NASA Technical Memorandum 102454

GRID2D/3D—A Computer Program for Generating Grid Systems in Complex-Shaped Two- and Three-Dimensional Spatial Domains

Part 2: User's Manual and Program Listing

R.T. Bailey
University of Florida
Gainesville, Florida

T.I-P. Shih
Carnegie Mellon University
Pittsburgh, Pennsylvania

H.L. Nguyen and R.J. Roelke
Lewis Research Center
Cleveland, Ohio

April 1990

CONTENTS

GRID2D/3D – A COMPUTER PROGRAM FOR GENERATING GRID SYSTEMS IN COMPLEX-SHAPED TWO- AND THREE-DIMENSIONAL SPATIAL DOMAINS

Part 2: User's Manual and Program Listing

R. T. Bailey
Department of Mechanical Engineering
University of Florida
Gainesville, Florida 32611

T. I-P. Shih
Department of Mechanical Engineering
Carnegie Mellon University
Pittsburgh, Pennsylvania 15213-3890

H. L. Nguyen and R. J. Roelke
National Aeronautics and Space Administration
Lewis Research Center
Cleveland, Ohio 44135

1.0 INTRODUCTION

A computer program has been written which utilizes the algebraic grid generation techniques described in Part 1 of this technical memorandum. This FORTRAN 77 program is known as GRID2D/3D and can be used to generate grid systems within both two- and three- dimensional (2-D and 3-D) spatial domains. The program was developed for use on an IBM PC, XT, or AT compatible computer but can be easily modified for use on a workstation or mainframe computer. This part of the technical memorandum (Part 2) is a user's manual for GRID2D/3D. A complete listing of the program is also provided.

1.1 Methods Used in GRID2D/3D

For a complete description of the grid generation techniques used in GRID2D/3D, the reader is encouraged to read Part 1 of this technical memorandum. Here, we mention only a few brief facts concerning the inner workings of the program. GRID2D/3D uses either the Two- or Four-Boundary Method to generate grid systems. The Six-Boundary Method is not yet a part of GRID2D/3D but may be included in future versions. Boundary curves are described parametrically by using tension spline interpolation, while boundary surfaces are described by using three-dimensional bidirectional Hermite interpolation.

1.2 Programs in GRID2D/3D

GRID2D/3D is actually made up of two programs – GRID2D and GRID3D. As the names imply, GRID2D generates grid systems for 2-D spatial domains, while GRID3D does the same for 3-D ones. The programs are quite similar, but each has its own distinctive features which will be described in Section 2.0. Two related programs, 3DSURF and PRGRID, deal with the viewing of grids once they have been generated. These programs will be discussed in Section 3.0

2.0 USING GRID2D/3D TO GENERATE GRID SYSTEMS

In this section, a description of how to generate grid systems for 2-D and 3-D spatial domains is presented. Instructions are given in the context of using GRID2D/3D on an IBM PC compatible computer; however, use of the program on a workstation or mainframe computer follows the same general procedure with only minor changes in file handling.

2.1 How to Generate Grid Systems in Two-Dimensional Spatial Domains

GRID2D/3D can be used to generate grids for 2-D spatial domains which meet the following criteria:

1. The domain of interest is a 2-D region with two or four specified boundary curves.
2. Each boundary curve is described by a set of discrete points which lie along the curve.

If these criteria have been met, the user must then answer the following questions:

1. Should the Two-Boundary Method or the Four-Boundary Method be used?
2. How many grid points are desired in the ξ and η directions?
3. Is any clustering of grid points needed?
4. What "K factors" (mentioned in Part 1 of this technical memorandum) should be used?

Once these questions have been answered, an input file should be constructed according to the format shown in figures 2-1 and 2-2. A guide to the grid control parameters mentioned in these figures is given in table 2-1.

The numbering of boundary curves is an important part of the input file, and the user should consult figure 2-3 to see how the numbered curves are mapped to the "transformed" domain. Figures 2-4 and 2-5 present sample 2-D grid input files for GRID2D/3D.

When running GRID2D/3D on a PC using MS-DOS, the user should first specify the input and output files using the DOS SET command. This is accomplished by typing

```
set 7=input.dat
set 8=output.dat
```

at the DOS prompt. The user should substitute the appropriate names for the input and output files. After this, the program can be run by typing

```
grid2d
```

at the DOS prompt.

The 2-D grid output files created by GRID2D/3D are printed out using the following algorithm (listed in pseudocode):

```
PrintOnOneLine (IL) (# of grid points in the ξ "direction")
PrintOnOneLine (JL) (# of grid points in the η "direction")
for i=1 to IL
   for j=1 to JL
      PrintOnOneLine  (x(i, j)   y(i, j)   1.0)
   end for j
end for i
```

The user should notice that a 1.0 has been tacked onto the end of each (x, y) grid point ordered pair. This was done to facilitate the use of the 3DSURF graphics program which is described in Section 3.0.

2.2 How to Generate Grid Systems in Three-Dimensional Spatial Domains

GRID2D/3D can be used to generate grids for 3-D spatial domains which meet the following criteria:

1. The domain of interest is a 3-D region with two or four specified boundary surfaces.

2. All boundary surfaces are four-sided with each side having four edge curves. Each of these edge curves is described by a set of discrete points which lie along the curve.

If these criteria have been met, the user must then answer the following questions:

1. Should the Two-Surface Method or the Four-Surface Method be used?
2. How many grid points are desired in the ξ, η, and ζ directions?
3. Is any clustering of grid points needed?
4. What "K factors" (mentioned in Part 1 of this technical memorandum) should be used?

Once these questions have been answered, an input file should be constructed according to the format shown in figure 2-6. Note that a 3-D grid input file for GRID2D/3D contains information about only one grid. This is different from a 2-D grid input file in which more than one grid can be specified. A guide to the grid control parameters in figure 2-6 was given previously in table 2-1.

The numbering of boundary curves is an important part of the input file, and the user should consult figure 2-7 to see how the numbered curves are mapped to the "transformed" domain. It should be noted that the boundary surfaces are formed from their edge curves as follows:

surface 1 - formed from curves 1 — 4
surface 2 - formed from curves 5 — 8
surface 3 - formed from curves 9 — 12
surface 4 - formed from curves 13 — 16

It should be further noted that some of the curves are identical. Thus, curves 1 and 9 are the same, as are curves 2 and 13, curves 5 and 10, and curves 6 and 14. The reason for this is that the authors felt that it was simpler to have the user repeat the necessary curves in the input file to preserve a logical structure (i.e., surface 1 possesses edge curves 1 through 4, surface 2 possesses edge curves 5 through 8, and so on). Figure 2-8 presents a sample 3-D grid input file for GRID2D/3D.

When running the program on a PC using MS-DOS, the user should first specify the input and output files using the DOS SET command. This is accomplished by typing

```
set 7=input.dat
set 8=output.dat
```

at the DOS prompt. The user should substitute the appropriate names for the input and output files. After this, the program can be run by typing

```
grid3d
```

at the DOS prompt.

The 3-D grid output files created by GRID2D/3D are printed out using the following algorithm (listed in pseudocode):

```
PrintOnOneLine (IL) (# of grid points in the ξ  "direction")
PrintOnOneLine (JL) (# of grid points in the η  "direction")
PrintOnOneLine (KL) (# of grid points in the ζ "direction")
for i=1 to IL
   for j=1 to JL
      for k=1 to KL
            PrintOnOneLine   (x(i, j)   y(i, j)   z(i, j, k))
      end for k
   end for j
end for i.
```

3.0 HOW TO VIEW THE TWO- AND THREE-DIMENSIONAL GRID SYSTEMS GENERATED

Two additional programs have been written to accompany GRID2D/3D. These two programs allow the user to see the grid which GRID2D/3D has produced as a graphics image on the computer screen and as hardcopy output from a Hewlett Packard HP-7470A pen plotter. The program which yields the graphics image/plotter output is called 3DSURF and is written in Turbo Pascal.

3DSURF is intended for use on an IBM PC, XT, or AT compatible computer and supports both CGA and EGA graphics. This program takes a 2-D grid file created by GRID2D/3D and draws the resulting grid on the CRT and the pen plotter (if desired); however, 3-D grid files created by GRID2D/3D are not suitable as input for 3DSURF. For this reason, another program, PRGRID, has been written. PRGRID uses a 3-D grid file created by GRID2D/3D to generate an output file suitable for use as input to 3DSURF. This section is intended to show how 3DSURF and PRGRID can be used.

3.1 How to Use 3DSURF

3DSURF allows the user to plot surface grids on the computer screen. Two-dimensional grid output files from GRID2D/3D are already in the proper format for use with 3DSURF. Three-dimensional grid output files from GRID2D/3D must be used as input for PRGRID as detailed in the next subsection to create files suitable for use with 3DSURF.

An input file for 3DSURF consists of a list of the grid point locations for one or more grid surfaces together with the number of grid points for each surface. The 2-D grid output files from GRID2D/3D have been "padded" in the sense that a z-coordinate of 1.0 has been assigned to all grid points generated by GRID2D/3D. Output files from PRGRID naturally have z-coordinates since they represent 3-D grids. The user should look at a 2-D grid output file from GRID2D/3D or an output file from PRGRID using an editor to see what these files actually look like.

Assuming the user has a suitable input file for 3DSURF (i.e., a 2-D grid output file from GRID2D/3D or an output file from PRGRID), 3DSURF can be invoked by typing

```
3dsurf
```

at the DOS prompt. The program will begin by asking the user for the input file name which the user should type in (including the appropriate drive information). The program will then inform the user how many surface grids are in the specified input file and ask how many of these surfaces the user actually wants to see. After the user responds, the program will ask for a list of the numbers of each surface to be displayed. The user should enter these numbers as a list on one line, each number separated from the next by a space. As an example, suppose that the user has just indicated that he or she wishes to view three surfaces out of a possible twelve in some file. If the surfaces that he or she wants to see are the third, fifth, and ninth in the file, then he or she should type

3 5 9

when asked for the surface numbers. Having been informed which surfaces are to be displayed, the program will prompt the user for what is referred to as the View Reference Point (VRP). This point represents the location of the user's eye as he or she views the surface or surfaces. For 2-D grids, an appropriate VRP is

0.0 0.0 1.0.

The VRP coordinates should be entered exactly as just described (i.e., as an ordered triple of real numbers separated by spaces, representing the x, y, and z locations of the VRP). A scale factor will also be requested. Numbers between 0.0 and 1.0 will reduce the image, while a value of 1.0 allows the program to automatically scale the image to fill the screen. Finally, the program will ask the user whether he or she wishes to generate pen plotter output. This is possible only if the PC is connected to a Hewlett Packard HP-7470A pen plotter (baud rate = 4800). The appropriate responses are "y" for yes and "n" for no. Having received a response, the program will generate a view of the grid as if the user were looking at it from the VRP. Parallel projection is used (i.e., no perspective), and no effort is made to hide lines or surfaces which should be hidden from view. Once a grid has been displayed, the user only needs to hit <RETURN> to produce a menu asking whether he or she wishes to continue.

3.2 How to Use PRGRID

Once a 3-D grid output file has been generated with GRID2D/3D, the user is often interested in evaluating the resulting grid visually using 3DSURF. As mentioned previously, 3-D grid output files from GRID2D/3D are not suitable as input files for 3DSURF. For this reason, the FORTRAN 77 program PRGRID has been written. PRGRID reads 3-D grid files created by GRID2D/3D and produces output files which can be used as input files for 3DSURF.

When running PRGRID, the user should first specify the input and output files using the DOS SET command. This is accomplished by typing

```
set 7=input.dat
set 8=output.dat
```

at the DOS prompt. The user should substitute the appropriate names for the input and output files. After this, the program can be run by typing

```
prgrid
```

at the DOS prompt. The user will be informed of the number of grid points in the ξ, η and ζ "directions" (IL, JL, and KL, respectively) and will be asked for three new numbers. These numbers are referred to as IL1, JL1, and KL1, and they represent the grid point numbers where an internal "chunk" will be removed from the grid to allow the user to see the "inside" of the 3-D grid (fig. 3-1). The output files created by PRGRID contain information about the 12 surfaces shown in figure 3-1. Since plotting the entire 3-D grid without hiding appropriate lines and surfaces would be confusing, PRGRID basically filters out all of the other grid points not contained in surfaces 1 through 12. As a consequence, its output files provide 3DSURF with information about only the outside "shell" of the 3-D grid with a "chunk" taken out to show the grid's interior features. Usually, the first nine surfaces are sufficient, but sometimes the other three (10–12) prove visually helpful. Here, we note that if the user specifies that IL1=IL, JL1=JL, and KL1=KL, then the output file from PRGRID will contain only the six surfaces which correspond to the six faces of the cube-shaped "transformed" domain.

4.0 EXAMPLES

In this section, we provide two sample input files for GRID2D/3D, together with the grids that were generated using these files. By examining these input files and the resulting grids, it is hoped that the user may become more familiar with the use of GRID2D/3D. The first file will be referred to as INLET.DAT and is a GRID2D/3D input file for generating a two-zone grid for a 2-D supersonic inlet. INLET.DAT is listed in figure 4-1, and the corresponding grid is shown in figure 4-2. The geometry of the inlet dictates the use of a zonal approach due to the separation of the interior and exterior flows. Clustering in the η "direction" was used to place more grid points near the solid surfaces in anticipation of complex boundary layer flow there. Other zonal configurations could be used to better resolve the physics of the flow (e.g., to facilitate shock capturing during the solution if the flow is supersonic).

The second example input file for GRID2D/3D will be referred to as STATOR.DAT and is actually four separate files—ZONE1.DAT, ZONE2.DAT, ZONE3.DAT, and ZONE4.DAT, listed in figures 4-3 through 4-6, respectively. These files create a four-zone grid between the blades of a 3-D axial turbine stator. The resulting grid is shown in figure 4-7. A single-zone grid was found to be unsatisfactory, so four zones were chosen to minimize grid skewness while maintaining (to as high a degree as possible) grid line orthogonality at the boundaries. Even with a judicious choice of zones, it was still necessary to smooth the ζ grid lines along the zonal interfaces. Such smoothing is often necessary in cases where the geometry contains sharp corners in the boundaries. A method for smoothing was given previously in Section 4.1 of Part 1 of this technical memorandum.

5.0 SUMMARY

Part 2 of this technical memorandum has presented a user's manual for GRID2D/3D – a versatile and efficient computer program for generating grid systems inside complex-shaped 2-D and 3-D spatial domains. The structures of the input and output files for GRID2D/3D have been discussed, and the details necessary to use GRID2D/3D for generating grid systems have been given. Two related programs, 3DSURF and PRGRID, have also been described. These programs are used to view the grid systems generated by GRID2D/3D on the CRT and may also be used to generate a hardcopy output of a grid on a Hewlett Packard HP-7470A pen plotter. Two example grids were presented showing both the input files and the resulting grids. Finally, a complete listing of GRID2D/3D is given in the Appendixes.

A.1 Listing of GRID2D

```
      PROGRAM Grid2D

C This program generates a two-dimensional algebraic grid system using
C transfinite Hermite interpolation. The geometric configuration of the
C grid is determined by information provided by the user at run-time via
C an input file. The user selects either the "two-boundary technique" or
C the "four-boundary technique" depending on whether two or four
C boundaries of the spatial domain need to be mapped correctly. Boundary
C curves are formed from sets of discrete data points using tension splines.

      PARAMETER (MxBCvs = 4, MxBPts = 50, MxGSiz = 51)

      INTEGER Tech, CrvNum, GrdNum, NBCrvs, NGrids, StrXi, StrEt,
     $        IL, JL, i, j, NDPts(MxBCvs)

      REAL EtStep, XiStep, S1XiRa, S2XiRa, S3EtRa, S4EtRa,
     $     k1, k2, k3, k4, BetaXi, BetaEt,
     $     Right(MxBPts), Diag(MxBPts), OfDiag(MxBPts),
     $     h1(MxGSiz), h2(MxGSiz), h3(MxGSiz),
     $     h4(MxGSiz), h5(MxGSiz), h6(MxGSiz),
     $     h7(MxGSiz), h8(MxGSiz), X1(MxGSiz), X2(MxGSiz),
     $     X3(MxGSiz), X4(MxGSiz), Y1(MxGSiz), Y2(MxGSiz),
     $     Y3(MxGSiz), Y4(MxGSiz),
     $     PX1PEt(MxGSiz), PX2PEt(MxGSiz),
     $     PY1PEt(MxGSiz), PY2PEt(MxGSiz),
     $     PX3PXi(MxGSiz), PY3PXi(MxGSiz),
     $     PX4PXi(MxGSiz), PY4PXi(MxGSiz),
     $     Tensn(MxBCvs),  x(MxBCvs,MxBPts),
     $     y(MxBCvs,MxBPts), s(MxBCvs,MxBPts),
     $     zx(MxBCvs,MxBPts), zy(MxBCvs,MxBPts),
     $     XMid(MxGSiz,MxGSiz), YMid(MxGSiz,MxGSiz)

      READ(7,*) NGrids
      WRITE(8,*) NGrids

      DO 20 GrdNum=1,NGrids

        READ(7,*) Tech
          NBCrvs=Tech

C Form the boundary curves by splining.

        DO 10 CrvNum=1,NBCrvs
          CALL RdGrIn(x,y,NDPts,CrvNum,Tensn,MxBCvs,MxBPts)
          CALL PTSpln(x,y,s,zx,zy,NDPts(CrvNum),CrvNum,Tensn(CrvNum),
     $         Right,Diag,OfDiag,MxBCvs,MxBPts)
 10     CONTINUE
```

```
C Calculate the grid point locations.

      CALL RdRgIn(IL,JL,StrXi,StrEt,Tech,NDPts,k1,k2,k3,k4,
     $            BetaXi,BetaEt,S1XiRa,S2XiRa,S3EtRa,S4EtRa,
     $            s,XiStep,EtStep,MxBCvs,MxBPts)


      CALL TwoBnd(XMid,YMid,IL,JL,k1,k2,BetaXi,BetaEt,
     $            S1XiRa,S2XiRa,XiStep,EtStep,h1,h2,h3,h4,
     $            X1,X2,Y1,Y2,PX1PEt,PX2PEt,PY1PEt,PY2PEt,
     $            x,y,s,zx,zy,NDPts,StrXi,StrEt,Tensn,MxBCvs,
     $            MxBPts,MxGSiz)
      ENDIF

      IF (Tech.EQ.4) THEN
        CALL FourBd(XMid,YMid,IL,JL,k3,k4,BetaXi,BetaEt,
     $              S3EtRa,S4EtRa,XiStep,EtStep,h1,h2,h3,h4,
     $              X1,X2,Y1,Y2,PX1PEt,PX2PEt,PY1PEt,PY2PEt,
     $              x,y,s,zx,zy,NDPts,StrXi,StrEt,Tensn,
     $              h5,h6,h7,h8,X3,X4,Y3,Y4,PX3PXi,PX4PXi,
     $              PY3PXi,PY4PXi,MxBCvs,MxBPts,MxGSiz)
      ENDIF

      CALL PrGrid(XMid,YMid,IL,JL,MxBCvs,MxBPts,MxGSiz)

 20   CONTINUE

      END

C===============================================================C

      SUBROUTINE RdGrIn (x,y,NPts,CrvNum,Tensn,MxBCvs,MxBPts)

C This procedure reads in the information concerning discrete points on
C the boundaries. The information is used for generating spline-fitted
C boundary approximation curves.

      INTEGER  CrvNum, i, NPts(MxBCvs)

      REAL  x(MxBCvs,MxBPts), y(MxBCvs,MxBPts),
     $      Tensn(MxBCvs)

      READ(7,*) Tensn(CrvNum)
      READ(7,*) NPts(CrvNum)

      DO 10 i=1,NPts(CrvNum)
        READ(7,*) x(CrvNum,i), y(CrvNum,i)
 10   CONTINUE

      RETURN
      END


C===============================================================C
```

```fortran
      SUBROUTINE CalcS (x,y,s,NPts,CrvNum,MxBCvs,MxBPts)

C This procedure calculates the spline parameter, s, as an approximate arc length.

      INTEGER  CrvNum, NPts, i

      REAL  x(MxBCvs,MxBPts), y(MxBCvs,MxBPts),
     $      s(MxBCvs,MxBPts)

      s(CrvNum,1)=0.0

      DO 10 i=2,NPts
        s(CrvNum,i)=s(CrvNum,i-1)
     $              +SQRT( (x(CrvNum,i)-x(CrvNum,i-1))**2
     $                    +(y(CrvNum,i)-y(CrvNum,i-1))**2)
 10   CONTINUE

      RETURN
      END
C==============================================================C

      SUBROUTINE SplMat (Diag,OfDiag,Right,w,s,NPts,T,CrvNum,
     $          MxBCvs,MxBPts)

C This procedure forms the parametric tension spline matrix for a
C particular boundary curve data set.

      INTEGER  CrvNum, NPts, i

      REAL  Diag(MxBPts), OfDiag(MxBPts), Right(MxBPts),
     $      w(MxBCvs,MxBPts), s(MxBCvs,MxBPts), T, h, hm

      Diag(1)=1.0
      OfDiag(1)=0.0
      Right(1)=0.0

      DO 10 i=2,NPts-1
        h=s(CrvNum,i+1)-s(CrvNum,i)
        hm=s(CrvNum,i)-s(CrvNum,i-1)
        Diag(i)=(T*COSH(T*hm)/SINH(T*hm)-1/hm+T*COSH(T*h)/SINH(T*h)
     $          -1/h)/T**2
         OfDiag(i)=(1/h-T/SINH(T*h))/T**2
         Right(i)=(w(CrvNum,i+1)-w(CrvNum,i))/h
     $            -(w(CrvNum,i)-w(CrvNum,i-1))/hm
 10   CONTINUE

      Diag(NPts)=1.0
      OfDiag(NPts-1)=0.0
      Right(NPts)=0.0

      RETURN
      END
C==============================================================C
```

```
      SUBROUTINE SplSlv (Diag,OfDiag,Right,Deriv2,NPts,CrvNum,
     $                   MxBCvs,MxBPts)

C This procedure solves the diagonally dominant parametric tension
C spline matrix for a given data set using the Gauss-Seidel iteration.
C Convergence is assumed after 20 iterations.

      INTEGER  NPts, CrvNum, i, j

      REAL  Diag(MxBPts), OfDiag(MxBPts), Right(MxBPts),
     $      Deriv2(MxBCvs,MxBPts)

C Initialize the second derivative matrix to all zeroes.

      DO 10 i=1,NPts
        Deriv2(CrvNum,i)=0.0
 10   CONTINUE

C Calculate the second derivative values using 20 iterations of
C the Gauss-Seidel method.

      DO 30 j=1,20
        DO 20 i=2,NPts-1
          Deriv2(CrvNum,i)=(Right(i)-OfDiag(i)*Deriv2(CrvNum,i+1)
     $                     -OfDiag(i-1)*Deriv2(CrvNum,i-1))
     $                     /Diag(i)
20      CONTINUE
30    CONTINUE

      RETURN
      END

C==================================================================C

      FUNCTION SplVal (s,w,Deriv2,sval,T,n,CrvNum,MxBCvs,MxBPts)

C This real function finds the w-value (x-value or y-value) corresponding
C to a specified s-value using the parametric tension spline curve
C generated for a particular boundary curve data set.

      INTEGER  n, CrvNum

      REAL  s(MxBCvs,MxBPts), w(MxBCvs,MxBPts),
     $      Deriv2(MxBCvs,MxBPts), sval, T, h, Interim,
     $      Temp1, Temp2

      Temp1=sval-s(CrvNum,n)

      h=s(CrvNum,n+1)-s(CrvNum,n)
      Temp2=s(CrvNum,n+1)-sval
      Interim=Deriv2(CrvNum,n)/T**2*SINH(T*Temp2)/SINH(T*h)
     $       +(w(CrvNum,n)-Deriv2(CrvNum,n)/T**2)*Temp2/h

      SplVal=Interim+Deriv2(CrvNum,n+1)/T**2*SINH(T*Temp1)
```

```
     $                  /SINH(T*h)+(w(CrvNum,n+1)
     $                  -Deriv2(CrvNum,n+1)/T**2)*Templ/h

      RETURN
      END

C===========================================================C

      SUBROUTINE PTSpln (x,y,s,XDeriv2,YDeriv2,NPts,CrvNum,Tensn,
     $                 Right,Diag,OfDiag,MxBCvs,MxBPts)

C This procedure forms the main routine for the parametric tension
C spline process.

      INTEGER  NPts, CrvNum

      REAL  Tensn, x(MxBCvs,MxBPts), y(MxBCvs,MxBPts),
     $      s(MxBCvs,MxBPts), XDeriv2(MxBCvs,MxBPts),
     $      YDeriv2(MxBCvs,MxBPts), Diag(MxBPts),
     $      OfDiag(MxBPts), Right(MxBPts)

      CALL CalcS(x,y,s,NPts,CrvNum,MxBCvs,MxBPts)

      CALL SplMat (Diag,OfDiag,Right,x,s,NPts,Tensn,CrvNum,
     $           MxBCvs,MxBPts)
      CALL SplSlv (Diag,OfDiag,Right,XDeriv2,NPts,CrvNum,
     $           MxBCvs,MxBPts)
      CALL SplMat (Diag,OfDiag,Right,y,s,NPts,Tensn,CrvNum,
     $           MxBCvs,MxBPts)
      CALL SplSlv (Diag,OfDiag,Right,YDeriv2,NPts,CrvNum,
     $           MxBCvs,MxBPts)

      RETURN
      END

C===========================================================C

      SUBROUTINE FindHs (h1,h2,h3,h4,n)

C This procedure computes the h factors used in Hermite interpolation.

      REAL  h1, h2, h3, h4, n

      h1= 2*n**3-3*n**2+1
      h2=-2*n**3+3*n**2
      h3= n**3-2*n**2+n
      h4= n**3-n**2

      RETURN
      END

C===========================================================C

      SUBROUTINE SplInt (n,s,SValue,NDPts,CurCrv,MxBCvs,MxBPts)
```

```
C This procedure finds the proper interval in which a point on a specified
C boundary lies.  The interval indicates which initial data points the
C point in question lies between and thus which spline coefficients to
C use.

      INTEGER  i, n, CurCrv, NDPts(MxBCvs)

      REAL  Temp, SValue, s(MxBCvs,MxBPts)

      n=1
      i=NDPts(CurCrv)

 10   IF ((n.EQ.1).AND.(i.GT.1)) THEN
        i=i-1
        Temp=SValue-s(CurCrv,i)

        IF (Temp.GT.0.0) THEN
          n=i
        ENDIF

        GOTO 10
      ENDIF

      RETURN
      END

C=============================================================C

      SUBROUTINE FAlNew (AlNew,Alpha,B,Str)

C This procedure computes the new Alpha value after stretching as
C AlNew.  Alpha is a dummy variable representing either Xi or Eta.

      INTEGER  Str

      REAL  AlNew, Alpha, B, Temp1, Temp2, B2

      AlNew=Alpha
      Temp1=(B+1)/(B-1)

      IF (Str.EQ.1) THEN
        Temp2=Temp1**(1-Alpha)
        AlNew=((B+1)-(B-1)*Temp2)/(Temp2+1)*1
      ENDIF

      IF (Str.EQ.2) THEN
        B2=0
        Temp2=Temp1**((Alpha-B2)/(1-B2))
        AlNew=((B+2*B2)*Temp2-B+2*B2)/((2*B2+1)*(1+Temp2))
      ENDIF



      IF (Str.EQ.3) THEN
```

```
      B2=0.5
      Temp2=Temp1**((Alpha-B2)/(1-B2))
      AlNew=((B+2*B2)*Temp2-B+2*B2)/((2*B2+1)*(1+Temp2))
    ENDIF

    RETURN
    END

C===============================================================C

    SUBROUTINE TwoBnd (XMid,YMid,IL,JL,k1,k2,BetaXi,BetaEt,S1XiRa,
   $              S2XiRa,XiStep,EtStep,h1,h2,h3,h4,X1,X2,Y1,Y2,
   $              PX1PEt,PX2PEt,PY1PEt,PY2PEt,x,y,s,zx,zy,
   $              NDPts,StrXi,StrEt,Tensn,MxBCvs,MxBPts,MxGSiz)

C This procedure calculates the grid point locations between two specified
C boundaries (1 and 2) using the "two-boundary technique".

    INTEGER  XCnt, YCnt, n1, n2, IL, JL, StrXi, StrEt, NDPts(MxBCvs)

    REAL  Xi, Eta, XiNew, EtaNew, XiStep, EtStep,
   $     S1XiRa, S2XiRa, S1, S2, k1, k2,
   $     BetaXi, BetaEt, PY1PXi, PX1PXi, PY2PXi, PX2PXi,
   $     x(MxBCvs,MxBPts), y(MxBCvs,MxBPts),
   $     s(MxBCvs,MxBPts), Tensn(MxBCvs),
   $     zx(MxBCvs,MxBPts), zy(MxBCvs,MxBPts),
   $     h1(MxGSiz), h2(MxGSiz), h3(MxGSiz), h4(MxGSiz),
   $     X1(MxGSiz), X2(MxGSiz), Y1(MxGSiz), Y2(MxGSiz),
   $     PX1PEt(MxGSiz), PX2PEt(MxGSiz),
   $     PY1PEt(MxGSiz), PY2PEt(MxGSiz),
   $     XMid(MxGSiz,MxGSiz), YMid(MxGSiz,MxGSiz)

C Calculate the grid point locations along boundaries 1 and 2.

    Xi=0.0

    DO 10 XCnt=1,IL
      CALL FAlNew(XiNew,Xi,BetaXi,StrXi)
      S1=XiNew*S1XiRa
      S2=XiNew*S2XiRa
      CALL SplInt(n1,s,S1,NDPts,1,MxBCvs,MxBPts)
      CALL SplInt(n2,s,S2,NDPts,2,MxBCvs,MxBPts)
      X1(XCnt)=SplVal(s,x,zx,S1,Tensn(1),n1,1,MxBCvs,MxBPts)
      X2(XCnt)=SplVal(s,x,zx,S2,Tensn(2),n2,2,MxBCvs,MxBPts)
      Y1(XCnt)=SplVal(s,y,zy,S1,Tensn(1),n1,1,MxBCvs,MxBPts)
      Y2(XCnt)=SplVal(s,y,zy,S2,Tensn(2),n2,2,MxBCvs,MxBPts)
      Xi=Xi+XiStep
 10 CONTINUE

C Calculate the h factors for the boundary 1-2 Hermite connecting
C curves.

    Eta=0.0

    DO 20 YCnt=1,JL
```

```
      CALL FAlNew(EtaNew,Eta,BetaEt,StrEt)
      CALL FindHs(h1(YCnt),h2(YCnt),h3(YCnt),h4(YCnt),EtaNew)
      Eta=Eta+EtStep
 20   CONTINUE

C Calculate the derivative values for forcing grid line orthogonality
C along boundaries 1 and 2.

      PX1PXi=(X1(2)-X1(1))/XiStep
      PX2PXi=(X2(2)-X2(1))/XiStep
      PY1PXi=(Y1(2)-Y1(1))/XiStep
      PY2PXi=(Y2(2)-Y2(1))/XiStep
      PX1PEt(1)=-k1*PY1PXi
      PX2PEt(1)=-k2*PY2PXi
      PY1PEt(1)= k1*PX1PXi
      PY2PEt(1)= k2*PX2PXi

      PX1PXi=(X1(IL)-X1(IL-1))/XiStep
      PX2PXi=(X2(IL)-X2(IL-1))/XiStep
      PY1PXi=(Y1(IL)-Y1(IL-1))/XiStep
      PY2PXi=(Y2(IL)-Y2(IL-1))/XiStep
      PX1PEt(IL)=-k1*PY1PXi
      PX2PEt(IL)=-k2*PY2PXi
      PY1PEt(IL)= k1*PX1PXi
      PY2PEt(IL)= k2*PX2PXi

      DO 30 XCnt=2,IL-1
        PX1PXi=(X1(XCnt+1)-X1(XCnt-1))/2/XiStep
        PX2PXi=(X2(XCnt+1)-X2(XCnt-1))/2/XiStep
        PY1PXi=(Y1(XCnt+1)-Y1(XCnt-1))/2/XiStep
        PY2PXi=(Y2(XCnt+1)-Y2(XCnt-1))/2/XiStep
        PX1PEt(XCnt)=-k1*PY1PXi
        PX2PEt(XCnt)=-k2*PY2PXi
        PY1PEt(XCnt)= k1*PX1PXi
        PY2PEt(XCnt)= k2*PX2PXi
 30   CONTINUE

C Calculate the interior grid point locations.

      DO 50 XCnt=1,IL
        DO 40 YCnt=1,JL
          XMid(XCnt,YCnt)= h1(YCnt)*X1(XCnt)
     $                    +h2(YCnt)*X2(XCnt)
     $                    +h3(YCnt)*PX1PEt(XCnt)
     $                    +h4(YCnt)*PX2PEt(XCnt)
          YMid(XCnt,YCnt)= h1(YCnt)*Y1(XCnt)
     $                    +h2(YCnt)*Y2(XCnt)
     $                    +h3(YCnt)*PY1PEt(XCnt)
     $                    +h4(YCnt)*PY2PEt(XCnt)
 40     CONTINUE
 50   CONTINUE

      RETURN
      END
```

```
C=====================================================C

      SUBROUTINE FourBd (XMid,YMid,IL,JL,k3,k4,BetaXi,BetaEt,
     $                   S3EtRa,S4EtRa,XiStep,EtStep,h1,h2,h3,h4,
     $                   X1,X2,Y1,Y2,PX1PEt,PX2PEt,PY1PEt,PY2PEt,
     $                   x,y,s,zx,zy,NDPts,StrXi,StrEt,Tensn,
     $                   h5,h6,h7,h8,X3,X4,Y3,Y4,PX3PXi,PX4PXi,
     $                   PY3PXi,PY4PXi,MxBCvs,MxBPts,MxGSiz)

C This procedure adjusts the grid so that the other two boundaries (3 and 4)
C are mapped correctly using the "four-boundary technique".

      INTEGER  XCnt, YCnt, i, j, n3, n4, IL, JL, StrXi, StrEt,
     $         NDPts(MxBCvs)

      REAL  Xi, Eta, XiNew, EtaNew, XiStep, EtStep, S3EtRa, S4EtRa,
     $      S3, S4, k3, k4, BetaXi, BetaEt,
     $      x(MxBCvs,MxBPts), y(MxBCvs,MxBPts), s(MxBCvs,MxBPts),
     $      zx(MxBCvs,MxBPts), zy(MxBCvs,MxBPts), Tensn(MxBCvs),
     $      h1(MxGSiz), h2(MxGSiz), h3(MxGSiz), h4(MxGSiz),
     $      h5(MxGSiz), h6(MxGSiz), h7(MxGSiz), h8(MxGSiz),
     $      X1(MxGSiz), X2(MxGSiz), Y1(MxGSiz), Y2(MxGSiz),
     $      X3(MxGSiz), X4(MxGSiz), Y3(MxGSiz), Y4(MxGSiz),
     $      PY3PEt, PX3PEt, PY4PEt, PX4PEt,
     $      P2Y00, P2Y01, P2Y10, P2Y11, P2X00, P2X01, P2X10, P2X11,
     $      PX3PXi(MxGSiz), PY3PXi(MxGSiz),
     $      PX4PXi(MxGSiz), PY4PXi(MxGSiz),
     $      PX1PEt(MxGSiz), PX2PEt(MxGSiz),
     $      PY1PEt(MxGSiz), PY2PEt(MxGSiz),
     $      XMid(MxGSiz,MxGSiz), YMid(MxGSiz,MxGSiz)

C Calculate the grid point locations along boundaries 3 and 4.

      Eta=0.0

      DO 10 YCnt=1,JL
        CALL FAlNew(EtaNew,Eta,BetaEt,StrEt)
        S3=EtaNew*S3EtRa
        S4=EtaNew*S4EtRa
        CALL SplInt(n3,s,S3,NDPts,3,MxBCvs,MxBPts)
        CALL SplInt(n4,s,S4,NDPts,4,MxBCvs,MxBPts)
        X3(YCnt)=SplVal(s,x,zx,S3,Tensn(3),n3,3,MxBCvs,MxBPts)
        X4(YCnt)=SplVal(s,x,zx,S4,Tensn(4),n4,4,MxBCvs,MxBPts)
        Y3(YCnt)=SplVal(s,y,zy,S3,Tensn(3),n3,3,MxBCvs,MxBPts)
        Y4(YCnt)=SplVal(s,y,zy,S4,Tensn(4),n4,4,MxBCvs,MxBPts)
        Eta=Eta+EtStep
 10   CONTINUE

C Calculate the h factors for the boundary 3-4 Hermite adjusting
C curves.

      Xi=0.0


      DO 20 XCnt=1,IL
```

```
      CALL FAlNew(XiNew,Xi,BetaXi,StrXi)
      CALL FindHs(h5(XCnt),h6(XCnt),h7(XCnt),h8(XCnt),XiNew)
      Xi=Xi+XiStep
 20   CONTINUE

C Calculate the derivative values for forcing grid line orthogonality
C along boundaries 3 and 4.

    PX3PEt=(X3(2)-X3(1))/EtStep
    PX4PEt=(X4(2)-X4(1))/EtStep
    PY3PEt=(Y3(2)-Y3(1))/EtStep
    PY4PEt=(Y4(2)-Y4(1))/EtStep
    PX3PXi(1)= k3*PY3PEt
    PX4PXi(1)= k4*PY4PEt
    PY3PXi(1)=-k3*PX3PEt
    PY4PXi(1)=-k4*PX4PEt

    PX3PEt=(X3(JL)-X3(JL-1))/EtStep
    PX4PEt=(X4(JL)-X4(JL-1))/EtStep
    PY3PEt=(Y3(JL)-Y3(JL-1))/EtStep
    PY4PEt=(Y4(JL)-Y4(JL-1))/EtStep
    PX3PXi(JL)= k3*PY3PEt
    PX4PXi(JL)= k4*PY4PEt
    PY3PXi(JL)=-k3*PX3PEt
    PY4PXi(JL)=-k4*PX4PEt

    DO 30 YCnt=2,JL-1
      PX3PEt=(X3(YCnt+1)-X3(YCnt-1))/2/EtStep
      PX4PEt=(X4(YCnt+1)-X4(YCnt-1))/2/EtStep
      PY3PEt=(Y3(YCnt+1)-Y3(YCnt-1))/2/EtStep
      PY4PEt=(Y4(YCnt+1)-Y4(YCnt-1))/2/EtStep
      PX3PXi(YCnt)= k3*PY3PEt
      PX4PXi(YCnt)= k4*PY4PEt
      PY3PXi(YCnt)=-k3*PX3PEt
      PY4PXi(YCnt)=-k4*PX4PEt
 30   CONTINUE

C Set the corner cross derivative terms to zero.

    P2X00=0.0
    P2X10=0.0
    P2X01=0.0
    P2X11=0.0
    P2Y00=0.0
    P2Y10=0.0
    P2Y01=0.0
    P2Y11=0.0

C Calculate the grid point locations everywhere.

    DO 50 i=1,IL
      DO 40 j=1,JL
        XMid(i,j)=XMid(i,j)
   $              +(X3(j)-h1(j)*X1(1)
   $                   -h2(j)*X2(1)
```

```
     $                -h3(j)*PX1PEt(1)
     $                -h4(j)*PX2PEt(1))*h5(i)
     $            +(X4(j)-h1(j)*X1(IL)
     $                -h2(j)*X2(IL)
     $                -h3(j)*PX1PEt(IL)
     $                -h4(j)*PX2PEt(IL))*h6(i)
     $            +(PX3PXi(j)-( h1(j)*PX3PXi(1)
     $                    +h2(j)*PX3PXi(JL)
     $                    +h3(j)*P2X00+h4(j)*P2X01))*h7(i)
     $          +(PX4PXi(j)-( h1(j)*PX4PXi(1)
     $                    +h2(j)*PX4PXi(JL)
     $                    +h3(j)*P2X10+h4(j)*P2X11))*h8(i)
          YMid(i,j)=YMid(i,j)
     $            +(Y3(j)-h1(j)*Y1(1)
     $                -h2(j)*Y2(1)
     $                -h3(j)*PY1PEt(1)
     $                -h4(j)*PY2PEt(1))*h5(i)
     $            +(Y4(j)-h1(j)*Y1(IL)
     $                -h2(j)*Y2(IL)
     $                -h3(j)*PY1PEt(IL)
     $                -h4(j)*PY2PEt(IL))*h6(i)
     $            +(PY3PXi(j)-( h1(j)*PY3PXi(1)
     $                    +h2(j)*PY3PXi(JL)
     $                    +h3(j)*P2Y00+h4(j)*P2Y01))*h7(i)
     $            +(PY4PXi(j)-( h1(j)*PY4PXi(1)
     $                    +h2(j)*PY4PXi(JL)
     $                    +h3(j)*P2Y10+h4(j)*P2Y11))*h8(i)
 40     CONTINUE
 50   CONTINUE

      RETURN
      END

C=============================================C

      SUBROUTINE PrGrid (XMid,YMid,IL,JL,MxBCvs,MxBPts,MxGSiz)

C This procedure prints (to output) the grid point x and y coordinates
C as ordered pairs.

      INTEGER  i, j, IL, JL

      REAL  XMid(MxGSiz,MxGSiz), YMid(MxGSiz,MxGSiz)

      WRITE(8,*) IL
      WRITE(8,*) JL


      DO 20 i=1,IL
        DO 10 j=1,JL
          WRITE(8,35) XMid(i,j),YMid(i,j),1.0
 10     CONTINUE
 20   CONTINUE

 35   FORMAT(1X,F10.6,3X,F10.6,3X,F3.1)
```

```
      RETURN
      END

C==================================================================C

      SUBROUTINE RdRgIn (IL,JL,StrXi,StrEt,Tech,NDPts,k1,k2,k3,k4,
     $                BetaXi,BetaEt,S1XiRa,S2XiRa,S3EtRa,S4EtRa,
     $                s,XiStep,EtStep,MxBCvs,MxBPts)

C This procedure reads in the grid control information.

      INTEGER  Tech, IL, JL, StrXi, StrEt, NDPts(MxBCvs)

      REAL  k1, k2, k3, k4, BetaXi, BetaEt, XiStep, EtStep,
     $      S1XiRa, S2XiRa, S3EtRa, S4EtRa, s(MxBCvs,MxBPts)

      READ(7,*) IL
      READ(7,*) JL

      READ(7,*) StrXi
      READ(7,*) StrEt

      READ(7,*) k1
      READ(7,*) k2

      IF (Tech.EQ.4) THEN
        READ(7,*) k3
        READ(7,*) k4
      ENDIF

      READ(7,*) BetaXi
      READ(7,*) BetaEt

      S1XiRa=s(1,NDPts(1))
      S2XiRa=s(2,NDPts(2))

      IF (Tech.EQ.4) THEN
         S3EtRa=s(3,NDPts(3))
         S4EtRa=s(4,NDPts(4))
      ENDIF

      XiStep=1.0/(IL-1)
      EtStep=1.0/(JL-1)

      RETURN
      END
```

A.2 Listing of GRID3D

```
      PROGRAM Grid3d

      PARAMETER (MxSrfs = 4, MxBPts = 23, MxGSiz = 23)

      INTEGER  CrvNum, SrfNum, NSurfs,
     $        StrXi, StrEt, StrZt, StrAA, StrBB, IL, JL, KL, AL, BL,
     $        i, j, k, NDPts(4)

      REAL  EtStep, XiStep, ZtStep, AAStep, BBStep,
     $     k1, k2, k3, k4, kXi1, kXi2, kEta1, kEta2, kZeta1, kZeta2,
     $     BetaXi, BetaEt, BetaZt, BetaAA, BetaBB,
     $     h1(MxGSiz), h2(MxGSiz), h3(MxGSiz), h4(MxGSiz),
     $     h5(MxGSiz), h6(MxGSiz), h7(MxGSiz), h8(MxGSiz),
     $     X1(MxGSiz), X2(MxGSiz), X3(MxGSiz), X4(MxGSiz),
     $     Y1(MxGSiz), Y2(MxGSiz), Y3(MxGSiz), Y4(MxGSiz),
     $     Z1(MxGSiz), Z2(MxGSiz), Z3(MxGSiz), Z4(MxGSiz),
     $     PXS1PE(MxGSiz,MxGSiz), PXS2PE(MxGSiz,MxGSiz),
     $     PYS1PE(MxGSiz,MxGSiz), PYS2PE(MxGSiz,MxGSiz),
     $     PZS1PE(MxGSiz,MxGSiz), PZS2PE(MxGSiz,MxGSiz),
     $     PXS3Zt(MxGSiz,MxGSiz), PXS4Zt(MxGSiz,MxGSiz),
     $     PYS3Zt(MxGSiz,MxGSiz), PYS4Zt(MxGSiz,MxGSiz),
     $     PZS3Zt(MxGSiz,MxGSiz), PZS4Zt(MxGSiz,MxGSiz)
      REAL  Tensn(4),
     $     Diag(MxBPts), OfDiag(MxBPts), Right(MxBPts),
     $     XDeriv2(4,MxBPts), YDeriv2(4,MxBPts),
     $     ZDeriv2(4,MxBPts),
     $     x(4,MxBPts),  y(4,MxBPts),
     $     z(4,MxBPts),  s(4,MxBPts),
     $    zx(4,MxBPts), zy(4,MxBPts),
     $    zz(4,MxBPts)
      REAL  PX1PBB(MxGSiz), PX2PBB(MxGSiz),
     $     PY1PBB(MxGSiz), PY2PBB(MxGSiz),
     $     PZ1PBB(MxGSiz), PZ2PBB(MxGSiz),
     $     PX1PAA(MxGSiz), PX2PAA(MxGSiz),
     $     PY1PAA(MxGSiz), PY2PAA(MxGSiz),
     $     PZ1PAA(MxGSiz), PZ2PAA(MxGSiz),
     $     PX3PBB(MxGSiz), PX4PBB(MxGSiz),
     $     PY3PBB(MxGSiz), PY4PBB(MxGSiz),
     $     PZ3PBB(MxGSiz), PZ4PBB(MxGSiz),
     $     PX3PAA(MxGSiz), PX4PAA(MxGSiz),
     $     PY3PAA(MxGSiz), PY4PAA(MxGSiz),
     $     PZ3PAA(MxGSiz), PZ4PAA(MxGSiz),
     $     XS(MxSrfs,MxGsiz,MxGsiz),
     $     YS(MxSrfs,MxGsiz,MxGsiz),
     $     ZS(MxSrfs,MxGsiz,MxGsiz),
     $     XPnt(MxGSiz,MxGSiz,MxGSiz),
     $     YPnt(MxGSiz,MxGSiz,MxGSiz),
     $     ZPnt(MxGSiz,MxGSiz,MxGSiz)
```

```
C Read in the grid control information.

      CALL RdGrIn(IL,JL,KL,StrXi,StrEt,StrZt,NSurfs,kXi1,kXi2,
     $            kEta1,kEta2,kZeta1,kZeta2,BetaXi,BetaEt,BetaZt,
     $            XiStep,EtStep,ZtStep)

C Calculate the boundary surface grid point locations.

      DO 20 SrfNum=1,NSurfs

C Form the boundary surface edge curves by splining.

        DO 10 CrvNum=1,4
          CALL RdCvIn(x,y,z,NDPts,CrvNum,Tensn,MxBPts)
          CALL PTSpln(x,y,z,s,zx,zy,zz,Diag,OfDiag,Right,NDPts,
     $              Tensn(CrvNum),CrvNum,MxBPts)
 10     CONTINUE

        IF (SrfNum.LE.2) THEN
          AAStep=ZtStep
          BBStep=XiStep
          StrAA=StrZt
          StrBB=StrXi
          BetaAA=BetaZt
          BetaBB=BetaXi
          k1=kXi1
          k2=kXi2
          k3=kZeta1
          k4=kZeta2
          AL=KL
          BL=IL
        ELSE
          AAStep=XiStep
          BBStep=EtStep
          StrAA=StrXi
          StrBB=StrEt
          BetaAA=BetaXi
          BetaBB=BetaEt
          k1=kEta1
          k2=kEta2
          k3=kXi1
          k4=kXi2
          AL=IL
          BL=JL
        ENDIF

C Calculate the boundary surface edge grid point locations.

        CALL EdgPts(X1,X2,X3,X4,Y1,Y2,Y3,Y4,Z1,Z2,Z3,Z4,AL,BL,
     $              AAStep,BBStep,x,y,z,s,zx,zy,zz,NDPts,Tensn,
     $              StrAA,StrBB,BetaAA,BetaBB,MxBPts,MxGSiz)

C Calculate the boundary surface edge derivative values.
```

```
      CALL EdgDer(PX1PAA,PX2PAA,PY1PAA,PY2PAA,PZ1PAA,PZ2PAA,
     $            PX3PBB,PX4PBB,PY3PBB,PY4PBB,PZ3PBB,PZ4PBB,
     $            X1,X2,X3,X4,Y1,Y2,Y3,Y4,Z1,Z2,Z3,Z4,AL,BL,
     $            AAStep,BBStep,MxGSiz)

C Calculate the boundary surface grid point locations.

      CALL TwoBnd(XS,YS,ZS,SrfNum,AL,BL,k1,k2,BetaAA,BetaBB,
     $            AAStep,BBStep,h1,h2,h3,h4,X1,X2,X3,X4,
     $            Y1,Y2,Y3,Y4,Z1,Z2,Z3,Z4,PX1PBB,PX2PBB,
     $            PY1PBB,PY2PBB,PZ1PBB,PZ2PBB,PX1PAA,PX2PAA,
     $            PY1PAA,PY2PAA,PZ1PAA,PZ2PAA,PX3PBB,PX4PBB,
     $            PY3PBB,PY4PBB,PZ3PBB,PZ4PBB,StrAA,StrBB,
     $            MxGSiz,MxSrfs)

      CALL ForBnd(XS,YS,ZS,SrfNum,AL,BL,k3,k4,BetaAA,BetaBB,
     $            AAStep,BBStep,h1,h2,h3,h4,h5,h6,h7,h8,
     $            X1,X2,X3,X4,Y1,Y2,Y3,Y4,Z1,Z2,Z3,Z4,
     $            PX1PBB,PX2PBB,PY1PBB,PY2PBB,PZ1PBB,PZ2PBB,
     $            PX1PAA,PX2PAA,PY1PAA,PY2PAA,PZ1PAA,PZ2PAA,
     $            PX3PBB,PX4PBB,PY3PBB,PY4PBB,PZ3PBB,PZ4PBB,
     $            PX3PAA,PX4PAA,PY3PAA,PY4PAA,PZ3PAA,PZ4PAA,
     $            StrAA,StrBB,MxGSiz,MxSrfs)

 20   CONTINUE

C Calculate the interior grid point locations.

      CALL TwoSrf(XPnt,YPnt,ZPnt,IL,JL,KL,kEta1,kEta2,
     $           BetaXi,BetaEt,BetaZt,
     $           XiStep,EtStep,ZtStep,XS,YS,ZS,h1,h2,h3,h4,
     $           PXS1PE,PXS2PE,PYS1PE,PYS2PE,PZS1PE,
     $           PZS2PE,StrXi,StrEt,StrZt,MxGSiz,MxSrfs)

      IF (NSurfs.EQ.4) THEN
        CALL ForSrf(XPnt,YPnt,ZPnt,IL,JL,KL,kZeta1,kZeta2,
     $             BetaXi,BetaEt,BetaZt,XS,YS,ZS,
     $             XiStep,EtStep,ZtStep,
     $             h1,h2,h3,h4,h5,h6,h7,h8,
     $             PXS1PE,PXS2PE,PYS1PE,PYS2PE,PZS1PE,PZS2PE,
     $             PXS3Zt,PXS4Zt,PYS3Zt,PYS4Zt,PZS3Zt,PZS4Zt,
     $             StrXi,StrEt,StrZt,MxGSiz,MxSrfs)
      ENDIF

      CALL PrGrid(XPnt,YPnt,ZPnt,IL,JL,KL,MxGSiz)

      END

C==============================================================C

      SUBROUTINE TwoSrf(XPnt,YPnt,ZPnt,IL,JL,KL,k1,k2,BetaXi,BetaEt,
     $             BetaZt,XiStep,EtStep,ZtStep,XS,YS,ZS,
     $             h1,h2,h3,h4,PXS1PE,PXS2PE,PYS1PE,PYS2PE,PZS1PE,
     $             PZS2PE,StrXi,StrEt,StrZt,MxGSiz,MxSrfs)
```

```
C This procedure calculates the grid point locations between two specified
C surfaces using the "two-boundary technique".

      INTEGER i, j, k, StrXi, StrEt, StrZt, IL, JL, KL

      REAL Xi, Eta, Zeta, XiNew, EtaNew, ZtaNew,
     $     PXS1Xi, PXS2Xi, PYS1Xi, PYS2Xi, PZS1Xi, PZS2Xi,
     $     PXS1Zt, PXS2Zt, PYS1Zt, PYS2Zt, PZS1Zt, PZS2Zt,
     $     k1, k2, BetaXi, BetaEt, BetaZt, XiStep, EtStep, ZtStep,
     $     h1(MxGSiz), h2(MxGSiz), h3(MxGSiz), h4(MxGSiz),
     $     PXS1PE(MxGSiz,MxGsiz), PXS2PE(MxGSiz,MxGsiz),
     $     PYS1PE(MxGSiz,MxGsiz), PYS2PE(MxGSiz,MxGsiz),
     $     PZS1PE(MxGSiz,MxGsiz), PZS2PE(MxGSiz,MxGsiz),
     $     XS(MxSrfs,MxGsiz,MxGsiz),
     $     YS(MxSrfs,MxGsiz,MxGsiz),
     $     ZS(MxSrfs,MxGsiz,MxGsiz),
     $     XPnt(MxGSiz,MxGsiz,MxGSiz),
     $     YPnt(MxGSiz,MxGsiz,MxGSiz),
     $     ZPnt(MxGSiz,MxGsiz,MxGSiz)

C Calculate the h factors for the boundary surface 1-2 Hermite
C connecting curves.

      Eta=0.0

      DO 50 j=1,JL
        CALL FAlNew(EtaNew,Eta,BetaEt,StrEt)
        CALL FindHs(h1(j),h2(j),h3(j),h4(j),EtaNew)
        Eta=Eta+EtStep
 50   CONTINUE

C Calculate the derivative values along the constant Xi/Zeta
C boundaries.

      PXS1Xi=(XS(1,1,2)-XS(1,1,1))/XiStep
      PXS2Xi=(XS(2,1,2)-XS(2,1,1))/XiStep
      PYS1Xi=(YS(1,1,2)-YS(1,1,1))/XiStep
      PYS2Xi=(YS(2,1,2)-YS(2,1,1))/XiStep
      PZS1Xi=(ZS(1,1,2)-ZS(1,1,1))/XiStep
      PZS2Xi=(ZS(2,1,2)-ZS(2,1,1))/XiStep
      PXS1Zt=(XS(1,2,1)-XS(1,1,1))/ZtStep
      PXS2Zt=(XS(2,2,1)-XS(2,1,1))/ZtStep
      PYS1Zt=(YS(1,2,1)-YS(1,1,1))/ZtStep
      PYS2Zt=(YS(2,2,1)-YS(2,1,1))/ZtStep
      PZS1Zt=(ZS(1,2,1)-ZS(1,1,1))/ZtStep
      PZS2Zt=(ZS(2,2,1)-ZS(2,1,1))/ZtStep
      PXS1PE(1,1)=-k1*(PYS1Xi*PZS1Zt-PZS1Xi*PYS1Zt)
      PXS2PE(1,1)=-k2*(PYS2Xi*PZS2Zt-PZS2Xi*PYS2Zt)
      PYS1PE(1,1)= k1*(PXS1Xi*PZS1Zt-PZS1Xi*PXS1Zt)
      PYS2PE(1,1)= k2*(PXS2Xi*PZS2Zt-PZS2Xi*PXS2Zt)
      PZS1PE(1,1)=-k1*(PXS1Xi*PYS1Zt-PYS1Xi*PXS1Zt)
      PZS2PE(1,1)=-k2*(PXS2Xi*PYS2Zt-PYS2Xi*PXS2Zt)

      PXS1Xi=(XS(1,1,IL)-XS(1,1,IL-1))/XiStep
      PXS2Xi=(XS(2,1,IL)-XS(2,1,IL-1))/XiStep
```

```
      PYS1Xi=(YS(1,1,IL)-YS(1,1,IL-1))/XiStep
      PYS2Xi=(YS(2,1,IL)-YS(2,1,IL-1))/XiStep
      PZS1Xi=(ZS(1,1,IL)-ZS(1,1,IL-1))/XiStep
      PZS2Xi=(ZS(2,1,IL)-ZS(2,1,IL-1))/XiStep
      PXS1Zt=(XS(1,2,IL)-XS(1,1,IL))/ZtStep
      PXS2Zt=(XS(2,2,IL)-XS(2,1,IL))/ZtStep
      PYS1Zt=(YS(1,2,IL)-YS(1,1,IL))/ZtStep
      PYS2Zt=(YS(2,2,IL)-YS(2,1,IL))/ZtStep
      PZS1Zt=(ZS(1,2,IL)-ZS(1,1,IL))/ZtStep
      PZS2Zt=(ZS(2,2,IL)-ZS(2,1,IL))/ZtStep
      PXS1PE(IL,1)=-k1*(PYS1Xi*PZS1Zt-PZS1Xi*PYS1Zt)
      PXS2PE(IL,1)=-k2*(PYS2Xi*PZS2Zt-PZS2Xi*PYS2Zt)
      PYS1PE(IL,1)= k1*(PXS1Xi*PZS1Zt-PZS1Xi*PXS1Zt)
      PYS2PE(IL,1)= k2*(PXS2Xi*PZS2Zt-PZS2Xi*PXS2Zt)
      PZS1PE(IL,1)=-k1*(PXS1Xi*PYS1Zt-PYS1Xi*PXS1Zt)
      PZS2PE(IL,1)=-k2*(PXS2Xi*PYS2Zt-PYS2Xi*PXS2Zt)

      DO 55 i=2,IL-1
        PXS1Xi=(XS(1,1,i+1)-XS(1,1,i-1))/2/XiStep
        PXS2Xi=(XS(2,1,i+1)-XS(2,1,i-1))/2/XiStep
        PYS1Xi=(YS(1,1,i+1)-YS(1,1,i-1))/2/XiStep
        PYS2Xi=(YS(2,1,i+1)-YS(2,1,i-1))/2/XiStep
        PZS1Xi=(ZS(1,1,i+1)-ZS(1,1,i-1))/2/XiStep
        PZS2Xi=(ZS(2,1,i+1)-ZS(2,1,i-1))/2/XiStep
        PXS1Zt=(XS(1,2,i)-XS(1,1,i))/ZtStep
        PXS2Zt=(XS(2,2,i)-XS(2,1,i))/ZtStep
        PYS1Zt=(YS(1,2,i)-YS(1,1,i))/ZtStep
        PYS2Zt=(YS(2,2,i)-YS(2,1,i))/ZtStep
        PZS1Zt=(ZS(1,2,i)-ZS(1,1,i))/ZtStep
        PZS2Zt=(ZS(2,2,i)-ZS(2,1,i))/ZtStep
        PXS1PE(i,1)=-k1*(PYS1Xi*PZS1Zt-PZS1Xi*PYS1Zt)
        PXS2PE(i,1)=-k2*(PYS2Xi*PZS2Zt-PZS2Xi*PYS2Zt)
        PYS1PE(i,1)= k1*(PXS1Xi*PZS1Zt-PZS1Xi*PXS1Zt)
        PYS2PE(i,1)= k2*(PXS2Xi*PZS2Zt-PZS2Xi*PXS2Zt)
        PZS1PE(i,1)=-k1*(PXS1Xi*PYS1Zt-PYS1Xi*PXS1Zt)
        PZS2PE(i,1)=-k2*(PXS2Xi*PYS2Zt-PYS2Xi*PXS2Zt)
 55   CONTINUE

      DO 70 k=2,KL-1
        PXS1Xi=(XS(1,k,2)-XS(1,k,1))/XiStep
        PXS2Xi=(XS(2,k,2)-XS(2,k,1))/XiStep
        PYS1Xi=(YS(1,k,2)-YS(1,k,1))/XiStep
        PYS2Xi=(YS(2,k,2)-YS(2,k,1))/XiStep
        PZS1Xi=(ZS(1,k,2)-ZS(1,k,1))/XiStep
        PZS2Xi=(ZS(2,k,2)-ZS(2,k,1))/XiStep
        PXS1Zt=(XS(1,k+1,1)-XS(1,k-1,1))/2/ZtStep
        PXS2Zt=(XS(2,k+1,1)-XS(2,k-1,1))/2/ZtStep
        PYS1Zt=(YS(1,k+1,1)-YS(1,k-1,1))/2/ZtStep
        PYS2Zt=(YS(2,k+1,1)-YS(2,k-1,1))/2/ZtStep
        PZS1Zt=(ZS(1,k+1,1)-ZS(1,k-1,1))/2/ZtStep
        PZS2Zt=(ZS(2,k+1,1)-ZS(2,k-1,1))/2/ZtStep
        PXS1PE(1,k)=-k1*(PYS1Xi*PZS1Zt-PZS1Xi*PYS1Zt)
        PXS2PE(1,k)=-k2*(PYS2Xi*PZS2Zt-PZS2Xi*PYS2Zt)
        PYS1PE(1,k)= k1*(PXS1Xi*PZS1Zt-PZS1Xi*PXS1Zt)
        PYS2PE(1,k)= k2*(PXS2Xi*PZS2Zt-PZS2Xi*PXS2Zt)
```

```
      PZS1PE(1,k)=-k1*(PXS1Xi*PYS1Zt-PYS1Xi*PXS1Zt)
      PZS2PE(1,k)=-k2*(PXS2Xi*PYS2Zt-PYS2Xi*PXS2Zt)

      PXS1Xi=(XS(1,k,IL)-XS(1,k,IL-1))/XiStep
      PXS2Xi=(XS(2,k,IL)-XS(2,k,IL-1))/XiStep
      PYS1Xi=(YS(1,k,IL)-YS(1,k,IL-1))/XiStep
      PYS2Xi=(YS(2,k,IL)-YS(2,k,IL-1))/XiStep
      PZS1Xi=(ZS(1,k,IL)-ZS(1,k,IL-1))/XiStep
      PZS2Xi=(ZS(2,k,IL)-ZS(2,k,IL-1))/XiStep
      PXS1Zt=(XS(1,k+1,IL)-XS(1,k-1,IL))/2/ZtStep
      PXS2Zt=(XS(2,k+1,IL)-XS(2,k-1,IL))/2/ZtStep
      PYS1Zt=(YS(1,k+1,IL)-YS(1,k-1,IL))/2/ZtStep
      PYS2Zt=(YS(2,k+1,IL)-YS(2,k-1,IL))/2/ZtStep
      PZS1Zt=(ZS(1,k+1,IL)-ZS(1,k-1,IL))/2/ZtStep
      PZS2Zt=(ZS(2,k+1,IL)-ZS(2,k-1,IL))/2/ZtStep
      PXS1PE(IL,k)=-k1*(PYS1Xi*PZS1Zt-PZS1Xi*PYS1Zt)
      PXS2PE(IL,k)=-k2*(PYS2Xi*PZS2Zt-PZS2Xi*PYS2Zt)
      PYS1PE(IL,k)= k1*(PXS1Xi*PZS1Zt-PZS1Xi*PXS1Zt)
      PYS2PE(IL,k)= k2*(PXS2Xi*PZS2Zt-PZS2Xi*PXS2Zt)
      PZS1PE(IL,k)=-k1*(PXS1Xi*PYS1Zt-PYS1Xi*PXS1Zt)
      PZS2PE(IL,k)=-k2*(PXS2Xi*PYS2Zt-PYS2Xi*PXS2Zt)

      DO 60 i=2,IL-1
        PXS1Xi=(XS(1,k,i+1)-XS(1,k,i-1))/2/XiStep
        PXS2Xi=(XS(2,k,i+1)-XS(2,k,i-1))/2/XiStep
        PYS1Xi=(YS(1,k,i+1)-YS(1,k,i-1))/2/XiStep
        PYS2Xi=(YS(2,k,i+1)-YS(2,k,i-1))/2/XiStep
        PZS1Xi=(ZS(1,k,i+1)-ZS(1,k,i-1))/2/XiStep
        PZS2Xi=(ZS(2,k,i+1)-ZS(2,k,i-1))/2/XiStep
        PXS1Zt=(XS(1,k+1,i)-XS(1,k-1,i))/2/ZtStep
        PXS2Zt=(XS(2,k+1,i)-XS(2,k-1,i))/2/ZtStep
        PYS1Zt=(YS(1,k+1,i)-YS(1,k-1,i))/2/ZtStep
        PYS2Zt=(YS(2,k+1,i)-YS(2,k-1,i))/2/ZtStep
        PZS1Zt=(ZS(1,k+1,i)-ZS(1,k-1,i))/2/ZtStep
        PZS2Zt=(ZS(2,k+1,i)-ZS(2,k-1,i))/2/ZtStep
        PXS1PE(i,k)=-k1*(PYS1Xi*PZS1Zt-PZS1Xi*PYS1Zt)
        PXS2PE(i,k)=-k2*(PYS2Xi*PZS2Zt-PZS2Xi*PYS2Zt)
        PYS1PE(i,k)= k1*(PXS1Xi*PZS1Zt-PZS1Xi*PXS1Zt)
        PYS2PE(i,k)= k2*(PXS2Xi*PZS2Zt-PZS2Xi*PXS2Zt)
        PZS1PE(i,k)=-k1*(PXS1Xi*PYS1Zt-PYS1Xi*PXS1Zt)
        PZS2PE(i,k)=-k2*(PXS2Xi*PYS2Zt-PYS2Xi*PXS2Zt)
60      CONTINUE
70    CONTINUE

      PXS1Xi=(XS(1,KL,2)-XS(1,KL,1))/XiStep
      PXS2Xi=(XS(2,KL,2)-XS(2,KL,1))/XiStep
      PYS1Xi=(YS(1,KL,2)-YS(1,KL,1))/XiStep
      PYS2Xi=(YS(2,KL,2)-YS(2,KL,1))/XiStep
      PZS1Xi=(ZS(1,KL,2)-ZS(1,KL,1))/XiStep
      PZS2Xi=(ZS(2,KL,2)-ZS(2,KL,1))/XiStep
      PXS1Zt=(XS(1,KL,1)-XS(1,KL-1,1))/ZtStep
      PXS2Zt=(XS(2,KL,1)-XS(2,KL-1,1))/ZtStep
      PYS1Zt=(YS(1,KL,1)-YS(1,KL-1,1))/ZtStep
      PYS2Zt=(YS(2,KL,1)-YS(2,KL-1,1))/ZtStep
      PZS1Zt=(ZS(1,KL,1)-ZS(1,KL-1,1))/ZtStep
```

```
      PZS2Zt=(ZS(2,KL,1)-ZS(2,KL-1,1))/ZtStep
      PXS1PE(1,KL)=-k1*(PYS1Xi*PZS1Zt-PZS1Xi*PYS1Zt)
      PXS2PE(1,KL)=-k2*(PYS2Xi*PZS2Zt-PZS2Xi*PYS2Zt)
      PYS1PE(1,KL)= k1*(PXS1Xi*PZS1Zt-PZS1Xi*PXS1Zt)
      PYS2PE(1,KL)= k2*(PXS2Xi*PZS2Zt-PZS2Xi*PXS2Zt)
      PZS1PE(1,KL)=-k1*(PXS1Xi*PYS1Zt-PYS1Xi*PXS1Zt)
      PZS2PE(1,KL)=-k2*(PXS2Xi*PYS2Zt-PYS2Xi*PXS2Zt)

      PXS1Xi=(XS(1,KL,IL)-XS(1,KL,IL-1))/XiStep
      PXS2Xi=(XS(2,KL,IL)-XS(2,KL,IL-1))/XiStep
      PYS1Xi=(YS(1,KL,IL)-YS(1,KL,IL-1))/XiStep
      PYS2Xi=(YS(2,KL,IL)-YS(2,KL,IL-1))/XiStep
      PZS1Xi=(ZS(1,KL,IL)-ZS(1,KL,IL-1))/XiStep
      PZS2Xi=(ZS(2,KL,IL)-ZS(2,KL,IL-1))/XiStep
      PXS1Zt=(XS(1,KL,IL)-XS(1,KL-1,IL))/ZtStep
      PXS2Zt=(XS(2,KL,IL)-XS(2,KL-1,IL))/ZtStep
      PYS1Zt=(YS(1,KL,IL)-YS(1,KL-1,IL))/ZtStep
      PYS2Zt=(YS(2,KL,IL)-YS(2,KL-1,IL))/ZtStep
      PZS1Zt=(ZS(1,KL,IL)-ZS(1,KL-1,IL))/ZtStep
      PZS2Zt=(ZS(2,KL,IL)-ZS(2,KL-1,IL))/ZtStep
      PXS1PE(IL,KL)=-k1*(PYS1Xi*PZS1Zt-PZS1Xi*PYS1Zt)
      PXS2PE(IL,KL)=-k2*(PYS2Xi*PZS2Zt-PZS2Xi*PYS2Zt)
      PYS1PE(IL,KL)= k1*(PXS1Xi*PZS1Zt-PZS1Xi*PXS1Zt)
      PYS2PE(IL,KL)= k2*(PXS2Xi*PZS2Zt-PZS2Xi*PXS2Zt)
      PZS1PE(IL,KL)=-k1*(PXS1Xi*PYS1Zt-PYS1Xi*PXS1Zt)
      PZS2PE(IL,KL)=-k2*(PXS2Xi*PYS2Zt-PYS2Xi*PXS2Zt)

      DO 75 i=2,IL-1
        PXS1Xi=(XS(1,KL,i+1)-XS(1,KL,i-1))/2/XiStep
        PXS2Xi=(XS(2,KL,i+1)-XS(2,KL,i-1))/2/XiStep
        PYS1Xi=(YS(1,KL,i+1)-YS(1,KL,i-1))/2/XiStep
        PYS2Xi=(YS(2,KL,i+1)-YS(2,KL,i-1))/2/XiStep
        PZS1Xi=(ZS(1,KL,i+1)-ZS(1,KL,i-1))/2/XiStep
        PZS2Xi=(ZS(2,KL,i+1)-ZS(2,KL,i-1))/2/XiStep
        PXS1Zt=(XS(1,KL,i)-XS(1,KL-1,i))/ZtStep
        PXS2Zt=(XS(2,KL,i)-XS(2,KL-1,i))/ZtStep
        PYS1Zt=(YS(1,KL,i)-YS(1,KL-1,i))/ZtStep
        PYS2Zt=(YS(2,KL,i)-YS(2,KL-1,i))/ZtStep
        PZS1Zt=(ZS(1,KL,i)-ZS(1,KL-1,i))/ZtStep
        PZS2Zt=(ZS(2,KL,i)-ZS(2,KL-1,i))/ZtStep
        PXS1PE(i,KL)=-k1*(PYS1Xi*PZS1Zt-PZS1Xi*PYS1Zt)
        PXS2PE(i,KL)=-k2*(PYS2Xi*PZS2Zt-PZS2Xi*PYS2Zt)
        PYS1PE(i,KL)= k1*(PXS1Xi*PZS1Zt-PZS1Xi*PXS1Zt)
        PYS2PE(i,KL)= k2*(PXS2Xi*PZS2Zt-PZS2Xi*PXS2Zt)
        PZS1PE(i,KL)=-k1*(PXS1Xi*PYS1Zt-PYS1Xi*PXS1Zt)
        PZS2PE(i,KL)=-k2*(PXS2Xi*PYS2Zt-PYS2Xi*PXS2Zt)
 75   CONTINUE

C Calculate the interior grid point locations.

      DO 100 k=1,KL
        DO 90 i=1,IL
          DO 80 j=1,JL
            XPnt(i,j,k)=h1(j)
     $             *XS(1,k,i)+h2(j)*XS(2,k,i)
```

```
     $              +h3(j)*PXS1PE(i,k)
     $              +h4(j)*PXS2PE(i,k)
          YPnt(i,j,k)=h1(j)
     $              *YS(1,k,i)+h2(j)*YS(2,k,i)
     $              +h3(j)*PYS1PE(i,k)
     $              +h4(j)*PYS2PE(i,k)
          ZPnt(i,j,k)=h1(j)
     $              *ZS(1,k,i)+h2(j)*ZS(2,k,i)
     $              +h3(j)*PZS1PE(i,k)
     $              +h4(j)*PZS2PE(i,k)
 80       CONTINUE
 90     CONTINUE
 100  CONTINUE

      RETURN
      END

C==================================================================C

      SUBROUTINE ForSrf(XPnt,YPnt,ZPnt,IL,JL,KL,k3,k4,BetaXi,BetaEt,
     $            BetaZt,XS,YS,ZS,XiStep,EtStep,ZtStep,
     $            h1,h2,h3,h4,h5,h6,h7,h8,
     $            PXS1PE,PXS2PE,PYS1PE,PYS2PE,PZS1PE,PZS2PE,
     $            PXS3Zt,PXS4Zt,PYS3Zt,PYS4Zt,PZS3Zt,PZS4Zt,
     $            StrXi,StrEt,StrZt,MxGSiz,MxSrfs)


C This procedure adjusts the grid so that the other two surfaces of the
C region are mapped correctly using the "four-boundary technique".

      INTEGER  i, j, k, StrXi, StrEt, StrZt, IL, JL, KL

      REAL  Xi, Eta, Zeta, XiNew, EtaNew, ZtaNew,
     $   h1(MxGSiz), h2(MxGSiz), h3(MxGSiz), h4(MxGSiz),
     $   h5(MxGSiz), h6(MxGSiz), h7(MxGSiz), h8(MxGSiz),
     $   PXS3Xi,  PXS4Xi, PYS3Xi,  PYS4Xi, PZS3Xi,  PZS4Xi,
     $   PXS3PE, PXS4PE, PYS3PE, PYS4PE, PZS3PE, PZS4PE,
     $   P2X00, P2X01, P2X10, P2X11, P2Y00, P2Y01, P2Y10, P2Y11,
     $   P2Z00, P2Z01, P2Z10, P2Z11
      REAL  k3, k4, BetaXi, BetaEt, BetaZt, XiStep, EtStep, ZtStep,
     $   PXS1PE(MxGSiz,MxGsiz), PXS2PE(MxGSiz,MxGsiz),
     $   PXS3Zt(MxGSiz,MxGsiz), PXS4Zt(MxGSiz,MxGsiz),
     $   PYS1PE(MxGSiz,MxGsiz), PYS2PE(MxGSiz,MxGsiz),
     $   PYS3Zt(MxGSiz,MxGsiz), PYS4Zt(MxGSiz,MxGsiz),
     $   PZS1PE(MxGSiz,MxGsiz), PZS2PE(MxGSiz,MxGsiz),
     $   PZS3Zt(MxGSiz,MxGsiz), PZS4Zt(MxGSiz,MxGsiz),
     $   XS(MxSrfs,MxGSiz,MxGsiz),
     $   YS(MxSrfs,MxGSiz,MxGsiz),
     $   ZS(MxSrfs,MxGSiz,MxGsiz),
     $   XPnt(MxGSiz,MxGsiz,MxGSiz),
     $   YPnt(MxGSiz,MxGsiz,MxGSiz),
     $   ZPnt(MxGSiz,MxGsiz,MxGSiz)

C Calculate the h factors for the boundary surface 3-4 Hermite
C adjusting curves.
```

```
      Zeta=0.0

      DO 40 k=1,KL
        CALL FAlNew(ZtaNew,Zeta,BetaZt,StrZt)
        CALL FindHs(h5(k),h6(k),h7(k),h8(k),ZtaNew)
        Zeta=Zeta+ZtStep
 40   CONTINUE

C Calculate the derivative values along the constant Xi/Eta
C boundaries.

      PXS3Xi=(XS(3,2,1)-XS(3,1,1))/XiStep
      PXS4Xi=(XS(4,2,1)-XS(4,1,1))/XiStep
      PYS3Xi=(YS(3,2,1)-YS(3,1,1))/XiStep
      PYS4Xi=(YS(4,2,1)-YS(4,1,1))/XiStep
      PZS3Xi=(ZS(3,2,1)-ZS(3,1,1))/XiStep
      PZS4Xi=(ZS(4,2,1)-ZS(4,1,1))/XiStep
      PXS3PE=(XS(3,1,2)-XS(3,1,1))/EtStep
      PXS4PE=(XS(4,1,2)-XS(4,1,1))/EtStep
      PYS3PE=(YS(3,1,2)-YS(3,1,1))/EtStep
      PYS4PE=(YS(4,1,2)-YS(4,1,1))/EtStep
      PZS3PE=(ZS(3,1,2)-ZS(3,1,1))/EtStep
      PZS4PE=(ZS(4,1,2)-ZS(4,1,1))/EtStep
      PXS3Zt(1,1)=-k3*(PYS3Xi*PZS3PE-PZS3Xi*PYS3PE)
      PXS4Zt(1,1)=-k4*(PYS4Xi*PZS4PE-PZS4Xi*PYS4PE)
      PYS3Zt(1,1)= k3*(PXS3Xi*PZS3PE-PZS3Xi*PXS3PE)
      PYS4Zt(1,1)= k4*(PXS4Xi*PZS4PE-PZS4Xi*PXS4PE)
      PZS3Zt(1,1)=-k3*(PXS3Xi*PYS3PE-PYS3Xi*PXS3PE)
      PZS4Zt(1,1)=-k4*(PXS4Xi*PYS4PE-PYS4Xi*PXS4PE)

      PXS3Xi=(XS(3,IL,1)-XS(3,IL-1,1))/XiStep
      PXS4Xi=(XS(4,IL,1)-XS(4,IL-1,1))/XiStep
      PYS3Xi=(YS(3,IL,1)-YS(3,IL-1,1))/XiStep
      PYS4Xi=(YS(4,IL,1)-YS(4,IL-1,1))/XiStep
      PZS3Xi=(ZS(3,IL,1)-ZS(3,IL-1,1))/XiStep
      PZS4Xi=(ZS(4,IL,1)-ZS(4,IL-1,1))/XiStep
      PXS3PE=(XS(3,IL,2)-XS(3,IL,1))/EtStep
      PXS4PE=(XS(4,IL,2)-XS(4,IL,1))/EtStep
      PYS3PE=(YS(3,IL,2)-YS(3,IL,1))/EtStep
      PYS4PE=(YS(4,IL,2)-YS(4,IL,1))/EtStep
      PZS3PE=(ZS(3,IL,2)-ZS(3,IL,1))/EtStep
      PZS4PE=(ZS(4,IL,2)-ZS(4,IL,1))/EtStep
      PXS3Zt(IL,1)=-k3*(PYS3Xi*PZS3PE-PZS3Xi*PYS3PE)
      PXS4Zt(IL,1)=-k4*(PYS4Xi*PZS4PE-PZS4Xi*PYS4PE)
      PYS3Zt(IL,1)= k3*(PXS3Xi*PZS3PE-PZS3Xi*PXS3PE)
      PYS4Zt(IL,1)= k4*(PXS4Xi*PZS4PE-PZS4Xi*PXS4PE)
      PZS3Zt(IL,1)=-k3*(PXS3Xi*PYS3PE-PYS3Xi*PXS3PE)
      PZS4Zt(IL,1)=-k4*(PXS4Xi*PYS4PE-PYS4Xi*PXS4PE)

      DO 45 i=2,IL-1
        PXS3Xi=(XS(3,i+1,1)-XS(3,i-1,1))/2/XiStep
        PXS4Xi=(XS(4,i+1,1)-XS(4,i-1,1))/2/XiStep
        PYS3Xi=(YS(3,i+1,1)-YS(3,i-1,1))/2/XiStep
        PYS4Xi=(YS(4,i+1,1)-YS(4,i-1,1))/2/XiStep
```

```
      PZS3Xi=(ZS(3,i+1,1)-ZS(3,i-1,1))/2/XiStep
      PZS4Xi=(ZS(4,i+1,1)-ZS(4,i-1,1))/2/XiStep
      PXS3PE=(XS(3,i,2)-XS(3,i,1))/EtStep
      PXS4PE=(XS(4,i,2)-XS(4,i,1))/EtStep
      PYS3PE=(YS(3,i,2)-YS(3,i,1))/EtStep
      PYS4PE=(YS(4,i,2)-YS(4,i,1))/EtStep
      PZS3PE=(ZS(3,i,2)-ZS(3,i,1))/EtStep
      PZS4PE=(ZS(4,i,2)-ZS(4,i,1))/EtStep
      PXS3Zt(i,1)=-k3*(PYS3Xi*PZS3PE-PZS3Xi*PYS3PE)
      PXS4Zt(i,1)=-k4*(PYS4Xi*PZS4PE-PZS4Xi*PYS4PE)
      PYS3Zt(i,1)= k3*(PXS3Xi*PZS3PE-PZS3Xi*PXS3PE)
      PYS4Zt(i,1)= k4*(PXS4Xi*PZS4PE-PZS4Xi*PXS4PE)
      PZS3Zt(i,1)=-k3*(PXS3Xi*PYS3PE-PYS3Xi*PXS3PE)
      PZS4Zt(i,1)=-k4*(PXS4Xi*PYS4PE-PYS4Xi*PXS4PE)
45   CONTINUE

     DO 60 j=2,JL-1
      PXS3Xi=(XS(3,2,j)-XS(3,1,j))/XiStep
      PXS4Xi=(XS(4,2,j)-XS(4,1,j))/XiStep
      PYS3Xi=(YS(3,2,j)-YS(3,1,j))/XiStep
      PYS4Xi=(YS(4,2,j)-YS(4,1,j))/XiStep
      PZS3Xi=(ZS(3,2,j)-ZS(3,1,j))/XiStep
      PZS4Xi=(ZS(4,2,j)-ZS(4,1,j))/XiStep
      PXS3PE=(XS(3,1,j+1)-XS(3,1,j-1))/2/EtStep
      PXS4PE=(XS(4,1,j+1)-XS(4,1,j-1))/2/EtStep
      PYS3PE=(YS(3,1,j+1)-YS(3,1,j-1))/2/EtStep
      PYS4PE=(YS(4,1,j+1)-YS(4,1,j-1))/2/EtStep
      PZS3PE=(ZS(3,1,j+1)-ZS(3,1,j-1))/2/EtStep
      PZS4PE=(ZS(4,1,j+1)-ZS(4,1,j-1))/2/EtStep
      PXS3Zt(1,j)=-k3*(PYS3Xi*PZS3PE-PZS3Xi*PYS3PE)
      PXS4Zt(1,j)=-k4*(PYS4Xi*PZS4PE-PZS4Xi*PYS4PE)
      PYS3Zt(1,j)= k3*(PXS3Xi*PZS3PE-PZS3Xi*PXS3PE)
      PYS4Zt(1,j)= k4*(PXS4Xi*PZS4PE-PZS4Xi*PXS4PE)
      PZS3Zt(1,j)=-k3*(PXS3Xi*PYS3PE-PYS3Xi*PXS3PE)
      PZS4Zt(1,j)=-k4*(PXS4Xi*PYS4PE-PYS4Xi*PXS4PE)

      PXS3Xi=(XS(3,IL,j)-XS(3,IL-1,j))/XiStep
      PXS4Xi=(XS(4,IL,j)-XS(4,IL-1,j))/XiStep
      PYS3Xi=(YS(3,IL,j)-YS(3,IL-1,j))/XiStep
      PYS4Xi=(YS(4,IL,j)-YS(4,IL-1,j))/XiStep
      PZS3Xi=(ZS(3,IL,j)-ZS(3,IL-1,j))/XiStep
      PZS4Xi=(ZS(4,IL,j)-ZS(4,IL-1,j))/XiStep
      PXS3PE=(XS(3,IL,j+1)-XS(3,IL,j-1))/2/EtStep
      PXS4PE=(XS(4,IL,j+1)-XS(4,IL,j-1))/2/EtStep
      PYS3PE=(YS(3,IL,j+1)-YS(3,IL,j-1))/2/EtStep
      PYS4PE=(YS(4,IL,j+1)-YS(4,IL,j-1))/2/EtStep
      PZS3PE=(ZS(3,IL,j+1)-ZS(3,IL,j-1))/2/EtStep
      PZS4PE=(ZS(4,IL,j+1)-ZS(4,IL,j-1))/2/EtStep
      PXS3Zt(IL,j)=-k3*(PYS3Xi*PZS3PE-PZS3Xi*PYS3PE)
      PXS4Zt(IL,j)=-k4*(PYS4Xi*PZS4PE-PZS4Xi*PYS4PE)
      PYS3Zt(IL,j)= k3*(PXS3Xi*PZS3PE-PZS3Xi*PXS3PE)
      PYS4Zt(IL,j)= k4*(PXS4Xi*PZS4PE-PZS4Xi*PXS4PE)
      PZS3Zt(IL,j)=-k3*(PXS3Xi*PYS3PE-PYS3Xi*PXS3PE)
      PZS4Zt(IL,j)=-k4*(PXS4Xi*PYS4PE-PYS4Xi*PXS4PE)
```

```
      DO 50 i=2,IL-1
        PXS3Xi=(XS(3,i+1,j)-XS(3,i-1,j))/2/XiStep
        PXS4Xi=(XS(4,i+1,j)-XS(4,i-1,j))/2/XiStep
        PYS3Xi=(YS(3,i+1,j)-YS(3,i-1,j))/2/XiStep
        PYS4Xi=(YS(4,i+1,j)-YS(4,i-1,j))/2/XiStep
        PZS3Xi=(ZS(3,i+1,j)-ZS(3,i-1,j))/2/XiStep
        PZS4Xi=(ZS(4,i+1,j)-ZS(4,i-1,j))/2/XiStep
        PXS3PE=(XS(3,i,j+1)-XS(3,i,j-1))/2/EtStep
        PXS4PE=(XS(4,i,j+1)-XS(4,i,j-1))/2/EtStep
        PYS3PE=(YS(3,i,j+1)-YS(3,i,j-1))/2/EtStep
        PYS4PE=(YS(4,i,j+1)-YS(4,i,j-1))/2/EtStep
        PZS3PE=(ZS(3,i,j+1)-ZS(3,i,j-1))/2/EtStep
        PZS4PE=(ZS(4,i,j+1)-ZS(4,i,j-1))/2/EtStep
        PXS3Zt(i,j)=-k3*(PYS3Xi*PZS3PE-PZS3Xi*PYS3PE)
        PXS4Zt(i,j)=-k4*(PYS4Xi*PZS4PE-PZS4Xi*PYS4PE)
        PYS3Zt(i,j)= k3*(PXS3Xi*PZS3PE-PZS3Xi*PXS3PE)
        PYS4Zt(i,j)= k4*(PXS4Xi*PZS4PE-PZS4Xi*PXS4PE)
        PZS3Zt(i,j)=-k3*(PXS3Xi*PYS3PE-PYS3Xi*PXS3PE)
        PZS4Zt(i,j)=-k4*(PXS4Xi*PYS4PE-PYS4Xi*PXS4PE)
50      CONTINUE
60    CONTINUE

      PXS3Xi=(XS(3,2,JL)-XS(3,1,JL))/XiStep
      PXS4Xi=(XS(4,2,JL)-XS(4,1,JL))/XiStep
      PYS3Xi=(YS(3,2,JL)-YS(3,1,JL))/XiStep
      PYS4Xi=(YS(4,2,JL)-YS(4,1,JL))/XiStep
      PZS3Xi=(ZS(3,2,JL)-ZS(3,1,JL))/XiStep
      PZS4Xi=(ZS(4,2,JL)-ZS(4,1,JL))/XiStep
      PXS3PE=(XS(3,1,JL)-XS(3,1,JL-1))/EtStep
      PXS4PE=(XS(4,1,JL)-XS(4,1,JL-1))/EtStep
      PYS3PE=(YS(3,1,JL)-YS(3,1,JL-1))/EtStep
      PYS4PE=(YS(4,1,JL)-YS(4,1,JL-1))/EtStep
      PZS3PE=(ZS(3,1,JL)-ZS(3,1,JL-1))/EtStep
      PZS4PE=(ZS(4,1,JL)-ZS(4,1,JL-1))/EtStep
      PXS3Zt(1,JL)=-k3*(PYS3Xi*PZS3PE-PZS3Xi*PYS3PE)
      PXS4Zt(1,JL)=-k4*(PYS4Xi*PZS4PE-PZS4Xi*PYS4PE)
      PYS3Zt(1,JL)= k3*(PXS3Xi*PZS3PE-PZS3Xi*PXS3PE)
      PYS4Zt(1,JL)= k4*(PXS4Xi*PZS4PE-PZS4Xi*PXS4PE)
      PZS3Zt(1,JL)=-k3*(PXS3Xi*PYS3PE-PYS3Xi*PXS3PE)
      PZS4Zt(1,JL)=-k4*(PXS4Xi*PYS4PE-PYS4Xi*PXS4PE)

      PXS3Xi=(XS(3,IL,JL)-XS(3,IL-1,JL))/XiStep
      PXS4Xi=(XS(4,IL,JL)-XS(4,IL-1,JL))/XiStep
      PYS3Xi=(YS(3,IL,JL)-YS(3,IL-1,JL))/XiStep
      PYS4Xi=(YS(4,IL,JL)-YS(4,IL-1,JL))/XiStep
      PZS3Xi=(ZS(3,IL,JL)-ZS(3,IL-1,JL))/XiStep
      PZS4Xi=(ZS(4,IL,JL)-ZS(4,IL-1,JL))/XiStep
      PXS3PE=(XS(3,IL,JL)-XS(3,IL,JL-1))/EtStep
      PXS4PE=(XS(4,IL,JL)-XS(4,IL,JL-1))/EtStep
      PYS3PE=(YS(3,IL,JL)-YS(3,IL,JL-1))/EtStep
      PYS4PE=(YS(4,IL,JL)-YS(4,IL,JL-1))/EtStep
      PZS3PE=(ZS(3,IL,JL)-ZS(3,IL,JL-1))/EtStep
      PZS4PE=(ZS(4,IL,JL)-ZS(4,IL,JL-1))/EtStep
      PXS3Zt(IL,JL)=-k3*(PYS3Xi*PZS3PE-PZS3Xi*PYS3PE)
      PXS4Zt(IL,JL)=-k4*(PYS4Xi*PZS4PE-PZS4Xi*PYS4PE)
```

```
      PYS3Zt(IL,JL)= k3*(PXS3Xi*PZS3PE-PZS3Xi*PXS3PE)
      PYS4Zt(IL,JL)= k4*(PXS4Xi*PZS4PE-PZS4Xi*PXS4PE)
      PZS3Zt(IL,JL)=-k3*(PXS3Xi*PYS3PE-PYS3Xi*PXS3PE)
      PZS4Zt(IL,JL)=-k4*(PXS4Xi*PYS4PE-PYS4Xi*PXS4PE)

      DO 65 i=2,IL-1
        PXS3Xi=(XS(3,i+1,JL)-XS(3,i-1,JL))/2/XiStep
        PXS4Xi=(XS(4,i+1,JL)-XS(4,i-1,JL))/2/XiStep
        PYS3Xi=(YS(3,i+1,JL)-YS(3,i-1,JL))/2/XiStep
        PYS4Xi=(YS(4,i+1,JL)-YS(4,i-1,JL))/2/XiStep
        PZS3Xi=(ZS(3,i+1,JL)-ZS(3,i-1,JL))/2/XiStep
        PZS4Xi=(ZS(4,i+1,JL)-ZS(4,i-1,JL))/2/XiStep
        PXS3PE=(XS(3,i,JL)-XS(3,i,JL-1))/EtStep
        PXS4PE=(XS(4,i,JL)-XS(4,i,JL-1))/EtStep
        PYS3PE=(YS(3,i,JL)-YS(3,i,JL-1))/EtStep
        PYS4PE=(YS(4,i,JL)-YS(4,i,JL-1))/EtStep
        PZS3PE=(ZS(3,i,JL)-ZS(3,i,JL-1))/EtStep
        PXS3Zt(i,JL)=-k3*(PYS3Xi*PZS3PE-PZS3Xi*PYS3PE)
        PXS4Zt(i,JL)=-k4*(PYS4Xi*PZS4PE-PZS4Xi*PYS4PE)
        PYS3Zt(i,JL)= k3*(PXS3Xi*PZS3PE-PZS3Xi*PXS3PE)
        PYS4Zt(i,JL)= k4*(PXS4Xi*PZS4PE-PZS4Xi*PXS4PE)
        PZS3Zt(i,JL)=-k3*(PXS3Xi*PYS3PE-PYS3Xi*PXS3PE)
        PZS4Zt(i,JL)=-k4*(PXS4Xi*PYS4PE-PYS4Xi*PXS4PE)
 65   CONTINUE

      P2X00=0.0
      P2X10=0.0
      P2X01=0.0
      P2X11=0.0
      P2Y00=0.0
      P2Y10=0.0
      P2Y01=0.0
      P2Y11=0.0
      P2Z00=0.0
      P2Z10=0.0
      P2Z01=0.0
      P2Z11=0.0

C Calculate the grid point locations everywhere.

      DO 90 k=1,KL
        DO 80 i=1,IL
          DO 70 j=1,JL
            XPnt(i,j,k)=XPnt(i,j,k)
     $          +(XS(3,i,j)-h1(j)*XS(1,1,i)
     $                  -h2(j)*XS(2,1,i)
     $                  -h3(j)*PXS1PE(i,1)
     $                  -h4(j)*PXS2PE(i,1))*h5(k)
     $          +(XS(4,i,j)-h1(j)*XS(1,KL,i)
     $                  -h2(j)*XS(2,KL,i)
     $                  -h3(j)*PXS1PE(i,KL)
     $                  -h4(j)*PXS2PE(i,KL))*h6(k)
     $          +(PXS3Zt(i,j)-(h1(j)*PXS3Zt(i,1)
     $                      +h2(j)*PXS3Zt(i,JL)
     $                      +h3(j)*P2X00+h4(j)*P2X01))*h7(k)
```

```
     $           +(PXS4Zt(i,j)-(h1(j)*PXS4Zt(i,1)
     $                       +h2(j)*PXS4Zt(i,JL)
     $                       +h3(j)*P2X10+h4(j)*P2X11))*h8(k)
            YPnt(i,j,k)=YPnt(i,j,k)
     $           +(YS(3,i,j)-h1(j)*YS(1,1,i)
     $                    -h2(j)*YS(2,1,i)
     $                    -h3(j)*PYS1PE(i,1)
     $                    -h4(j)*PYS2PE(i,1))*h5(k)
     $           +(YS(4,i,j)-h1(j)*YS(1,KL,i)
     $                    -h2(j)*YS(2,KL,i)
     $                    -h3(j)*PYS1PE(i,KL)
     $                    -h4(j)*PYS2PE(i,KL))*h6(k)
     $           +(PYS3Zt(i,j)-(h1(j)*PYS3Zt(i,1)
     $                       +h2(j)*PYS3Zt(i,JL)
     $                       +h3(j)*P2Y00+h4(j)*P2Y01))*h7(k)
     $           +(PYS4Zt(i,j)-(h1(j)*PYS4Zt(i,1)
     $                       +h2(j)*PYS4Zt(i,JL)
     $                       +h3(j)*P2Y10+h4(j)*P2Y11))*h8(k)
            ZPnt(i,j,k)=ZPnt(i,j,k)
     $           +(ZS(3,i,j)-h1(j)*ZS(1,1,i)
     $                    -h2(j)*ZS(2,1,i)
     $                    -h3(j)*PZS1PE(i,1)
     $                    -h4(j)*PZS2PE(i,1))*h5(k)
     $           +(ZS(4,i,j)-h1(j)*ZS(1,KL,i)
     $                    -h2(j)*ZS(2,KL,i)
     $                    -h3(j)*PZS1PE(i,KL)
     $                    -h4(j)*PZS2PE(i,KL))*h6(k)
     $           +(PZS3Zt(i,j)-(h1(j)*PZS3Zt(i,1)
     $                       +h2(j)*PZS3Zt(i,JL)
     $                       +h3(j)*P2Z00+h4(j)*P2Z01))*h7(k)
     $           +(PZS4Zt(i,j)-(h1(j)*PZS4Zt(i,1)
     $                       +h2(j)*PZS4Zt(i,JL)
     $                       +h3(j)*P2Z10+h4(j)*P2Z11))*h8(k)
 70         CONTINUE
 80       CONTINUE
 90     CONTINUE

      RETURN
      END

C=========================================================C

      SUBROUTINE PrGrid (XPnt,YPnt,ZPnt,IL,JL,KL,MxGSiz)

C This procedure prints (to output) the grid point x, y, and z coordinates.

      INTEGER  i, j, k, IL, JL, KL


      REAL  XPnt(MxGSiz,MxGsiz,MxGSiz),
     $      YPnt(MxGSiz,MxGsiz,MxGSiz),
     $      ZPnt(MxGSiz,MxGsiz,MxGSiz)

      WRITE(8,*) IL
      WRITE(8,*) JL
```

```
      WRITE(8,*) KL

      DO 30 i=1,IL
        DO 20 j=1,JL
          DO 10 k=1,KL
            WRITE(8,35) XPnt(i,j,k),YPnt(i,j,k),ZPnt(i,j,k)
10        CONTINUE
20      CONTINUE
30    CONTINUE

35    FORMAT(1X,F10.6,3x,F10.6,3X,F10.6)

      RETURN
      END

C==================================================C

      SUBROUTINE RdGrIn(IL,JL,KL,StrXi,StrEt,StrZt,NSurfs,kXi1,kXi2,
     $            kEta1,kEta2,kZeta1,kZeta2,BetaXi,BetaEt,BetaZt,
     $            XiStep,EtStep,ZtStep)

C This procedure reads in the desired grid information for grid control.

      INTEGER  IL, JL, KL, StrXi, StrEt, StrZt

      REAL  kXi1, kXi2, kEta1, kEta2,  kZeta1, kZeta2,
     $      BetaXi, BetaEt, BetaZt, XiStep, EtStep, ZtStep

      READ(7,*) NSurfs

      READ(7,*) IL
      READ(7,*) JL
      READ(7,*) KL

      READ(7,*) StrXi
      READ(7,*) StrEt
      READ(7,*) StrZt

      READ(7,*) kXi1
      READ(7,*) kXi2
      READ(7,*) kEta1
      READ(7,*) kEta2
      READ(7,*) kZeta1
      READ(7,*) kZeta2

      READ(7,*) BetaXi
      READ(7,*) BetaEt
      READ(7,*) BetaZt

      XiStep=1.0/(IL-1)
      EtStep=1.0/(JL-1)
      ZtStep=1.0/(KL-1)

      RETURN
      END
```

```
C=============================================================C

      SUBROUTINE RdCvIn (x,y,z,NDPts,CrvNum,Tensn,MxBPts)

C This SUBROUTINE reads in the information concerning discrete points on
C the boundaries.  This information is used for generating spline-fitted
C boundary approximation curves.

      INTEGER  CrvNum, i, NDPts(4)

      REAL  x(4,MxBPts), y(4,MxBPts),
     $      z(4,MxBPts), Tensn(4)

      READ(7,*) Tensn(CrvNum)
      READ(7,*) NDPts(CrvNum)

      DO 10 i=1,NDPts(CrvNum)
        READ(7,*) x(CrvNum,i), y(CrvNum,i) ,z(CrvNum,i)
 10   CONTINUE

      RETURN
      END
C=============================================================C

      SUBROUTINE CalcS (x,y,z,s,NDPts,CrvNum,MxBPts)

C This SUBROUTINE calculates the spline parameter, s, as an approximate
C arc length.

      INTEGER  NDPts(4), CrvNum, i

      REAL  x(4,MxBPts), y(4,MxBPts),
     $      z(4,MxBPts), s(4,MxBPts)

      s(CrvNum,1)=0.0

      DO 10 i=2,NDPts(CrvNum)
       s(CrvNum,i)=s(CrvNum,i-1)
     $               +SQRT( (x(CrvNum,i)-x(CrvNum,i-1))**2
     $                   +(y(CrvNum,i)-y(CrvNum,i-1))**2
     $                   +(z(CrvNum,i)-z(CrvNum,i-1))**2)
 10   CONTINUE

      RETURN
      END

C=============================================================C

      SUBROUTINE SplMat (Diag,OfDiag,Right,w,s,NDPts,T,CrvNum,MxBPts)

C This SUBROUTINE forms the parametric tension spline matrix for a
C particular boundary curve data set.
```

```
      INTEGER  i, NDPts(4), CrvNum

      REAL Diag(MxBPts), OfDiag(MxBPts), Right(MxBPts),
     $     w(4,MxBPts), s(4,MxBPts), T, h, hm

      Diag(1)=1.0
      OfDiag(1)=0.0
      Right(1)=0.0

      DO 10 i=2,NDPts(CrvNum)-1
        h=s(CrvNum,i+1)-s(CrvNum,i)
        hm=s(CrvNum,i)-s(CrvNum,i-1)
        Diag(i)=(T*COSH(T*hm)/SINH(T*hm)-1/hm+T*COSH(T*h)/SINH(T*h)
     $          -1/h)/T**2
        OfDiag(i)=(1/h-T/SINH(T*h))/T**2
        Right(i)= (w(CrvNum,i+1)-w(CrvNum,i))/h
     $            -(w(CrvNum,i)-w(CrvNum,i-1))/hm
 10   CONTINUE

      Diag(NDPts(CrvNum))=1.0
      OfDiag(NDPts(CrvNum)-1)=0.0
      Right(NDPts(CrvNum))=0.0

      RETURN
      END

C================================================================C

      SUBROUTINE SplSlv (Diag,OfDiag,Right,Deriv2,NDPts,CrvNum,MxBPts)

C This SUBROUTINE solves the diagonally dominant parametric tension
C spline matrix for a given data set using the Gauss-Seidel iteration.
C Convergence is assumed after 20 iterations.

      INTEGER  i, j, NDPts(4), CrvNum

      REAL Diag(MxBPts), OfDiag(MxBPts), Right(MxBPts),
     $     Deriv2(4,MxBPts)

C Initialize the second derivative matrix to all zeroes.

      DO 10 i=1,NDPts(CrvNum)
        Deriv2(CrvNum,i)=0.0
 10   CONTINUE
```

```
C Calculate the second derivative values using 20 iterations of
C the Gauss-Seidel method.

      DO 30 j=1,20
        DO 20 i=2,NDPts(CrvNum)-1
          Deriv2(CrvNum,i)=(Right(i)-OfDiag(i)*Deriv2(CrvNum,i+1)
     $                     -OfDiag(i-1)*Deriv2(CrvNum,i-1))
     $                     /Diag(i)
 20     CONTINUE
 30   CONTINUE

      RETURN
      END

C==============================================================C

      FUNCTION SplVal (s,w,Deriv2,sval,T,n,CrvNum,MxBPts)

C This real function finds the w-value (x-value or y-value) corresponding
C to a specified s-value using the parametric tension spline curve
C generated for a particular boundary curve data set.

      INTEGER  n, CrvNum

      REAL  s(4,MxBPts), w(4,MxBPts), Deriv2(4,MxBPts),
     $      sval, T, h, Interim, Temp1, Temp2


      Temp1=sval-s(CrvNum,n)
      h=s(CrvNum,n+1)-s(CrvNum,n)
      Temp2=s(CrvNum,n+1)-sval
      Interim=Deriv2(CrvNum,n)/T**2*SINH(T*Temp2)/SINH(T*h)
     $        +(w(CrvNum,n)-Deriv2(CrvNum,n)/T**2)*Temp2/h
      SplVal=Interim+Deriv2(CrvNum,n+1)/T**2*SINH(T*Temp1)
     $             /SINH(T*h)+(w(CrvNum,n+1)
     $             -Deriv2(CrvNum,n+1)/T**2)*Temp1/h

      RETURN
      END

C==============================================================C

      SUBROUTINE PTSpln(x,y,z,s,XDeriv2,YDeriv2,ZDeriv2,Diag,OfDiag,
     $            Right,NDPts,Tensn,CrvNum,MxBPts)

C This SUBROUTINE forms the main routine for the parametric tension
C spline process.

      INTEGER  NDPts(4), CrvNum

      REAL Diag(MxBPts), OfDiag(MxBPts), Right(MxBPts),
     $    XDeriv2(4,MxBPts), YDeriv2(4,MxBPts),
     $    ZDeriv2(4,MxBPts), Tensn,
     $    x(4,MxBPts), y(4,MxBPts),
     $    z(4,MxBPts), s(4,MxBPts)
```

```
      CALL CalcS(x,y,z,s,NDPts,CrvNum,MxBPts)
      CALL SplMat(Diag,OfDiag,Right,x,s,NDPts,Tensn,CrvNum,MxBPts)
      CALL SplSlv(Diag,OfDiag,Right,XDeriv2,NDPts,CrvNum,MxBPts)
      CALL SplMat(Diag,OfDiag,Right,y,s,NDPts,Tensn,CrvNum,MxBPts)
      CALL SplSlv(Diag,OfDiag,Right,YDeriv2,NDPts,CrvNum,MxBPts)
      CALL SplMat(Diag,OfDiag,Right,z,s,NDPts,Tensn,CrvNum,MxBPts)
      CALL SplSlv(Diag,OfDiag,Right,ZDeriv2,NDPts,CrvNum,MxBPts)

      RETURN
      END

C==============================================================C

      SUBROUTINE FindHs(h1,h2,h3,h4,n)

C This SUBROUTINE computes the h factors used in Hermite interpolation.

      REAL  h1, h2, h3, h4, n

      h1= 2*n**3-3*n**2+1
      h2=-2*n**3+3*n**2
      h3= n**3-2*n**2+n
      h4= n**3-n**2

      RETURN
      END

C==============================================================C

      SUBROUTINE SplInt(n,s,SValue,NDPts,CurCrv,MxBPts)

C This SUBROUTINE finds the proper interval in which a point on a specified
C boundary lies.  The interval indicates which initial data points the
C point in question lies between and thus which spline coefficients to use.

      INTEGER  i, n, CurCrv, NDPts(4)

      REAL  Temp, SValue, s(4,MxBPts)

      n=1
      i=NDPts(CurCrv)

 10   IF ((n.EQ.1).AND.(i.GT.1)) THEN
        i=i-1
        Temp=SValue-s(CurCrv,i)

        IF (Temp.GT.0.0) THEN
          n=i
        ENDIF

        GOTO 10
      ENDIF
```

```
      RETURN
      END

C==================================================================C

      SUBROUTINE FAlNew(AlNew,Alpha,B,Str)

C This SUBROUTINE computes the new Alpha value after stretching as
C AlNew. Alpha is a dummy variable representing either Xi, Eta or Zeta.

      INTEGER Str

      REAL Alpha, Temp1, Temp2, B2, AlNew, B

      AlNew=Alpha
      Temp1=(B+1)/(B-1)

      IF (Str.EQ.1) THEN
        Temp2=Temp1**(1-Alpha)
        AlNew=((B+1)-(B-1)*Temp2)/(Temp2+1)*1
      ENDIF

      IF (Str.EQ.2) THEN
        B2=0
        Temp2=Temp1**((Alpha-B2)/(1-B2))
        AlNew=((B+2*B2)*Temp2-B+2*B2)/((2*B2+1)*(1+Temp2))
      ENDIF

      IF (Str.EQ.3) THEN
        B2=0.5
        Temp2=Temp1**((Alpha-B2)/(1-B2))
        AlNew=((B+2*B2)*Temp2-B+2*B2)/((2*B2+1)*(1+Temp2))
      ENDIF

      RETURN
      END

C==================================================================C

      SUBROUTINE EdgPts(X1,X2,X3,X4,Y1,Y2,Y3,Y4,Z1,Z2,Z3,Z4,AL,BL,
     $          AAStep,BBStep,x,y,z,s,zx,zy,zz,NDPts,Tensn,
     $          StrAA,StrBB,BetaAA,BetaBB,MxBPts,MxGSiz)

C This SUBROUTINE calculates the grid point locations along the surface
C edges.

      INTEGER ACt, BCt, n1, n2, n3, n4,
     $     AL, BL, StrAA, StrBB, NDPts(4)

      REAL AA, BB, AANew, BBNew, S1, S2, S3, S4, BBStep, AAStep,
     $  S1AAR, S2AAR, S3BBR, S4BBR,
     $  X1(MxGSiz), X2(MxGSiz), X3(MxGSiz), X4(MxGSiz),
     $  Y1(MxGSiz), Y2(MxGSiz), Y3(MxGSiz), Y4(MxGSiz),
     $  Z1(MxGSiz), Z2(MxGSiz), Z3(MxGSiz), Z4(MxGSiz),
```

```
     $      x(4,MxBPts),  y(4,MxBPts),  z(4,MxBPts),
     $      s(4,MxBPts), zx(4,MxBPts), zy(4,MxBPts),
     $      zz(4,MxBPts), Tensn(4), BetaAA, BetaBB

      S1AAR=s(1,NDPts(1))
      S2AAR=s(2,NDPts(2))
      S3BBR=s(3,NDPts(3))
      S4BBR=s(4,NDPts(4))

C Calculate the grid point locations along boundaries 1 and 2.

      AA=0.0

      DO 10 ACt=1,AL
        CALL FAlNew(AANew,AA,BetaAA,StrAA)
        S1=AANew*S1AAR
        S2=AANew*S2AAR
        CALL SplInt(n1,s,S1,NDPts,1,MxBPts)
        CALL SplInt(n2,s,S2,NDPts,2,MxBPts)
        X1(ACt)=SplVal(s,x,zx,S1,Tensn(1),n1,1,MxBPts)
        X2(ACt)=SplVal(s,x,zx,S2,Tensn(2),n2,2,MxBPts)
        Y1(ACt)=SplVal(s,y,zy,S1,Tensn(1),n1,1,MxBPts)
        Y2(ACt)=SplVal(s,y,zy,S2,Tensn(2),n2,2,MxBPts)
        Z1(ACt)=SplVal(s,z,zz,S1,Tensn(1),n1,1,MxBPts)
        Z2(ACt)=SplVal(s,z,zz,S2,Tensn(2),n2,2,MxBPts)
        AA=AA+AAStep
 10   CONTINUE

C Calculate the grid point locations along boundaries 3 and 4.

      BB=0.0

      DO 20 BCt=1,BL
        CALL FAlNew(BBNew,BB,BetaBB,StrBB)
        S3=BBNew*S3BBR
        S4=BBNew*S4BBR
        CALL SplInt(n3,s,S3,NDPts,3,MxBPts)
        CALL SplInt(n4,s,S4,NDPts,4,MxBPts)
        X3(BCt)=SplVal(s,x,zx,S3,Tensn(3),n3,3,MxBPts)
        X4(BCt)=SplVal(s,x,zx,S4,Tensn(4),n4,4,MxBPts)
        Y3(BCt)=SplVal(s,y,zy,S3,Tensn(3),n3,3,MxBPts)
        Y4(BCt)=SplVal(s,y,zy,S4,Tensn(4),n4,4,MxBPts)
        Z3(BCt)=SplVal(s,z,zz,S3,Tensn(3),n3,3,MxBPts)
        Z4(BCt)=SplVal(s,z,zz,S4,Tensn(4),n4,4,MxBPts)
        BB=BB+BBStep
 20   CONTINUE

      RETURN
      END

C==============================================================C

      SUBROUTINE EdgDer(PX1PAA,PX2PAA,PY1PAA,PY2PAA,PZ1PAA,PZ2PAA,
     $               PX3PBB,PX4PBB,PY3PBB,PY4PBB,PZ3PBB,PZ4PBB,
     $               X1,X2,X3,X4,Y1,Y2,Y3,Y4,Z1,Z2,Z3,Z4,
```

```
     $              AL,BL,AAStep,BBStep,MxGSiz)

      INTEGER  ACt, BCt, AL, BL

      REAL  AAStep, BBStep, PX3PBB(MxGSiz), PX4PBB(MxGSiz),
     $     PY3PBB(MxGSiz), PY4PBB(MxGSiz), PZ3PBB(MxGSiz),
     $     PZ4PBB(MxGSiz), PX1PAA(MxGSiz), PX2PAA(MxGSiz),
     $     PY1PAA(MxGSiz), PY2PAA(MxGSiz), PZ1PAA(MxGSiz),
     $     PZ2PAA(MxGSiz), X1(MxGSiz), X2(MxGSiz), X3(MxGSiz),
     $     X4(MxGSiz), Y1(MxGSiz), Y2(MxGSiz), Y3(MxGSiz),
     $     Y4(MxGSiz), Z1(MxGSiz), Z2(MxGSiz), Z3(MxGSiz),
     $     Z4(MxGSiz)

C Calculate the derivative values along the constant AA boundaries.

      PX1PAA(1)=(X1(2)-X1(1))/AAStep
      PX2PAA(1)=(X2(2)-X2(1))/AAStep
      PY1PAA(1)=(Y1(2)-Y1(1))/AAStep
      PY2PAA(1)=(Y2(2)-Y2(1))/AAStep
      PZ1PAA(1)=(Z1(2)-Z1(1))/AAStep
      PZ2PAA(1)=(Z2(2)-Z2(1))/AAStep

      PX1PAA(AL)=(X1(AL) -X1(AL-1))/AAStep
      PX2PAA(AL)=(X2(AL) -X2(AL-1))/AAStep
      PY1PAA(AL)=(Y1(AL) -Y1(AL-1))/AAStep
      PY2PAA(AL)=(Y2(AL) -Y2(AL-1))/AAStep
      PZ1PAA(AL)=(Z1(AL) -Z1(AL-1))/AAStep
      PZ2PAA(AL)=(Z2(AL) -Z2(AL-1))/AAStep

      DO 10 ACt=2,AL-1
        PX1PAA(ACt)= (X1(ACt+1)-X1(ACt-1))/2/AAStep
        PX2PAA(ACt)= (X2(ACt+1)-X2(ACt-1))/2/AAStep
        PY1PAA(ACt)= (Y1(ACt+1)-Y1(ACt-1))/2/AAStep
        PY2PAA(ACt)= (Y2(ACt+1)-Y2(ACt-1))/2/AAStep
        PZ1PAA(ACt)= (Z1(ACt+1)-Z1(ACt-1))/2/AAStep
        PZ2PAA(ACt)= (Z2(ACt+1)-Z2(ACt-1))/2/AAStep
 10   CONTINUE

C Calculate the derivative values along the constant BB boundaries.

      PX3PBB(1)= (X3(2)-X3(1))/BBStep
      PX4PBB(1)= (X4(2)-X4(1))/BBStep
      PY3PBB(1)= (Y3(2)-Y3(1))/BBStep
      PY4PBB(1)= (Y4(2)-Y4(1))/BBStep
      PZ3PBB(1)= (Z3(2)-Z3(1))/BBStep
      PZ4PBB(1)= (Z4(2)-Z4(1))/BBStep

      PX3PBB(BL)=(X3(BL) -X3(BL-1))/BBStep
      PX4PBB(BL)=(X4(BL) -X4(BL-1))/BBStep
      PY3PBB(BL)=(Y3(BL) -Y3(BL-1))/BBStep
      PY4PBB(BL)=(Y4(BL) -Y4(BL-1))/BBStep
      PZ3PBB(BL)=(Z3(BL) -Z3(BL-1))/BBStep
      PZ4PBB(BL)=(Z4(BL) -Z4(BL-1))/BBStep

      DO 20 BCt=2,BL-1
```

```
      PX3PBB(BCt)= (X3(BCt+1)-X3(BCt-1))/2/BBStep
      PX4PBB(BCt)= (X4(BCt+1)-X4(BCt-1))/2/BBStep
      PY3PBB(BCt)= (Y3(BCt+1)-Y3(BCt-1))/2/BBStep
      PY4PBB(BCt)= (Y4(BCt+1)-Y4(BCt-1))/2/BBStep
      PZ3PBB(BCt)= (Z3(BCt+1)-Z3(BCt-1))/2/BBStep
      PZ4PBB(BCt)= (Z4(BCt+1)-Z4(BCt-1))/2/BBStep
 20   CONTINUE

      RETURN
      END

C=====================================================================C

      SUBROUTINE TwoBnd(XS,YS,ZS,SrfNum,AL,BL,k1,k2,BetaAA,BetaBB,
     $            AAStep,BBStep,h1,h2,h3,h4,X1,X2,X3,X4,
     $            Y1,Y2,Y3,Y4,Z1,Z2,Z3,Z4,PX1PBB,PX2PBB,
     $            PY1PBB,PY2PBB,PZ1PBB,PZ2PBB,PX1PAA,PX2PAA,
     $            PY1PAA,PY2PAA,PZ1PAA,PZ2PAA,PX3PBB,PX4PBB,
     $            PY3PBB,PY4PBB,PZ3PBB,PZ4PBB,StrAA,StrBB,
     $            MxGSiz,MxSrfs)

C This SUBROUTINE calculates the interior grid point locations between two
C specified boundaries (1 and 2) using transfinite Hermite interpolation.

      INTEGER  ACt, BCt, AL, BL, StrAA, StrBB, SrfNum

      REAL  AA, BB, AANew, BBNew,
     $    Box1i, Box1j, Box1k, Box2i, Box2j, Box2k,
     $    Paren1i, Paren1j, Paren1k, Paren2i, Paren2j, Paren2k,
     $    k1, k2, BetaAA, BetaBB, BBStep, AAStep,
     $    h1(MxGSiz), h2(MxGSiz), h3(MxGSiz), h4(MxGSiz),
     $    X1(MxGSiz), X2(MxGSiz), X3(MxGSiz), X4(MxGSiz),
     $    Y1(MxGSiz), Y2(MxGSiz), Y3(MxGSiz), Y4(MxGSiz),
     $    Z1(MxGSiz), Z2(MxGSiz), Z3(MxGSiz), Z4(MxGSiz)
      REAL  PX1PBB(MxGSiz), PX2PBB(MxGSiz),
     $    PY1PBB(MxGSiz), PY2PBB(MxGSiz),
     $    PZ1PBB(MxGSiz), PZ2PBB(MxGSiz),
     $    PX1PAA(MxGSiz), PX2PAA(MxGSiz),
     $    PY1PAA(MxGSiz), PY2PAA(MxGSiz),
     $    PZ1PAA(MxGSiz), PZ2PAA(MxGSiz),
     $    PX3PBB(MxGSiz), PX4PBB(MxGSiz),
     $    PY3PBB(MxGSiz), PY4PBB(MxGSiz),
     $    PZ3PBB(MxGSiz), PZ4PBB(MxGSiz),
     $    XS(MxSrfs,MxGSiz,MxGSiz),
     $    YS(MxSrfs,MxGSiz,MxGSiz),
     $    ZS(MxSrfs,MxGSiz,MxGSiz)


C Calculate the h factors for the boundary 1-2 Hermite connecting
C curves.

      BB=0.0

      DO 10 BCt=1,BL
        CALL FAlNew(BBNew,BB,BetaBB,StrBB)
```

```
      CALL FindHs(h1(BCt),h2(BCt),h3(BCt),h4(BCt),BBNew)
      BB=BB+BBStep
 10   CONTINUE

C Calculate the derivative values for grid line orthogonality.

     AA=0.0

     DO 20 ACt=1,AL
      CALL FAlNew(AANew,AA,BetaAA,StrAA)
      Box1i=    AA*( PY1PAA(AL)*PZ4PBB(1)
   $               -PZ1PAA(AL)*PY4PBB(1))
   $       +(1-AA)*( PY1PAA(1) *PZ3PBB(1)
   $               -PZ1PAA(1) *PY3PBB(1))
      Box1j=    AA*( PX1PAA(AL)*PZ4PBB(1)
   $               -PZ1PAA(AL)*PX4PBB(1))
   $       +(1-AA)*( PX1PAA(1) *PZ3PBB(1)
   $               -PZ1PAA(1) *PX3PBB(1))
      Box1k=    AA*( PX1PAA(AL)*PY4PBB(1)
   $               -PY1PAA(AL)*PX4PBB(1))
   $       +(1-AA)*( PX1PAA(1) *PY3PBB(1)
   $               -PY1PAA(1) *PX3PBB(1))
      Box2i=    AA*( PY2PAA(AL)*PZ4PBB(BL)
   $               -PZ2PAA(AL)*PY4PBB(BL))
   $       +(1-AA)*( PY2PAA(1) *PZ3PBB(BL)
   $               -PZ2PAA(1) *PY3PBB(BL))
      Box2j=    AA*( PX2PAA(AL)*PZ4PBB(BL)
   $               -PZ2PAA(AL)*PX4PBB(BL))
   $       +(1-AA)*( PX2PAA(1) *PZ3PBB(BL)
   $               -PZ2PAA(1) *PX3PBB(BL))
      Box2k=    AA*( PX2PAA(AL)*PY4PBB(BL)
   $               -PY2PAA(AL)*PX4PBB(BL))
   $       +(1-AA)*( PX2PAA(1) *PY3PBB(BL)
   $               -PY2PAA(1) *PX3PBB(BL))
      Paren1i=PY1PAA(ACt)*Box1k+PZ1PAA(ACt)*Box1j
      Paren1j=PX1PAA(ACt)*Box1k-PZ1PAA(ACt)*Box1i
      Paren1k=PX1PAA(ACt)*Box1j+PY1PAA(ACt)*Box1i
      Paren2i=PY2PAA(ACt)*Box2k+PZ2PAA(ACt)*Box2j
      Paren2j=PX2PAA(ACt)*Box2k-PZ2PAA(ACt)*Box2i
      Paren2k=PX2PAA(ACt)*Box2j+PY2PAA(ACt)*Box2i
      PX1PBB(ACt)=-k1*Paren1i
      PX2PBB(ACt)=-k2*Paren2i
      PY1PBB(ACt)= k1*Paren1j
      PY2PBB(ACt)= k2*Paren2j
      PZ1PBB(ACt)= k1*Paren1k
      PZ2PBB(ACt)= k2*Paren2k

      AA=AA+AAStep
 20   CONTINUE

C Calculate the interior grid point locations.

     DO 40 ACt=1,AL
      DO 30 BCt=1,BL
       XS(SrfNum,ACt,BCt)= h1(BCt)*X1(ACt)+h2(BCt)*X2(ACt)
```

```
     $                +h3(BCt)*PX1PBB(ACt)+h4(BCt)*PX2PBB(ACt)
       YS(SrfNum,ACt,BCt)= h1(BCt)*Y1(ACt)+h2(BCt)*Y2(ACt)
     $                +h3(BCt)*PY1PBB(ACt)+h4(BCt)*PY2PBB(ACt)
       ZS(SrfNum,ACt,BCt)= h1(BCt)*Z1(ACt)+h2(BCt)*Z2(ACt)
     $                +h3(BCt)*PZ1PBB(ACt)+h4(BCt)*PZ2PBB(ACt)
 30     CONTINUE
 40   CONTINUE

      RETURN
      END

C==================================================================C

      SUBROUTINE ForBnd(XS,YS,ZS,SrfNum,AL,BL,k3,k4,BetaAA,BetaBB,
     $          AAStep,BBStep,h1,h2,h3,h4,h5,h6,h7,h8,
     $          X1,X2,X3,X4,Y1,Y2,Y3,Y4,Z1,Z2,Z3,Z4,
     $          PX1PBB,PX2PBB,PY1PBB,PY2PBB,PZ1PBB,PZ2PBB,
     $          PX1PAA,PX2PAA,PY1PAA,PY2PAA,PZ1PAA,PZ2PAA,
     $          PX3PBB,PX4PBB,PY3PBB,PY4PBB,PZ3PBB,PZ4PBB,
     $          PX3PAA,PX4PAA,PY3PAA,PY4PAA,PZ3PAA,PZ4PAA,
     $          StrAA,StrBB,MxGSiz,MxSrfs)

C This SUBROUTINE adjusts the grid so that the other two boundaries
C (3 and 4) of the surface are mapped correctly using transfinite Hermite
C interpolation.

      INTEGER  ACt, BCt, AL, BL, StrAA, StrBB, i, j, SrfNum

      REAL  AA, BB, AANew, BBNew,
     $    Box3i, Box3j, Box3k, Box4i, Box4j, Box4k,
     $    Paren3i, Paren3j, Paren3k, Paren4i, Paren4j, Paren4k,
     $    P2Y00, P2Y01, P2Y10, P2Y11, P2X00, P2X01, P2X10, P2X11,
     $    P2Z00, P2Z01, P2Z10, P2Z11,
     $    k3, k4, BetaAA, BetaBB, BBStep, AAStep,
     $    h1(MxGSiz), h2(MxGSiz), h3(MxGSiz), h4(MxGSiz),
     $    h5(MxGSiz), h6(MxGSiz), h7(MxGSiz), h8(MxGSiz),
     $    X1(MxGSiz), X2(MxGSiz), X3(MxGSiz), X4(MxGSiz),
     $    Y1(MxGSiz), Y2(MxGSiz), Y3(MxGSiz), Y4(MxGSiz),
     $    Z1(MxGSiz), Z2(MxGSiz), Z3(MxGSiz), Z4(MxGSiz)
      REAL  PX1PBB(MxGSiz), PX2PBB(MxGSiz),
     $    PY1PBB(MxGSiz), PY2PBB(MxGSiz),
     $    PZ1PBB(MxGSiz), PZ2PBB(MxGSiz),
     $    PX1PAA(MxGSiz), PX2PAA(MxGSiz),
     $    PY1PAA(MxGSiz), PY2PAA(MxGSiz),
     $    PZ1PAA(MxGSiz), PZ2PAA(MxGSiz),
     $    PX3PBB(MxGSiz), PX4PBB(MxGSiz),
     $    PY3PBB(MxGSiz), PY4PBB(MxGSiz),
     $    PZ3PBB(MxGSiz), PZ4PBB(MxGSiz),
     $    PX3PAA(MxGSiz), PX4PAA(MxGSiz),
     $    PY3PAA(MxGSiz), PY4PAA(MxGSiz),
     $    PZ3PAA(MxGSiz), PZ4PAA(MxGSiz),
     $    XS(MxSrfs,MxGSiz,MxGSiz),
     $    YS(MxSrfs,MxGSiz,MxGSiz),
     $    ZS(MxSrfs,MxGSiz,MxGSiz)
```

```
C Calculate the h factors for the boundary 3-4 Hermite adjusting
C curves.

      AA=0.0

      DO 10 ACt=1,AL
         CALL FAlNew(AANew,AA,BetaAA,StrAA)
         CALL FindHs(h5(ACt),h6(ACt),h7(ACt),h8(ACt),AANew)
         AA=AA+AAStep
 10   CONTINUE

C Calculate the derivative values for grid line orthogonality.

      BB=0.0

      DO 20 BCt=1,BL
        CALLFAlNew(BBNew,BB,BetaBB,StrBB)
        Box3i=      BB*( PY2PAA(1)*PZ3PBB(BL)
     $                  -PZ2PAA(1)*PY3PBB(BL))
     $        +(1-BB)*( PY1PAA(1)*PZ3PBB(1)
     $                  -PZ1PAA(1)*PY3PBB(1))
        Box3j=      BB*( PX2PAA(1)*PZ3PBB(BL)
     $                  -PZ2PAA(1)*PX3PBB(BL))
     $        +(1-BB)*( PX1PAA(1)*PZ3PBB(1)
     $                  -PZ1PAA(1)*PX3PBB(1))
        Box3k=      BB*( PX2PAA(1)*PY3PBB(BL)
     $                  -PY2PAA(1)*PX3PBB(BL))
     $        +(1-BB)*( PX1PAA(1)*PY3PBB(1)
     $                  -PY1PAA(1)*PX3PBB(1))
        Box4i=      BB*( PY2PAA(1)*PZ4PBB(BL)
     $                  -PZ2PAA(1)*PY4PBB(BL))
     $        +(1-BB)*( PY1PAA(1)*PZ4PBB(1)
     $                  -PZ1PAA(1)*PY4PBB(1))
        Box4j=      BB*( PX2PAA(1)*PZ4PBB(BL)
     $                  -PZ2PAA(1)*PX4PBB(BL))
     $        +(1-BB)*( PX1PAA(1)*PZ4PBB(1)
     $                  -PZ1PAA(1)*PX4PBB(1))
        Box4k=      BB*( PX2PAA(1)*PY4PBB(BL)
     $                  -PY2PAA(1)*PX4PBB(BL))
     $        +(1-BB)*( PX1PAA(1)*PY4PBB(1)
     $                  -PY1PAA(1)*PX4PBB(1))
        Paren3i=PZ3PBB(BCt)*Box3j+PY3PBB(BCt)*Box3k
        Paren3j=PZ3PBB(BCt)*Box3i-PX3PBB(BCt)*Box3k
        Paren3k=PY3PBB(BCt)*Box3i+PX3PBB(BCt)*Box3j
        Paren4i=PZ4PBB(BCt)*Box4j+PY4PBB(BCt)*Box4k
        Paren4j=PZ4PBB(BCt)*Box4i-PX4PBB(BCt)*Box4k
        Paren4k=PY4PBB(BCt)*Box4i+PX4PBB(BCt)*Box4j
        PX3PAA(BCt)= k3*Paren3i
        PX4PAA(BCt)= k4*Paren4i
        PY3PAA(BCt)= k3*Paren3j
        PY4PAA(BCt)= k4*Paren4j
        PZ3PAA(BCt)=-k3*Paren3k
        PZ4PAA(BCt)=-k4*Paren4k

        BB=BB+BBStep
```

```
 20 CONTINUE

C Set the cross-derivative terms equal to zero.

      P2X00=0.0
      P2X10=0.0
      P2X01=0.0
      P2X11=0.0
      P2Y00=0.0
      P2Y10=0.0
      P2Y01=0.0
      P2Y11=0.0
      P2Z00=0.0
      P2Z10=0.0
      P2Z01=0.0
      P2Z11=0.0

C Calculate the grid point locations everywhere.

      DO 40 i=1,AL
        DO 30 j=1,BL
          XS(SrfNum,i,j)=XS(SrfNum,i,j)
     $          +(X3(j)-h1(j)*X1(1)
     $               -h2(j)*X2(1)
     $               -h3(j)*PX1PBB(1)
     $               -h4(j)*PX2PBB(1))*h5(i)
     $          +(X4(j)-h1(j)*X1(AL)
     $               -h2(j)*X2(AL)
     $               -h3(j)*PX1PBB(AL)
     $               -h4(j)*PX2PBB(AL))*h6(i)
     $          +(PX3PAA(j)-( h1(j)*PX3PAA(1)
     $                   +h2(j)*PX3PAA(BL)
     $                   +h3(j)*P2X00+h4(j)*P2X01))*h7(i)
     $          +(PX4PAA(j)-( h1(j)*PX4PAA(1)
     $                   +h2(j)*PX4PAA(BL)
     $                   +h3(j)*P2X10+h4(j)*P2X11))*h8(i)
          YS(SrfNum,i,j)=YS(SrfNum,i,j)
     $          +(Y3(j)-h1(j)*Y1(1)
     $               -h2(j)*Y2(1)
     $               -h3(j)*PY1PBB(1)
     $               -h4(j)*PY2PBB(1))*h5(i)
     $          +(Y4(j)-h1(j)*Y1(AL)
     $               -h2(j)*Y2(AL)
     $               -h3(j)*PY1PBB(AL)
     $               -h4(j)*PY2PBB(AL))*h6(i)
     $          +(PY3PAA(j)-( h1(j)*PY3PAA(1)
     $                   +h2(j)*PY3PAA(BL)
     $                   +h3(j)*P2Y00+h4(j)*P2Y01))*h7(i)
     $          +(PY4PAA(j)-( h1(j)*PY4PAA(1)
     $                   +h2(j)*PY4PAA(BL)
     $                   +h3(j)*P2Y10+h4(j)*P2Y11))*h8(i)

          ZS(SrfNum,i,j)=ZS(SrfNum,i,j)
     $          +(Z3(j)-h1(j)*Z1(1)
     $               -h2(j)*Z2(1)
```

```
     $               -h3(j)*PZ1PBB(1)
     $               -h4(j)*PZ2PBB(1))*h5(i)
     $           +(Z4(j)-h1(j)*Z1(AL)
     $               -h2(j)*Z2(AL)
     $               -h3(j)*PZ1PBB(AL)
     $               -h4(j)*PZ2PBB(AL))*h6(i)
     $           +(PZ3PAA(j)-( h1(j)*PZ3PAA(1)
     $                   +h2(j)*PZ3PAA(BL)
     $                   +h3(j)*P2Z00+h4(j)*P2Z01))*h7(i)
     $           +(PZ4PAA(j)-( h1(j)*PZ4PAA(1)
     $                   +h2(j)*PZ4PAA(BL)
     $                   +h3(j)*P2Z10+h4(j)*P2Z11))*h8(i)
 30      CONTINUE
 40    CONTINUE

      RETURN
      END

C=========================================================C
```

TABLE 2-1. - GUIDE TO GRID CONTROL PARAMETERS LISTED IN FIGURES 2-1, 2-2, AND 2-6.

PARAMETER	RANGE	TRIAL VALUE	DESCRIPTION
StretchTypeXi StretchTypeEta StretchTypeZeta	0 - No clustering 1 - Clustering near lower surface 2 - Clustering near upper surface 3 - Clustering near both surfaces	n/a	Controls type of clustering in the Xi, Eta, and Zeta "directions"
kXi1 kXi2 kEta1 kEta2 kZeta1 kZeta2	$0.0 \leq k \leq \infty$ less curvature ... more curvature	0.2	Controls curvature of the grid lines in the Xi, Eta, and Zeta "directions"
BetaXi BetaEta BetaZeta	$1.0 < \text{Beta} < \infty$ more clustering ... less clustering	1.1	Controls amount of clustering in the Xi, Eta, and Zeta "directions"
Tension	$0.0 < \text{Tension} < \infty$ more curvature ... less curvature	1.0	Controls curvature of the boundary spline curves
Method	2 - Two-Boundary Method 4 - Four-Boundary Method	n/a	Controls which method is used to form the grid

n — Number of Grids
Information for Grid 1
Information for Grid 2
Information for Grid 3

.
.
.

Information for Grid n

FIGURE 2-1. - DIAGRAM SHOWING OVERALL LAYOUT OF 2-D GRID INPUT FILE FOR GRID2D/3D.

t - Method for Grid n
Tension for Boundary Curve 1
Number of Discrete Points for Boundary Curve 1

List of discrete points for Boundary Curve 1.
List is made up of xy pairs, each pair on its own line.

Tension for Boundary Curve 2
Number of Discrete Points for Boundary Curve 2

List of discrete points for Boundary Curve 2.
List is made up of xy pairs, each pair on its own line.

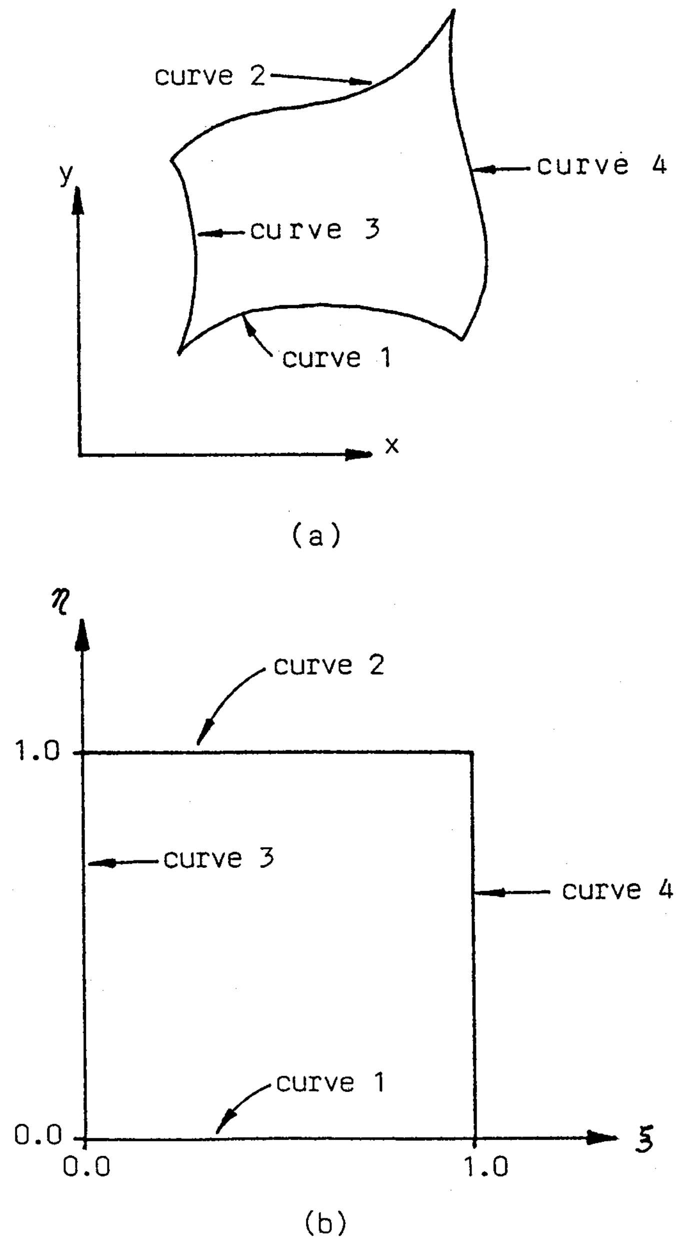
.
.
.

Tension for Boundary Curve t
Number of Discrete Points for Boundary Curve t

List of discrete points for Boundary Curve t.
List is made up of xy pairs, each pair on its own line.

IL - Number of Xi grid points for grid n
JL - Number of Eta grid points for grid n
StretchTypeXi
StretchTypeEta
KXi1
KXi2
KEta1 (necessary only if t=4)
KEta2 (necessary only if t=4)
BetaXi
BetaEta

FIGURE 2-2.- DIAGRAM SHOWING DETAILS OF SECTION MARKED "INFORMATION FOR GRID n" IN FIGURE 2-1.



(a) IN x-y COORDINATE SYSTEM.

(b) IN ξ - η COORDINATE SYSTEM (TRANSFORMED DOMAIN).

FIGURE 2-3. - SPATIAL DOMAIN.

```
2               Number of grids
2               Method for grid 1 ======================
2.0             Tension for curve 1 of grid 1 -----------------
3               Number of nodal points for curve 1 of grid 1
0.0    0.0
5.0    0.5
10.0   0.0
2.0             Tension for curve 2 of grid 1 -----------------
4               Number of nodal points for curve 2 of grid 1
0.0    2.0
3.0    2.0
6.0    1.5
10.0   1.75
21              Number of Xi grid points for grid 1
7               Number of Eta grid points for grid 1
0               StretchTypeXi for grid 1
3               StretchTypeEta for grid 1
0.4             kXi1 for grid 1
0.4             kXi2 for grid 1
1.05            BetaXi for grid 1
1.05            BetaEta for grid 1
2               Method for grid 2 ======================
2.0             Tension for curve 2 of grid 2 -----------------
3               Number of nodal points for curve 1 of grid 2
0.0    2.0
5.0    2.5
10.0   3.0
2.0             Tension for curve 2 of grid 2 -----------------
2               Number of nodal points for curve 2 of grid 2
0.0    5.0
10.0   5.0
21              Number of Xi grid points for grid 2
7               Number of Eta grid points for grid 2
0               StretchTypeXi for grid 2
3               StretchTypeEta for grid 2
0.4             kXi1 for grid 2
0.4             kXi2 for grid 2
1.05            BetaXi for grid 2
1.05            BetaEta for grid 2
```

FIGURE 2-4. - SAMPLE 2-D GRID INPUT FILE FOR GRID2D/3D.

```
1           Number of grids
2           Method for grid 1 ======================
2.0         Tension for curve 1 of grid 1 ----------------
2           Number of nodal points for curve 1 of grid 1
1.0   0.0
9.0   0.0
10.0  0.0
2.0         Tension for curve 2 of grid 1 ----------------
6           Number of nodal points for curve 2 of grid 1
1.0   7.0
2.0   7.0
4.0   8.0
6.0   8.0
8.0   7.0
9.0   7.0
2.0         Tension for curve 3 of grid 1 ----------------
6           Number of nodal points for curve 3 of grid 1
1.0   0.0
1.0   1.0
0.3   3.0
0.3   4.0
1.0   6.0
1.0   7.0
2.0         Tension for curve 4 of grid 1 ----------------
6           Number of nodal points for curve 4 of grid 1
9.0   0.0
9.0   1.0
9.7   3.0
9.7   4.0
9.0   6.0
9.0   7.0
21          Number of Xi grid points for grid 1
21          Number of Eta grid points for grid 1
0           StretchTypeXi for grid 1
0           StretchTypeEta for grid 1
0.4         kXi1 for grid 1
0.4         kXi2 for grid 1
0.4         kEta1 for grid 1
0.4         kEta2 for grid 1
1.05        BetaXi for grid 1
1.05        BetaEta for grid 1
```

FIGURE 2-5. - SECOND SAMPLE 2-D GRID INPUT FILE FOR GRID2D/3D.

t - Method
IL - Number of Xi grid points
JL - Number of Eta grid points
KL - Number of Zeta grid points
StretchTypeXi
StretchTypeEta
StretchTypeZeta
KXi1
KXi2
KEta1
KEta2
KZeta1
KZeta2
BetaXi
BetaEta
BetaZeta
Tension for Boundary Curve 1
Number of Discrete Points for Boundary Curve 1

List of discrete points for Boundary Curve 1.
List is made up of xyz triples, each triple on its own line.

Tension for Boundary Curve 2
Number of Discrete Points for Boundary Curve 2

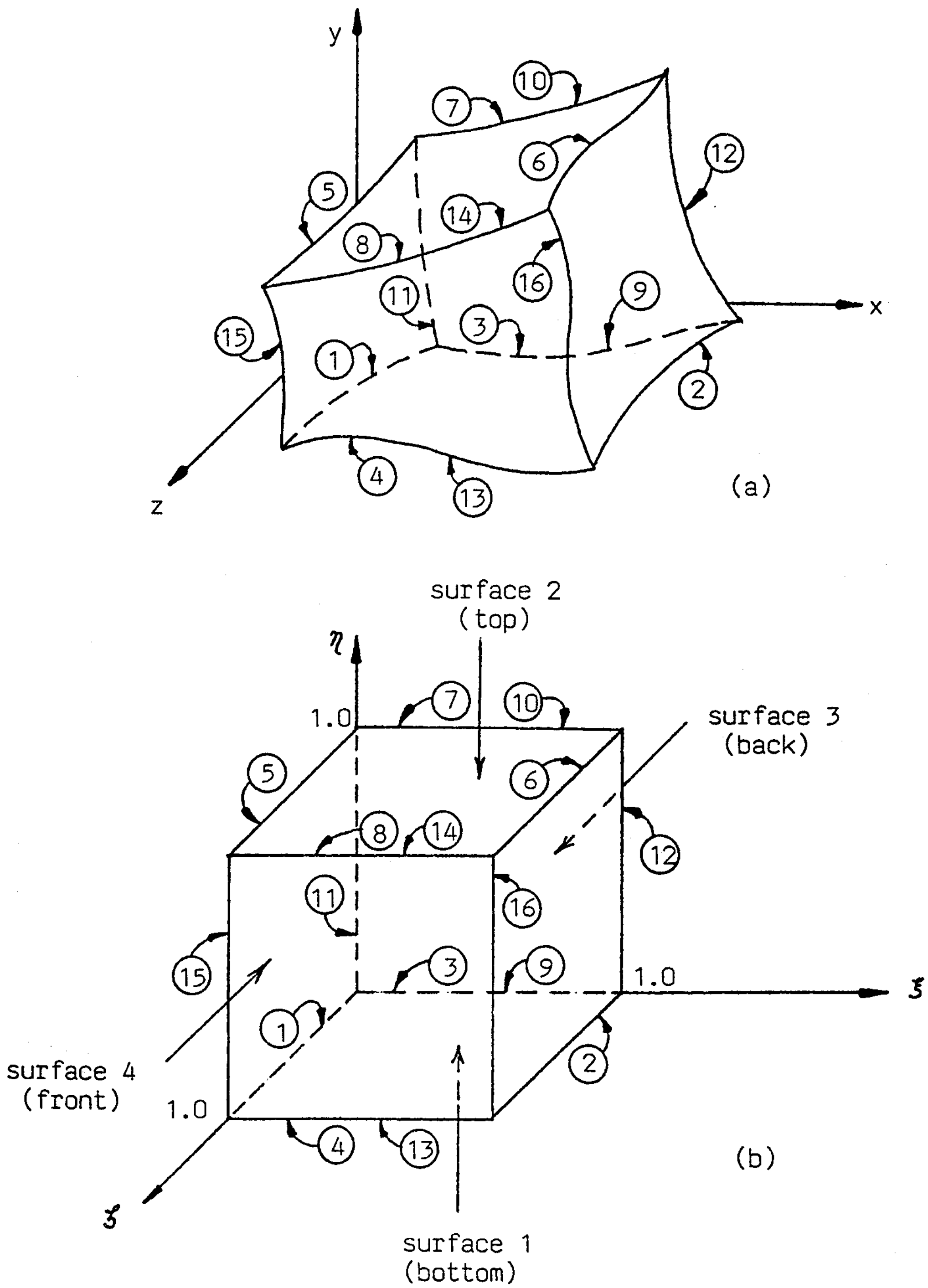
List of discrete points for Boundary Curve 2.
List is made up of xyz triples, each triple on its own line.

.
.
.

Tension for Boundary Curve 4t
Number of Discrete Points for Boundary Curve 4t

List of discrete points for Boundary Curve 4t.
List is made up of xyz triples, each triple on its own line.

FIGURE 2-6. - DIAGRAM SHOWING LAYOUT OF 3-D GRID INPUT FILE FOR GRID2D/3D.



(a) IN x-y-z COORDINATE SYSTEM.

(b) IN ξ-η-ζ COORDINATE SYSTEM (TRANSFORMED DOMAIN).

FIGURE 2-7. - SPATIAL DOMIAN.

```
2              Method ==============================
21             Number of Xi grid points
21             Number of Eta grid points
21             Number of Zeta grid points
0              StretchTypeXi
0              StretchTypeEta
0              StretchTypeZeta
0.4            kXi1
0.4            kXi2
0.4            kEta1
0.4            kEta2
0.4            kZeta1
0.4            kZeta2
1.05           BetaXi
1.05           BetaEta
1.05           BetaZeta
2.0            Tension for curve 1 --------------------------
2              Number of nodal points for curve 1
0.0     0.0    0.0
6.0     0.0    0.0
2.0            Tension for curve 2 --------------------------
2              Number of nodal points for curve 2
0.0     0.0    10.0
6.0     0.0    10.0
2.0            Tension for curve 3 --------------------------
2              Number of nodal points for curve 3
0.0     0.0    0.0
0.0     0.0    10.0
2.0            Tension for curve 4 --------------------------
2              Number of nodal points for curve 4
6.0     0.0    0.0
6.0     0.0    10.0
2.0            Tension for curve 5 --------------------------
2              Number of nodal points for curve 5
0.0     6.0    0.0
6.0     6.0    0.0
2.0            Tension for curve 6 --------------------------
2              Number of nodal points for curve 6
0.0     6.0    10.0
6.0     6.0    10.0
2.0            Tension for curve 7 --------------------------
2              Number of nodal points for curve 7
0.0     6.0    0.0
0.0     6.0    10.0
2.0            Tension for curve 8 --------------------------
2              Number of nodal points for curve 8
6.0     6.0    0.0
6.0     6.0    10.0
```

FIGURE 2-8. - SAMPLE 3-D GRID INPUT FILE FOR GRID2D/3D.

```
2           Number of Grids
2           Technique for Grid 1 =======================
2.0         Tension for Curve 1 of Grid 1 --------------------
7           Number of Nodal Points for Curve 1 of Grid 1
  0.0   0.0
  0.5   0.0
  3.0   0.5
  5.0   0.5
  8.0   0.0
  9.0   0.0
 10.0   0.0
2.0         Tension for Curve 2 of Grid 1 --------------------
8           Number of Nodal Points for Curve 2 of Grid 1
  0.0   2.0
  2.0   2.0
  3.0   2.0
  4.0   1.9
  6.0   1.5
  8.0   1.75
  9.0   1.75
 10.0   1.75
21          Number of Xi  Grid Points for Grid 1 -------------
11          Number of Eta Grid Points for Grid 1
0           Xi  Direction Stretching Type for Grid 1
3           Eta Direction Stretching Type for Grid 1
0.2         kXi1 for Grid 1
0.2         kXi2 for Grid 1
1.005       Stretching Parameter BetaXi  for Grid 1
1.005       Stretching Parameter BetaEta for Grid 1
2           Technique for Grid 2 =======================
2.0         Tension for Curve 1 of Grid 2 --------------------
7           Number of Nodal Points for Curve 1 of Grid 2
  0.0   2.0
  2.0   2.0
  3.0   2.0
  5.0   2.5
  8.0   3.0
  9.0   3.0
 10.0   3.0
2.0         Tension for Curve 2 of Grid 2 --------------------
2           Number of Nodal Points for Curve 2 of Grid 2
  0.0   5.0
 10.0   5.0
21          Number of Xi  Grid Points for Grid 2 -------------
11          Number of Eta Grid Points for Grid 2
0           Xi  Direction Stretching Type for Grid 2
1           Eta Direction Stretching Type for Grid 2
0.4         kXi1 for Grid 2
0.4         kXi2 for Grid 2
1.01        Stretching Parameter BetaXi  for Grid 2
1.01        Stretching Parameter BetaEta for Grid 2
```

FIGURE 4-1. - LISTINGS OF 2-D GRID INPUT FILE INLET.DAT.

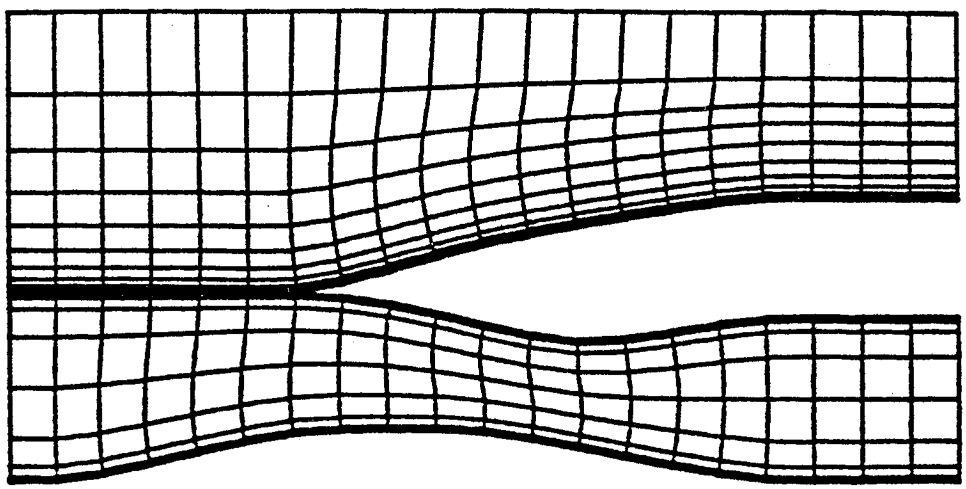


FIGURE 4.2. - GRID GENERATED BY GRID2D/3D USING INPUT FILE INLET.DAT (FIG. 4-1).

```
4          Technique
9          IL
21         JL
21         KL
3          StretchTypeXi
3          StretchTypeEta
0          StretchTypeZeta
2.0        kXi1
2.0        kXi2
2.0        kEta1
2.0        kEta2
2.0        kZeta1
2.0        kZeta2
1.01       BetaXi
1.01       BetaEta
1.05       BetaZeta
1.0        Tension 1 ---------------
2          Number of Points 1
- 0.095      0.32183      0.0
 0.0         0.32183      0.0
1.0        Tension 2 ---------------
2          Number of Points 2
- 0.095      0.375        0.0
 0.0         0.375        0.0
1.0        Tension 3 ---------------
2  Number of Points 3
- 0.095      0.32183      0.0
- 0.095      0.375        0.0
1.0        Tension 4 ---------------
2          Number of Points 4
 0.0         0.32183      0.0
 0.0         0.375        0.0
1.0        Tension 5 ---------------
2          Number of Points 5
- 0.095      0.31247      0.07704
 0.0         0.31247      0.07704
1.0        Tension 6 ---------------
2          Number of Points 6
- 0.095      0.36410      0.08977
 0.0         0.36410      0.08977
1.0        Tension 7 ---------------
2          Number of Points 7
- 0.095      0.31247      0.07704
- 0.095      0.36410      0.08977
1.0        Tension 8 ---------------
2          Number of Points 8
 0.0         0.31247      0.07704
 0.0         0.36410      0.08977
```

FIGURE 4-3. - LISTING OF 3-D GRID INPUT FILE ZONE1.DAT.

1.0	Tension 9 ---------------	
2	Number of Points 9	
- 0.095	0.32183	0.0
- 0.095	0.375	0.0
1.0	Tension 10 --------------	
2	Number of Points 10	
- 0.095	0.31247	0.07704
- 0.095	0.36410	0.08977
1.0	Tension 11 --------------	
4	Number of Points 11	
- 0.095	0.32183	0.0
- 0.095	0.32104	0.02253
- 0.095	0.31751	0.05257
- 0.095	0.31247	0.07704
1.0	Tension 12 --------------	
4	Number of Points 12	
- 0.095	0.375	0.0
- 0.095	0.37408	0.02625
- 0.095	0.36996	0.06125
- 0.095	0.36410	0.08977
1.0	Tension 13 --------------	
2	Number of Points 13	
0.0	0.32183	0.0
0.0	0.375	0.0
1.0	Tension 14 --------------	
2	Number of Points 14	
0.0	0.31247	0.07704
0.0	0.36410	0.08977
2.0	Tension 15 --------------	
6	Number of Points 15	
0.0	0.32183	0.0
- 0.00437	0.32174	0.00751
- 0.00787	0.32104	0.02253
- 0.00787	0.31751	0.05257
- 0.00437	0.31465	0.06758
0.0	0.31247	0.07704
2.0	Tension 16 --------------	
6	Number of Points 16	
0.0	0.375	0.0
- 0.00437	0.37490	0.00875
- 0.00787	0.37409	0.02625
- 0.00787	0.36996	0.06125
- 0.00437	0.36664	0.07875
0.0	0.36410	0.08977

FIGURE 4-3. - CONCLUDED.

4	Technique	
9	IL	
21	JL	
21	KL	
3	StretchTypeXi	
3	StretchTypeEta	
0	StretchTypeZeta	
2.0	kXi1	
2.0	kXi2	
2.0	kEta1	
2.0	kEta2	
2.0	kZeta1	
2.0	kZeta2	
1.01	BetaXi	
1.01	BetaEta	
1.05	BetaZeta	
2.0	Tension 1 ---------------	
12	Number of Points 1	
0.00000	0.32183	0.00000
0.00474	0.32177	0.00637
0.00945	0.32167	0.01025
0.01416	0.32149	0.01472
0.01886	0.32122	0.01986
0.02357	0.32079	0.02582
0.02828	0.32016	0.03270
0.03299	0.31926	0.04062
0.03770	0.31800	0.04948
0.04241	0.31638	0.05899
0.04712	0.31438	0.06884
0.05297	0.31292	0.07521
2.0	Tension 2 ---------------	
12	Number of Points 2	
0.00000	0.37500	0.00000
0.00474	0.37493	0.00742
0.00945	0.37481	0.01194
0.01416	0.37461	0.01715
0.01886	0.37429	0.02314
0.02357	0.37379	0.03008
0.02828	0.37306	0.03810
0.03299	0.37200	0.04733
0.03770	0.37054	0.05765
0.04241	0.36865	0.06873
0.04712	0.36632	0.08021
0.05297	0.36462	0.08763

FIGURE 4-4. - LISTING OF 3-D GRID INPUT FILE ZONE2.DAT.

```
1.0        Tension 3 ---------------
2          Number of Points 3
 0.00000     0.32183      0.00000
 0.00000     0.37500      0.00000
1.0        Tension 4 ---------------
2          Number of Points 4
 0.05297     0.31292      0.07521
 0.05297     0.36462      0.08763
2.0        Tension 5 ---------------
6          Number of Points 5
 0.00000     0.31247      0.07704
 0.01063     0.31392      0.07091
 0.01533     0.31367      0.07200
 0.02004     0.31310      0.07447
 0.02475     0.31204      0.07879
 0.02946     0.31032      0.08531
2.0        Tension 6 ---------------
6          Number of Points 6
 0.00000     0.36410      0.08977
 0.01063     0.36578      0.08263
 0.01533     0.36550      0.08389
 0.02004     0.36482      0.08677
 0.02475     0.36359      0.09181
 0.02946     0.36158      0.09941
1.0        Tension 7 ---------------
2          Number of Points 7
 0.00000     0.31247      0.07704
 0.00000     0.36410      0.08977
1.0        Tension 8 ---------------
2          Number of Points 8
 0.02946     0.31032      0.08531
 0.02946     0.36158      0.09941
1.0        Tension 9 ---------------
2          Number of Points 9
 0.00000     0.32183      0.00000
 0.00000     0.37500      0.00000
1.0        Tension 10 --------------
2          Number of Points 10
 0.00000     0.31247      0.07704
 0.00000     0.36410      0.08977
2.0        Tension 11 --------------
6          Number of Points 11
 0.0         0.32183      0.0
- 0.00437    0.32174      0.00751
- 0.00787    0.32104      0.02253
- 0.00787    0.31751      0.05257
- 0.00437    0.31465      0.06758
 0.0         0.31247      0.07704
```

FIGURE 4-4. - CONTINUED.

2.0	Tension 12 --------------	
6	Number of Points 12	
0.0	0.375	0.0
- 0.00437	0.37490	0.00875
- 0.00787	0.37409	0.02625
- 0.00787	0.36996	0.06125
- 0.00437	0.36664	0.07875
0.0	0.36410	0.08977
1.0	Tension 13 --------------	
2	Number of Points 13	
0.05297	0.31292	0.07521
0.05297	0.36462	0.08763
1.0	Tension 14 --------------	
2	Number of Points 14	
0.02946	0.31032	0.08531
0.02946	0.36158	0.09941
2.0	Tension 15 --------------	
4	Number of Points 15	
0.05297	0.31292	0.07521
0.04513	0.31209	0.07858
0.03730	0.31122	0.08194
0.02946	0.31032	0.08531
2.0	Tension 16 --------------	
4	Number of Points 16	
0.05297	0.36462	0.08763
0.04513	0.36365	0.09156
0.03730	0.36264	0.09548
0.02946	0.36158	0.09941

FIGURE 4-4. - CONCLUDED.

```
4           Technique
9           IL
21          JL
21          KL
3           StretchTypeXi
3           StretchTypeEta
0           StretchTypeZeta
2.0         kXi1
2.0         kXi2
2.0         kEta1
2.0         kEta2
2.0         kZeta1
2.0         kZeta2
1.01        BetaXi
1.01        BetaEta
1.05        BetaZeta
2.0         Tension 1 ---------------
4           Number of Points 1
 0.05297     0.31292     0.07521
 0.06125     0.30710     0.09624
 0.07875     0.30133     0.11302
 0.0875      0.30058     0.115
2.0         Tension 2 ---------------
4           Number of Points 2
 0.05297     0.36462     0.08763
 0.06125     0.35784     0.11214
 0.07875     0.35112     0.13169
 0.0875      0.35024     0.134
1.0         Tension 3 ---------------
2           Number of Points 3
 0.05297     0.31292     0.07521
 0.05297     0.36462     0.08763
1.0         Tension 4 ---------------
2           Number of Points 4
 0.0875      0.30058     0.115
 0.0875      0.35024     0.134
2.0         Tension 5 ---------------
6           Number of Points 5
 0.02946     0.31032     0.08531
 0.03417     0.30768     0.09437
 0.03888     0.30398     0.10568
 0.04359     0.29913     0.11873
 0.04830     0.29306     0.13301
 0.05297     0.28582     0.14792
2.0         Tension 6 ---------------
6           Number of Points 6
 0.02946     0.36158     0.09941
 0.03417     0.35852     0.10996
 0.03888     0.35421     0.12314
 0.04359     0.34855     0.13834
 0.04830     0.34147     0.15499
 0.05297     0.33304     0.17236
```

FIGURE 4-5. - LISTING OF 3-D GRID INPUT FILE ZONE3.DAT.

```
1.0         Tension 7 ---------------
2           Number of Points 7
 0.02946     0.31032      0.08531
 0.02946     0.36158      0.09941
1.0         Tension 8 ---------------
2           Number of Points 8
 0.05297     0.28582      0.14792
 0.05297     0.33304      0.17236
1.0         Tension 9 ---------------
2           Number of Points 9
 0.05297     0.31292      0.07521
 0.05297     0.36462      0.08763
1.0         Tension 10 --------------
2           Number of Points 10
 0.02946     0.31032      0.08531
 0.02946     0.36158      0.09941
2.0         Tension 11 --------------
4           Number of Points 11
 0.05297     0.31292      0.07521
 0.04513     0.31209      0.07858
 0.03730     0.31122      0.08194
 0.02946     0.31032      0.08531
2.0         Tension 12 --------------
4           Number of Points 12
 0.05297     0.36462      0.08763
 0.04513     0.36365      0.09156
 0.03730     0.36264      0.09548
 0.02946     0.36158      0.09941
1.0         Tension 13 --------------
2           Number of Points 13
 0.0875      0.30058      0.115
 0.0875      0.35024      0.134
1.0         Tension 14 --------------
2           Number of Points 14
 0.05297     0.28582      0.14792
 0.05297     0.33304      0.17236
2.0         Tension 15 --------------
5           Number of Points 15
 0.0875      0.30058      0.115
 0.08531     0.29856      0.12015
 0.07875     0.29207      0.13517
 0.07        0.28847      0.14268
 0.05297     0.28582      0.14792
2.0         Tension 16 --------------
5           Number of Points 16
 0.0875      0.35024      0.134
 0.08531     0.34789      0.14
 0.07875     0.34032      0.1575
 0.07        0.33613      0.16625
 0.05297     0.33304      0.17236
```

FIGURE 4-5. - CONCLUDED.

```
4           Technique
9           IL
21          JL
21          KL
3           StretchTypeXi
3           StretchTypeEta
0           StretchTypeZeta
2.0         kXi1
2.0         kXi2
2.0         kEta1
2.0         kEta2
2.0         kZeta1
2.0         kZeta2
1.01        BetaXi
1.01        BetaEta
1.05        BetaZeta
2.0         Tension 1 ---------------
3           Number of Points 1
 0.0875      0.30058      0.115
 0.105       0.30058      0.115
 0.1225      0.30058      0.115
2.0         Tension 2 ---------------
3           Number of Points 2
 0.0875      0.35024      0.134
 0.105       0.35024      0.134
 0.1225      0.35024      0.134
1.0         Tension 3 ---------------
2           Number of Points 3
 0.0875      0.30058      0.115
 0.0875      0.35024      0.134
1.0         Tension 4 ---------------
2           Number of Points 4
 0.1225      0.30058      0.115
 0.1225      0.35024      0.134
2.0         Tension 5 ---------------
6           Number of Points 5
 0.05297     0.28582      0.14792
 0.06125     0.27391      0.16897
 0.07875     0.26247      0.18623
 0.0875      0.26140      0.18773
 0.105       0.26140      0.18773
 0.1225      0.26140      0.18773
2.0         Tension 6 ---------------
6           Number of Points 6
 0.05297     0.33304      0.17236
 0.06125     0.31916      0.19688
 0.07875     0.30584      0.217
 0.0875      0.30459      0.21875
 0.105       0.30459      0.21875
 0.1225      0.30459      0.21875
```

FIGURE 4-6. - LISTING OF 3-D GRID INPUT FILE ZONE4.DAT.

```
1.0         Tension 7 ---------------
2           Number of Points 7
 0.05297     0.28582      0.14792
 0.05297     0.33304      0.17236
1.0         Tension 8 ---------------
2           Number of Points 8
 0.1225      0.26140      0.18773
 0.1225      0.30459      0.21875
1.0         Tension 9 ---------------
2           Number of Points 9
 0.0875      0.30058      0.115
 0.0875      0.35024      0.134
1.0         Tension 10 --------------
2           Number of Points 10
 0.05297     0.28582      0.14792
 0.05297     0.33304      0.17236
2.0         Tension 11 --------------
5           Number of Points 11
 0.0875      0.30058      0.115
 0.08531     0.29856      0.12015
 0.07875     0.29207      0.13517
 0.07        0.28847      0.14268
 0.05297     0.28582      0.14792
2.0         Tension 12 --------------
5           Number of Points 12
 0.0875      0.35024      0.134
 0.08531     0.34789      0.14
 0.07875     0.34032      0.1575
 0.07        0.33613      0.16625
 0.05297     0.33304      0.17236
1.0         Tension 13 --------------
2           Number of Points 13
 0.1225      0.30058      0.115
 0.1225      0.35024      0.134
1.0         Tension 14 --------------
2           Number of Points 14
 0.1225      0.26140      0.18773
 0.1225      0.30459      0.21875
1.0         Tension 15 --------------
4           Number of Points 15
 0.1225      0.30058      0.115
 0.1225      0.29207      0.13517
 0.1225      0.27619      0.16521
 0.1225      0.26140      0.18773
1.0         Tension 16 --------------
4           Number of Points 16
 0.1225      0.35024      0.134
 0.1225      0.34032      0.1575
 0.1225      0.32182      0.1925
 0.1225      0.30459      0.21875
```

FIGURE 4-6. - CONCLUDED.

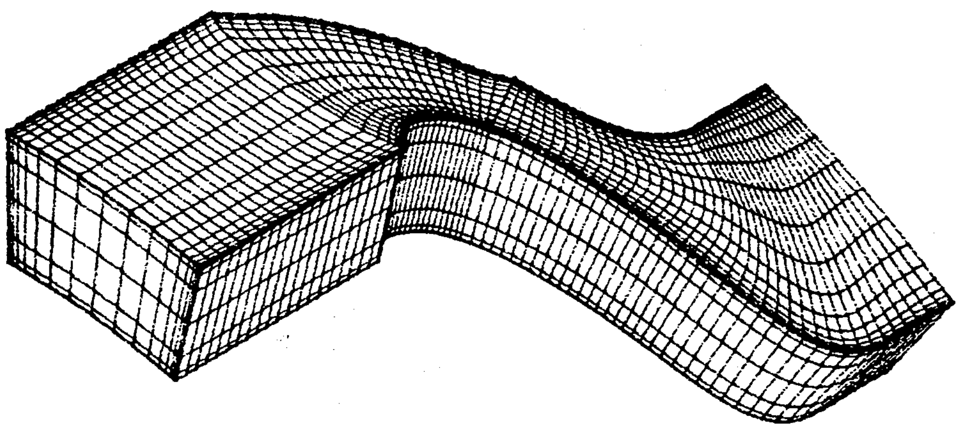


FIGURE 4-7. - GRID GENERATED BY GRID2D/3D USING 3-D GRID INPUT FILE STATOR.DAT (A COMBINATION OF FILES ZONE1.DAT, ZONE2.DAT, ZONE3.DAT, AND ZONE4.DAT).

NASA
National Aeronautics and Space Administration

Report Documentation Page

1. Report No. NASA TM-102454	2. Government Accession No.	3. Recipient's Catalog No.
4. Title and Subtitle GRID2D/3D—A Computer Program for Generating Grid Systems in Complex-Shaped Two- and Three-Dimensional Spatial Domains Part 2: User's Manual and Program Listing		5. Report Date April 1990
		6. Performing Organization Code
7. Author(s) R.T. Bailey, T.I-P. Shih, H.L. Nguyen, and R.J. Roelke		8. Performing Organization Report No. E-5242
		10. Work Unit No. 535-05-01
9. Performing Organization Name and Address National Aeronautics and Space Administration Lewis Research Center Cleveland, Ohio 44135-3191		11. Contract or Grant No.
		13. Type of Report and Period Covered Technical Memorandum
12. Sponsoring Agency Name and Address National Aeronautics and Space Administration Washington, D.C. 20546-0001		14. Sponsoring Agency Code

15. Supplementary Notes

R.T. Bailey, Dept. of Mechanical Engineering, University of Florida, Gainesville, Florida 32611; T.I-P. Shih, Dept. of Mechanical Engineering, Carnegie Mellon University, Pittsburgh, Pennsylvania 15213; H.L. Nguyen and R.J. Roelke, NASA Lewis Research Center.

16. Abstract

An efficient computer program, called GRID2D/3D, has been developed to generate single and composite grid systems within geometrically complex two- and three-dimensional (2- and 3-D) spatial domains that can deform with time. GRID2D/3D generates single grid systems by using algebraic grid generation methods based on transfinite interpolation in which the distribution of grid points within the spatial domain is controlled by stretching functions. All single grid systems generated by GRID2D/3D can have grid lines that are continuous and differentiable everywhere up to the second-order. Also, grid lines can intersect boundaries of the spatial domain orthogonally. GRID2D/3D generates composite grid systems by patching together two or more single grid systems. The patching can be discontinuous or continuous. For continuous composite grid systems, the grid lines are continuous and differentiable everywhere up to the second-order except at interfaces where different single grid systems meet. At interfaces where different single grid systems meet, the grid lines are only differentiable up to the first-order. For 2-D spatial domains, the boundary curves are described by using either cubic or tension spline interpolation. For 3-D spatial domains, the boundary surfaces are described by using either linear Coon's interpolation, bi-hyperbolic spline interpolation, or a new technique referred to as 3-D bi-directional Hermite interpolation. Since grid systems generated by algebraic methods can have grid lines that overlap one another, GRID2D/3D contains a graphics package for evaluating the grid systems generated. With the graphics package, the user can generate grid systems in an interactive manner with the grid generation part of GRID2D/3D. GRID2D/3D is written in FORTRAN 77 and can be run on any IBM PC, XT, or AT compatible computer. In order to use GRID2D/3D on workstations or mainframe computers, some minor modifications must be made in the graphics part of the program; no modifications are needed in the grid generation part of the program. This technical memorandum describes the theory and method used in GRID2D/3D. Part 1 of this technical memorandum, under a separate cover, contains the computer program, GRID2D/3D, and a user's manual.

17. Key Words (Suggested by Author(s)) Grid generation Computational fluid mechanics Turbomachinery	18. Distribution Statement Unclassified – Unlimited Subject Category 61

19. Security Classif. (of this report)	20. Security Classif. (of this page)	21. No. of pages	22. Price*
Unclassified	Unclassified	73	A04

NASA FORM 1626 OCT 86

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