      COMMON/SVALS/NSREG,NSEQNS(MXSREG),SUL(MXSREG),SUU(MXSREG),
      *          SVL(MXSREG),SVU(MXSREG),NSU(MXSREG),NSV(MXSREG),
      *          CCS(10,MXSREG),TRNSS(3,MXSREG),IZAX(MXSREG),
      *          IORDU(MXSREG),IORDV(MXSREG),NCCEQS

C-----
C List of local variables:
C-----
      INTEGER ICOUNT      ! counter used to draw segmenting
      INTEGER JCOUNT     ! counter used to draw segmenting
      INTEGER BEGINPOINT  ! first point of cylinder
      INTEGER ENDPOINT    ! last point of cylinder
      INTEGER NUMVPOINTS  ! number of points to use when drawing V arcs
      INTEGER NUMUPOINTS  ! number of points to use when drawing U lines

      REAL I, J           ! scratch variables
      REAL X, Y, Z        ! current calculated x,y,z values (local)
      REAL VSTEP          ! angle between each V arc point drawn
      REAL USTEP          ! distance between each U segment drawn

C-----
C Determine the number of points to plot for each arc and the step angle
C between each point. For a full  $2 * \text{PI}$  (360 degree) arc, 48 points
C will be used. The minimum number of points used is 4.
C-----
      IF (SUBDIV .EQ. 1) THEN
        NUMVPOINTS = 4 * NSV(IRG)
        NUMUPOINTS = NSU(IRG)
      ELSE
        I = 24.0 * (SVU(IRG) - SVL(IRG)) / PI
        NUMVPOINTS = 4 * INT(I / 4.0 + .99)
        NUMUPOINTS = 1
      END IF

      VSTEP = (SVU(IRG) - SVL(IRG)) / NUMVPOINTS

```

NRL Memorandum Report 6709

```

USTEP = (SUW(IRG) - SUL(IRG)) / NUMPOINTS

J = SUL(IRG)
BEGINPOINT = N + 1
DO JCOUNT = 0, NUMPOINTS
C-----
C  Calculate the first point of the circle on the top of cylinder
C-----
      X = CCS(1,IRG) * COS(SVL(IRG))
      Y = CCS(1,IRG) * SIN(SVL(IRG))
      Z = J
      CALL PUT_POINT(IRG, N, X, Y, Z)
      ENDPOINT = N
C-----
C  Draw the rest of the circle on the top of the cylinder
C-----
      I = SVL(IRG) + VSTEP
      DO ICOUNT = 1, NUMVPOINTS
        X = CCS(1,IRG) * COS(I)
        Y = CCS(1,IRG) * SIN(I)
        CALL PUT_POINT(IRG, N, X, Y, Z)
        WRITE(CON_UNIT,*) N - 1, N, COLOR
        I = I + VSTEP
      END DO
      J = J + USTEP
END DO

C-----
C  Draw vertical lines to simulate a 3D cylinder, then exit.
C-----
      DO ICOUNT = BEGINPOINT, BEGINPOINT + NUMVPOINTS, 4
        WRITE(CON_UNIT,*) ICOUNT, ENDPOINT, COLOR
        ENDPOINT = ENDPOINT + 4
      END DO

RETURN
END

```

Siders and Bolling

```

C-----
C
C ELLIPTICAL_CYLINDER() - written by D. Mitchell Bolling Jr.
C                               TRI/TESSCO, Inc.
C                               August 1989
C
C Purpose: ELLIPTICAL_CYLINDER is called by PLOTCHIEF.
C It generates the points and lines required for drawing a cylinder
C or section of a cylinder where circumference is elliptical in shape.
C
C Subroutines used: PUT_POINT()
C
C Modification Log:
C-----

      SUBROUTINE ELLIPTICAL_CYLINDER(IRG, N, COLOR, CON_UNIT, SUBDIV)

      PARAMETER PI = 3.1415926536
      PARAMETER MXSREG = 500      ! Maximum number of surface regions
C-----
C List of subroutine parameters:
C-----
      INTEGER IRG      ! CHIEF surface region id number (1 < IRG < NSREG)
      INTEGER N        ! coordinate number of last point plotted
      INTEGER COLOR    ! number representing color to draw cylinder
      INTEGER CON_UNIT ! lun number for connections file
      INTEGER SUBDIV   ! if set, then use NSU and NSV for subdivisions
C-----
C List of global variables:
C-----
      REAL      SUL(MXSREG)      ! lower limits of U
      REAL      SUU(MXSREG)      ! upper limits of U
      REAL      SVL(MXSREG)      ! lower limits of V
      REAL      SVU(MXSREG)      ! upper limits of V
      REAL      CCS(10,MXSREG)   ! constants needed for CHIEF equations

      COMMON/SVALS/NSREG, NSEQNS(MXSREG), SUL(MXSREG), SUU(MXSREG),
      *          SVL(MXSREG), SVU(MXSREG), NSU(MXSREG), NSV(MXSREG),
      *          CCS(10,MXSREG), TRNSS(3,MXSREG), IZAX(MXSREG),
      *          IORDU(MXSREG), IORDV(MXSREG), NCCEQS
C-----
C List of local variables:
C-----
      INTEGER ICOUNT      ! counter used to draw segmenting
      INTEGER JCOUNT     ! counter used to draw segmenting
      INTEGER BEGINPOINT  ! first point of cylinder
      INTEGER ENDPOINT    ! last point of cylinder
      INTEGER NUMVPOINTS  ! number of points to use when drawing V arc

      REAL I, J          ! scratch variables
      REAL X, Y, Z       ! current calculated x, y, z values (local)
      REAL VSTEP         ! angle between each V arc point drawn
      REAL USTEP         ! distance between each U segment drawn
C-----
C Determine the number of points to plot for each arc and the step angle
C between each point. For a full 2 * PI (360 degree) arc, 48 points
C will be used. The minimum number of points used is 4.
C-----
      IF (SUBDIV .EQ. 1) THEN
        NUMVPOINTS = 4 * NSV(IRG)
        NUMUPOINTS = NSU(IRG)
      ELSE
        I = 24.0 * (SVU(IRG) - SVL(IRG)) / PI
        NUMVPOINTS = 4 * INT(I / 4 + .99)
        NUMUPOINTS = 1
      END IF

      VSTEP = (SVU(IRG) - SVL(IRG)) / NUMVPOINTS

```

NRL Memorandum Report 6709

```

USTEP = (SUU(IRG) - SUL(IRG)) / NUMPOINTS

J = SUL(IRG)
BEGINPOINT = N + 1
DO JCOUNT = 0, NUMPOINTS
C-----
C Calculate the first point of an ellipse of cylinder.
C-----
      X = CCS(5,IRG) * COS(SVL(IRG))
      Y = CCS(6,IRG) * SIN(SVL(IRG))
      Z = J
      CALL PUT_POINT(IRG, N, X, Y, Z)
      ENDPOINT = N

C-----
C Draw the rest of the ellipse of cylinder.
C-----
      I = SVL(IRG) + VSTEP
      DO ICOUNT = 1, NUMVPOINTS
        X = CCS(5,IRG) * COS(I)
        Y = CCS(6,IRG) * SIN(I)
        CALL PUT_POINT(IRG, N, X, Y, Z)
        WRITE(CON_UNIT,*) N - 1, N, COLOR
        I = I + VSTEP
      END DO
      J = J + USTEP
      END DO

C-----
C Draw extra vertical lines to show solidity, then exit.
C-----
      DO ICOUNT = BEGINPOINT, BEGINPOINT + NUMVPOINTS, 4
        WRITE(CON_UNIT,*) I, ENDPOINT, COLOR
        ENDPOINT = ENDPOINT + 4
      END DO

      RETURN
      END

```

Siders and Bolling

```

C-----
C
C   SPHEROID()      - written by D. Mitchell Bolling Jr.
C                   TRI/TESSCO, Inc.
C                   August 1989
C
C Purpose: SPHEROID is called by PLOTCHIEF or PLOTCHIEF in order to
C generate the points and lines for drawing a SPHEROID or section
C of a SPHEROID.
C
C Subroutines used: PUT_POINT()
C
C Modification Log:
C   June 1990 - PLOTCHIEF would not fully close circles that were
C   slightly less than PI for a SPHEROID example. Determined that
C   it was due to floating point roundoff error within the incremental
C   loops. Changed indexes from REAL to INTEGER.
C-----

      SUBROUTINE SPHEROID(IRG, N, COLOR, CON_UNIT, SUBDIV)

C-----
C List of subroutine parameters:
C-----
      INTEGER IRG      ! CHIEF surface region id number (1 < IRG < NSREG)
      INTEGER N        ! coordinate number of next point to be drawn
      INTEGER COLOR    ! number associated with color to draw spheroid
      INTEGER CON_UNIT ! lun number for connections file
      INTEGER SUBDIV   ! if set, then use NSU and NSV for subdivisions

      PARAMETER PI = 3.1415926536
      PARAMETER MXSREG = 500 ! Maximum number of surface regions

C-----
C List of global variables used:
C-----
      REAL      SUL(MXSREG) ! lower limits of U
      REAL      SUU(MXSREG) ! upper limits of U
      REAL      SVL(MXSREG) ! lower limits of V
      REAL      SVU(MXSREG) ! upper limits of V
      REAL      CCS(10,MXSREG) ! constants needed for CHIEF equations

      COMMON/SVALS/NSREG, NSEQNS(MXSREG), SUL(MXSREG), SUU(MXSREG),
      *          SVL(MXSREG), SVU(MXSREG), NSU(MXSREG), NSV(MXSREG),
      *          CCS(10,MXSREG), TRNSS(3,MXSREG), IZAX(MXSREG),
      *          IORDU(MXSREG), IORDV(MXSREG), NCCEQS

C-----
C List of local variables:
C-----
      INTEGER NUMPOINTS ! number of points used when drawing U arcs
      INTEGER NUMVPOINTS ! number of points used when drawing V arcs
      INTEGER UCOUNT    ! counter for U angle
      INTEGER VCOUNT   ! counter for V angle

      REAL USTEP      ! angle between each point around arc in U direction
      REAL VSTEP      ! angle between each point around arc in V direction
      REAL X, Y, Z    ! current calculated x, y and z coordinates
      REAL U          ! current angle corresponding to current UCOUNT
      REAL V          ! current angle corresponding to current VCOUNT
      REAL T          ! temporary variable

C-----
C Determine the number of points to plot for each arc and the step angle
C between each point. For a full  $2 * \pi$  arc, 48 points will be used.
C An arc of less than  $\pi / 6$  will use 4 points.
C-----
      IF (SUBDIV.EQ. 1) THEN
        NUMPOINTS = 4 * NSU(IRG)
        NUMVPOINTS = 4 * NSV(IRG)

```

NRL Memorandum Report 6709

```

ELSE
  T = 24.0 * (SUU(IRG) - SUL(IRG)) / PI
  NUMPOINTS = 4 * INT(T / 4.0 + .99)
  T = 24.0 * (SVU(IRG) - SVL(IRG)) / PI
  NUMVPOINTS = 4 * INT(T / 4.0 + .99)
END IF
USTEP = (SUU(IRG) - SUL(IRG)) / NUMPOINTS
VSTEP = (SVU(IRG) - SVL(IRG)) / NUMVPOINTS

C-----
C Draw circles in U direction roughly every PI/8 radians (30 degrees).
C If SUBDIV is set, then draw NSU(IRG) circles in the U direction.
C-----
  U = SUL(IRG)
  DO UCOUNT = 0, NUMPOINTS, 4
C-----
C Plot the first point.
C-----
    X = CCS(1,IRG) * SIN(U) * COS(SVL(IRG))
    Y = CCS(1,IRG) * SIN(U) * SIN(SVL(IRG))
    Z = CCS(1,IRG) * COS(U)
    CALL PUT_POINT(IRG, N, X, Y, Z)
    V = SVL(IRG) + VSTEP
    DO VCOUNT = 1, NUMVPOINTS
C-----
C Connect a line to the next point.
C-----
      X = CCS(1,IRG) * SIN(U) * COS(V)
      Y = CCS(1,IRG) * SIN(U) * SIN(V)
      CALL PUT_POINT(IRG, N, X, Y, Z)
      WRITE(CON_UNIT,*) N - 1, N, COLOR
C-----
C Draw another point on the circle in the U direction.
C-----
      V = V + VSTEP
    END DO
  END DO

C-----
C Move down and start drawing the next circle.
C-----
  U = U + USTEP * 4
END DO

C-----
C Draw circles in V direction roughly every PI/8 radians (30 degrees)
C If SUBDIV is set, then draw NSV(IRG) circles in the U direction.
C-----
  V = SVL(IRG)
  DO VCOUNT = 0, NUMVPOINTS, 4
C-----
C Plot first point of circle in V direction.
C-----
    X = CCS(1,IRG) * SIN(SUL(IRG)) * COS(V)
    Y = CCS(1,IRG) * SIN(SUL(IRG)) * SIN(V)
    Z = CCS(1,IRG) * COS(SUL(IRG))
    CALL PUT_POINT(IRG, N, X, Y, Z)
    U = SUL(IRG) + USTEP
    DO UCOUNT = 1, NUMPOINTS
C-----
C Connect a line to the next point.
C-----
      X = CCS(1,IRG) * SIN(U) * COS(V)
      Y = CCS(1,IRG) * SIN(U) * SIN(V)
      Z = CCS(1,IRG) * COS(U)
      CALL PUT_POINT(IRG, N, X, Y, Z)
      WRITE(CON_UNIT,*) N - 1, N, COLOR
C-----
C Move down the circle in V direction
C-----

```

Siders and Bolling

```
      U = U + USTEP  
    END DO
```

```
C-----  
C Move across and start drawing the next circle.  
C-----
```

```
      V = V + VSTEP * 4  
    END DO
```

```
  RETURN  
END
```

NRL Memorandum Report 6709

```

C-----
C
C  PROLATE_OBLATE_SPHEROID() - written by D. Mitchell Bolling Jr.
C                             TRI/TESSCO, Inc.
C                             August 1989
C
C  Purpose:  PROLATE_OBLATE_SPHEROID is called by PLOTCID and PLOTCHIEF in
C  order to generate the points and lines for drawing a prolate/oblate
C  spheroid or section of a prolate/oblate spheroid.
C
C  Subroutines used:  PUT_POINT()
C
C  Modification Log:
C  June 1990 - PLOTCID would not fully close circles that were
C  slightly less than PI. Determined that it was due to floating
C  point roundoff error within the incremental loops. Changed them
C  from REAL to INTEGER
C-----

```

```

SUBROUTINE PROLATE_OBLATE_SPHEROID(IRG, N, COLOR,
*                                CON_UNIT, SUBDIV)

```

```

INTEGER IRG      ! CHIEF surface region id number (1 < IRG < NSREG)
INTEGER N        ! coordinate number of next point to be drawn
INTEGER COLOR    ! # associated with color to draw prolate/oblate spheroid
INTEGER CON_UNIT ! lun number for connections file
INTEGER SUBDIV   ! if set, then use NSU and NSV for subdivisions

```

```

PARAMETER PI = 3.1415926536
PARAMETER MXSREG = 500      ! Maximum number of surface regions

```

```

C-----
C  List of global variables used:
C-----

```

```

C      REAL      SUL(MXSREG)      ! lower limits of U
C      REAL      SUU(MXSREG)      ! upper limits of U
C      REAL      SVL(MXSREG)      ! lower limits of V
C      REAL      SVU(MXSREG)      ! upper limits of V
C      REAL      CCS(10,MXSREG)   ! constants needed for CHIEF equations
C      REAL      NSU(MXSREG)      ! number of subdivisions in U direction
C      REAL      NSV(MXSREG)      ! number of subdivisions in V direction

```

```

COMMON/SVALS/NSREG,NSEQNS(MXSREG),SUL(MXSREG),SUU(MXSREG),
*          SVL(MXSREG),SVU(MXSREG),NSU(MXSREG),NSV(MXSREG),
*          CCS(10,MXSREG),TRNSS(3,MXSREG),IZAX(MXSREG),
*          IORDU(MXSREG),IORDV(MXSREG),NCCEQS

```

```

C-----
C  List of local variables:
C-----

```

```

INTEGER NUMPOINTS ! number of points used when drawing U arcs
INTEGER NUMVPOINTS ! number of points used when drawing V arcs
INTEGER UCOUNT     ! counter for U angle
INTEGER VCOUNT     ! counter for V angle

REAL USTEP         ! angle between each point around arc in U direction
REAL VSTEP         ! angle between each point around arc in V direction
REAL X, Y, Z       ! current calculated x, y, and z coordinates (local)
REAL U             ! current angle corresponding to current UCOUNT
REAL V             ! current angle corresponding to current VCOUNT
REAL T             ! temporary variable

```

```

C-----
C  If SUBDIV is set, then number of points becomes 4 X number of
C  subdivisions, otherwise use the default subdivisions and determine
C  the number of points to plot for each arc and the step angle
C  between each point. For a full 2 * PI arc, 48 points will be used
C  An arc of less than PI / 6 will use 4 points.
C-----

```

```

IF (SUBDIV EQ. 1) THEN
  NUMPOINTS = 4 * NSU(IRG)
  NUMVPOINTS = 4 * NSV(IRG)

```

Siders and Bolling

```

ELSE
  T = 24.0 * (SUU(IRG) - SUL(IRG)) / PI
  NUMPOINTS = 4 * INT(T / 4.0 + .99)
  T = 24.0 * (SVU(IRG) - SVL(IRG)) / PI
  NUMVPOINTS = 4 * INT(T / 4.0 + .99)
END IF

USTEP = (SUU(IRG) - SUL(IRG)) / NUMPOINTS
VSTEP = (SVU(IRG) - SVL(IRG)) / NUMVPOINTS

C-----
C Draw circles in U direction roughly every PI / 6 radians (30 degrees).
C If SUBDIV is set, then draw NSU(IRG) circles in the U direction
C-----
  U = SUL(IRG)
  DO UCOUNT = 0, NUMPOINTS, 4

C-----
C Plot the first point.
C-----
    X = CCS(5,IRG) * SIN(U) * COS(SVL(IRG))
    Y = CCS(5,IRG) * SIN(U) * SIN(SVL(IRG))
    Z = CCS(3,IRG) * COS(U)
    CALL PUT_POINT(IRG, N, X, Y, Z)
    V = SVL(IRG) + VSTEP
    DO VCOUNT = 1, NUMVPOINTS

C-----
C Connect a line to the next point.
C-----
      X = CCS(5,IRG) * SIN(U) * COS(V)
      Y = CCS(5,IRG) * SIN(U) * SIN(V)
      CALL PUT_POINT(IRG, N, X, Y, Z)
      WRITE(CON_UNIT,*) N - 1, N, COLOR

C-----
C Draw another point on the circle in the U direction
C-----
      V = V + VSTEP
    END DO

C-----
C Move down and start drawing the next circle.
C-----
    U = U + USTEP * 4
  END DO

C-----
C Draw circles in V direction roughly every PI / 6 radians (30 degrees).
C-----
  V = SVL(IRG)
  DO VCOUNT = 0, NUMVPOINTS, 4

C-----
C Plot first point of circle in V direction
C-----
    X = CCS(5,IRG) * SIN(SUL(IRG)) * COS(V)
    Y = CCS(5,IRG) * SIN(SUL(IRG)) * SIN(V)
    Z = CCS(3,IRG) * COS(SUL(IRG))
    CALL PUT_POINT(IRG, N, X, Y, Z)
    U = SUL(IRG) + USTEP
    DO UCOUNT = 1, NUMPOINTS

C-----
C connect a line to the next point.
C-----
      X = CCS(5,IRG) * SIN(U) * COS(V)
      Y = CCS(5,IRG) * SIN(U) * SIN(V)
      Z = CCS(3,IRG) * COS(U)
      CALL PUT_POINT(IRG, N, X, Y, Z)
      WRITE(CON_UNIT,*) N - 1, N, COLOR

C-----

```

NRL Memorandum Report 6709

C Move down the circle in V direction.

C-----
 U = U + USTEP
 END DO

C-----
C Move across and start drawing the next circle.
C-----

V = V + VSTEP * 4
 END DO

RETURN
 END

Siders and Bolling

```

C-----
C
C TOROID() - written by D. Mitchell Bolling Jr.
C           TRI/TESSCO, Inc.
C           August 1989
C
C Purpose: TOROID is called by PLOTCID or PLOTCHIEF in order to
C generate the points and lines for drawing a toroid or section
C of a toroid.
C
C Subroutines used: PUT_POINT()
C
C Modification Log:
C   June 1990 - PLOTCID would not fully close circles that were
C   slightly less than PI for a spherical example. Determined that
C   it was due to floating point roundoff error within the incremental
C   loops. Changed them from REAL to INTEGER
C-----

```

SUBROUTINE TOROID(IRG, N, COLOR, CON_UNIT, SUBDIV)

```

C-----
C List of subroutine parameters:
C-----
C   INTEGER IRG      ! CHIEF surface region id number (1 < IRG < NSREG)
C   INTEGER N        ! coordinate number of next point to be drawn
C   INTEGER COLOR    ! number associated with color to draw TOROID
C   INTEGER CON_UNIT ! lun number for connections file
C   INTEGER SUBDIV   ! if set, then use NSU and NSV for subdivisions

```

```

C   PARAMETER PI = 3.1415926536
C   PARAMETER MXSREG = 500 ! Maximum number of surface regions

```

```

C-----
C List of global variables:
C-----
C   REAL    SUL(MXSREG) ! lower limits of U
C   REAL    SUU(MXSREG) ! upper limits of U
C   REAL    SVL(MXSREG) ! lower limits of V
C   REAL    SVU(MXSREG) ! upper limits of V
C   REAL    CCS(10,MXSREG) ! constants needed for CHIEF equations

```

```

C   COMMON/SVALS/NSREG,NSEQNS(MXSREG),SUL(MXSREG),SUU(MXSREG),
C   +          SVL(MXSREG),SVU(MXSREG),NSU(MXSREG),NSV(MXSREG),
C   +          CCS(10,MXSREG),TRNSS(3,MXSREG),IZAX(MXSREG),
C   +          IORDU(MXSREG),IORDV(MXSREG),NCCEQS

```

```

C-----
C List of local variables:
C-----
C   INTEGER NUMUPOINTS ! number of points used when drawing U arcs
C   INTEGER NUMVPOINTS ! number of points used when drawing V arcs
C   INTEGER UCOUNT    ! counter for U angle
C   INTEGER VCOUNT    ! counter for V angle
C
C   REAL USTEP ! angle between each point around arc in U direction
C   REAL VSTEP ! angle between each point around arc in V direction
C   REAL X, Y, Z ! current calculated x, y and z coordinates (local)
C   REAL U      ! current angle corresponding to current UCOUNT
C   REAL V      ! current angle corresponding to current VCOUNT
C   REAL T      ! temporary variable

```

```

C-----
C If SUBDIV is set, then the number of points becomes 4 * number of
C subdivisions, otherwise use the default subdivisions and determine
C the number of points to plot for each arc and the step angle
C between each point. For a full 2 * PI arc, 48 points will be used.
C An arc of less than PI / 8 will use 4 points.
C-----

```

```

C   IF (SUBDIV .EQ. 1) THEN

```

NRL Memorandum Report 6709

```

NUMPOINTS = 4 * NSU(IRG)
NUMVPOINTS = 4 * NSV(IRG)
ELSE
  T = 24.0 * (SUU(IRG) - SUL(IRG)) / PI
  NUMPOINTS = 4 * INT(T / 4.0 + .99)
  T = 24.0 * (SVU(IRG) - SVL(IRG)) / PI
  NUMVPOINTS = 4 * INT(T / 4.0 + .99)
END IF

USTEP = (SUU(IRG) - SUL(IRG)) / NUMPOINTS
VSTEP = (SVU(IRG) - SVL(IRG)) / NUMVPOINTS

C-----
C Draw the large circles in U direction roughly every PI/8 radians
C (30 degrees).
C-----
  U = SUL(IRG)
  DO UCOUNT = 0, NUMPOINTS, 4

C-----
C Plot the first point.
C-----
    X = CCS(3,IRG) * COS(SVL(IRG)) /
      * (CCS(1,IRG) - COS(U))
    Y = CCS(3,IRG) * SIN(SVL(IRG)) /
      * (CCS(1,IRG) - COS(U))
    Z = CCS(3,IRG) * CCS(4,IRG) * SIN(U) /
      * (CCS(1,IRG) - COS(U))
    CALL PUT_POINT(IRG, N, X, Y, Z)
    V = SVL(IRG) + VSTEP
    DO VCOUNT = 1, NUMVPOINTS

C-----
C Connect a line to the next point.
C-----
      X = CCS(3,IRG) * COS(V) /
        * (CCS(1,IRG) - COS(U))
      Y = CCS(3,IRG) * SIN(V) /
        * (CCS(1,IRG) - COS(U))
      CALL PUT_POINT(IRG, N, X, Y, Z)
      WRITE(CON_UNIT,*) N - 1, N, COLOR

C-----
C Draw another point on the circle in the U direction.
C-----
      V = V + VSTEP
    END DO

C-----
C Move down and start drawing the next circle.
C-----
    U = U + USTEP * 4
  END DO

C-----
C Draw the small circles in V direction roughly every PI / 8 radians
C (30 degrees).
C-----
  V = SVL(IRG)
  DO VCOUNT = 0, NUMVPOINTS, 4

C-----
C Plot first point of circle in V direction.
C-----
    X = CCS(3,IRG) * COS(V) /
      * (CCS(1,IRG) - COS(SUL(IRG)))
    Y = CCS(3,IRG) * SIN(V) /
      * (CCS(1,IRG) - COS(SUL(IRG)))
    Z = CCS(3,IRG) * CCS(4,IRG) * SIN(SUL(IRG)) /
      * (CCS(1,IRG) - COS(SUL(IRG)))
    CALL PUT_POINT(IRG, N, X, Y, Z)
    U = SUL(IRG) + USTEP
  END DO

```

Siders and Bolling

```

DO UCOUNT = 1, NUMPOINTS
C-----
C Connect a line to the next point.
C-----
      X = CCS(3,IRG) * COS(V) /
        (CCS(1,IRG) - COS(U))
      +
      Y = CCS(3,IRG) * SIN(V) /
        (CCS(1,IRG) - COS(U))
      +
      Z = CCS(3,IRG) * CCS(4,IRG) * SIN(U) /
        (CCS(1,IRG) - COS(U))
      +
      CALL PUT_POINT(IRG, N, X, Y, Z)
      WRITE(CON_UNIT,*) N - 1, N, COLOR

C-----
C Draw another point on the circle in the U direction.
C-----
      U = U + USTEP
      END DO

C-----
C Move across and start drawing the next circle.
C-----
      V = V + VSTEP * 4
      END DO

      RETURN
      END

```

NRL Memorandum Report 6709

```

C-----
C
C QUADRILATERAL() - written by D. Mitchell Bolling Jr.
C                      TRI/TESSCO, Inc.
C                      August 1989
C
C Purpose: QUADRILATERAL is called by PLOTCID or PLOTCHIEF. It
C generates the points and lines required for drawing a quadrilateral.
C
C Subroutines used: PUT_POINT()
C
C Modification Log:
C-----
C
C      SUBROUTINE QUADRILATERAL(IRG, N, COLORN, CON_UNIT, SUBDIV)
C
C      PARAMETER MXSREG = 500      ! Maximum number of surface regions
C      PARAMETER MAXCOR = 1000     ! Maximum number of finite element nodes
C-----
C List of subroutine parameters:
C-----
C      INTEGER IRG      ! surface region id number (1 < IRG < NSREG)
C      INTEGER N        ! current coordinate number
C      INTEGER COLORN    ! number representing the color to draw plane
C      INTEGER CON_UNIT  ! lun number for connections file
C      INTEGER SUBDIV    ! not used in this routine
C-----
C List of global variables:
C-----
C      REAL          CCS(10,MXSREG) ! constants needed for CHIEF equations
C
C      COMMON/VSVALS/NSREG,NSEQNS(MXSREG),SUL(MXSREG),SUU(MXSREG),
C      *          SVL(MXSREG),SVU(MXSREG),NSU(MXSREG),NSV(MXSREG),
C      *          CCS(10,MXSREG),TRNSS(3,MXSREG),IZAX(MXSREG),
C      *          IORDU(MXSREG),IORDV(MXSREG),NCCEQS
C
C      COMMON/COORD/COORDS(MAXCOR,3)
C-----
C List of local variables:
C-----
C      REAL X, Y, Z ! Work variables
C
C      INTEGER COLOR ! holds color and line type (negative for dotted)
C-----
C If IZAX is negative, then make color negative to show inward normal.
C The graphics driver should draw a dotted line when colors are negative.
C-----
C      COLOR = SIGN(COLORN,IZAX(IRG))
C-----
C Starting at node 1, draw lines to 5, 2, 6, 3, 7, 4, 8 and back to 1.
C
C      1 <---- 8 <---- 4
C      |          |
C      V          V
C      5          7
C      |          |
C      V          V
C      2 ----> 6 ----> 3
C-----
C
C      X = COORDS(CCS(1, IRG), 1)
C      Y = COORDS(CCS(1, IRG), 2)
C      Z = COORDS(CCS(1, IRG), 3)
C      CALL PUT_POINT(IRG, N, X, Y, Z)
C
C      IF (CCS(5,IRG) NE. 0) THEN
C      X = COORDS(CCS(5, IRG), 1)
C      Y = COORDS(CCS(5, IRG), 2)

```

Siders and Bolling

```

      Z = COORDS(CCS(5, IRG), 3)
      CALL PUT_POINT(IRG, N, X, Y, Z)
      WRITE(CON_UNIT,*) N - 1, N, COLOR
END IF

      X = COORDS(CCS(2, IRG), 1)
      Y = COORDS(CCS(2, IRG), 2)
      Z = COORDS(CCS(2, IRG), 3)
      CALL PUT_POINT(IRG, N, X, Y, Z)
      WRITE(CON_UNIT,*) N - 1, N, COLOR

      IF (CCS(5,IRG) .NE. 0) THEN
        X = COORDS(CCS(8, IRG), 1)
        Y = COORDS(CCS(8, IRG), 2)
        Z = COORDS(CCS(8, IRG), 3)
        CALL PUT_POINT(IRG, N, X, Y, Z)
        WRITE(CON_UNIT,*) N - 1, N, COLOR
      END IF

      X = COORDS(CCS(3, IRG), 1)
      Y = COORDS(CCS(3, IRG), 2)
      Z = COORDS(CCS(3, IRG), 3)
      CALL PUT_POINT(IRG, N, X, Y, Z)
      WRITE(CON_UNIT,*) N - 1, N, COLOR

      IF (CCS(5,IRG) .NE. 0) THEN
        X = COORDS(CCS(7, IRG), 1)
        Y = COORDS(CCS(7, IRG), 2)
        Z = COORDS(CCS(7, IRG), 3)
        CALL PUT_POINT(IRG, N, X, Y, Z)
        WRITE(CON_UNIT,*) N - 1, N, COLOR
      END IF

      X = COORDS(CCS(4, IRG), 1)
      Y = COORDS(CCS(4, IRG), 2)
      Z = COORDS(CCS(4, IRG), 3)
      CALL PUT_POINT(IRG, N, X, Y, Z)
      WRITE(CON_UNIT,*) N - 1, N, COLOR

      IF (CCS(5,IRG) .NE. 0) THEN
        X = COORDS(CCS(8, IRG), 1)
        Y = COORDS(CCS(8, IRG), 2)
        Z = COORDS(CCS(8, IRG), 3)
        CALL PUT_POINT(IRG, N, X, Y, Z)
        WRITE(CON_UNIT,*) N - 1, N, COLOR
      END IF

      X = COORDS(CCS(1, IRG), 1)
      Y = COORDS(CCS(1, IRG), 2)
      Z = COORDS(CCS(1, IRG), 3)
      CALL PUT_POINT(IRG, N, X, Y, Z)
      WRITE(CON_UNIT,*) N - 1, N, COLOR

      RETURN
      END

```

NRL Memorandum Report 6709

```

C-----
C
C  AXISYMMETRIC() - written by D. Mitchell Bolling Jr.
C                      TRI/TESSCO, Inc.
C                      August 1989
C
C  Purpose:  AXISYMMETRIC is called by PLOTCID or PLOTCHIEF.  It
C  generates the points and lines required for drawing an axisymmetric.
C
C  Subroutines used:  PUT_POINT()
C
C  Modification Log:
C-----

      SUBROUTINE AXISYMMETRIC(IRG, N, COLOR, CON_UNIT, SUBDIV)

      PARAMETER MXSREG = 500      ! Maximum number of surface regions
      PARAMETER MAXCOR = 1000     ! Maximum number of finite element nodes
      PARAMETER PI = 3.1415926536

C-----
C  List of subroutine parameters:
C-----
      INTEGER IRG      ! surface region id number (1 < IRG < NSREG)
      INTEGER N        ! current coordinate number
      INTEGER COLOR     ! number representing the color to draw plane
      INTEGER CON_UNIT  ! lun number for connections file
      INTEGER SUBDIV    ! not used in this routine

C-----
C  List of global variables:
C-----
      REAL      CCS(10,MXSREG)  ! constants needed for CHIEF equations
      REAL      COORDS(MAXCOR,3) ! holds finite element coordinates

      COMMON/SVALS/NSREG,NSEQNS(MXSREG),SUL(MXSREG),SUU(MXSREG),
      +          SVL(MXSREG),SVU(MXSREG),NSU(MXSREG),NSV(MXSREG),
      +          CCS(10,MXSREG),TRNSS(3,MXSREG),IZAX(MXSREG),
      +          IORDU(MXSREG),IORDV(MXSREG),NCCEQS

      COMMON/CORD/COORDS(MAXCOR,3)

C-----
C  List of local variables:
C-----
      INTEGER J        ! scratch counter
      INTEGER NUMPOINTS ! number of points to use when drawing V arc

      REAL X, Y, Z, R  ! work variables
      REAL I          ! angle counter
      REAL STEP       ! angle between each V arc point drawn

C-----
C  Determine the number of points to plot for each arc and the step angle
C  between each point.  For a full 2 * PI (360 degree) arc, 48 points
C  will be used.  The minimum number of points used is 4.
C-----
      I = 24.0 * (SVU(IRG) - SVL(IRG)) / PI
      NUMPOINTS = 4 * INT(I / 4.0 + .99)
      STEP = (SVU(IRG) - SVL(IRG)) / NUMPOINTS

C-----
C  Draw first points.
C-----
      R = COORDS(CCS(1, IRG), 1)
      Z = COORDS(CCS(1, IRG), 2)
      X = COS(SVL(IRG)) * R
      Y = SIN(SVL(IRG)) * R
      CALL PUT_POINT(IRG, N, X, Y, Z)

      R = COORDS(CCS(2, IRG), 1)
      Z = COORDS(CCS(2, IRG), 2)
      X = COS(SVL(IRG)) * R

```

Siders and Bolling

```

Y = SIN(SVL(IRG)) * R
CALL PUT_POINT(IRG, N, X, Y, Z)

```

```

C-----
C If the third node is zero, then CCS(1,IRG) and CCS(2,IRG) are end
C nodes, otherwise, CCS(2,IRG) is a midnode.
C-----
      IF (CCS(3,IRG) .NE. 0) THEN
        R = COORDS(CCS(3, IRG), 1)
        Z = COORDS(CCS(3, IRG), 2)
        X = COS(SVL(IRG)) * R
        Y = SIN(SVL(IRG)) * R
        CALL PUT_POINT(IRG, N, X, Y, Z)
      END IF

```

```

C-----
C Draw circles perpendicular to the Z axis.
C-----
      I = SVL(IRG) + STEP
      DO J = 1, NUMPOINTS
        R = COORDS(CCS(1, IRG), 1)
        Z = COORDS(CCS(1, IRG), 2)
        X = COS(I) * R
        Y = SIN(I) * R
        CALL PUT_POINT(IRG, N, X, Y, Z)

        R = COORDS(CCS(2, IRG), 1)
        Z = COORDS(CCS(2, IRG), 2)
        X = COS(I) * R
        Y = SIN(I) * R
        CALL PUT_POINT(IRG, N, X, Y, Z)
      END DO

```

```

C-----
C If the third node is zero, then CCS(1,IRG) and CCS(2,IRG) are end
C nodes, otherwise, CCS(2,IRG) is a midnode.
C-----
      IF (CCS(3,IRG) .NE. 0) THEN
        R = COORDS(CCS(3, IRG), 1)
        Z = COORDS(CCS(3, IRG), 2)
        X = COS(I) * R
        Y = SIN(I) * R
        CALL PUT_POINT(IRG, N, X, Y, Z)
      END IF

```

```

C-----
C Connect the 3 points with the previous 3 points.
C-----
      WRITE(CON_UNIT,*) N - 5, N - 2, COLOR
      WRITE(CON_UNIT,*) N - 4, N - 1, COLOR
      WRITE(CON_UNIT,*) N - 3, N, COLOR
    ELSE

```

```

C-----
C or connect the 2 points with the previous 2 points.
C-----
      WRITE(CON_UNIT,*) N - 3, N - 1, COLOR
      WRITE(CON_UNIT,*) N - 2, N, COLOR
    END IF
    I = I + STEP
  END DO

```

```

C-----
C Draw cross sections parallel to the Z axis to show solidity.
C-----
      I = SVL(IRG)
      DO J = 0, NUMPOINTS, 4
        R = COORDS(CCS(1, IRG), 1)
        Z = COORDS(CCS(1, IRG), 2)
        X = COS(I) * R
        Y = SIN(I) * R
        CALL PUT_POINT(IRG, N, X, Y, Z)

        R = COORDS(CCS(2, IRG), 1)

```

NRL Memorandum Report 6709

```

Z = COORDS(CCS(2, IRG), 2)
X = COS(I) * R
Y = SIN(I) * R
CALL PUT_POINT(IRG, N, X, Y, Z)
WRITE(CON_UNIT,*) N - 1, N, COLOR

```

```

C-----
C If the third node is zero, then CCS(1,IRG) and CCS(2,IRG) are end
C nodes, otherwise, CCS(2,IRG) is a midnode.
C-----

```

```

      IF (CCS(3,IRG) .NE. 0) THEN
        R = COORDS(CCS(3, IRG), 1)
        Z = COORDS(CCS(3, IRG), 2)
        X = COS(I) * R
        Y = SIN(I) * R
        CALL PUT_POINT(IRG, N, X, Y, Z)
        WRITE(CON_UNIT,*) N - 1, N, COLOR
      END IF
      I = I + STEP * 4
    END DO

    RETURN
  END

```

Siders and Bolling

```

C-----
C
C TRIANGLE() - written by D. Mitchell Bolling Jr.
C               TRI/TESSCO, Inc.
C               August 1989
C
C Purpose: TRIANGLE is called by PLOTCID or PLOTCHIEF. It
C generates the points and lines required for drawing a triangle.
C
C Subroutines used: PUT_POINT()
C
C Modification Log:
C-----

      SUBROUTINE TRIANGLE(IRG, N, COLOR, CON_UNIT, SUBDIV)

      PARAMETER MXSREG = 500      ! Maximum number of surface regions
      PARAMETER MAXCOR = 1000     ! Maximum number of finite element nodes

C-----
C List of subroutine parameters:
C-----
      INTEGER IRG      ! surface region id number (1 < IRG < NSREG)
      INTEGER N        ! current coordinate number
      INTEGER COLOR    ! number representing the color to draw plane
      INTEGER CON_UNIT ! lun number for connections file
      INTEGER SUBDIV   ! not used in this subroutine

C-----
C List of global variables:
C-----
      REAL      CCS(10,MXSREG) ! constants needed for CHIEF equations

      COMMON/SVALS/NSREG, NSEQNS(MXSREG), SUL(MXSREG), SUU(MXSREG),
      *          SVL(MXSREG), SVU(MXSREG), NSU(MXSREG), NSV(MXSREG),
      *          CCS(10,MXSREG), TRNSS(3,MXSREG), IZAX(MXSREG),
      *          IORDU(MXSREG), IORDV(MXSREG), NCCEQS

      COMMON/COORD/COORDS(MAXCOR,3)

C-----
C List of local variables:
C-----
      REAL X, Y, Z ! Work variables

C-----
C Starting at node 1, draw lines to 4, 2, 5, 3, 6 and back to 1.
C-----
      X = COORDS(CCS(1, IRG), 1)
      Y = COORDS(CCS(1, IRG), 2)
      Z = COORDS(CCS(1, IRG), 3)
      CALL PUT_POINT(IRG, N, X, Y, Z)

      IF (CCS(4,IRG) .NE. 0) THEN
        X = COORDS(CCS(4, IRG), 1)
        Y = COORDS(CCS(4, IRG), 2)
        Z = COORDS(CCS(4, IRG), 3)
        CALL PUT_POINT(IRG, N, X, Y, Z)
        WRITE(CON_UNIT,*) N - 1, N, COLOR
      END IF

      X = COORDS(CCS(2, IRG), 1)
      Y = COORDS(CCS(2, IRG), 2)
      Z = COORDS(CCS(2, IRG), 3)
      CALL PUT_POINT(IRG, N, X, Y, Z)
      WRITE(CON_UNIT,*) N - 1, N, COLOR

      IF (CCS(4,IRG) .NE. 0) THEN
        X = COORDS(CCS(5, IRG), 1)
        Y = COORDS(CCS(5, IRG), 2)
        Z = COORDS(CCS(5, IRG), 3)
        CALL PUT_POINT(IRG, N, X, Y, Z)
        WRITE(CON_UNIT,*) N - 1, N, COLOR
      
```

NRL Memorandum Report 6709

END IF

```
X = COORDS(CCS(3, IRG), 1)
Y = COORDS(CCS(3, IRG), 2)
Z = COORDS(CCS(3, IRG), 3)
CALL PUT_POINT(IRG, N, X, Y, Z)
WRITE(CON_UNIT,*) N - 1, N, COLOR
```

```
IF (CCS(4, IRG) .NE. 0) THEN
  X = COORDS(CCS(8, IRG), 1)
  Y = COORDS(CCS(8, IRG), 2)
  Z = COORDS(CCS(8, IRG), 3)
  CALL PUT_POINT(IRG, N, X, Y, Z)
  WRITE(CON_UNIT,*) N - 1, N, COLOR
END IF
```

```
X = COORDS(CCS(1, IRG), 1)
Y = COORDS(CCS(1, IRG), 2)
Z = COORDS(CCS(1, IRG), 3)
CALL PUT_POINT(IRG, N, X, Y, Z)
WRITE(CON_UNIT,*) N - 1, N, COLOR
```

```
RETURN
END
```

Siders and Bolling

```

C-----
C
C CONE() - written by D. Mitchell Bolling Jr.
C          TRI/TESSCO, Inc.
C          July 1990
C
C Purpose: CONE is called by PLOTCHIEF or PLOTCHIEF.
C It generates the points and lines required for drawing a cone
C or section of a cone.
C
C Subroutines used: PUT_POINT()
C
C Modification Log:
C-----

      SUBROUTINE CONE(IRG, N, COLOR, CON_UNIT, SUBDIV)

      PARAMETER PI = 3.1415926535
      PARAMETER MXSREG = 500 ! Maximum number of surface regions

C-----
C List of subroutine parameters:
C-----
      INTEGER IRG      ! CHIEF surface region id number (1 < IRG < NSREG)
      INTEGER N        ! coordinate number of last point plotted
      INTEGER COLOR    ! number representing color to draw this shape
      INTEGER CON_UNIT ! lun number for connections file
      INTEGER SUBDIV   ! if set, then use NSU and NSV for subdivisions

C-----
C List of global variables:
C-----
      REAL      SUL(MXSREG) ! lower limits of U
      REAL      SUU(MXSREG) ! upper limits of U
      REAL      SVL(MXSREG) ! lower limits of V
      REAL      SVU(MXSREG) ! upper limits of V
      REAL      CCS(10,MXSREG) ! constants needed for CHIEF equations

      COMMON/SVALS/NSREG, NSECRS(MXSREG), SUL(MXSREG), SUU(MXSREG),
      *          SVL(MXSREG), SVU(MXSREG), NSU(MXSREG), NSV(MXSREG),
      *          CCS(10,MXSREG), TRNSS(3,MXSREG), IZAX(MXSREG),
      *          IORDU(MXSREG), IORDV(MXSREG), NCCEQS

C-----
C List of local variables:
C-----
      INTEGER VCOUNT ! counter used to draw segmenting
      INTEGER UCOUNT ! counter used to draw segmenting
      INTEGER BEGINPOINT ! first point of circle
      INTEGER ENDPPOINT ! last point of circle
      INTEGER NUMVPOINTS ! number of points to use when drawing V lines
      INTEGER NUMUPOINTS ! number of points to use when drawing U arcs

      REAL I, U, V ! scratch variables
      REAL X, Y, Z ! current calculated x,y,z values (local)
      REAL VSTEP ! distance between each V segment drawn
      REAL USTEP ! angle between each U arc point drawn

C-----
C Determine the number of points to plot for each arc and the step angle
C between each point. For a full 2 * PI (360 degree) arc, 48 points
C will be used. The minimum number of points used is 4.
C-----
      IF (SUBDIV .EQ. 1) THEN
        NUMUPOINTS = 4 * NSU(IRG)
        NUMVPOINTS = NSV(IRG)
      ELSE
        I = 24.0 * (SUU(IRG) - SUL(IRG)) / PI
        NUMUPOINTS = 4 * INT(I / 4.0 + .99)
        NUMVPOINTS = 1
      END IF

      VSTEP = (SVU(IRG) - SVL(IRG)) / NUMVPOINTS

```

NRL Memorandum Report 6709

```

USTEP = (SUU(IRG) - SUL(IRG)) / NUMPOINTS

V = SVL(IRG)
BEGINPOINT = N + 1
DO VCOUNT = 0, NUMVPOINTS
C-----
C Calculate the first point of the circle on the top of cylinder
C-----
      X = CCS(1,IRG) / CCS(2,IRG) * (CCS(2,IRG) - V)
      * COS(SUL(IRG))
      Y = CCS(1,IRG) / CCS(2,IRG) * (CCS(2,IRG) - V)
      * SIN(SUL(IRG))
      Z = V
      CALL PUT_POINT(IRG, N, X, Y, Z)
      ENDPOINT = N
C-----
C Draw the rest of the circle on the top of the cylinder
C-----
      U = SUL(IRG) + USTEP
      DO UCOUNT = 1, NUMPOINTS
      X = CCS(1,IRG) / CCS(2,IRG) * (CCS(2,IRG) - V)
      * COS(U)
      Y = CCS(1,IRG) / CCS(2,IRG) * (CCS(2,IRG) - V)
      * SIN(U)
      CALL PUT_POINT(IRG, N, X, Y, Z)
      WRITE(CON_UNIT,*) N - 1, N, COLOR
      U = U + USTEP
      END DO
      V = V + VSTEP
      END DO

C-----
C Draw vertical lines to simulate a 3D cylinder, then exit.
C-----
      DO UCOUNT = BEGINPOINT, BEGINPOINT + NUMPOINTS, 4
      WRITE(CON_UNIT,*) UCOUNT, ENDPOINT, COLOR
      ENDPOINT = ENDPOINT + 4
      END DO

      RETURN
      END

```

Siders and Bolling

```

C-----
C
C CONNECT_ARROWS() - written by D. Mitchell Bolling Jr.
C                   TRI/TESSCO, Inc.
C                   P.O. BOX 568458
C                   Orlando, FL 32856
C                   January 1990
C
C Purpose: There are 22 points needed to draw the reference axes,
C arrowheads, and the axis labels. The connectivity of these points
C are added to the connectivity file containing the CHIEF figure.
C
C Subroutines used: NONE
C
C Modification Log:
C-----

      SUBROUTINE CONNECT_ARROWS(CON_UNIT, N)

C-----
C List of subroutine parameters:
C-----
      INTEGER CON_UNIT ! run number for connections file
      INTEGER N        ! keeps track of the next point to be created.

      WRITE(CON_UNIT,*) N + 1, N + 2, 5      ! arrow shafts for X, Y, Z axes
      WRITE(CON_UNIT,*) N + 1, N + 3, 5
      WRITE(CON_UNIT,*) N + 1, N + 4, 5

      WRITE(CON_UNIT,*) N + 5, N + 2, 5      ! X arrowhead
      WRITE(CON_UNIT,*) N + 2, N + 6, 5
      WRITE(CON_UNIT,*) N + 6, N + 5, 5

      WRITE(CON_UNIT,*) N + 7, N + 3, 5      ! Y arrowhead
      WRITE(CON_UNIT,*) N + 3, N + 8, 5
      WRITE(CON_UNIT,*) N + 8, N + 7, 5

      WRITE(CON_UNIT,*) N + 9, N + 4, 5      ! Z arrowhead
      WRITE(CON_UNIT,*) N + 4, N + 10, 5
      WRITE(CON_UNIT,*) N + 10, N + 9, 5

      WRITE(CON_UNIT,*) N + 11, N + 12, 5    ! X label
      WRITE(CON_UNIT,*) N + 13, N + 14, 5

      WRITE(CON_UNIT,*) N + 15, N + 16, 5    ! Y label
      WRITE(CON_UNIT,*) N + 18, N + 17, 5
      WRITE(CON_UNIT,*) N + 18, N + 18, 5

      WRITE(CON_UNIT,*) N + 19, N + 20, 5    ! Z label
      WRITE(CON_UNIT,*) N + 20, N + 21, 5
      WRITE(CON_UNIT,*) N + 21, N + 22, 5

      RETURN
      END

```

NRL Memorandum Report 6709

```

C
C
C  ROT3D() - Written by Mitch Bolling of
C             TRI/TESSCO, Inc.
C             August 1989
C
C
C  ROT3D() is a subroutine that inputs 3 dimensional coordinates and 3
C  angles of rotation and returns a list of coordinates that have been
C  rotated about the X, Y, and Z axes by the amount specified. The
C  rotations are done by 3 separate procedures. The first to rotate around
C  the X axis, the second to rotate around the Y axis and the third to
C  rotate around the Z axis
C
C-----

```

SUBROUTINE ROT3D(N, AX, AY, AZ)

PARAMETER MAXCOORDS = 10000

```

C-----
C  List of parameters used:
C-----
      INTEGER N          ! number of coordinates to plot

      REAL AX           ! angle of rotation about the X axis (in degrees)
      REAL AY           ! angle of rotation about the Y axis (in degrees)
      REAL AZ           ! angle of rotation about the Z axis (in degrees)

      REAL A
      REAL B
      REAL C
      REAL L
      REAL V

```

```
C-----
C List of global variables used:
C-----
      REAL      COORD3D      ! array of 3d points
      REAL      ROTATED      ! 3d points after rotations
```

COMMON / ROT INFO / COORD3D(3, MAXCOORDS), ROTATED(3, MAXCOORDS)

```

C-----
C  List of local variables used:
C
C  Temporary work matrices used to hold rotated homogeneous coordinates
C-----

```

```

REAL U1(4, MAXCOORDS)
REAL U2(4, MAXCOORDS)

REAL TX(4,4)      ! describes rotation about X axis.
REAL TZ(4,4)      ! describes rotation about Z axis.

REAL RX(4,4)      ! describes rotation about X axis until axis of
                  ! rotation is in the XZ plane.

REAL RXI(4,4)     ! describes inverse rotation about X axis until axis
                  ! of rotation is back in original position.

REAL RY(4,4)      ! describes rotation about Y axis until Z axis
                  ! corresponds to axis of rotation.

REAL RYI(4,4)     ! describes inverse rotation of Y axis

REAL TMP1(4,4)    ! temporary work matrices used to hold these rotation
REAL TMP2(4,4)    ! matrices.

REAL X            ! angle of rotation about the X axis (in radians)
REAL Y            ! angle of rotation about the Y axis (in radians)
REAL Z            ! angle of rotation about the Z axis (in radians)

INTEGER I, J      ! scratch counter variables (I for row, J for column)

```

Siders and Bolling

C-----
 C First 3 points defined in U will represent the X, Y, and Z axes,
 C respectively and are used to keep track of the orientation of these
 C axes as the figure is rotated.
 C-----

```
U1(1, 1) = 1.0
U1(2, 1) = 0.0
U1(3, 1) = 0.0
U1(4, 1) = 0.0
```

```
U1(1, 2) = 0.0
U1(2, 2) = 1.0
U1(3, 2) = 0.0
U1(4, 2) = 0.0
```

```
U1(1, 3) = 0.0
U1(2, 3) = 0.0
U1(3, 3) = 1.0
U1(4, 3) = 0.0
```

C-----
 C Convert the coordinate matrix into a homogeneous matrix.
 C-----

```
DO I = 1, N
  DO J = 1, 3
    U1(J, I + 3) = COORD3D(J, I)
  END DO
  U1(4, I) = 1.0
END DO
N = N + 3
```

C-----
 C Convert rotation angles from degrees into radians
 C-----

```
X = -AX / 57.29578
Y = -AY / 57.29578
Z = -AZ / 57.29578
```

C-----
 C Define the 4 X 4 matrix that will be used for rotating the data
 C about the X axis
 C-----

```
TX(1,1) = 1.0
TX(2,1) = 0.0
TX(3,1) = 0.0
TX(4,1) = 0.0
```

```
TX(1,2) = 0.0
TX(2,2) = COS(X)
TX(3,2) = -SIN(X)
TX(4,2) = 0.0
```

```
TX(1,3) = 0.0
TX(2,3) = SIN(X)
TX(3,3) = COS(X)
TX(4,3) = 0.0
```

```
TX(1,4) = 0.0
TX(2,4) = 0.0
TX(3,4) = 0.0
TX(4,4) = 1.0
```

C-----
 C Multiply the unrotated coordinates (U1) by the rotation matrix (TX)
 C to produce the rotated coordinates (U2) about the X axis.
 C-----

```
DO I = 1, N
  DO J = 1, 4
    U2(J, I) = 0.0
  END DO
  DO K = 1, 4
    U2(J, I) = U2(J, I) + U1(K, I) * TX(K, J)
  END DO
END DO
```

NRL Memorandum Report 6709

END DO
END DO
END DO

C-----
C This is the beginning of the rotation about the Y axis. The
C coordinates, after a rotation about the X axis, are stored in U2
C The variables B, C, and V are the lengths of vectors in the YZ plane
C that are used to rotate the Y axis so that it is aligned with the
C Z axis.
C-----

B = U2(2, 2)
C = U2(3, 2)
V = SQRT(B*B + C*C)

C-----
C Define rotation matrix used to align the Y axis with the Z axis.
C-----

RX(1,1) = 1.0
RX(2,1) = 0.0
RX(3,1) = 0.0
RX(4,1) = 0.0

RX(1,2) = 0.0
RX(2,2) = C / V
RX(3,2) = -B / V
RX(4,2) = 0.0

RX(1,3) = 0.0
RX(2,3) = B / V
RX(3,3) = C / V
RX(4,3) = 0.0

RX(1,4) = 0.0
RX(2,4) = 0.0
RX(3,4) = 0.0
RX(4,4) = 1.0

C-----
C Define the 4 X 4 matrix that will be used for rotating the data
C about the Z axis
C-----

TZ(1,1) = COS(Y)
TZ(2,1) = -SIN(Y)
TZ(3,1) = 0.0
TZ(4,1) = 0.0

TZ(1,2) = SIN(Y)
TZ(2,2) = COS(Y)
TZ(3,2) = 0.0
TZ(4,2) = 0.0

TZ(1,3) = 0.0
TZ(2,3) = 0.0
TZ(3,3) = 1.0
TZ(4,3) = 0.0

TZ(1,4) = 0.0
TZ(2,4) = 0.0
TZ(3,4) = 0.0
TZ(4,4) = 1.0

C-----
C This 4 x 4 matrix reverses the rotation about the X axis
C (Inverse of RX) This puts the Y axis back to its previous orientation
C-----

RXI(1,1) = 1.0
RXI(2,1) = 0.0
RXI(3,1) = 0.0
RXI(4,1) = 0.0

RXI(1,2) = 0.0

Siders and Bolling

```

RXI(2,2) = C / V
RXI(3,2) = B / V
RXI(4,2) = 0.0

```

```

RXI(1,3) = 0.0
RXI(2,3) = -B / V
RXI(3,3) = C / V
RXI(4,3) = 0.0

```

```

RXI(1,4) = 0.0
RXI(2,4) = 0.0
RXI(3,4) = 0.0
RXI(4,4) = 1.0

```

```

C-----
C To simplify the matrix manipulations, first multiply the rotation
C matrices (RX, TZ and RXI) together. Then multiply the coordinates (U2)
C with the combined product (TMP2) U1 = U2 * (RX * TZ * RXI)
C
C Multiply RX by TZ
C-----

```

```

DO I = 1, 4
  DO J = 1, 4
    TMP1(I,J) = 0.0
    DO K = 1, 4
      TMP1(I,J) = TMP1(I,J) + RX(I,K) * TZ(K,J)
    END DO
  END DO
END DO

```

```

C-----
C Multiply the resulting product by RXI to produce TMP2
C (RX * TZ * RXI)
C-----

```

```

DO I = 1, 4
  DO J = 1, 4
    TMP2(I,J) = 0.0
    DO K = 1, 4
      TMP2(I,J) = TMP2(I,J) + TMP1(I,K) * RXI(K,J)
    END DO
  END DO
END DO

```

```

C-----
C Multiply the coordinates (U2) that have been rotated about the X axis
C by the rotation matrix (TMP2) to produce coordinates that have been
C rotated about the Y axis (U1).
C-----

```

```

DO I = 1, N
  DO J = 1, 4
    U1(J, I) = 0.0
    DO K = 1, 4
      U1(J, I) = U1(J, I) + U2(K, I) * TMP2(K, J)
    END DO
  END DO
END DO

```

```

C-----
C This is the beginning of the rotation about the Z axis. The
C coordinate after a rotation about the Y axis is stored in U1. The
C variables B, C, and V are the lengths of vectors in the YZ plane
C that are used to rotate the Y axis so that it is aligned with the Z
C axis. The variables V, A, and L are lengths of vectors in the XZ
C plane used to rotate the Z axis
C-----

```

```

A = U1(1, 3)
B = U1(2, 3)
C = U1(3, 3)

V = SQRT(B*B + C*C)
L = SQRT(A*A + B*B + C*C)

```

NRL Memorandum Report 6709

C-----
 C Define rotation the matrices used to align the Z axis with the global
 C Z axis before rotation.
 C-----

$RX(1,1) = 1.0$
 $RX(2,1) = 0.0$
 $RX(3,1) = 0.0$
 $RX(4,1) = 0.0$

$RX(1,2) = 0.0$
 $RX(2,2) = C / V$
 $RX(3,2) = -B / V$
 $RX(4,2) = 0.0$

$RX(1,3) = 0.0$
 $RX(2,3) = B / V$
 $RX(3,3) = C / V$
 $RX(4,3) = 0.0$

$RX(1,4) = 0.0$
 $RX(2,4) = 0.0$
 $RX(3,4) = 0.0$
 $RX(4,4) = 1.0$

$RY(1,1) = V / L$
 $RY(2,1) = 0.0$
 $RY(3,1) = -A / L$
 $RY(4,1) = 0.0$

$RY(1,2) = 0.0$
 $RY(2,2) = 1.0$
 $RY(3,2) = 0.0$
 $RY(4,2) = 0.0$

$RY(1,3) = A / L$
 $RY(2,3) = 0.0$
 $RY(3,3) = V / L$
 $RY(4,3) = 0.0$

$RY(1,4) = 0.0$
 $RY(2,4) = 0.0$
 $RY(3,4) = 0.0$
 $RY(4,4) = 1.0$

C-----
 C Define the 4 X 4 matrix that will be used for rotating the data
 C about the Z axis
 C-----

$TZ(1,1) = \cos(Z)$
 $TZ(2,1) = -\sin(Z)$
 $TZ(3,1) = 0.0$
 $TZ(4,1) = 0.0$

$TZ(1,2) = \sin(Z)$
 $TZ(2,2) = \cos(Z)$
 $TZ(3,2) = 0.0$
 $TZ(4,2) = 0.0$

$TZ(1,3) = 0.0$
 $TZ(2,3) = 0.0$
 $TZ(3,3) = 1.0$
 $TZ(4,3) = 0.0$

$TZ(1,4) = 0.0$
 $TZ(2,4) = 0.0$
 $TZ(3,4) = 0.0$
 $TZ(4,4) = 1.0$

C-----
 C These 2 inverse rotation matrices are used to move the Z axis back
 C to its original position (Inverse of RX and RY)
 C-----

Siders and Bolling

```

C-----
RXI(1,1) = 1.0
RXI(2,1) = 0.0
RXI(3,1) = 0.0
RXI(4,1) = 0.0

RXI(1,2) = 0.0
RXI(2,2) = C / V
RXI(3,2) = B / V
RXI(4,2) = 0.0

RXI(1,3) = 0.0
RXI(2,3) = -B / V
RXI(3,3) = C / V
RXI(4,3) = 0.0

RXI(1,4) = 0.0
RXI(2,4) = 0.0
RXI(3,4) = 0.0
RXI(4,4) = 1.0

C-----
C Inverse of RY.
C-----
RYI(1,1) = V / L
RYI(2,1) = 0.0
RYI(3,1) = A / L
RYI(4,1) = 0.0

RYI(1,2) = 0.0
RYI(2,2) = 1.0
RYI(3,2) = 0.0
RYI(4,2) = 0.0

RYI(1,3) = -A / L
RYI(2,3) = 0.0
RYI(3,3) = V / L
RYI(4,3) = 0.0

RYI(1,4) = 0.0
RYI(2,4) = 0.0
RYI(3,4) = 0.0
RYI(4,4) = 1.0

C-----
C First multiply the rotation matrices (RX, RY, TZ, RYI, and RXI) together.
C Then multiply the coordinates (U2) with the combined product (TMP2)
C
C Multiply RX by RY
C-----
      DO I = 1, 4
        DO J = 1, 4
          TMP1(I,J) = 0.0
          DO K = 1, 4
            TMP1(I,J) = TMP1(I,J) + RX(I,K) * RY(K,J)
          END DO
        END DO
      END DO

C-----
C Multiply the resulting product by TZ to produce TMP2 (RX * RY * TZ)
C-----
      DO I = 1, 4
        DO J = 1, 4
          TMP2(I,J) = 0.0
          DO K = 1, 4
            TMP2(I,J) = TMP2(I,J) + TMP1(I,K) * TZ(K,J)
          END DO
        END DO
      END DO
C-----

```

NRL Memorandum Report 6709

C Multiply the resulting product by RYI to produce TMP1
C (RX * RY * TZ * RYI)

```
C-----
DO I = 1, 4
  DO J = 1, 4
    TMP1(I,J) = 0.0
    DO K = 1, 4
      TMP1(I,J) = TMP1(I,J) + TMP2(I,K) * RYI(K,J)
    END DO
  END DO
END DO
```

C-----
C Multiply the resulting product by RXI to produce TMP2
C (RX * RY * TZ * RYI * RXI)

```
C-----
DO I = 1, 4
  DO J = 1, 4
    TMP2(I,J) = 0.0
    DO K = 1, 4
      TMP2(I,J) = TMP2(I,J) + TMP1(I,K) * RXI(K,J)
    END DO
  END DO
END DO
```

C-----
C Multiply the coordinates (U1) rotated about the X and Y axes by
C the rotation matrix (TMP2) to produce coordinates that have been
C rotated about the Z axis (U2).

```
C-----
DO I = 1, N
  DO J = 1, 4
    U2(J, I) = 0.0
    DO K = 1, 4
      U2(J, I) = U2(J, I) + U1(K, I) * TMP2(K, J)
    END DO
  END DO
END DO
```

C-----
C Output the matrix of rotated X, Y, and Z coordinates, omitting the
C homogeneous coordinates.

```
C-----
N = N - 3
DO I = 1, N
  DO J = 1, 3
    ROTATED(J, I) = U2(J, I + 3)
  END DO
END DO

RETURN
END
```

Siders and Bolling

```

C-----
C
C  DRAW_ARROWS() - Written by Mitch Bolling of
C                  TRI/TESSCO, Inc.
C                  January 1990
C
C  DRAW_ARROWS() is a subroutine that generates the points needed for
C  drawing the arrow heads and letters (X, Y, Z) of the reference axes.
C-----

      SUBROUTINE DRAW_ARROWS(CRD_UNIT, N, AX, AY, AZ)

      PARAMETER MAXCOORDS = 10000

C-----
C  List of parameters used:
C-----
      INTEGER CRD_UNIT ! run number for coordinate file
      INTEGER N        ! number of coordinates currently being plotted

      REAL AX          ! angle of rotation about the X axis (in degrees)
      REAL AY          ! angle of rotation about the Y axis (in degrees)
      REAL AZ          ! angle of rotation about the Z axis (in degrees)

C-----
C  List of global variables used:
C-----
      REAL COORD3D      ! array of unrotated 3d points
      REAL ROTATED      ! 3d points after rotations

      COMMON / ROT_INFO / COORD3D(3, MAXCOORDS), ROTATED(3, MAXCOORDS)

C-----
C  List of local variables used:
C-----
      INTEGER I, J      ! scratch counter variables (I for row, J for column)

      REAL ARROWS(3, 4) ! 4 points that define the reference axes
      REAL MINX          ! leftmost point of rotated object
      REAL MINY          ! bottom point of rotated object
      REAL MINZ          ! deepest point of rotated object
      REAL MAXX          ! rightmost point of rotated object
      REAL MAXY          ! top point of rotated object
      REAL MAXZ          ! highest point of rotated object
      REAL MAXDIFF       ! length of side of box drawn around CHIEF figure
      REAL MAXREFSIZE    ! length of box drawn around reference axes
      REAL XCENTER       ! global x position for tip of arrowhead
      REAL YCENTER       ! global y position for tip of arrowhead
      REAL ZCENTER       ! global z position for tip of arrowhead
      REAL HEADX         ! x offset for arrowhead base or axis label
      REAL HEADY         ! y offset for arrowhead base or axis label
      REAL ARROWLEN      ! length of XY vector of an arrow

      DATA ARROWS / 0.0, 0.0, 0.0, ! center
      * 0.85, 0.0, 0.0, ! x shaft
      * 0.0, 0.85, 0.0, ! y shaft
      * 0.0, 0.0, 0.85 / ! z shaft

C-----
C  Scaling the reference axis with respect to rotated view of chief
C  figure.
C-----
      MINX = ROTATED(1, 1)
      MINY = ROTATED(2, 1)
      MINZ = ROTATED(3, 1)
      MAXX = MINX
      MAXY = MINY
      MAXZ = MINZ

      DO I = 2, N
        IF (ROTATED(1, I) .LT. MINX) MINX = ROTATED(1, I)

```

NRL Memorandum Report 6709

```

IF (ROTATED(2, I) .LT. MINY) MINY = ROTATED(2, I)
IF (ROTATED(3, I) .LT. MINZ) MINZ = ROTATED(3, I)
IF (ROTATED(1, I) .GT. MAXX) MAXX = ROTATED(1, I)
IF (ROTATED(2, I) .GT. MAXY) MAXY = ROTATED(2, I)
IF (ROTATED(3, I) .GT. MAXZ) MAXZ = ROTATED(3, I)
END DO

```

C-----
C Find the largest difference of rotated coordinates.
C-----

```

MAXDIFF = MAXX - MINX
IF ((MAXY - MINY) .GT. MAXDIFF) MAXDIFF = MAXY - MINY
IF ((MAXZ - MINZ) .GT. MAXDIFF) MAXDIFF = MAXZ - MINZ

```

C-----
C make scale of axes 1/8th of size of CHIEF figure.
C-----

```

MAXREFSIZE = MAXDIFF / 8

```

```

N = 4
DO I = 1, N
  COORD3D(1, I) = ARROWS(1, I) * MAXREFSIZE
  COORD3D(2, I) = ARROWS(2, I) * MAXREFSIZE
  COORD3D(3, I) = ARROWS(3, I) * MAXREFSIZE
END DO

```

C-----
C Rotate reference axes in the same way as the object was rotated.
C-----

```

CALL ROT3D(N, AX, AY, AZ)

```

C-----
C Now write rotated reference axes in bottom left corner. These
C points will become the tips of the reference arrows.
C-----

```

DO I = 1, N
  WRITE(CRD_UNIT, *) ROTATED(1, I) * MINX - MAXREFSIZE,
    + ROTATED(2, I) * MINY - MAXREFSIZE,
    + ROTATED(3, I) * MINZ - MAXREFSIZE
END DO

```

C-----
C Create points used to draw the base of arrowhead for each axis, one
C point on each side of the shaft. These points and the arrow tip
C will be connected in a triangle to form the arrowhead.
C-----

```

DO I = 1, 4
  ARROWLEN = SQRT(ROTATED(1, I)**2 + ROTATED(2, I)**2)

```

C-----
C First, locate tip of arrows on reference axis in global coordinates.
C-----

```

XCENTER = ROTATED(1, I) * MINX - MAXREFSIZE
YCENTER = ROTATED(2, I) * MINY - MAXREFSIZE
ZCENTER = ROTATED(3, I) * MINZ - MAXREFSIZE

```

C-----
C Decide when reference axis vector is smaller than arrowhead. If so
C then write the two arrowhead base points on top of the arrow tip
C (In this way, the arrowhead will not appear when an axis is seen
C almost head on.)
C-----

```

IF (ARROWLEN .LT. MAXREFSIZE / 3) THEN
  WRITE(CRD_UNIT, *) XCENTER, YCENTER, ZCENTER
  WRITE(CRD_UNIT, *) XCENTER, YCENTER, ZCENTER
ELSE

```

C-----
C Dividing by ARROWLEN makes it a unit vector. Multiply by MAXREFSIZE
C to determine how large arrowhead will be from tip to base. (this way,
C the arrowheads for all three axes will be of the same length.)
C-----

```

HEADX = MAXREFSIZE/4 * ROTATED(1, I) / ARROWLEN

```

Siders and Bolling

```

HEADY = MAXREFSIZE/4 * ROTATED(2,1) / ARROWLEN

C-----
C Create the two points of the base of the arrowhead.
C-----
      WRITE(CRD_UNIT, *) XCENTER - HEADX + HEADY/4,
      *                  YCENTER - HEADY - HEADX/4,
      *                  ZCENTER

      WRITE(CRD_UNIT, *) XCENTER - HEADX - HEADY/4,
      *                  YCENTER - HEADY + HEADX/4,
      *                  ZCENTER
      END IF
      END 00

C-----
C Create points used to draw the letter 'X'.
C-----
      ARROWLEN = SQRT(ROTATED(1,2)**2 + ROTATED(2,2)**2)
      IF (ARROWLEN .EQ. 0.0) ARROWLEN = 1.0

      XCENTER = ROTATED(1,2) + MINX - MAXREFSIZE
      YCENTER = ROTATED(2,2) + MINY - MAXREFSIZE
      ZCENTER = ROTATED(3,2) + MINZ - MAXREFSIZE

C-----
C Move the same distance away from the tip of arrow as the arrowhead
C base is, but move in the opposite direction.
C-----
      HEADX = MAXREFSIZE/4 * ROTATED(1,2) / ARROWLEN
      HEADY = MAXREFSIZE/4 * ROTATED(2,2) / ARROWLEN

C-----
C Write the points describing and 'X' to the coordinate file.
C-----
      WRITE(CRD_UNIT, *) XCENTER + HEADX + MAXREFSIZE/12,
      *                  YCENTER + HEADY + MAXREFSIZE/12,
      *                  ZCENTER

      WRITE(CRD_UNIT, *) XCENTER + HEADX - MAXREFSIZE/12,
      *                  YCENTER + HEADY - MAXREFSIZE/12,
      *                  ZCENTER

      WRITE(CRD_UNIT, *) XCENTER + HEADX - MAXREFSIZE/12,
      *                  YCENTER + HEADY + MAXREFSIZE/12,
      *                  ZCENTER

      WRITE(CRD_UNIT, *) XCENTER + HEADX + MAXREFSIZE/12,
      *                  YCENTER + HEADY - MAXREFSIZE/12,
      *                  ZCENTER

C-----
C Create points used to draw the letter 'Y'.
C-----
      ARROWLEN = SQRT(ROTATED(1,3)**2 + ROTATED(2,3)**2)
      IF (ARROWLEN .EQ. 0.0) ARROWLEN = 1.0

C-----
C Move the same distance away from the tip of arrow as the arrowhead
C base is, but move in the opposite direction.
C-----
      XCENTER = ROTATED(1,3) + MINX - MAXREFSIZE
      YCENTER = ROTATED(2,3) + MINY - MAXREFSIZE
      ZCENTER = ROTATED(3,3) + MINZ - MAXREFSIZE

      HEADX = MAXREFSIZE/4 * ROTATED(1,3) / ARROWLEN
      HEADY = MAXREFSIZE/4 * ROTATED(2,3) / ARROWLEN

C-----
C Write the points describing and 'Y' to the coordinate file
C-----
      WRITE(CRD_UNIT, *) XCENTER + HEADX - MAXREFSIZE/12,

```

NRL Memorandum Report 6709

```

+          YCENTER + HEADY + MAXREFSIZE/12,
+          ZCENTER

WRITE(CRD_UNIT, *) XCENTER + HEADX,
+          YCENTER + HEADY,
+          ZCENTER

WRITE(CRD_UNIT, *) XCENTER + HEADX + MAXREFSIZE/12,
+          YCENTER + HEADY + MAXREFSIZE/12,
+          ZCENTER

WRITE(CRD_UNIT, *) XCENTER + HEADX,
+          YCENTER + HEADY - MAXREFSIZE/12,
+          ZCENTER

C-----
C  Create points used to draw the letter 'Z'.
C-----
      ARROWLEN = SQRT(ROTATED(1,4)**2 + ROTATED(2,4)**2)
      IF (ARROWLEN .EQ. 0.0) ARROWLEN = 1.0

      XCENTER = ROTATED(1,4) + MINX - MAXREFSIZE
      YCENTER = ROTATED(2,4) + MINY - MAXREFSIZE
      ZCENTER = ROTATED(3,4) + MINZ - MAXREFSIZE

C-----
C  Move the same distance away from the tip of arrow as the arrowhead
C  base is, but move in the opposite direction.
C-----
      HEADX = MAXREFSIZE/4 * ROTATED(1,4) / ARROWLEN
      HEADY = MAXREFSIZE/4 * ROTATED(2,4) / ARROWLEN

C-----
C  Write the points describing and 'Z' to the coordinate file.
C-----
      WRITE(CRD_UNIT, *) XCENTER + HEADX - MAXREFSIZE/12,
+          YCENTER + HEADY + MAXREFSIZE/12,
+          ZCENTER

      WRITE(CRD_UNIT, *) XCENTER + HEADX + MAXREFSIZE/12,
+          YCENTER + HEADY + MAXREFSIZE/12,
+          ZCENTER

      WRITE(CRD_UNIT, *) XCENTER + HEADX - MAXREFSIZE/12,
+          YCENTER + HEADY - MAXREFSIZE/12,
+          ZCENTER

      WRITE(CRD_UNIT, *) XCENTER + HEADX + MAXREFSIZE/12,
+          YCENTER + HEADY - MAXREFSIZE/12,
+          ZCENTER

      RETURN
      END

```