IN-09-CR
185521
41P

Status Report for NASA Langley Grant NAG-1-1133, Supplement 3: Method of Characteristics Design of a Supersonic Wind Tunnel Nozzle with Square Cross-Section

Principal Investigator:
Steven P. Schneider
Assistant Professor of Aerodynamics
School of Aeronautics and Astronautics
Purdue University
West Lafayette, IN 47907-1282

Period Covered: 9/1/92 to 1/1/93

(NASA-CR-194359) METHOD OF CHARACTERISTICS DESIGN OF A SUPERSONIC WIND TUNNEL NOZZLE WITH SQUARE CROSS-SECTION, SUPPLEMENT 3 Status Report, 1 Sep. 1992 - 1 Jan. 1993 (Purdue Univ.) 41 p

N94-12824

Unclas

G3/09 0185521

Summary

Nozzle design codes developed earlier by the author under NAG-1-1133 were modified and used in order to design a supersonic wind tunnel nozzle with square cross-sections. As part of the design process, a post-processing code that generates square nozzles from the output of the Sivells code was written. The method is based on the results of an axisymmetric method of characteristics code and an axisymmetric boundary layer code. These design codes are documented in this report.

Introduction

As part of earlier work aimed at development of supersonic quiet wind tunnels with axisymmetric or 2D cross-sections, the author adapted and used the Sivells wind tunnel nozzle design code (J.C. Sivells, AEDC-TR-78-63, 1978) and the Harris boundary layer code (Harris and Blanchard, NASA TM 83207, 1982). The codes were adapted to run in Fortran-77 with simplified input/output. These codes have since been used at NASA Langley. A requirement for design of quiet nozzles with square cross-sections was later identified at Langley. The work reported on here used these existing codes to carry out an approximate method of characteristics design of two such square nozzles. The postprocessing code developed for this work is described in this report.

Description of Design Method

The square nozzles were designed based on the results of Sivells axisymmetric method of characteristics solution for a supersonic nozzle. The design method was suggested by Ivan E. Beckwith. Internal streamlines were determined for the axisymmetric nozzle. For the purpose of quiet-flow design, it is important to note that Sivell's code can produce nozzles with a section of radial flow - see figure 3 in the reference above. The Sivells code was already capable of generating internal streamlines for nozzles, although a one-line modification was necessary in order to successfully generate these streamlines for general nozzles with non-zero lengths of radial flow. The Sivells code was modified slightly to write this information to a file in a simple form to be read by a post-processing code. The streamlines of the axisymmetric flow are thus determined, along with the flow properties along them.

At the exit plane of the nozzle a square cross-section is then located within the axisymmetric flow. The streamlines associated with each point on the square cross-section are identified. These streamlines are then traced backwards towards the throat, giving at each streamwise station a new cross-section of the nozzle, which is identified. The cross-sections upstream of the exit are of course not exactly square. An approximate square is however drawn, by drawing the square with the same cross-sectional area as that of the original cross-section. This process continues back to the throat region. Thus, the post-processing code produces an array of the side length of the square versus the axial distance, for the square nozzle.

A boundary layer correction is then performed, based on the Harris code. Flow data on the streamline in the centerplane is used as input to the Harris code running in axisymmetric mode to produce boundary layer displacement thicknesses as a function of streamwise distance. These thicknesses are used as approximate thicknesses for the boundary layer at all locations in the real 3D nozzle at the given streamwise location. Thus, this thickness is just added to the half-side length of the square to give the design half-side length.

Results

The work described here produced a slightly modified Sivells code (again, modified only in terms of the input/output form, with the addition of one statement to allow correct creation of streamlines for general geometries). It also produced a Fortran-77 post-processing code which manipulates geometry, the Sivells output, and the isentropic flow relations to give square nozzle designs. The code produces files for input to the Harris code, and uses the Harris output to approximately correct the square nozzles for the boundary layer growth. It is this code that is documented in this report.

This code is currently being further modified to include a Hopkins-Hill transonic solution technique for the region upstream of the throat, to allow the design of square supersonic nozzles with an upstream bleed slot designed in the same way as earlier quiet nozzles. These modifications will be reported on at a later date.

The Appendix to this report contains the user's manual for the post-processing code, along with a brief description of how to obtain an electronic version of the source code.

Appendix

An electronic version of the source for the Fortran-77 post-processing code can be obtained from the Experimental Methods Branch at the NASA Langley Research Center, Hampton, VA, 23665. Contact Dr. Steven Wilkinson or Dr. Gregory Jones. Alternatively, contact Prof. Steven P. Schneider, Aeronautical and Astronautical Engineering, Purdue University, West Lafayette, IN 47907-1282, email `steves@ecn.purdue.edu`, telephone (317) 494-3343.

The following pages contain the manual for the post-processing code, written by Micheal J. Moen.

SNODEC: A POST PROCESSING CODE FOR THE DESIGN OF SLOW-EXPANSION SQUARE SUPERSONIC NOZZLES

Michael J. Moen, M.S.

Purdue University, West Lafayette, IN

November 15, 1992

Abstract

A computer code is written to aid in the design of a square supersonic nozzle. The code utilizes the output of an existing method of characteristic code that was written by Sivells [6] to design axisymmetric nozzles. Sivells' code is run in an inviscid mode to provide a record of streamlines. A square cross-section is defined in the nozzle exit plane and the streamlines that lie on the square wall are tracked upstream to the nozzle throat. Some bowing in the wall cross-sections is seen to develop during the tracking process. To account for the area change due to the wall bowing, the cross-sections are made perfectly square through a numerical integration. The boundary layer code called VGBLP written by Harris and Blanchard [3] is then used to estimate the boundary layer thickness based on the square dimensions and the corresponding δ^* is added to the wall dimensions to provide a final nozzle geometry. Every cross-section of the nozzle geometry is square and the code outputs the geometry as the wall height from the nozzle centerline for stations along the streamwise axis.

Introduction

In order to design a suitable nozzle for a high-speed, low disturbance wind tunnel, high quality mean flows as well as very low levels and low frequencies of fluctuating disturbances must be achieved. One way of obtaining such flow conditions is to maintain a laminar boundary layer on the nozzle walls as far downstream as is possible. In order to maintain a laminar boundary layer in rapid expansion nozzles, wall smoothness must be maximized and machining tolerances must be kept very tight. In the case of axisymmetric nozzles, the machining tolerances and wall smoothness required to obtain a test rhombus of useful size can be prohibitively expensive depending upon the material chosen for the nozzle. One proposed way of improving the size of the test rhombus in a high-speed tunnel without having to pay the price in machining tolerances is to utilize a three-dimensional slow-expansion nozzle design. Design of a slow-expansion nozzle is achieved by first designing a nozzle to meet a specific Mach distribution in the inviscid region. The inviscid contour is then corrected by adding on the displacement thickness of the boundary layer. The three-dimensional nozzle provides another improvement in the tunnel capabilities by potentially decreasing the effect of wall waviness on boundary layer transition thereby increasing the region of uniform flow in the test section.

This work is interested in developing a code that is useful in aiding the design of slow-expansion, three-dimensional square nozzles. A two-dimensional and axisymmetric design code based on the Method of Characteristics was written in 1978 by Sivells [6] for Arnold Engineering Development Center (AEDC). While this code is not capable of designing square nozzle geometries, it can provide the streamline locations for an axisymmetric tunnel and these streamlines can then be used to approximate a square nozzle design. By defining a square cross-section in the exit plane of an axisymmetric nozzle, the streamlines located on the square wall section can be tracked all the way back to the nozzle throat and beyond. The cross-sections that result from the tracking process typically have some amount of bowing in the wall. The bowing in the wall can be eliminated by numerically integrating the cross-sectional area and adjusting the geometry to correspond to a perfectly square cross-section with an identical area. Next, the boundary layer code written by Harris and Blanchard [3] called VGBLP is used to calculate the boundary layer displacement thickness δ^* for the nozzle wall and this displacement thickness is added to the inviscid contour. The result of the design process is a square nozzle geometry that should match the Mach number requirement of the axisymmetric design. The reliability of the calculation and flow field will be determined through a parabolized Navier-Stokes code by Korte and McRae [4].

Further benefits in designing a square supersonic nozzle can be seen in that there is no complex geometry transitioning from a axisymmetric nozzle to a square test section. Square test sections are much better from the standpoint of optical flow visualization and wall waviness problems. Though vortices are known to restrict the size of the uniform flow test region somewhat, the use of the slow-expansion design should alleviate the effects of these vortices.

Using The Code

The SNODEC code is intended to be used with two other previously developed codes. One code, referred to as Sivells' code [6], is used to produce an inviscid axisymmetric nozzle contour with continuous curvature that has uniform parallel flow at the nozzle exit. The other code called VGBLP, written by Harris and Blanchard [3], calculates the boundary layer growth on the contour and this correction is added to the wall dimension to create a slow-expansion nozzle. The SNODEC code was written as a patch to derive the square nozzle geometry from the axisymmetric geometry produced by Sivells' code and then utilize the boundary layer correction to create a three-dimensional nozzle geometry. The standard procedure of operations is stated as follows:

1. Run Sivells' code to design the desired inviscid axisymmetric nozzle contour.

2. Run SNODEC to define the square cross-section in the exit plane and then track the streamlines up the x axis to produce square cross-sections.

3. Run VGBLP to calculate the required boundary layer thickness correction for the contour.

4. Run SNODEC to adjust the square cross-sections to provide a final three-dimensional nozzle geometry.

The process for running Sivells' code and VGBLP are described in their separate manuals [6, 3]. SNODEC will search for, or provide the necessary files in order to run efficiently with the other two codes. When the Sivells' code has been run, the streamline output is written to a file with the name *.bl (* indicates a file root name). Before SNODEC can be run, this extension must be changed to *.st. When SNODEC is executed, the user is first prompted for the file root name that all data files should be named with:

```
What is the root name of the file to be converted?
```

The user responds by typing in the file root name of the *.st file. Next the user is prompted with:

```
How many streamlines were tracked on the half wall?
```

When the input file was created for Sivells' code, the user was asked to specify a number of streamlines that should calculated. The number that was specified for the Sivells' code input file should be entered in response to this question. Next, the user is prompted with:

```
How many cross sections downstream of the
throat should be calculated?
```

The number that the user types in defines the number of cross-sections that will be calculated in between the nozzle throat and the nozzle exit plane. In addition, the step size that is defined by determining the number of cross-sections between the throat and exit plane is applied to calculate how many cross-sections upstream of the throat will be calculated by SNODEC. The distance upstream of the throat that SNODEC calculates is controlled within the source code with the variable `rtup`. By default, SNODEC is set to calculate additional cross-sections one throat radius upstream of the throat (`rtup=1.0`). Due to poor accuracy of the transition region solution in Sivells' code, cross-sections further than one throat radius upstream of the throat may contain erroneous data. Therefore, it is suggested to use `rtup` as it has been set in the source code. Furthermore, it is advised that no more than 200 cross-sections be calculated in the region between the throat and the exit plane. It is possible to calculate more cross-sections, but too many points will strain the limits of accuracy for the interpolation procedure that is used to locate and calculate streamwise points.

Once this data has been entered, SNODEC creates a file with the extension *.stp that contains the streamline data that will be interpolated for each streamwise cross-section. Next, the program presents five options for operation. Options 1 through 4 must be executed in the order that they appear because each option creates files that are used by the following options. The options are presented as:

```
Choose an option:

(1) Interpolate streamlines to uniform stations
    and write axisymmetric geometry
(2) Convert axisymmetric geometry to square
    geometry and write cross sectional info
(3) Perform numerical integration to square
    off cross sections
(4) Make boundary layer correction to cross
    sections (requires that Harris code be run
    using previously created *.bl output file)
(5) Exit
```

When option 1 is chosen, SNODEC reads in the streamline data from *.st and interpolates the values of Y and P_e/P_o for each cross-section X along each streamline. The interpolation data is then written to a file called *.int. In addition to the interpolation file, a file is also written to be used for the VGBLP program called *.bl, as well as a file containing data for the Navier-Stokes calculation called *.ccr. Option 1 also writes a file called *.cir containing the data for the axisymmetric nozzle geometry as produced by Sivells' code.

When option 2 is chosen next, SNODEC takes the interpolated data and creates a file called *.noz containing the data for the three-dimensional geometry with square cross-sections that results from the interpolation and streamline tracking process. Option 2 also creates several files for a more in depth analysis of the resulting geometry. These files include a file called *.csa that contains the geometry of each individual cross-section, a file called *.vpc that shows the pressure distribution along the half wall of each cross-section, a file called *.vpm that represents the pressure contours along the nozzle half wall, a file called *.pdf that contains data for the pressure difference in between the wall centerline streamline and the wall corner streamline along the streamwise direction and a file called *.zdf that shows the difference in between the height of the wall centerline streamline and the wall corner streamline along the streamwise direction.

When option 3 is chosen, SNODEC numerically integrates each cross-section with a simple trapezoidal rule in order to calculate the area. The dimensions of each cross-section are then adjusted so that the cross-section retains the same area, but has the dimensions of a perfect square. Option 3 writes the dimensions of the newly squared cross-sections to a file called *.csb.

Before option 4 is chosen, SNODEC must be exited using option 5. At this point, VGBLP can be run using the *.bl file that was produced earlier with option 1. Once VGBLP is executed and the output is obtained in the *.prt file, SNODEC should be executed again. After answering the initial questions in SNODEC, options 1 through 3 can be bypassed to choose option 4. Option 4 takes the file *.prt from VGBLP, interpolates it and adds the δ^* correction on to the wall geometry from the file *.csb. This final data is written to a file called *.csc in the form of a wall profile as well as a file called *.csd in the form of individual cross-section geometries. Following this, SNODEC may be exited.

SNODEC Output Files

Most of the output files that are created by SNODEC are written in a specific format so that they may be plotted using the Tecplot program. The following text describes the format and content of each file in the order they are created or used by SNODEC. In all cases except for where specified, the coordinates are normalized as follows.

$$X=x/r_{throat} \qquad Y=y/r_{throat} \qquad Z=z/r_{throat} \qquad R=r/r_{throat} \qquad \Theta=\theta$$

The coordinates are shown in Figure 1.

{filename}.st contains the raw streamline data for each streamline as well as the wall geometry as produced by Sivells' code. The streamwise X stations for each streamline are not aligned to each other so interpolation is required.

{filename}.stp contains the streamline data for each streamline as well as the wall geometry as produced by Sivells' code. This data file is a version of {filename}.st that can be plotted with Tecplot in order to check the Sivells' code output (see Figure 2). Each streamline is represented in X-R coordinates. This file contains no information on the angular component of the streamline locations.

{filename}.int contains the streamline data in its interpolated form. The file has data for the specified number of streamlines at the specified number of cross-sections in X-R-Θ coordinates where Θ is in degrees. This file is used by other subroutines to provide the interpolated data.

{filename}.ccr contains a record of a streamline along the axisymmetric geometry wall that can be used for a Navier-Stokes calculation. The data represents the streamline with the X-R coordinates This file can be plotted with Tecplot.

{filename}.bl contains data on the three-dimensional centerwall streamline along with the data P_e/P_o for the boundary layer calculation. Because the Cartesian Y component of the centerline streamline is zero, data is provided for only the X and Z components of the streamline. The file format is set to be compatible with the VGBLP program.

{filename}.cir contains the axisymmetric nozzle geometry in X-Y-Z coordinates based on the output of the Sivells' code. This file can be plotted with Tecplot (see Figure 3). It includes data on P_e/P_o so that pressure contours may be plotted on the nozzle geometry.

{filename}.noz contains the three-dimensional nozzle geometry in X-Y-Z coordinates before wall bowing correction and boundary layer correction. This file can be plotted with Tecplot (see

Figure 4). It includes data on P_e/P_o so that pressure contours may be plotted on the nozzle geometry.

{filename}.csa contains the Y and Z coordinates for the cross-section at each X location of the three-dimensional nozzle to show the amount of wall bowing that is occurring. This file can be plotted with Tecplot (see Figure 5).

{filename}.vpc contains the value of $(P_e-P_c)/P_o$ along the half wall of the cross-section at each X location where the distance to the wall is represented as Y/Y_{max}. This file can be plotted with Tecplot (see Figure 6).

{filename}.vpm contains the value of $(P_e-P_c)/P_o$ for the half wall mesh. This differs from the three-dimensional nozzle contour data in that the Z data is not included. The value of $(P_e-P_c)/P_o$ replaces the Z value for each X-Y location instead. This file can be plotted with Tecplot to physically show the non-dimensional pressure contour along the half wall (see Figure 7).

{filename}.zdf contains data for the difference in the altitude Z for the streamlines located at the wall centerline and the wall corner along the streamwise axis X. This gives an indication to the amount of wall bowing that is occurring at each cross-section. This file can be plotted with Tecplot (see Figure 8).

{filename}.pdf contains data for the difference in the pressure ratio P_e/P_o for the streamlines located at the wall centerline and the wall corner along the streamwise axis X. This gives an indication of the uniformity of the pressure distribution in the nozzle exit as well as each cross-section (see Figure 9).

{filename}.csb contains data for the nozzle contour with the wall bowing correction using only the X and Z components. This file is later combined with the boundary layer correction data to produce the final nozzle geometry. This file can be plotted with Tecplot.

{filename}.re contains the Reynolds' number information that is used to scale the boundary layer output in VGBLP and in the case of SNODEC, the geometry output.

{filename}.prt contains the boundary layer thickness correction δ^* for the nozzle wall contour. This data is interpolated to match the cross-section locations already in use by SNODEC. The data is in the dimension of feet.

{filename}.csc contains data for the three-dimensional geometry with both the wall bowing correction and the boundary layer correction. This is the final form of the nozzle geometry. The data is represented as the X streamwise location and the wall centerline height Z as measured from the nozzle X axis. For this file, the geometry is in the dimension of feet. This file can be plotted as a wall profile with Tecplot (see Figure 10).

{filename}.csd contains data of the Y and Z coordinates of the cross-sections along the streamwise X locations for the three-dimensional geometry with both the wall bowing correction and the boundary layer correction. For this file, the geometry is in the dimension of feet. This file can be plotted with Tecplot (see Figure 11).

References

1. Beckwith, I.E., Ridyard, H.W.; "The Aerodynamic Design of High Mach Number Nozzles Utilizing Axisymmetric Flow With Application to a Nozzle of Square Test Section"; NACA-TN-2711, 1952.

2. Beckwith, I.E., Chen, F.J., Wilkinson, S.P., Malik, M.R., Tuttle, D.G.; "Design and Operational Features of Low-Disturbance Wind Tunnels at NASA Langley for Mach Numbers from 3.5 to 18"; AIAA-90-1391, 1990.

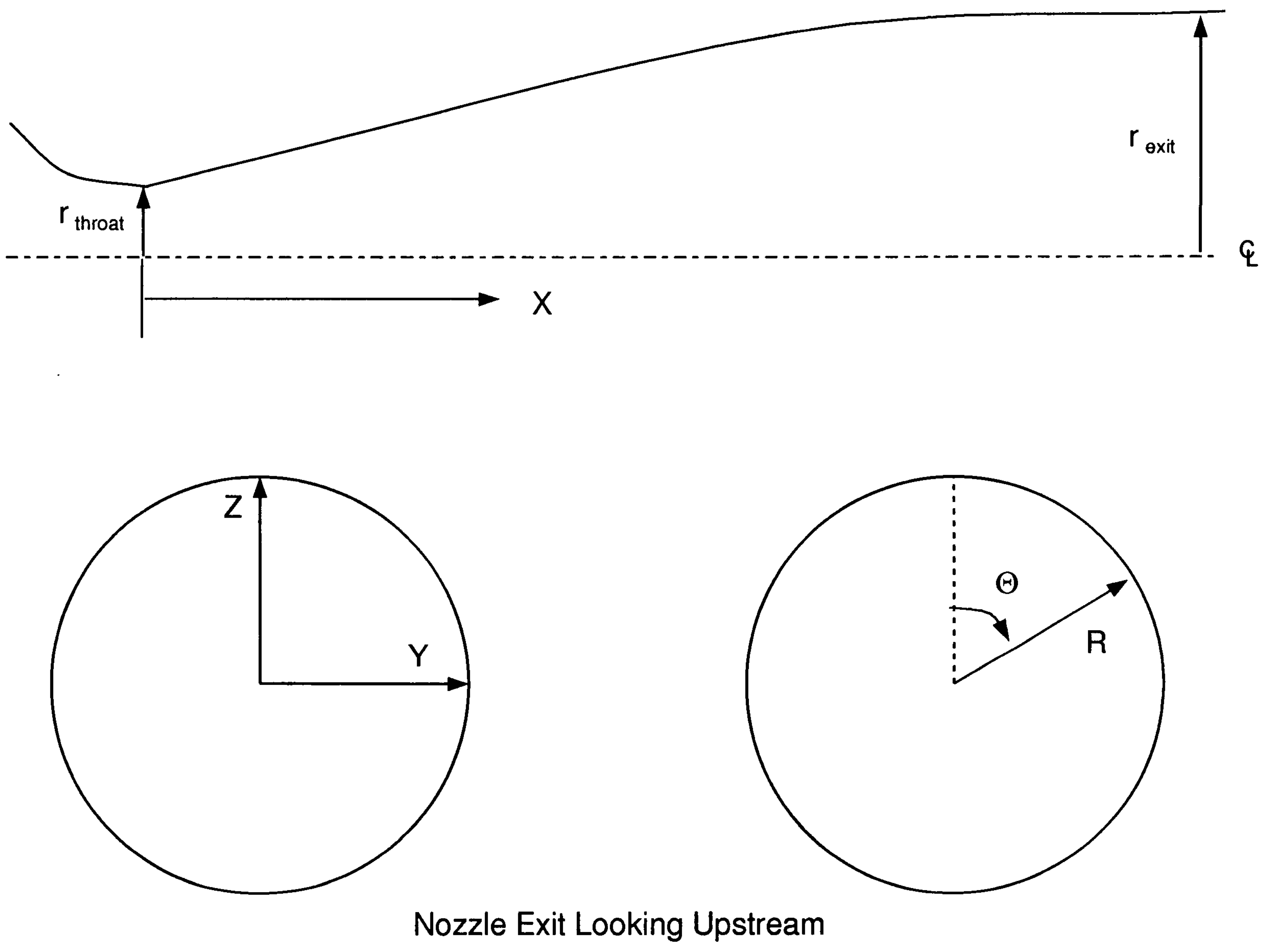
3. Harris, J.E., Blanchard, D.K.; "Computer Program for Solving Laminar, Transitional, or Turbulent Compressible Boundary-Layer Equations for Two-Dimensional and Axisymmetric Flow"; NASA-TM-83207, 1982.

4. Korte, J.J., McRae, D.S.; "Explicit Upwind Algorithim for the Parabolized Navier-Stokes Equations"; AIAA-88-0716, 1988.

5. Sivells, J.C.; "Aerodynamic Design of Axisymmetric Hypersonic Wind Tunnel Nozzles"; Journal of Spacecraft, V. 7, No. 11, November 1970, pp. 1292-1299.

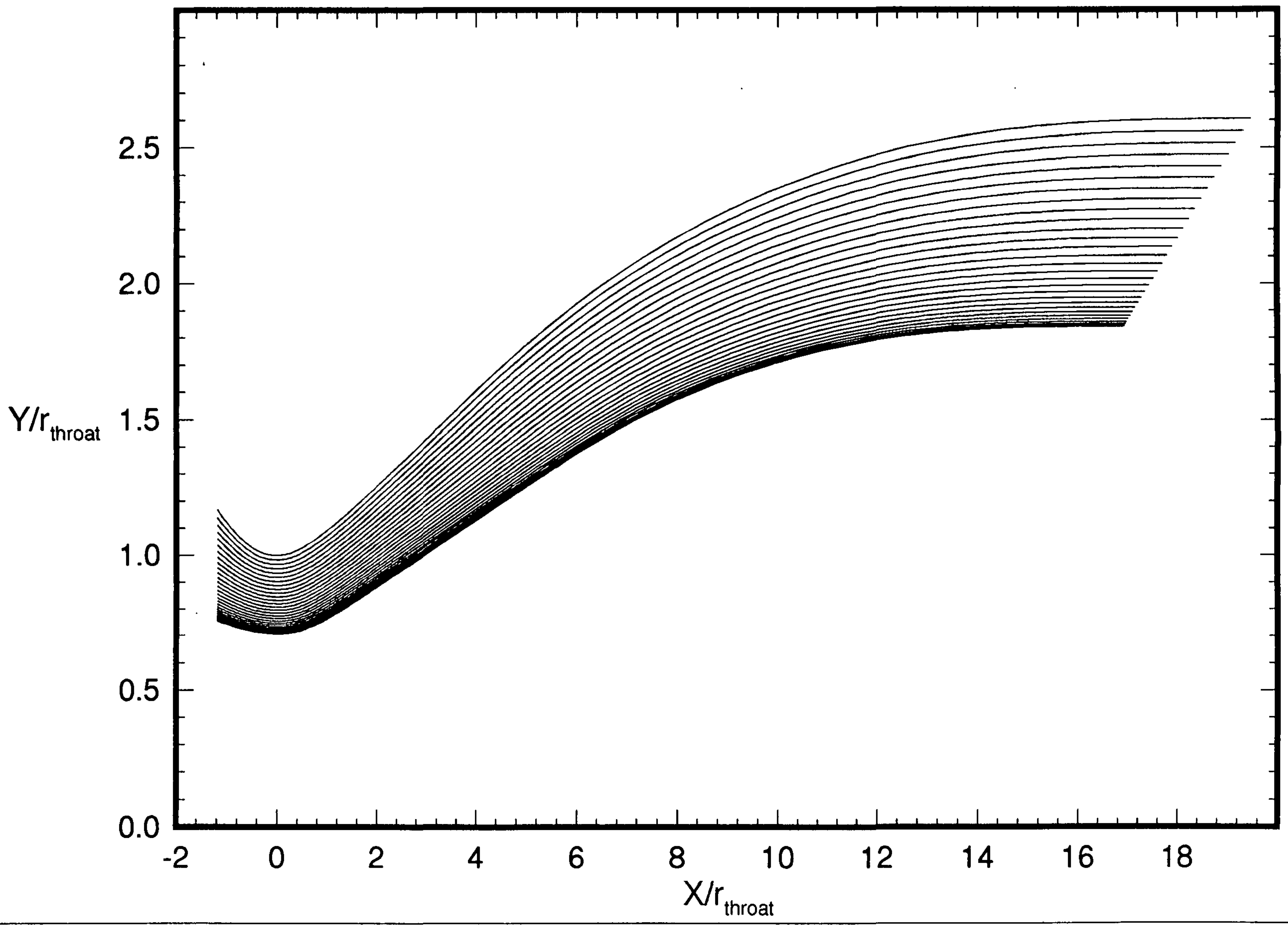
6. Sivells, J.C.; "A Computer Program for the Aerodynamic Design of Axisymmetric and Planar Nozzles for Supersonic and Hypersonic Wind Tunnels"; AEDC-TR-78-63, 1978.

Fig. 1 Coordinate Systems Used By SNODEC



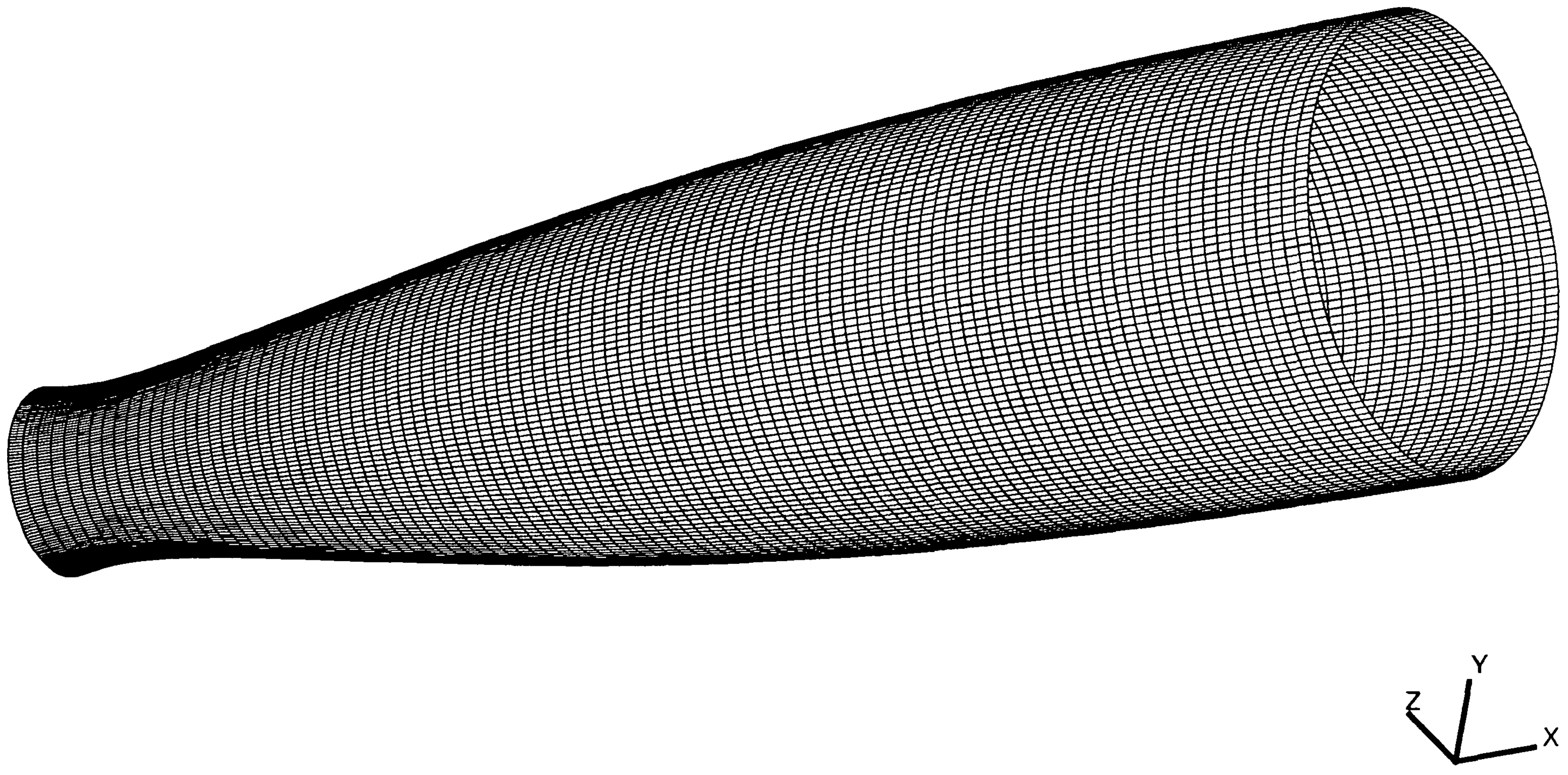
(2D) || Print || stp.plt || Streamline Plot For Supersonic Nozzle

Fig. 2 Mach 3.5 Axisymmetric Nozzle Streamlines Before Interpolation



(3D) || Print || cir.plt || Axisymmetric Nozzle

Fig. 3 Mach 3.5 Axisymmetric Nozzle Before 3-D Conversion



(3D) || Print || noz.plt || Wall Mesh With Pressure Contour

Fig. 4 Mach 3.5 3-D Nozzle Resulting From Streamline Tracking Conversion

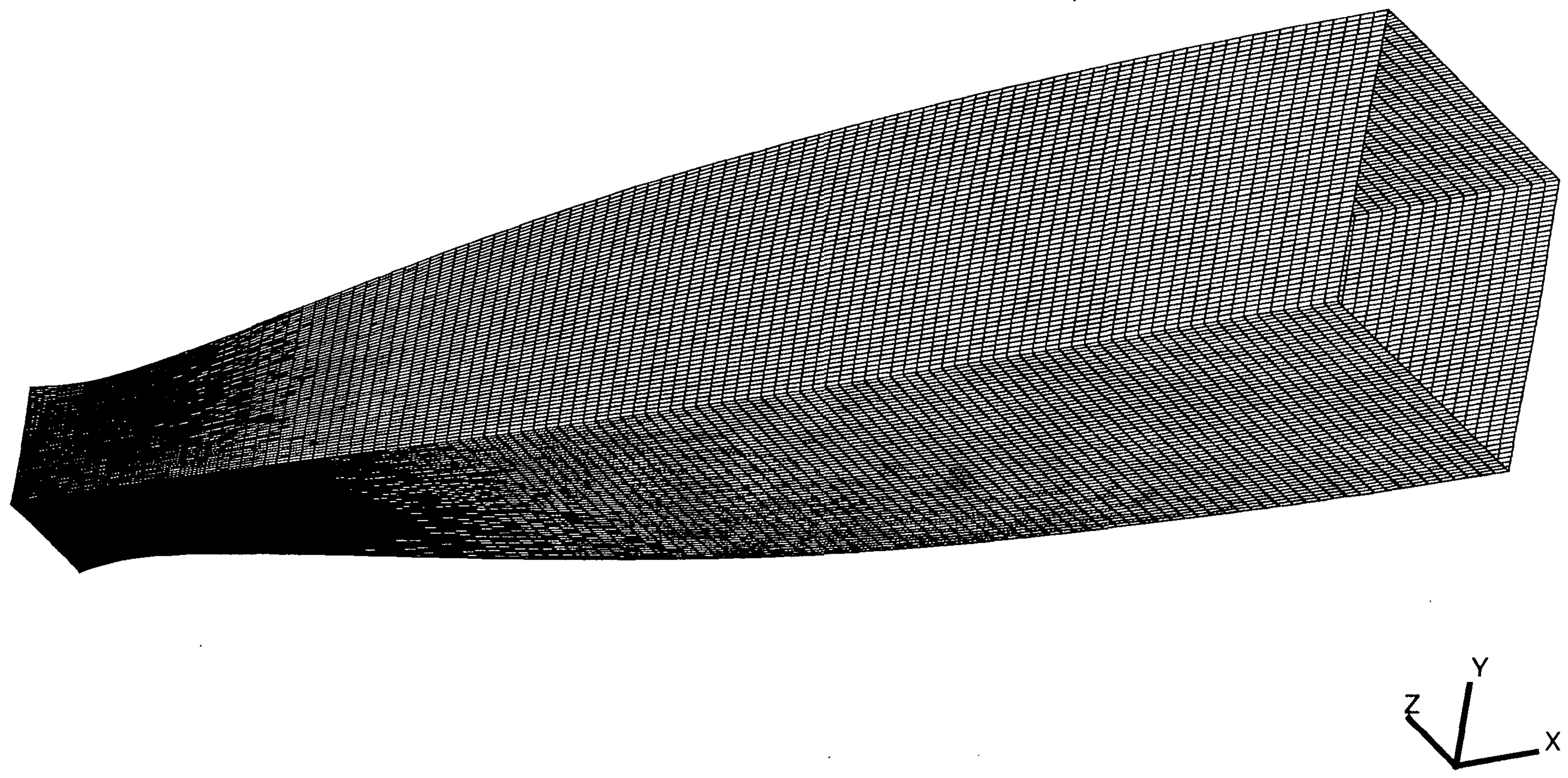


Fig. 5 Mach 3.5 3-D Nozzle Cross Sections Before Wall Bowing And Boundary Layer Corrections

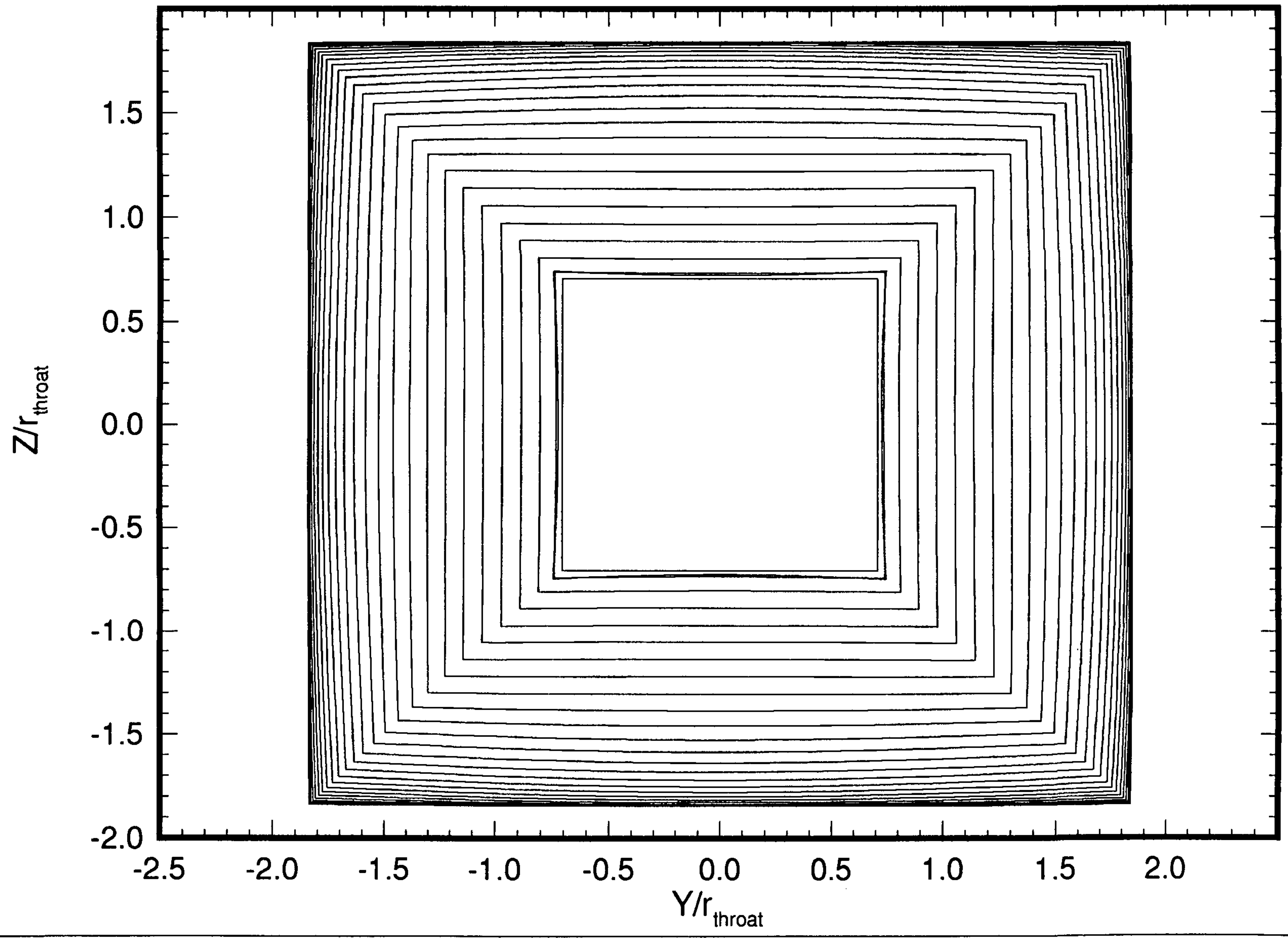
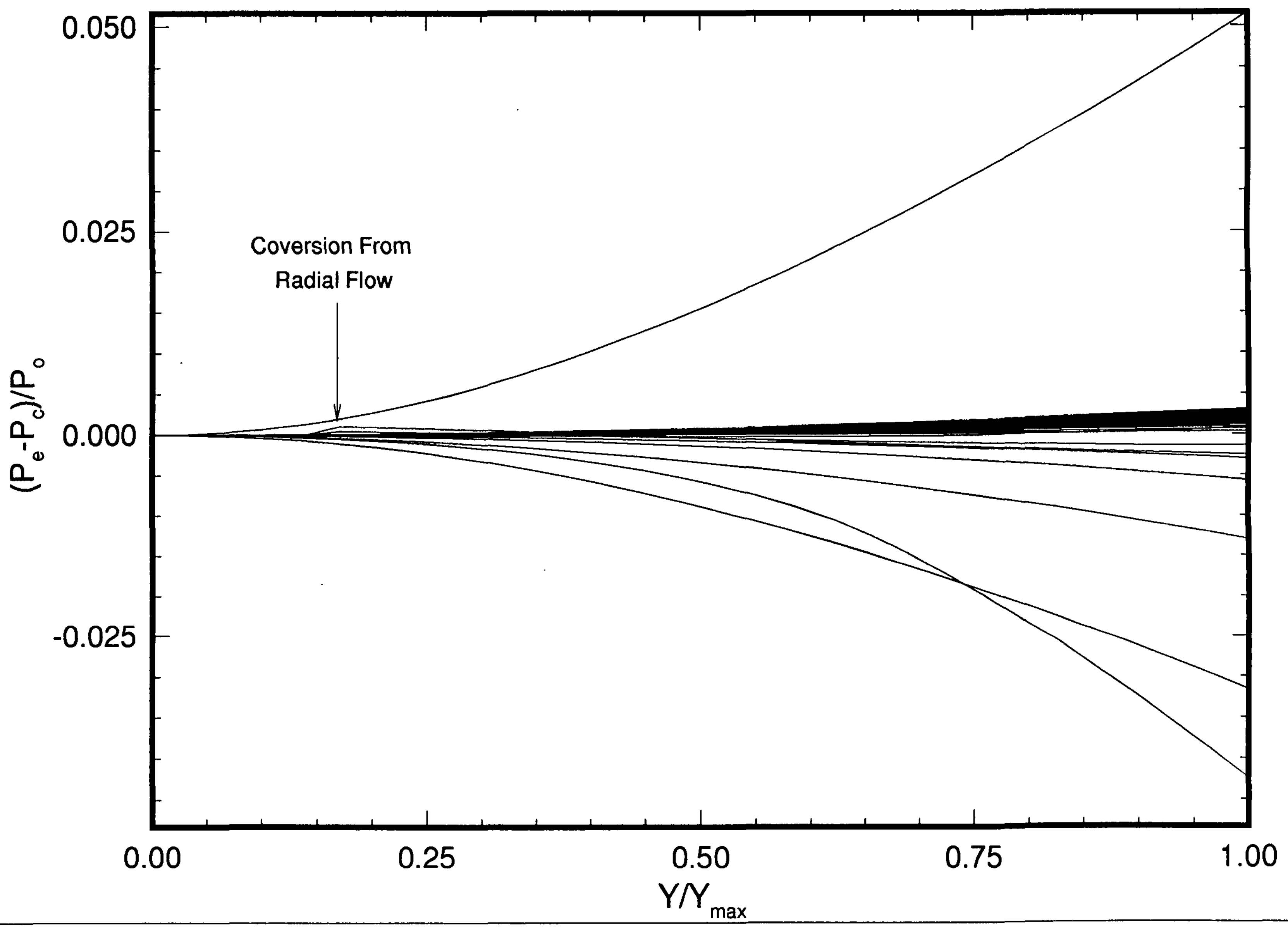


Fig. 6 Pressure Profiles On The Mach 3.5 3-D Nozzle Half Wall Cross Sections



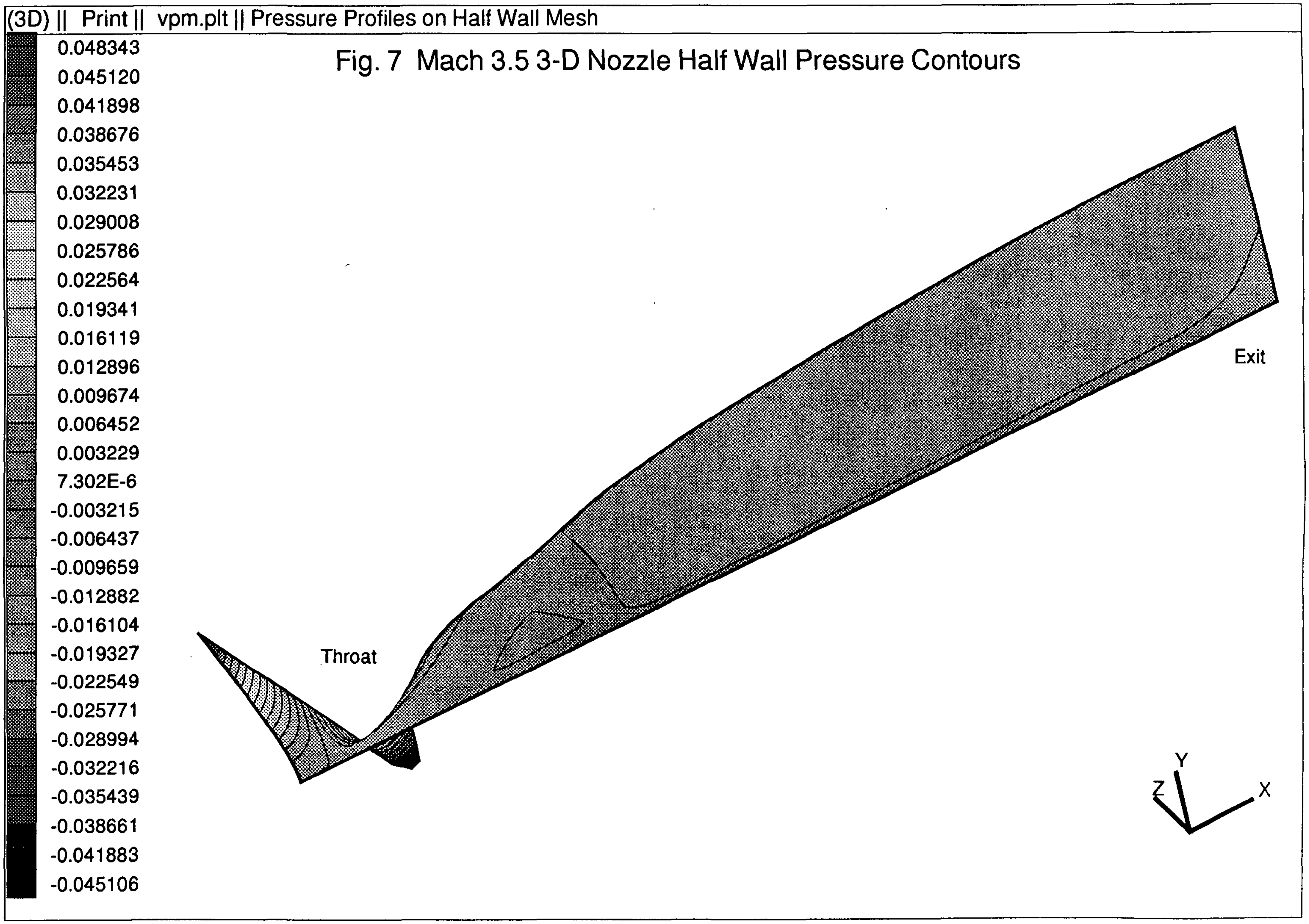
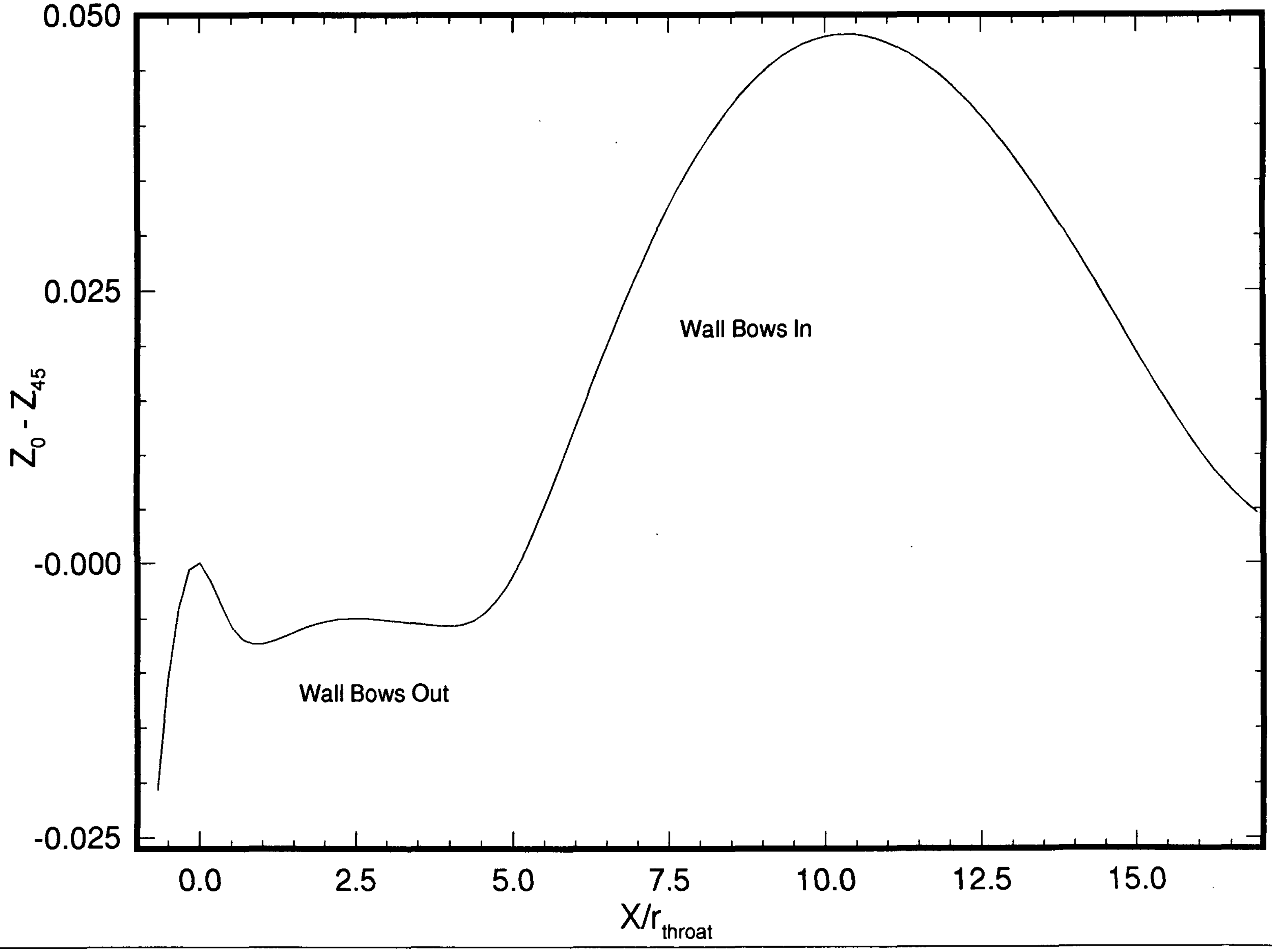


Fig. 7 Mach 3.5 3-D Nozzle Half Wall Pressure Contours

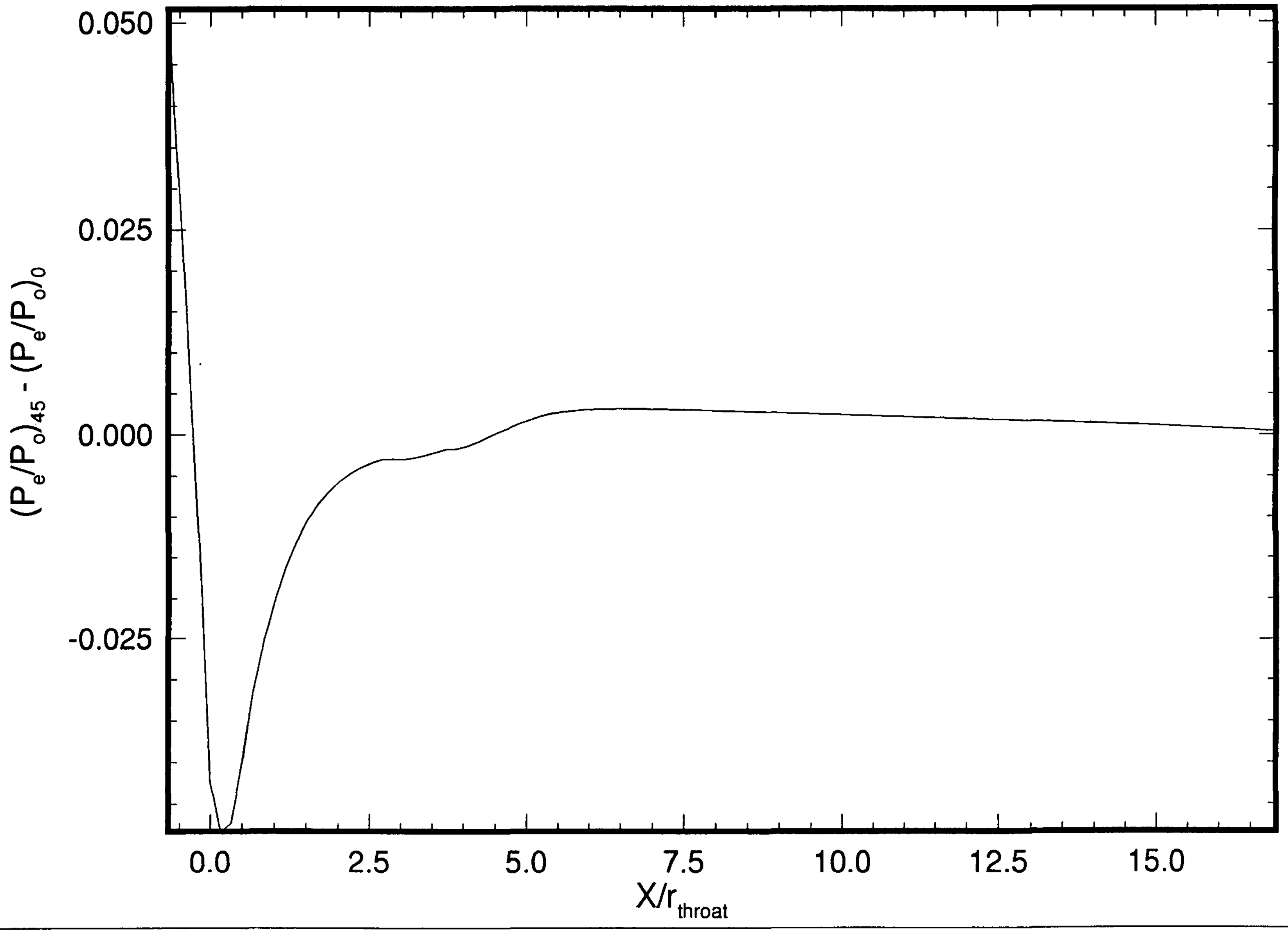
(2D) || Print || zdf.plt || Wall Bowing Along Nozzle Axis

Fig. 8 Wall Bowing Due To Streamline Tracking of Square Geometry



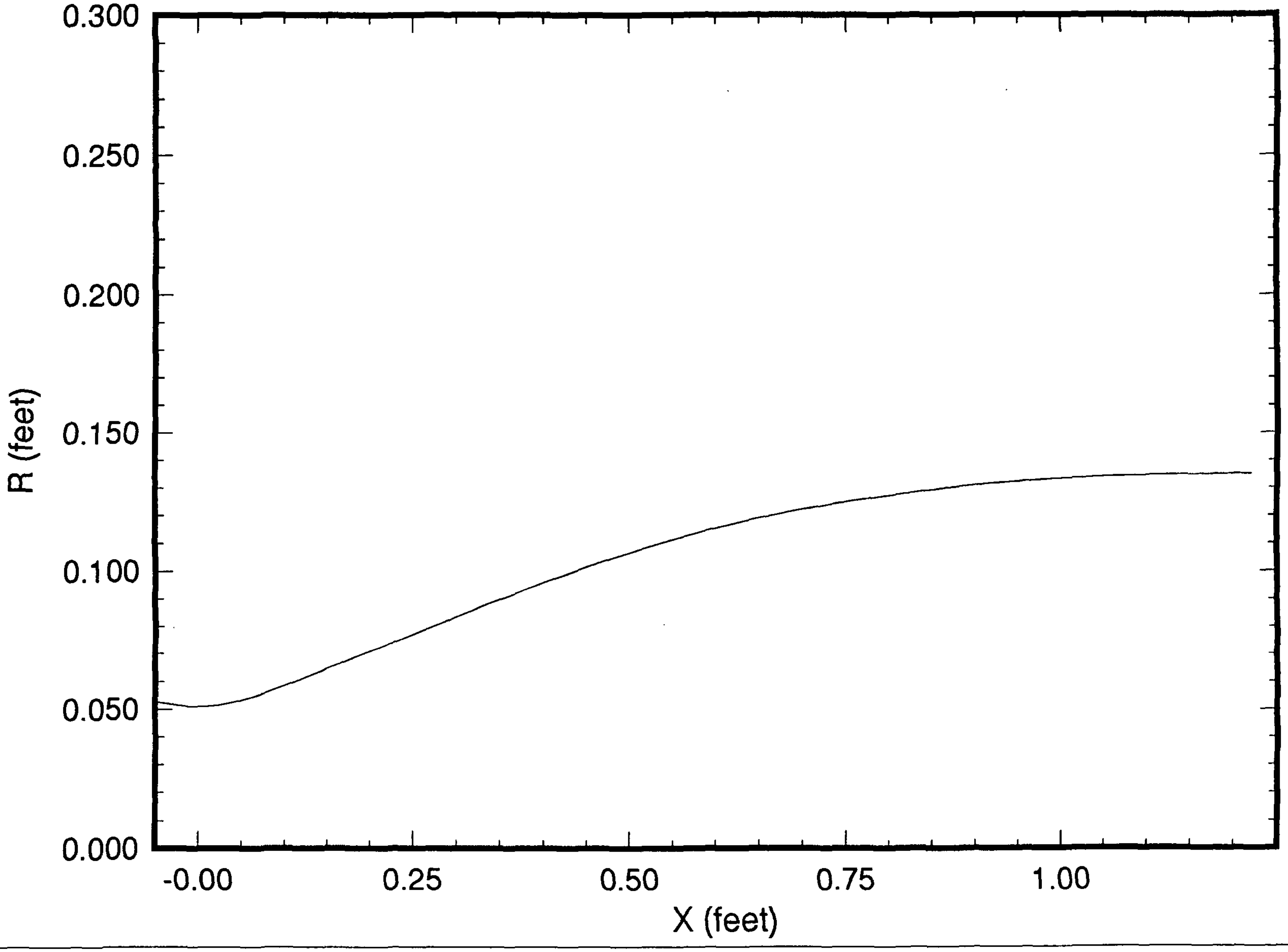
(2D) || Print || pdf.plt || Wall Pressure Differentials

Fig. 9 Pressure Differential on Half Wall For Streamwise Cross Sections



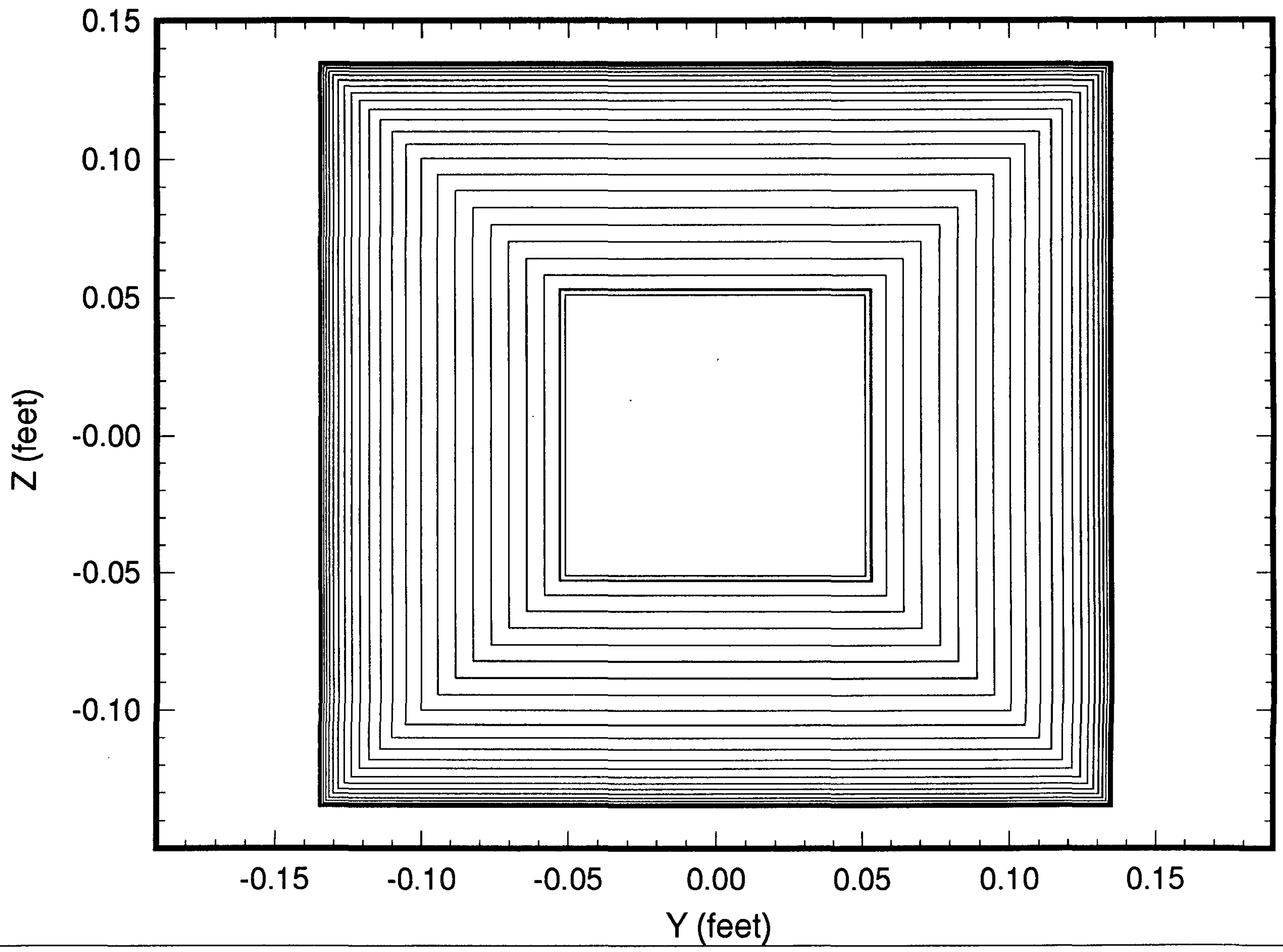
(2D) || Print || csc.plt || Squared, B.L. Corrected Nozzle Profile

Fig. 10 Mach 3.5 3-D Nozzle Wall Profile With Squared Cross Sections And Boundary Layer Correction



(2D) || Print || csd.plt || Squared, B.L. Corrected Cross Sections

Fig. 11 Mach 3.5 3-D Nozzle Squared Cross Sections With Boundary Layer Correction



SNODEC Source Code

```
c    This program takes in streamline data from Sivells' code
c    and converts converts an axisymmetric inviscid nozzle design
c    into a square three dimensional nozzle.  It also uses output
c    from Harris and Blanchard's VGBLP program to make the
c    boundary layer correction to the wall geometry.  The resulting
c    data files can be plotted using Tecplot.
c    Written by Michael J. Moen 10-1-92.
c
c    Begin main program SNODEC (square nozzle design code)
c
c    Set up program and prompt for geometry calculation data.
c    Istrm is entered as the number of streamlines calculated
c    by the Sivells code. Icrss is entered as the number of
c    cross sections wanted past the throat, but the program
c    will adjust to calculate more cross sections so the the
c    transonic region upstream of the throat may be included.
c    Before this program is run, the Sivells' *.bl file must be
c    renamed to *.st.
c
     implicit real(a-h,l-z)
     integer iopt
     character*20 iname
     common istrm,icrss,icr
     write(*,*) 'What is the root name of the file to be converted?'
     read(*,10) iname
     write(*,*) 'How many streamlines were tracked on the half wall?'
     read(*,*) istrm
     write(*,*) 'How many cross-sections downstream of the'
     write(*,*) 'throat should be calculated?'
     read(*,*) icrss
     call rawtotp(iname)
c
c    Presents options for conversion.
c
   1 write(*,*) ' '
     write(*,*) 'Choose an option:'
     write(*,*) ' '
     write(*,*) '(1) Interpolate streamlines to uniform stations'
     write(*,*) '    and write axisymmetric geometry'
     write(*,*) '(2) Convert axisymmetric geometry to square'
     write(*,*) '    geometry and write cross sectional info'
     write(*,*) '(3) Perform numerical integration to square'
     write(*,*) '    off cross sections'
     write(*,*) '(4) Make boundary layer correction to cross'
     write(*,*) '    sections (requires that Harris code be run'
     write(*,*) '    using previously created *.bl output file)'
     write(*,*) '(5) Exit'
     write(*,*) ' '
     read(*,*) iopt
     if(iopt.eq.1)then
       call geompp(iname)
```

```
        go to 1
      elseif(iopt.eq.2)then
        call intertotp(iname)
        go to 1
      elseif(iopt.eq.3)then
        call areacalc(iname)
        go to 1
      elseif(iopt.eq.4)then
        call blcorrect(iname)
        go to 1
      elseif(iopt.eq.5)then
        go to 500
      elseif(iopt.lt.1.or.iopt.gt.5)then
        write(*,*) ''
        write(*,*) 'Not an option.  Pick again.'
        write(*,*) ''
        go to 1
      endif
  500 stop
   10 format(a20)
      end


ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c                                                             c
c    This makes a preliminary file of streamlines called      c
c    *.stp, which can be used by tecplot.                     c
c                                                             c
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc


      subroutine rawtotp(iname)
      implicit real(a-h,l-z)
      integer istrm,icrss
      character*20 iname,ist,istp
      character*12 itle,inull
      common istrm,icrss,icr
      dimension x(50,500),y(50,500),peo(50,500)
      dimension ipts(50)
      ist=iname
      istp=iname
      ileng=index(iname,'  ') -1
      ist(ileng+1:ileng+3) = '.st'
      istp(ileng+1:ileng+4) = '.stp'
      open(unit=1, file=ist, status='old')
      open(unit=2, file=istp, status='unknown')
c
c    First loop reads in nozzle wall geometry.
c
      read(1,95) itle,xbin,xcin,sf,frip
      read(1,96) ipts(istrm)
      write(*,*) ''
      write(*,*) 'Reading wall geometry for',ipts(istrm),' points'
      write(2,*) 'TITLE="Streamline Plot For Supersonic Nozzle"'
      write(2,*) 'VARIABLES="X","R","Pe/Po"'
      write(2,*) 'ZONE T="Wall Countour",I=',ipts(istrm),',J=1,F=POINT'
      do 5 j=1,ipts(istrm)
        read(1,*) x(istrm,j),y(istrm,j),peo(istrm,j)
        write(2,99) x(istrm,j),y(istrm,j),peo(istrm,j)
    5 continue
c
c     Following loops read in streamlines.
c
      do 20 i=1,istrm-1
        read(1,102) inull
        read(1,103) ipts(i)
        write(2,*) 'ZONE T="Streamline",I=',ipts(i),',J=1,F=POINT'
        write(*,*) 'Reading streamline set of',ipts(i),' points'
        do 10 j=1,ipts(i)
          read(1,*) x(i,j),y(i,j),peo(i,j)
          write(2,99) x(i,j),y(i,j),peo(i,j)
   10   continue
   20 continue
      write(*,*) ' '
      write(*,*) 'Writing ordered streamline set to ',istp
c
c     Adjust cross section to include area upstream of throat.  When
c     rtup is 1.0, the program will look one throat radius upstream of
c     the throat.  If this value were changed to 2.0, it would look
c     two throat radii upstream, etc.
c
      rtup=1.0
      xstt=x(1,ipts(1))
      xcrss=xstt/float(icrss)
      icr=icrss+aint(rtup/xcrss)
c
c     Closes the files.
c
      close(1)
      close(2)
   95 format(2x,a12,15x,f11.6,6x,f11.6,6x,f11.6,7x,f11.6)
   96 format(/ i6)
   99 format(3(1x,e14.7))
  101 format(a20)
  102 format(/ a20)
  103 format(i6)
      return
      end


cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c                                                            c
c     This option converts the *.st file into a file called  c
c     *.int, which contains the interpolated streamlines     c
c     at uniform stations.  This option also creates the     c
c     *.bl file to be used by VGBLP.                         c
c                                                            c
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
```

```
      subroutine geompp(iname)
      implicit real(a-h,o-z)
      character*20 iname,ist,iint,ibl,icir,iccr
      character*12 itle,inull
      common istrm,icrss,icr
      dimension ipts(50)
      dimension x(50,500),y(50,500),peo(50,500)
      dimension xa(3),ya(3),pa(3)
      dimension xs(50,500),ys(50,500),ps(50,500),theta(50,500)
      dimension xd(181,500),yd(181,500),zd(181,500),pd(181,500)
      ist=iname
      iint=iname
      ibl=iname
      icir=iname
      iccr=iname
      ileng=index(iname,' ') -1
      ist(ileng+1:ileng+3) = '.st'
      iint(ileng+1:ileng+4) = '.int'
      ibl(ileng+1:ileng+3) = '.bl'
      icir(ileng+1:ileng+4) = '.cir'
      iccr(ileng+1:ileng+4) = '.ccr'
      open(unit=1, file=ist, status='old')
      open(unit=2, file=iint, status='unknown')
      open(unit=4, file=ibl, status='unknown')
      open(unit=8, file=icir, status='unknown')
      open(unit=3, file=iccr, status='unkown')
c
c     First loop reads in nozzle wall geometry.
c
      read(1,200) itle,xbin,xcin,sf,frip
      read(1,201) ipts(istrm)
      do 5 j=1,ipts(istrm)
        read(1,*) x(istrm,j),y(istrm,j),peo(istrm,j)
    5 continue
c
c     Following loops read in streamlines.
c
      do 20 i=1,istrm-1
        read(1,203) inull
        read(1,204) ipts(i)
        do 10 j=1,ipts(i)
          read(1,*) x(i,j),y(i,j),peo(i,j)
   10   continue
   20 continue
c
c     Locates starting coordinates and determines size
c     of steps along x axis.
c
      xstt=x(1,ipts(1))
      ystt=y(1,ipts(1))
      write(*,*) ''
      write(*,*) 'Interpolation will track streamlines for square'
      write(*,*) 'half wall (square eighth space) starting at exit'
      write(*,*)
      write(*,*) 'Starting x-coord = ',xstt
      write(*,*) 'Starting y-coord = ',ystt
      write(*,*) ' '
      write(*,*) 'Beginning interpolation and writing to ',iint
      dz=ystt/float(istrm-1)
      xcrss=xstt/float(icrss)
c
c     Interpolates stations on x along with y and pe/po.
c
      do 40 i=1,istrm
       do 30 j=1,icr
       xloc=xstt-xcrss*(float(j)-1.0)
c
c     Locates the points bounding the x station of interest.
c
       ml=0
       mu=ipts(i)+1
   25   if(mu-ml.gt.1)then
        mm=(mu+ml)/2
        if((x(i,ipts(i)).gt.x(i,1)).eqv.(xloc.gt.x(i,mm)))then
          ml=mm
        else
          mu=mm
        endif
       go to 25
       endif
       m=ml
c
c     Set up an array to feed in points for interpolation.
c
       xa(1)=x(i,m)
       xa(2)=x(i,m+1)
       xa(3)=x(i,m+2)
       ya(1)=y(i,m)
       ya(2)=y(i,m+1)
       ya(3)=y(i,m+2)
       pa(1)=peo(i,m)
       pa(2)=peo(i,m+1)
       pa(3)=peo(i,m+2)
       if(xa(2).eq.0.0.and.xa(3).eq.0.0)then
        k=1
       elseif(xa(2).gt.0.0.and.xa(3).eq.0.0)then
        k=2
       else
        k=3
       endif
c
c     Interpolate points on each streamline.
c
       thet=atan(dz*(float(i)-1.0)/ystt)
       theta(i,j)=thet*180.0/3.14159
       call polint(xa,ya,k,xloc,yloc,dy)
```

```
         xs(i,j)=xloc
         ys(i,j)=yloc
         call polint(xa,pa,k,xloc,ploc,dp)
         ps(i,j)=ploc
         write(2,205) xs(i,j),ys(i,j),theta(i,j),ps(i,j)
  30   continue
  40 continue
c
c    Write axisymmetric profile to a file
c    for a Navier-Stokes calculation.
c
      write(*,*) ''
      write(*,*) 'Writing axisymmetric wall streamline to ',iccr
      write(*,*) 'for Navier Stokes calculation'
      write(3,*) 'TITLE="Navier-Stokes Calculation Data"'
      write(3,*) 'VARIABLES="X","R"'
      write(3,*) 'ZONE T="Axisymmetric",I=',icr,',J=1,F=POINT'
      do 50 j=icr,1,-1
        write(3,205) xs(i,j),ys(i,j)
  50 continue
c
c    Write points for the center wall streamline to a
c    separate file called *.bl to be used for calculating
c    the boundary layer correction.
c
      write(*,*) ''
      write(*,*) 'Writing center wall streamline to ',ibl
      write(*,*) 'for VGBLP boundary layer calculation'
      write(4,*) 'This file is used by the Harris code to calculate'
      write(4,*) 'the streamline shift for the square nozzle'
      write(4,*) icr
      do 70 j=icr,1,-1
        write(4,206) xs(1,j),ys(1,j),ps(1,j)
  70 continue
c
c    Write points out to a axisymmetric nozzle file called
c    *.cir just for looks.
c
      write(*,*) ''
      write(*,*) 'Writing data to ',icir
      write(*,*) 'for axisymmetric nozzle geometry'
      write(8,*) 'TITLE="Axisymmetric Nozzle"'
      write(8,*) 'VARIABLES="X","Y","Z","Pe/Po"'
      write(8,*) 'ZONE T="Nozzle",I=181,J=',icr,',F=POINT'
      do 90 j=1,icr
        do 80 i=1,181
          xd(i,j)=xs(istrm,j)
          yd(i,j)=ys(istrm,j)*sin(2.0*(float(i)-1.0)*3.14159/180.)
          zd(i,j)=ys(istrm,j)*cos(2.0*(float(i)-1.0)*3.14159/180.)
          pd(i,j)=ps(istrm,j)
          write(8,*) xd(i,j),yd(i,j),zd(i,j),pd(i,j)
  80   continue
  90 continue
c
```

```
c   Closes the files.
c
    close(1)
    close(2)
    close(3)
    close(4)
    close(8)
 200 format(2x,a12,15x,f11.6,6x,f11.6,6x,f11.6,7x,f11.6)
 201 format(/ i6)
 202 format(a20)
 203 format(/ a20)
 204 format(i6)
 205 format(4(1x,e14.7))
 206 format(3(1x,e14.7))
    return
    end


cccccccccccccccccccccccccccccccccccccccccccccccccccccc
c                                                    c
c   Subroutine polint does the interpolation on the  c
c   streamline to give y and pe/po data for each x.  c
c   Based on Numerical Recipes routine.              c
c                                                    c
cccccccccccccccccccccccccccccccccccccccccccccccccccccc


    subroutine polint(xa,ya,n,x,y,dy)
    parameter (nmax=10)
    dimension xa(n),ya(n),c(nmax),d(nmax)
    ns=1
    dif=abs(x-xa(1))
    do 10 j=1,n
      dift=abs(x-xa(j))
      if(dift.lt.dif)then
        ns=j
        dif=dift
      endif
      c(j)=ya(j)
      d(j)=ya(j)
  10 continue
    y=ya(ns)
    ns=ns-1
    do 30 m=1,n-1
      do 20 j=1,n-m
        ho=xa(j)-x
        hp=xa(j+m)-x
        w=c(j+1)-d(j)
        den=ho-hp
        if(den.eq.0.0)pause
        den=w/den
        d(j)=hp*den
        c(j)=ho*den
  20    continue
```

```
        if(2*ns.lt.n-m)then
          dy=c(ns+1)
        else
          dy=d(ns)
          ns=ns-1
        endif
        y=y+dy
   30 continue
      return
      end


cccccccccccccccccccccccccccccccccccccccccccccccccccc
c                                                  c
c   This option takes the *.int file and creates   c
c   files for cross sectional area, pressure       c
c   information and full geometry.  All files can  c
c   be used by tecplot.                            c
c                                                  c
cccccccccccccccccccccccccccccccccccccccccccccccccccc


      subroutine intertotp(iname)
      implicit real(a-h,o-z)
      character*20 iname,iint,inoz,icsa,ivpc,ivpm,izdf,ipdf
      common istrm,icrss,icr
      integer istrm,icrss
      dimension x(50,500),y(50,500),theta(50,500),peo(50,500)
      dimension yd(50,500),pd(50,500),dz(500)
      dimension xc(500,500),yc(500,500),zc(500,500),pc(500,500)
      iint=iname
      inoz=iname
      icsa=iname
      ivpc=iname
      ivpm=iname
      izdf=iname
      ipdf=iname
      ileng=index(iname,' ') -1
      iint(ileng+1:ileng+4) = '.int'
      inoz(ileng+1:ileng+4) = '.noz'
      icsa(ileng+1:ileng+4) = '.csa'
      ivpc(ileng+1:ileng+4) = '.vpc'
      ivpm(ileng+1:ileng+4) = '.vpm'
      izdf(ileng+1:ileng+4) = '.zdf'
      ipdf(ileng+1:ileng+4) = '.pdf'
      open(unit=1, file=iint, status='old')
      open(unit=2, file=inoz, status='unknown')
      open(unit=3, file=icsa, status='unknown')
      open(unit=9, file=ivpc, status='unknown')
      open(unit=10, file=ivpm, status='unknown')
      open(unit=11, file=izdf, status='unknown')
      open(unit=12, file=ipdf, status='unknown')
      ifull=istrm*8-7
      write(*,*) ''
      write(*,*) 'Writing data to ',inoz
      write(*,*) 'for 3-D nozzle geometry'
      write(2,*) 'TITLE="Wall Mesh With Pressure Contour"'
      write(2,*) 'VARIABLES="X","Y","Z","Pe/Po"'
      write(2,*) 'ZONE T="Streamlines",I=',icr,',J=',ifull,',F=POINT'
c
c     Creates streamline values for whole nozzle geometry
c     by translating streamline values to points of symmetry.
c     The output is to a file called *.noz.
c
      do 20 i=1,istrm
       do 10 j=1,icr
        read(1,*) x(i,j),y(i,j),theta(i,j),peo(i,j)
        xc(i,j)=x(i,j)
        yc(i,j)=y(i,j)*sin(theta(i,j)*3.14159/180.)
        zc(i,j)=y(i,j)*cos(theta(i,j)*3.14159/180.)
        pc(i,j)=peo(i,j)
        write(2,301) xc(i,j),yc(i,j),zc(i,j),pc(i,j)
   10   continue
   20 continue
      do 40 i=1,istrm-1
       do 30 j=1,icr
        xc(istrm+i,j)=xc(1,j)
        yc(istrm+i,j)=zc(istrm-i,j)
        zc(istrm+i,j)=yc(istrm-i,j)
        pc(istrm+i,j)=pc(istrm-i,j)
        k=istrm+i
        write(2,301) xc(k,j),yc(k,j),zc(k,j),pc(k,j)
   30   continue
   40 continue
      do 60 i=1,istrm-1
       do 50 j=1,icr
        xc(2*istrm-1+i,j)=xc(1,j)
        yc(2*istrm-1+i,j)=yc(2*istrm-1-i,j)
        zc(2*istrm-1+i,j)=-zc(2*istrm-1-i,j)
        pc(2*istrm-1+i,j)=pc(2*istrm-1-i,j)
        k=2*istrm-1+i
        write(2,301) xc(k,j),yc(k,j),zc(k,j),pc(k,j)
   50   continue
   60 continue
      do 80 i=1,istrm-1
       do 70 j=1,icr
        xc(3*istrm-2+i,j)=xc(1,j)
        yc(3*istrm-2+i,j)=yc(istrm-i,j)
        zc(3*istrm-2+i,j)=-zc(istrm-i,j)
        pc(3*istrm-2+i,j)=pc(istrm-i,j)
        k=3*istrm-2+i
        write(2,301) xc(k,j),yc(k,j),zc(k,j),pc(k,j)
   70   continue
   80 continue
      do 100 i=1,istrm-1
       do 90 j=1,icr
        xc(4*istrm-3+i,j)=xc(1,j)
        yc(4*istrm-3+i,j)=-yc(4*istrm-3-i,j)
        zc(4*istrm-3+i,j)=zc(4*istrm-3-i,j)
        pc(4*istrm-3+i,j)=pc(4*istrm-3-i,j)
        k=4*istrm-3+i
        write(2,301) xc(k,j),yc(k,j),zc(k,j),pc(k,j)
 90     continue
 100 continue
      do 120 i=1,istrm-1
       do 110 j=1,icr
        xc(5*istrm-4+i,j)=xc(1,j)
        yc(5*istrm-4+i,j)=-yc(3*istrm-2-i,j)
        zc(5*istrm-4+i,j)=zc(3*istrm-2-i,j)
        pc(5*istrm-4+i,j)=pc(3*istrm-2-i,j)
        k=5*istrm-4+i
        write(2,301) xc(k,j),yc(k,j),zc(k,j),pc(k,j)
 110    continue
 120 continue
      do 140 i=1,istrm-1
       do 130 j=1,icr
        xc(6*istrm-5+i,j)=xc(1,j)
        yc(6*istrm-5+i,j)=-yc(2*istrm-1-i,j)
        zc(6*istrm-5+i,j)=zc(2*istrm-1-i,j)
        pc(6*istrm-5+i,j)=pc(2*istrm-1-i,j)
        k=6*istrm-5+i
        write(2,301) xc(k,j),yc(k,j),zc(k,j),pc(k,j)
 130    continue
 140 continue
      do 160 i=1,istrm-1
       do 150 j=1,icr
        xc(7*istrm-6+i,j)=xc(1,j)
        yc(7*istrm-6+i,j)=yc(5*istrm-4-i,j)
        zc(7*istrm-6+i,j)=-zc(5*istrm-4-i,j)
        pc(7*istrm-6+i,j)=pc(5*istrm-4-i,j)
        k=7*istrm-6+i
        write(2,301) xc(k,j),yc(k,j),zc(k,j),pc(k,j)
 150    continue
 160 continue
c
c     Creates individual cross sections for file.
c     Each cross section is shown, but it has not
c     yet been corrected for wall curvature.  The
c     resulting file is called *.csa.
c
      write(*,*) ''
      write(*,*) 'Writing data to ',icsa
      write(*,*) 'for 3-D geometry cross-sections'
      write(*,*) '(includes wall bowing)'
      write(3,*) 'TITLE="Wall Cross Sections"'
      write(3,*) 'VARIABLES="Y","Z","X"'
      do 180 j=icr,1,-1
       write(3,*) 'ZONE T="Section",I=',ifull,',J=1,F=POINT'
       do 170 i=1,ifull
        write(3,301) yc(i,j),zc(i,j),xc(i,j)
 170    continue
 180 continue
```

```
c
c    Creates pressure differential for each cross section
c    of square half wall streamlines.  Pressure differential
c    calculation is done by substracting centerline pe/po
c    from the pe/po of a streamline on a given cross-section.
c    The resulting file is called *.vpc.
c
      write(*,*) ' '
      write(*,*) 'Writing data to ',ivpc
      write(*,*) 'for half wall pressure profiles at each'
      write(*,*) 'cross section'
      write(9,*) 'TITLE="Pressure Profiles on Wall Cross Sections"'
      write(9,*) 'VARIABLES="Y/Ym","Pe-Pc/Po","X"'
      do 200 j=1,icr
        write(9,*) 'ZONE T="Section",I=',istrm,',J=1,F=POINT'
        do 190 i=1,istrm
          yd(i,j)=yc(i,j)/yc(istrm,j)
          pd(i,j)=pc(i,j)-pc(1,j)
          write(9,302) yd(i,j),pd(i,j),xc(i,j)
 190    continue
 200  continue
c
c    Creates pressure differential mesh for half wall
c    section.  The pressure differential is calculated
c    by subtracting centerline pe/po from the pe/po of a
c    streamline on a given cross-section.  The resulting
c    file is called *.vpm.
c
      write(*,*) ' '
      write(*,*) 'Writing data to ',ivpm
      write(*,*) 'for overall half wall pressure contours'
      write(10,*) 'TITLE="Pressure Profiles on Half Wall Mesh"'
      write(10,*) 'VARIABLES="X","Y","Pe-Pc/Po"'
      write(10,*) 'ZONE T="Section",I=',icr,',J=',istrm,',F=POINT'
      do 220 i=1,istrm
        do 210 j=1,icr
          write(10,303) xc(i,j),yc(i,j),pd(i,j)
 210  continue
 220  continue
c
c    Creates a file for the difference in z between the
c    streamline at the wall centerline and the wall corner
c    along the streamwise direction x.  The resulting
c    file is called *.zdf.
c
      write(*,*) ' '
      write(*,*) 'Writing data to ',izdf
      write(*,*) 'for wall centerline to wall corner'
      write(*,*) 'height differences'
      write(11,*) 'TITLE="Wall Bowing Along Nozzle Axis"'
      write(11,*) 'VARIABLES="X","dZ"'
      write(11,*) 'ZONE T="Profile",I=',icr,',J=1,F=POINT'
      do 230 j=1,icr
        dz(j)=zc(1,j)-zc(istrm,j)
      write(11,302) xc(1,j),dz(j)
  230 continue
c
c   Creates a file for the difference in pe/po between the
c   streamline at the wall centerline and the wall corner
c   along the streamwise direction x.  The resulting
c   file is called *.pdf.
c
     write(*,*) ''
     write(*,*) 'Writing data to ',ipdf
     write(*,*) 'for wall centerline to wall corner'
     write(*,*) 'pressure differences'
     write(12,*) 'TITLE="Wall Pressure Differentials"'
     write(12,*) 'VARIABLES="X","dP"'
     write(12,*) 'ZONE T="Profile",I=',icr,',J=1,F=POINT'
     do 240 j=1,icr
      write(12,302) xc(1,j),pd(istrm,j)
  240 continue
c
c   Closes the files.
c
     close(1)
     close(2)
     close(3)
     close(9)
     close(10)
     close(11)
     close(12)
  301 format(4(1x,e14.7))
  302 format(2(1x,e14.7))
  303 format(3(1x,e14.7))
     return
     end


cccccccccccccccccccccccccccccccccccccccccccccccccccccc
c                                                    c
c   This option calculates the area of an eigth      c
c   of a square cross-section.  It then adjusts      c
c   the wall height to account for the wall          c
c   curvature.  It uses a trapezoidal rule for       c
c   the area integration.                            c
c                                                    c
cccccccccccccccccccccccccccccccccccccccccccccccccccccc


     subroutine areacalc(iname)
     implicit double precision(a-h,o-z)
     character*20 iname,iint,icsb
     common istrm,icrss,icr
     integer istrm,icrss,icr
     dimension x(50,500),y(50,500),theta(50,500),peo(50,500)
     dimension yc(50,500),zc(50,500)
     dimension area(500),rn(500)
```

```
      iint=iname
      icsb=iname
      ileng=index(iname,' ') -1
      iint(ileng+1:ileng+4) = '.int'
      icsb(ileng+1:ileng+4) = '.csb'
      open(unit=1, file=iint, status='old')
      open(unit=2, file=icsb, status='unknown')
c
c     Read in the streamline points to be used
c     for trapezoidal integration.
c
      write(*,*) ' '
      write(*,*) 'Numerically integrating cross sectional area'
      write(*,*) 'and adjusting height to accomodate wall bowing'
      do 20 i=1,istrm
        do 10 j=1,icr
          read(1,*) x(i,j),y(i,j),theta(i,j),peo(i,j)
          yc(i,j)=y(i,j)*sin(theta(i,j)*3.14159/180.)
          zc(i,j)=y(i,j)*cos(theta(i,j)*3.14159/180.)
   10   continue
   20 continue
c
c     Integrate the area for each cross section using
c     a basic trapezoidal rule.
c
      do 40 j=1,icr
        sum=0.0
        do 30 i=1,istrm-1
          s=(yc(i+1,j)-yc(i,j))*(zc(i+1,j)+zc(i,j))/2.0
          sum=sum+s
   30   continue
        area(j)=sum-(yc(istrm,j)**2.0)/2.0
        rn(j)=dsqrt(area(j)*2.0)
   40 continue
c
c     Creates file for squared profile called *.csb.
c     This data file can also be used for a Navier-
c     Stokes Calculation.
c
      write(*,*) ' '
      write(*,*) 'Writing data to ',icsb
      write(*,*) 'for squared geometry profile with'
      write(*,*) 'no wall bowing but no b.l. correction.'
      write(*,*) 'Data can also be used for Navier-Stokes'
      write(*,*) 'calculation.'
      write(2,*) 'TITLE="Squared Wall Profile"'
      write(2,*) 'VARIABLES="X","Z"'
      write(2,*) 'ZONE T="Profile",I=',icr,',J=1,F=POINT'
      do 50 j=icr,1,-1
        write(2,401) x(1,j),rn(j)
   50 continue
c
c     Closes the files.
c
```

```
   close(1)
   close(2)
 401 format(2(1x,e14.7))
    return
    end


ccccccccccccccccccccccccccccccccccccccccccccccccccccc
c                                                     c
c   This option takes the squared cross section file  c
c   *.csb, and takes into account the boundary layer  c
c   correction to adjust the wall geometry.           c
c                                                     c
ccccccccccccccccccccccccccccccccccccccccccccccccccccc


    subroutine blcorrect(iname)
    implicit real(a-h,o-z)
    character*20 iname,inul1,icsb,icsc,iprt,ire,icsd
    common istrm,icrss,icr
    integer istrm,icrss,icr
    dimension xbl(500),delst(500)
    dimension x(500),y(500)
    dimension yf(500),xr(500),yr(500),ds(500)
    dimension xa(3),da(3)
    iprt=iname
    icsb=iname
    icsc=iname
    ire=iname
    icsd=iname
    ileng=index(iname,' ') -1
    iprt(ileng+1:ileng+4) = '.prt'
    icsb(ileng+1:ileng+4) = '.csb'
    icsc(ileng+1:ileng+4) = '.csc'
    ire(ileng+1:ileng+3) = '.re'
    icsd(ileng+1:ileng+4) = '.csd'
    open(unit=1, file=iprt, status='old')
    open(unit=2, file=icsb, status='old')
    open(unit=3, file=icsc, status='unknown')
    open(unit=4, file=ire, status='old')
    open(unit=8, file=icsd, status='unknown')
c
c   Reads in delta star correction from Harris code
c   output file.
c
    read(1,502) ipts
    do 10 j=ipts,1,-1
      read(1,*) xbl(j),dum2,dum3,dum4,delst(j)
  10 continue
c
c   Reads in wall geometry from squared wall file *.csb
c   and calculates new wall geometry based on boundary
c   layer correction,
c
```

```
      read(2,503) inull
      do 20 j=1,icr
       read(2,*) x(j),y(j)
   20 continue
c
c     Reads scaling data from Reynold's number information file
c     and converts dimensionless geometry to actual geometry.
c
      read(4,*) inull
      read(4,*) resc
      do 40 j=1,icr
       yr(j)=y(j)*resc
       xr(j)=x(j)*resc
   40 continue
c
c     Writes additional points to delta star string to
c     match solution point geometry to nozzle geometry.
c
      xbl(ipts+1)=xr(icr)
      delst(ipts+1)=0.0
c
c     Enter interpolation loop.
c
      do 50 j=1,icr
       xloc=xr(j)
c
c     Locates the points bounding the x station of interest.
c
       ml=0
       mu=ipts+1
   45  if(mu-ml.gt.1)then
        mm=(mu+ml)/2
        if((xbl(ipts).gt.xbl(1)).eqv.(xloc.gt.xbl(mm)))then
         ml=mm
        else
         mu=mm
        endif
       go to 45
       endif
       m=ml
c
c     Set up an array to feed in points for interpolation.
c
       xa(1)=xbl(m)
       xa(2)=xbl(m+1)
       xa(3)=xbl(m+2)
       da(1)=delst(m)
       da(2)=delst(m+1)
       da(3)=delst(m+2)
       if(xa(2).eq.0.0.and.xa(3).eq.0.0)then
        k=1
       elseif(xa(2).gt.0.0.and.xa(3).eq.0.0)then
        k=2
       else
```

```
        k=3
       endif
c
c     Interpolates solution stations of Harris code output to the
c     calculation stations of this conversion code.
c
       call polint(xa,da,k,xloc,dloc,dd)
       ds(j)=dloc
  50 continue
c
c     Creates file for boundary layer corrected cross
c     sections called *.csc.
c
      write(*,*) ' '
      write(*,*) 'Writing data to ',icsc
      write(*,*) 'for boundary layer corrected square'
      write(*,*) 'nozzle wall profile.  This data can'
      write(*,*) 'also be use for a Navier-Stokes'
      write(*,*) 'calculation.'
      write(3,*) 'TITLE="Squared, B.L. Corrected Nozzle Profile"'
      write(3,*) 'VARIABLES="X","Z"'
      write(3,*) 'ZONE T="Profile",I=',icr,',J=1,F=POINT'
      do 60 j=1,icr
       yf(j)=ds(j)+yr(j)
       write(3,501) xr(j),yf(j)
  60 continue
c
c     Creates file for boundary layer corrected cross
c     sections called *.csd.
c
      write(*,*) ' '
      write(*,*) 'Writing data to ',icsd
      write(*,*) 'for boundary layer corrected square'
      write(*,*) 'nozzle cross sections'
      write(8,*) 'TITLE="Squared, B.L. Corrected Cross Sections"'
      write(8,*) 'VARIABLES="Y","Z","X"'
      do 70 j=1,icr
       write(8,*) 'ZONE T="Section",I=5,J=1,F=POINT'
       write(8,*) yf(j),yf(j),xr(j)
       write(8,*) yf(j),-yf(j),xr(j)
       write(8,*) -yf(j),-yf(j),xr(j)
       write(8,*) -yf(j),yf(j),xr(j)
       write(8,*) yf(j),yf(j),xr(j)
  70 continue
c
c     Close the files.
c
      close(1)
      close(2)
      close(3)
      close(8)
 501 format(2(1x,e14.7))
 502 format(// i6)
 503 format(// a20)
```

```
504 format(1x,a20)
    return
    end
```