P-29

NASA Technical Memorandum 102705

INTERACTIVE GRID GENERATION PROGRAM FOR CAP-TSD

SAMUEL R. BLAND

October 1990

National Aeronautics and Space Administration

Langley Research Center
Hampton, Virginia 23665

INTERACTIVE GRID GENERATION PROGRAM FOR CAP-TSD

Samuel R. Bland

Unsteady Aerodynamics Branch
NASA Langley Research Center
Hampton, Virginia 23665-5225

Abstract

This document describes a grid generation program for use with the CAP-TSD transonic small disturbance code. The program runs interactively in FORTRAN on the Sun Workstation. A fifth-degree polynomial is used to map the grid index onto the computational coordinate. The grid is plotted to aid in the assessment of its quality and may be saved on a file in NAMELIST format.

Approach

In the transonic small disturbance code CAP-TSD[1] the computational grid consists of independent meshes in each of the computational coordinates XI, ETA, and ZETA with indices I, J, and K. That is, XI(I) is a one-dimensional array which maps the integers I = 1 to NXT onto the computational coordinate XI, with similar expressions for the other two arrays. Typically, the streamwise XI and vertical ZETA meshes extend 10 to 40 chord lengths from the configuration and the spanwise ETA mesh extends 2 to 4 span lengths.

Each of the meshes may be built up from a sequence of independent meshes for each region of the configuration. For a simple wing, the XI mesh might consist of three regions: (1) a mesh extending from the upstream boundary of the computational domain to the wing leading edge; (2) one from the leading edge to the trailing edge; (3) one from the trailing edge to the downstream boundary. It is desirable that configuration changes such as wing, tail, and body edges lie either on or half-way between the mesh points. For the simple wing, the up and downstream far-field boundaries would lie on mesh points with indices 1 and NXT, the wing leading edge would lie half-way between two mesh points, and the trailing edge would coincide with a mesh point.

In general, the mesh spacing should be finer at region boundaries and in other regions of rapid changes in the flow. Typically, mesh spacing on a lifting surface might range from 0.2 to 2.0 percent chord in XI and from 0.5 to 5.0 percent chord in ETA. The ZETA mesh spacing at the lifting surface should be comparable to that used chordwise with the addition of two points lying very close above and below the surface at 0.001 percent chord. The non-reflecting boundary conditions will be better approximated by the difference equations if the mesh spacing is kept below one chord length at all far-field boundaries.

Method

In this grid generation program, the mesh for each independent region is constructed from a fifth-degree polynomial which furnishes the computational coordinate (called X in the program) as a function of the index of the mesh point (called I). This quintic polynomial has six independent constants which are chosen from the following three conditions at each endpoint of the region: (1) the mesh index at the computational coordinate of the region boundary; (2) the mesh spacing at the boundary; (3) zero second derivative of the polynomial at the boundary. This last condition insures mesh smoothness at the boundaries. The index at the region boundary would be an integer for the case in which the mesh point lies on the boundary (e.g., at the wing trailing edge), but would be a half-integer for the case in which the boundary falls between mesh points (e.g., at the leading edge).

Although the meshes for each region may be computed independently, one should maintain continuity and smoothness at the region boundaries by using the same mesh spacing at a common boundary. The program facilitates this condition by allowing for the computation of the mesh for more than one region at a time.

Computer Program

The computer program was originally written in BASIC for the IBM-PC with EGA graphics card. The program described herein is written in FORTRAN for the Sun Workstation using Sun CGI graphics. A listing of the program is given in the Appendix.

The program begins by displaying a screen of general information. You are asked to enter ⟨Return⟩ to continue. The program then asks for the number of regions in the mesh you wish to design. As with all the numeric input, a default is provided. You are next asked for three quantities at each region boundary: (1) the computational coordinate of the boundary; (2) the mesh spacing at the boundary; (3) the mesh index of the boundary point (integer or half-integer, greater than zero). In each case, the current value of the quantity is displayed and may be left unchanged by entering ⟨Return⟩.

When the entry of the data for each boundary is complete, the program writes these defining values on the file "param.tem". The program then lists the mesh index, coordinate, first, and second differences on the screen. Negative values for the first difference indicate that the mesh has folded on itself — fewer points in the region or smaller spacings at the boundaries will fix this.

Response to the next prompt will clear the screen and plot the mesh as horizontal lines against X. The mesh index I is shown on the horizontal axis and the mesh function, first difference, and second difference are plotted against the index. The defining parameters for the mesh and its maximum and minimum spacings are printed at the top of the screen. The region boundaries are shown as tic marks with the boundary numbers along the right edge of the frame. The plot is in color; a black-on-white version may be obtained by changing "call colortable8" to "call colortable2" in the source code. To clear the plot screen you must enter "go" and ⟨Return⟩. Any other response will exit the program.

After exiting from the plot, you are asked whether you wish to save this grid on a file. This option will write the mesh coordinates in a five decimal place format, five to a line separated by commas, on your named file. Later, this file may be edited into the CAP-TSD input namelist. You are asked next if you wish to exit the program. If your mesh was not satisfactory, you may respond to the prompt simply with ⟨Return⟩ and the program will re-enter its input phase. Your current values will be displayed and you may make any desired changes.

Output of Computer Program

As examples of the mesh generation program output, results for the five default grids are shown. The meshes are described in terms of applications to particular configurations. However, these meshes are too coarse and the far-field boundaries are too close to provide accurate results.

For each example, a screen dump of the black-on-white plot of the mesh is shown. The command "screendump | rasfilter8to1 | lpr -s -v &", typed in a window other than the one containing the plot, will dump the entire screen to the printer. The screen used in the examples is one with the largest square area for plotting available on the Sun Workstation. It is 900 pixels tall and 880 pixels wide. Use of too small a window will lead to distortions in the lettering and other problems.

The default mesh for one region is shown in figure 1. This mesh is intended to illustrate vertical ZETA mesh properties for an isolated wing. The spacing is fine near the wing surface (X = 0) and stretches smoothly to a large spacing at the boundary (X = 8). The grid function X is plotted with long dashes, the spacing dX with short dashes, and the second difference ddX with dot-dashes. The origin for the latter curve is arbitrary, but this difference does approach zero at the region boundaries. For accurate calculations with CAP-TSD, the mesh would need two to three times the number of points and mesh extent as illustrated here.

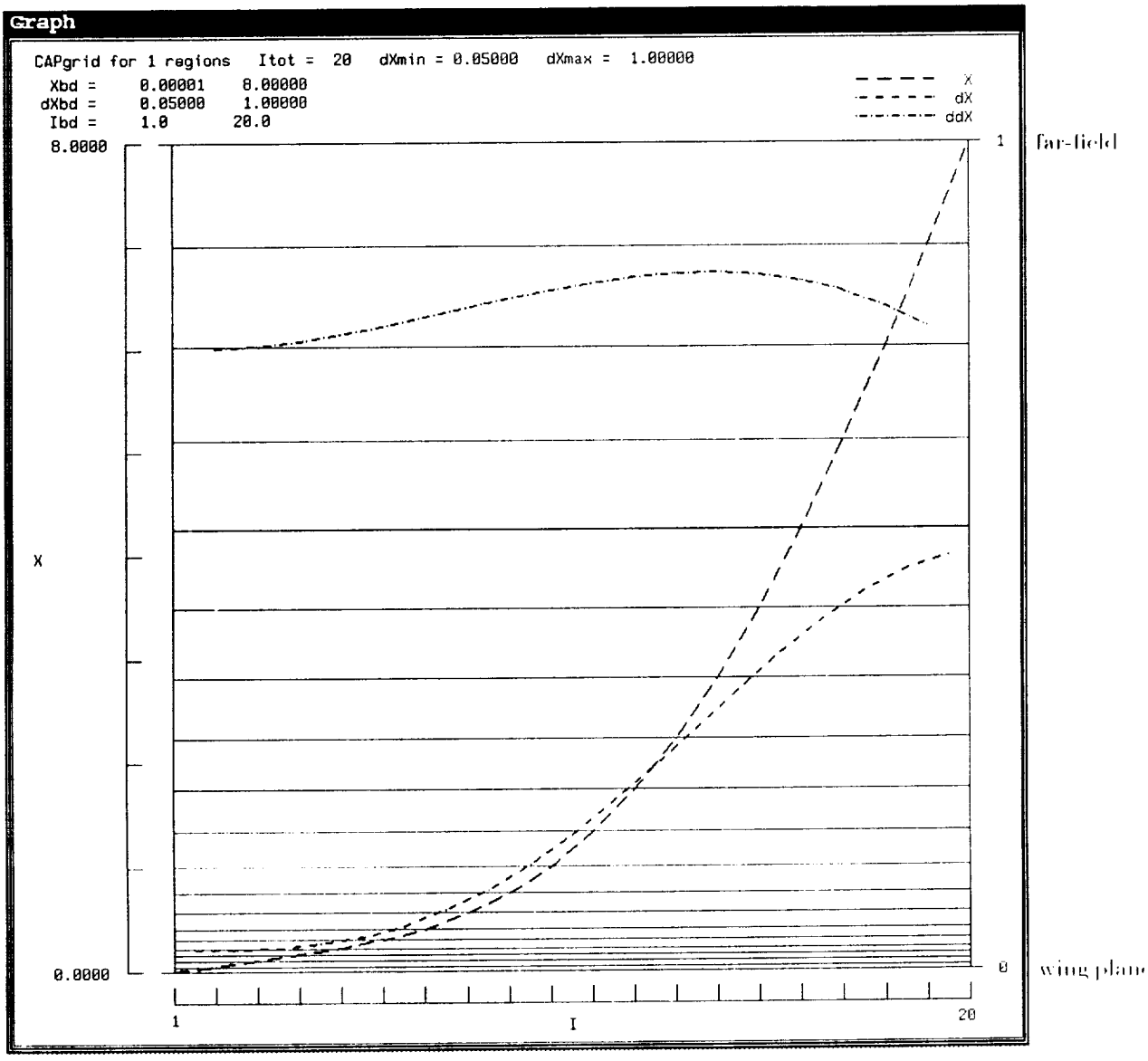

Figure 1.– Plot of default mesh with 1 region.

The default mesh for two regions is shown in figure 2. This mesh is typical of a very coarse spanwise ETA mesh for a simple wing. The spacing decreases smoothly from the root (X = 0) to the tip (X = 1) and then increases to the far-field boundary (X = 3). The wing tip lies between two mesh points (X = 10.5). The number of points on the wing was chosen to provide nearly uniform spacing near the root. The number of points off the wing was chosen to provide similar stretchings about the tip (the curve for dX is symmetrical there) and to achieve the maximum spacing at the far-field boundary. All of the curves possess the nice property of smoothness.

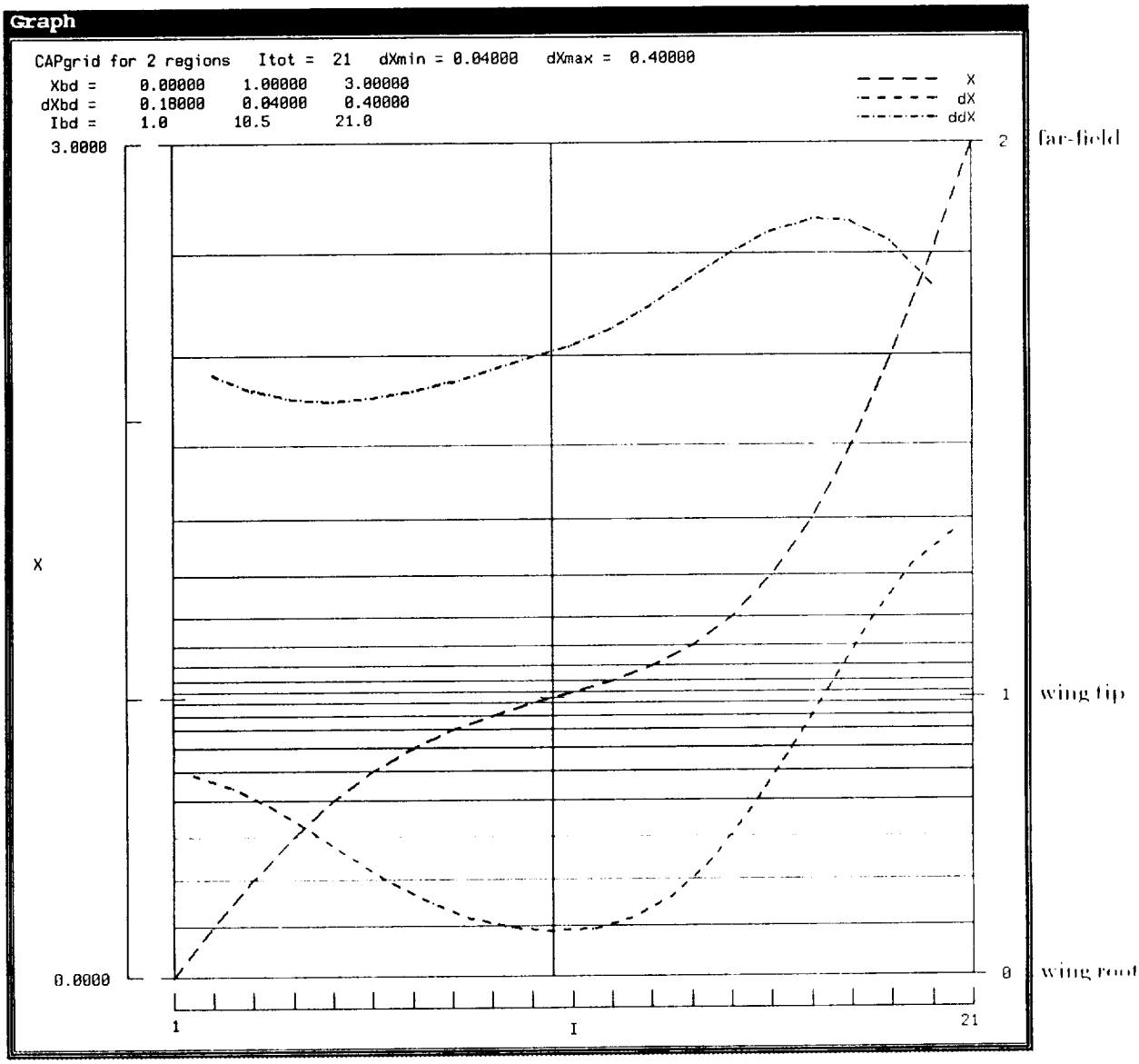

Figure 2.– Plot of default mesh with 2 regions.

The default mesh for three regions is shown in figure 3. This mesh illustrates the features of a streamwise XI mesh for an isolated wing, but it has too few points, too close far-field boundaries, and too large a spacing on the wing to be practical. The wing lies in the middle region with leading edge at X = 0 and trailing edge at X = 1. The leading edge lies between mesh points 13 and 14 and the trailing edge at point 28. The mesh is stretched somewhat over the middle of the wing chord and has its finest spacing at the wing edges. It achieves maximum spacing at the up and downstream boundaries.

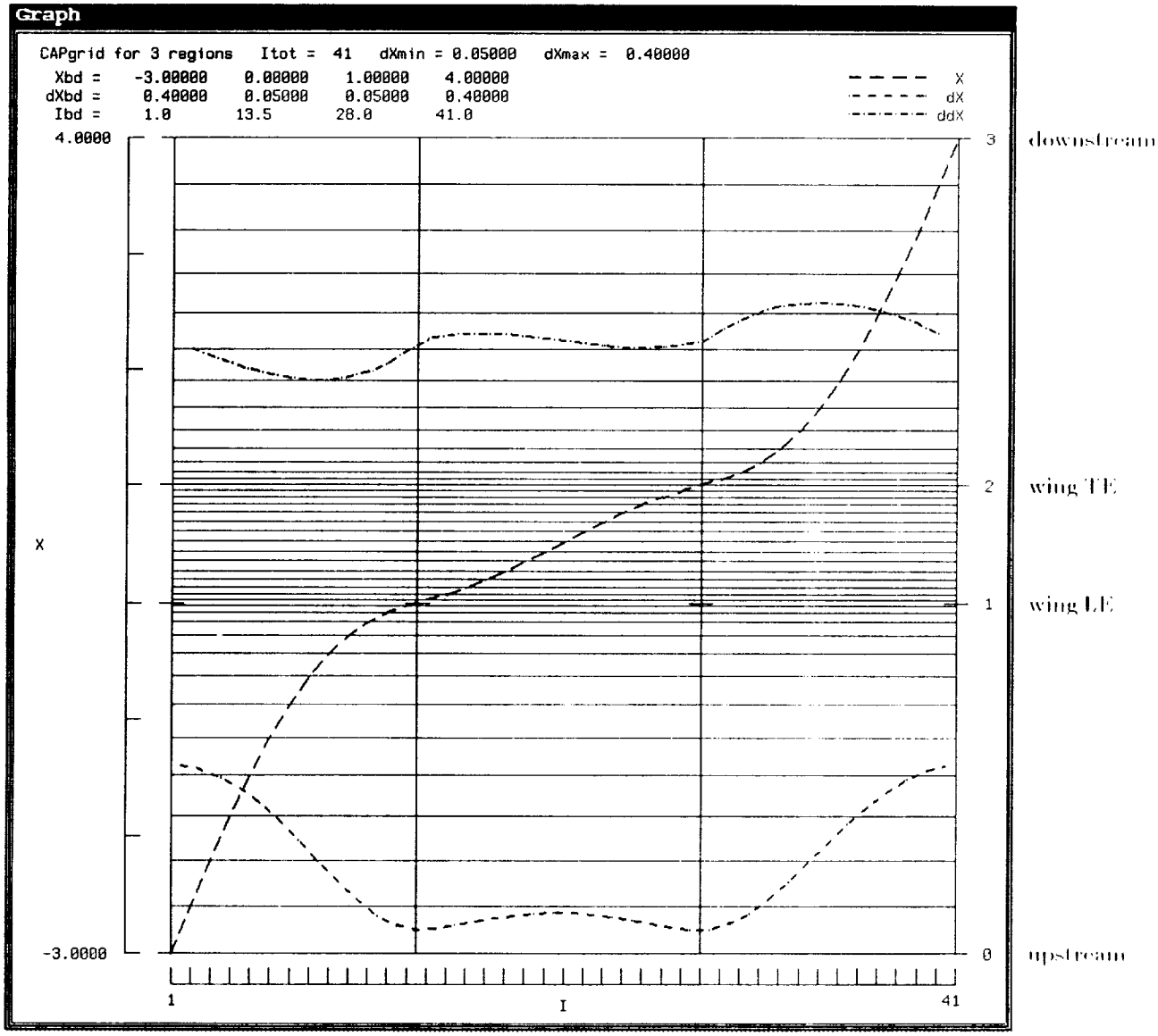

Figure 3.– Plot of default mesh with 3 regions.

The default mesh for four regions is shown in figure 4. This example illustrates a coarse spanwise ETA mesh for a fuselage/wing/tail configuration. The first mesh point lies on the centerline at X = 0. The side of the fuselage is at X = 0.1 and lies between mesh points. The tail tip is at X = 0.5 and the wing tip at X = 1.0, with both lying between mesh points. The far-field boundary is at X = 2.0. The mesh is clustered somewhat near the lifting surface tips with spacings of three percent of the configuration semispan, and stretches symmetrically away from them. There are three mesh points on the (half) fuselage, nine on the tail, and 21 on the wing.

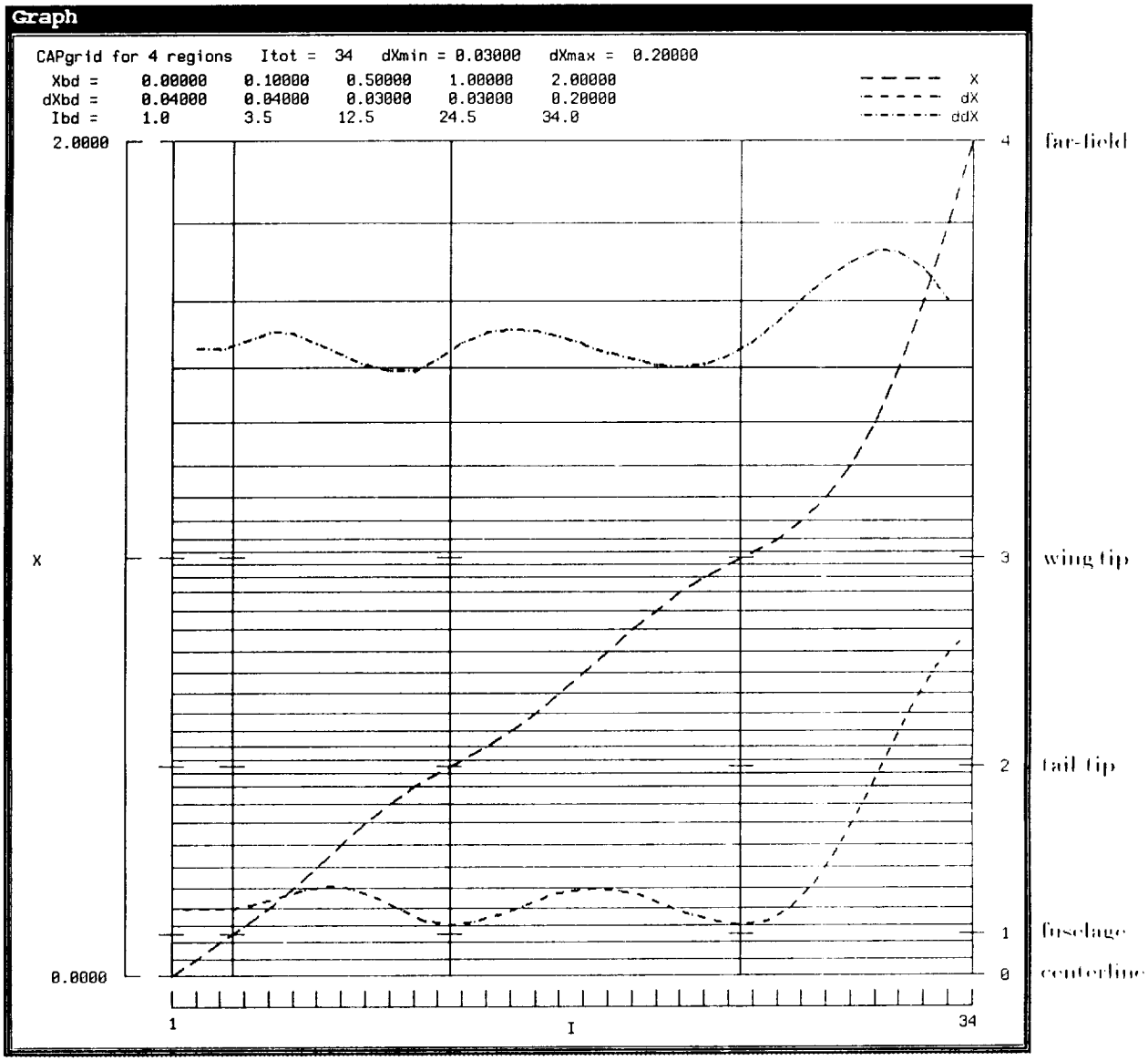

Figure 4.– Plot of default mesh with 4 regions.

The default mesh for five regions is shown in figure 5. This mesh illustrates the considerations in designing a streamwise XI mesh for a wing/tail configuration. There are a total of 47 points with 13 on the wing (which lies between X = 0.0 and 1.0), 5 in the wing-to-tail gap, and 8 points on the tail (which lies between X = 1.5 and 2.0). A practical mesh for such a configuration would need at least three to four times the number of points shown, and should have the far-field boundaries at least ten chord lengths away.

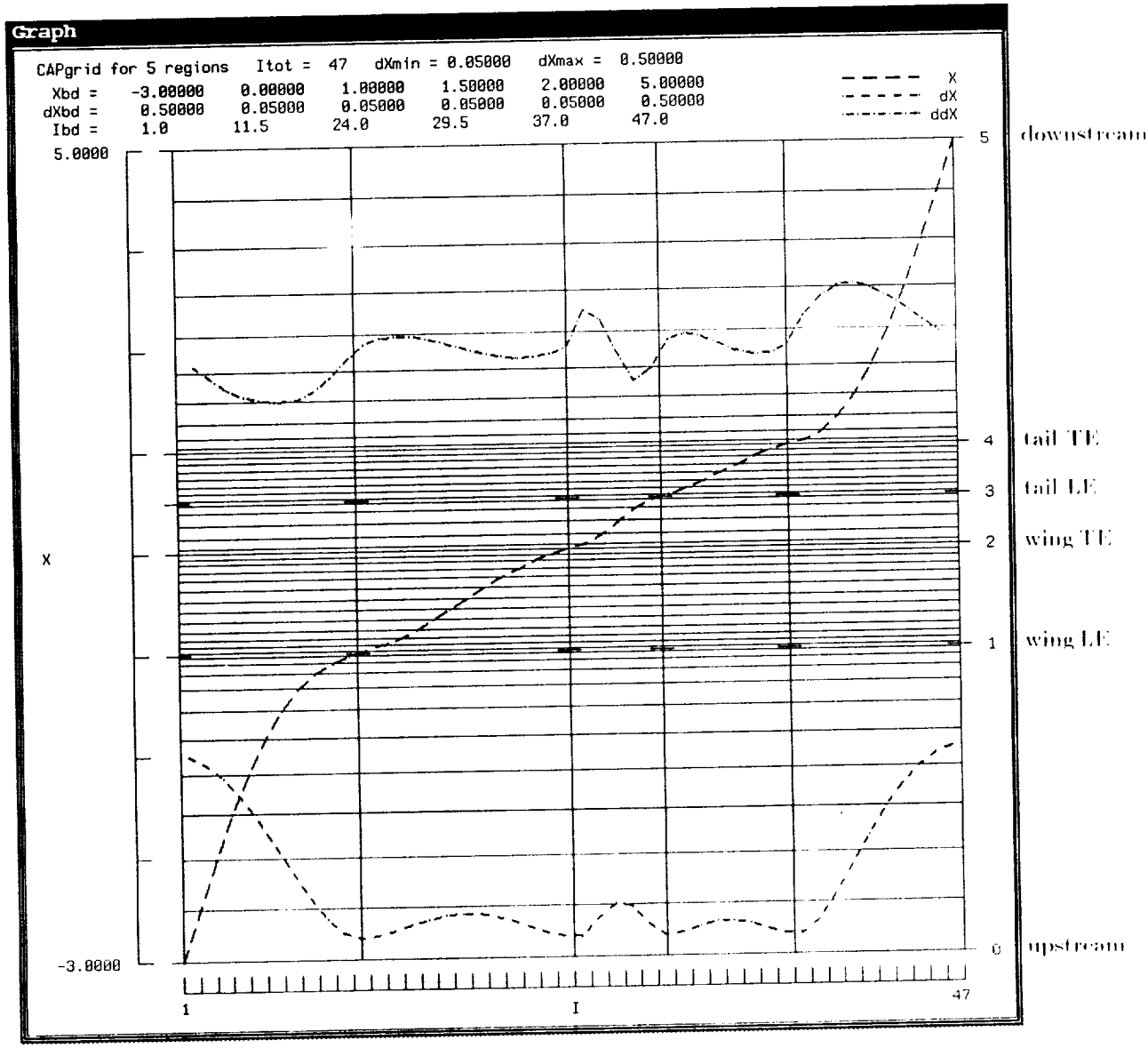

Figure 5.– Plot of default mesh with 5 regions.

Conclusion

An interactive program to aid in designing grids for use with the CAP-TSD computer code has been described. The program uses a fifth-degree polynomial to define the computational coordinate as a function of the grid point index. The program default grids, which illustrate some of the considerations in producing suitable grids for use with CAP-TSD, are shown.

Reference

1. Batina, J. T., Seidel, D. A., Bland, S. R., and Bennett, R. M., "Unsteady Transonic Flow Calculations for Realistic Aircraft Configurations," *Journal of Aircraft*, Vol. 26, Jan. 1989, pp. 21–28.

```
      program capgrid
c
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c                                                                       c
c     Grid generation for multiple regions with restart                 c
c                                                                       c
c     SUN FORTRAN version - 90/08/09                                    c
c                                                                       c
c     The compile statement is:                                         c
c                                                                       c
c   f77 -o capgrid capgrid.f -lcgi77 -lcgi -lsunwindow -lpixrect -lm    c
c                                                                       c
c     Samuel R. Bland     UAB/SDyD        (804) 864-2272              c
c     NASA Langley Research Center                                      c
c                                                                       c
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c
      common / bound / nreg, xb(0:5), rib(0:5), db(0:5), kb(0:5),
     $                 rir(0:5), irm(0:5), ril(0:5), ilp(0:5),
     $                 xr(0:5), xrm(0:5), xl(0:5), xlp(0:5)
      common / mesh  / x(999), dx(999), ddx(999),
     $                 imin, imax, itot, xmin, xmax, dxmin, dxmax
c
      character zzz$*8
      character yn$*8
      character gridfile$*40
c
      integer kb(0:5), ilp(0:5), irm(0:5)
c
      real ril(0:5), rir(0:5)
      real xb(0:5), rib(0:5), db(0:5)
      real xl(0:5), xlp(0:5), xrm(0:5), xr(0:5)
c
      real*8 x(999), dx(999), ddx(999)
      real*8 c(6)
c
      jmax = 6
      nreg = 2
      iter = 0
c
      call descrip
c
c     call colortable8
      call colortable2
c
c     Begin grid iteration
c
 100  continue
c
         call enterinput
c
         call mapping (jmax,c)
c
         call diffs
```

```
c
          call printscrn
c
          call plotgrid
c
          print '(//,a,$)', ' Save the grid on a file (y/n)? '
          read '(a8)', yn$
c
          if (yn$ .eq. 'y') then
             print '(a)', '          Input the file name'
             read '(a40)', gridfile$
c
             call writegrid (gridfile$)
c
             print '(/)'
             print '(a,$)', ' Enter  <Return>  to continue  '
             read '(a8)', zzz$
          endif
c
          print '(//,a,$)', ' Do you want to  EXIT  the program (y/n)? '
          read '(a8)',  yn$
c
          if (yn$ .ne. 'y') then
             goto 100
          endif
c
       print '(/)'
       print '(a)',' ***************    END OF PROGRAM    ***************'
       print '(/)'
c
       end

       subroutine enterinput
c
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c                                                                      c
c      Enter the region boundary coordinate, grid spacing, and         c
c         boundary index.  Compute the grid coordinate and index for   c
c         each region.                                                 c
c                                                                      c
c      The definitions are:                                            c
c                                                                      c
c         n          Region boundary index (0-nreg)                    c
c         xb(n)      Coordinate of the region boundary                 c
c         db(n)      Grid spacing at the region boundary               c
c         rib(n)     Real grid index at region boundary (integer or    c
c                       half integer                                   c
c         kb(n)      Flag to indicate whether boundary point is a      c
c                       grid point                                     c
c                                                                      c
c         ril(n)     Real index of left boundary point                 c
c         rir(n)     Real index of right boundary point                c
c         ilp(n)     Index of leftmost point wholly within region      c
c         irm(n)     Index of rightmost point wholly within region     c
c                                                                      c
c         xl(n)      Coordinate of left boundary point                 c
c         xr(n)      Coordinate of right boundary point                c
c         xlp(n)     Coordinate of leftmost point wholly within region c
c         xrm(n)     Coordinate of rightmost point wholly within region c
```

```
c                                                                       c
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c
      common / bound / nreg, xb(0:5), rib(0:5), db(0:5), kb(0:5),
     $                 rir(0:5), irm(0:5), ril(0:5), ilp(0:5),
     $                 xr(0:5), xrm(0:5), xl(0:5), xlp(0:5)
      common / mesh  / x(999), dx(999), ddx(999),
     $                 imin, imax, itot, xmin, xmax, dxmin, dxmax
c
      character*8 inp$
      character*1 n$
      real*8 x(999), dx(999), ddx(999)
c
      iter = iter + 1
c
      print '(/)'
      print '(a)',' ************   CAP-TSD GRID DESIGN   **************'
      print '(/)'
      print '(a)', ' You may leave any parameter unchanged by'
      print '(a)', '    responding to the input request with  <Return>'
      print '(/)'
c
      print '(a,i3,a,$)', ' Number of regions =', nreg, ' ? '
      read '(a8)', inp$
      iflag = 0
      do 10 k = 0, 9
         if (iflag .eq. 0) then
            write (n$, '(i1)') k
            if (index(inp$,n$) .ne. 0) then
               iflag = 1
               iter = 1
               read (inp$, '(i8)') nreg
            endif
         endif
 10   continue
c
      if (iter .eq. 1) then
         call iniparam
      endif
c
      do 400 n = 0, nreg
         print '(/,a,i2)', ' Data at boundary ', n
c
         print '(a,f10.5,a,$)', '     Coordinate =', xb(n), ' ? '
         read '(a8)', inp$
         if (index(inp$,'.') .ne. 0) then
            read (inp$, '(f10.5)') xb(n)
         else
            iflag = 0
            do 100 k = 0, 9
               if (iflag .eq. 0) then
                  write (n$, '(i1)') k
                  if (index(inp$,n$) .ne. 0) then
                     iflag = 1
                     read (inp$, '(i8)') iin
                     xb(n) = iin
                  endif
               endif
100         continue
         endif
```

```
c
          print '(a,f10.5,a,$)', '         Spacing =', db(n), ' ? '
          read '(a8)', inp$
          if (index(inp$,'.') .ne. 0) then
             read (inp$, '(f10.5)') db(n)
          else
             iflag = 0
             do 200 k = 0, 9
                if (iflag .eq. 0) then
                   write (n$, '(i1)') k
                   if (index(inp$,n$) .ne. 0) then
                      iflag = 1
                      read (inp$, '(i8)') iin
                      db(n) = iin
                   endif
                endif
 200         continue
          endif
c
          print '(a,f10.1,a,$)', '           Index =', rib(n), ' ? '
          read '(a8)', inp$
          if (index(inp$,'.') .ne. 0) then
             read (inp$, '(f10.5)') rib(n)
          else
             iflag = 0
             do 300 k = 0, 9
                if (iflag .eq. 0) then
                   write (n$, '(i1)') k
                   if (index(inp$,n$) .ne. 0) then
                      iflag = 1
                      read (inp$, '(i8)') iin
                      rib(n) = iin
                   endif
                endif
 300         continue
          endif
c
          if (rib(n) .eq. int(rib(n))) then
             kb(n) = 1
          else
             kb(n) = 2
             rib(n) = int(rib(n)) + .5
          endif
c
 400   continue
c
       if (kb(0) .eq. 1) then
          imin = rib(0)
       else
          imin = rib(0) + .5
       endif
c
       if (kb(nreg) .eq. 1) then
          imax = rib(nreg)
       else
          imax = rib(nreg) - .5
       endif
c
```

```
      itot = imax - imin + 1
      xmin = xb(0)
      xmax = xb(nreg)
c
c     Save the parameters that define the grid on file "param.tem"
c
      call outparam
c
      print '(//,a)', ' Grid definition is saved on file  "param.tem"'
c
c     Compute region boundary parameters
c
      do 500 n = 0, nreg
         rir(n) = rib(n)
         ril(n + 1) = rib(n)
         xr(n) = xb(n)
         xl(n + 1) = xb(n)
         if (kb(n) .eq. 1) then
            irm(n) = rib(n) - 1.
            ilp(n+1) = rib(n) + 1.
            xrm(n) = xb(n) - db(n)
            xlp(n+1) = xb(n) + db(n)
         else
            irm(n) = rib(n) - .5
            ilp(n+1) = rib(n) + .5
            xrm(n) = xb(n) - .5 * db(n)
            xlp(n+1) = xb(n) + .5 * db(n)
         endif
 500  continue
c
      return
      end

      subroutine diffs
c
ccccccccccccccccccccccccccccccccccccccccccccccc
c                                              c
c     Compute the 1st and 2nd differences      c
c                                              c
ccccccccccccccccccccccccccccccccccccccccccccccc
c
      common / mesh  / x(999), dx(999), ddx(999),
     $                 imin, imax, itot, xmin, xmax, dxmin, dxmax
c
      real*8 x(999), dx(999), ddx(999)
c
      do 100 i = imin, imax - 1
         dx(i) = x(i+1) - x(i)
 100  continue
c
      do 110 i = imin + 1, imax - 1
         ddx(i) = x(i+1) - 2. * x(i) + x(i-1)
 110  continue
c
```

```
      dxmin = 1000.
      dxmax = 0.
      do 120 i = imin, imax - 1
         if (dx(i) .lt. dxmin) then
            dxmin = dx(i)
         endif
         if (dx(i) .gt. dxmax) then
            dxmax = dx(i)
         endif
 120  continue
c
      return
      end

      subroutine printscrn
c
ccccccccccccccccccccccccccccccccccccccccccccccccc
c                                                 c
c     Print the mesh values and differences       c
c                                                 c
ccccccccccccccccccccccccccccccccccccccccccccccccc
c
      common / mesh  / x(999), dx(999), ddx(999),
     $                 imin, imax, itot, xmin, xmax, dxmin, dxmax
c
      real*8 x(999), dx(999), ddx(999)
c
      print '(/,a)', '  I        X(I)       dX(I)      ddX(I)'
c
      print '(i3,2f12.5)', imin, x(imin), dx(imin)
      do 100 i = imin + 1, imax - 1
         print '(i3,3f12.5)', i, x(i), dx(i), ddx(i)
 100  continue
      print '(i3,f12.5,/)', imax, x(imax)
c
      return
      end

      subroutine colortable2
c
ccccccccccccccccccccccccccccccccccccccccccccccccc
c                                                 c
c     Define a  2  color table with the values:   c
c                                                 c
c           0  white     1-7  black              c
c                                                 c
ccccccccccccccccccccccccccccccccccccccccccccccccc
c
      common / color / lred, lgrn, lblu, ncolors
c
      integer lred(8), lgrn(8), lblu(8)
c
      ncolors = 8
c
      lred(1) = 255
      lgrn(1) = 255
      lblu(1) = 255
c
```

```
      do 100 i = 2, 8
         lred(i) = 0
         lgrn(i) = 0
         lblu(i) = 0
 100  continue
c
      return
      end

      subroutine colortable8
c
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c                                                             c
c     Define an  8  color table with the values:              c
c                                                             c
c           0  dark blue      4  red                         c
c           1  blue           5  magenta                     c
c           2  green          6  yellow                      c
c           3  cyan           7  white                       c
c                                                             c
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c
      common / color / lred, lgrn, lblu, ncolors
c
      integer lred(8), lgrn(8), lblu(8)
      integer lc(2)
c
      ncolors = 8
      lc(1) =   0
      lc(2) = 255
c
      i = 0
      do 300 ir = 1, 2
         do 200 ig = 1, 2
            do 100 ib = 1, 2
               i = i + 1
               lred(i) = lc(ir)
               lgrn(i) = lc(ig)
               lblu(i) = lc(ib)
 100        continue
 200     continue
 300  continue
c
      lblu(1) = 160
c
      return
      end

      subroutine plotgrid
c
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c                                                                             c
c     Open the  CGI  graphics and plot the grid, the grid function,           c
c        and the 1st and 2nd differences.                                     c
c                                                                             c
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c
```

```
      common / bound / nreg, xb(0:5), rib(0:5), db(0:5), kb(0:5),
     $                 rir(0:5), irm(0:5), ril(0:5), ilp(0:5),
     $                 xr(0:5), xrm(0:5), xl(0:5), xlp(0:5)
      common / mesh  / x(999), dx(999), ddx(999),
     $                 imin, imax, itot, xmin, xmax, dxmin, dxmax
      common / color / lred, lgrn, lblu, ncolors
      common / plot  / lxof, lyof, rxs, ris, rxds, rxdds
c
      character*80 ntex$
      character*8 zzz$
      integer lred(8), lgrn(8), lblu(8)
      integer lxp(999), lyp(999)
      real*8 x(999), dx(999), ddx(999)
c
      print '(a)', ' The program will plot the grid next.  To end the'
      print '(a)', '    plot without exiting the program type  "go"'
      print '(a)', '    then  <Return>  after the plot is completed.'
      print '(/,a,$)',' Enter  <Return>  to continue  '
      read '(a8)', zzz$
      print '(/)'
c
      call cfopencgi ()
c
      call cfopenvws (name,0,0,0,1,CGPIXWINDD,ncolors,
     $                     'Color',0,0,0,0,5,0)
c
      call cfcotable (0,lred,lgrn,lblu,ncolors)
c
      call cflnspecmode (0)
c
      call setupplot
c
c     Draw the grid
c
      call cflnwidth (1.)
      call cflntype (0)
      call cflncolor (2)
      do 100 i = imin, imax
         lxp(1) = lxof
         lxp(2) = lxp(1) + ris * (itot - 1)
         lyp(1) = lyof + rxs * (x(i) - xmin)
         lyp(2) = lyp(1)
         call cfpolyline (lxp,lyp,2)
 100  continue
c
c     Plot X
c
      call cflnwidth (2.)
      call cflntype (5)
      call cflncolor (7)
      ip = 0
      do 110 i = imin, imax
         ip = ip + 1
         lxp(ip) = lxof + ris * (i - imin)
         lyp(ip) = lyof + rxs * (x(i) - xmin)
 110  continue
      call cfpolyline (lxp,lyp,ip)
```

```
c
c       Plot dX
c
        call cflnwidth (2.)
        call cflntype (2)
        call cflncolor (3)
        ip = 0
        do 120 i = imin, imax - 1
           ip = ip + 1
           lxp(ip) = lxof + ris * (i - imin + .5)
           lyp(ip) = lyof + rxds * dx(i)
 120    continue
        call cfpolyline (lxp,lyp,ip)
c
c       Plot ddX
c
        call cflnwidth (2.)
        call cflntype (3)
        call cflncolor (6)
        ip = 0
        do 130 i = imin + 1, imax - 1
           ip = ip + 1
           lxp(ip) = lxof + ris * (i - imin)
           lyp(ip) = lyof + rxs * ((xmax - xmin) * .75) + rxdds * ddx(i)
 130    continue
        call cfpolyline (lxp,lyp,ip)
c
c       Draw and label region boundaries
c
        call cflnwidth (1.)
        call cflntype (0)
        call cflncolor (5)
        call cftextcolor (7)
        call cftextfontix (4)
        lxt = 32000
        do 210 n = 0, nreg
           lyp(1) = lyof + rxs * (xb(n) - xmin)
           lyp(2) = lyp(1)
           do 200 n1 = 0, nreg
              lxp(1) = lxof + ris * (rib(n1) - imin) - 400
              lxp(2) = lxp(1) + 800
              call cfpolyline (lxp,lyp,2)
 200       continue
           lyt = lyp(1) - 180
           write (ntex$,'(i1)') n
           call cftext (lxt,lyt,ntex$)
           lxp(1) = lxof + ris * (rib(n) - imin)
           lxp(2) = lxp(1)
           lyp(1) = lyof + rxs * (xb(0) - xmin)
           lyp(2) = lyof + rxs * (xb(nreg) - xmin)
           call cfpolyline (lxp,lyp,2)
 210    continue
c
        pause
c
        call cfclosevws (name)
c
        call cfclosecgi ()
c
        return
        end
```

```
      subroutine setupplot
c
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c                                                                      c
c     Define the plot origin and scale factors and draw and label      c
c        the axes.                                                     c
c                                                                      c
c     The plot units are in the range  0-32767  for  x  and  y         c
c        and use the largest square available in the current window.   c
c                                                                      c
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c
      common / bound / nreg, xb(0:5), rib(0:5), db(0:5), kb(0:5),
     $                 rir(0:5), irm(0:5), ril(0:5), ilp(0:5),
     $                 xr(0:5), xrm(0:5), xl(0:5), xlp(0:5)
      common / mesh  / x(999), dx(999), ddx(999),
     $                 imin, imax, itot, xmin, xmax, dxmin, dxmax
      common / color / lred, lgrn, lblu, ncolors
      common / plot  / lxof, lyof, rxs, ris, rxds, rxdds
c
      character*80 ntex$
      integer lred(8), lgrn(8), lblu(8)
      integer lxp(999), lyp(999)
      real*8 x(999), dx(999), ddx(999)
c
      lxof = 5200
      lyof = 2300
      rxs = 27000. / (xmax - xmin)
      ris = 25900. / (imax - imin)
      rxds  = 4. * rxs
      rxdds = 8. * rxs
c
c     Draw rectangle around plotting area
c
      call cfperimspecmode (0)
      call cfperimwidth (0.)
      call cfperimtype (0)
      call cfperimcolor (4)
      call cfrectangle (32717,50,50,32717)
c
c     Write legend on plot
c
      call cftextcolor (7)
      call cftextfontix (4)
c
      lxt = 800
      lyt = 31900
      write (ntex$, '(a,i1,a,i4,3x,a,f7.5,3x,a,f8.5)')
     $              'CAPgrid for ', nreg, ' regions   Itot =',
     $               itot, 'dXmin = ', dxmin, 'dXmax = ', dxmax
      call cftext (lxt,lyt,ntex$)
c
      lxt = 1000
      lyt = lyt - 800
      call cftext (lxt,lyt,' Xbd =')
c
      lyt = lyt - 600
      call cftext (lxt,lyt,'dXbd =')
```

```
c
      lyt = lyt - 600
      call cftext (lxt,lyt,' Ibd =')
c
      lxt = lxt + 2000
      lyt = lyt + 2 * 600
      write (ntex$, '(6(f11.5))')    (xb(n), n = 0, nreg)
      call cftext (lxt,lyt,ntex$)
c
      lyt = lyt - 600
      write (ntex$, '(6(f11.5))')    (db(n), n = 0, nreg)
      call cftext (lxt,lyt,ntex$)
c
      lyt = lyt - 600
      write (ntex$, '(6(f7.1,4x))') (rib(n), n = 0, nreg)
      call cftext (lxt,lyt,ntex$)
c
c     Draw key on plot
c
      lxt = 30400
      call cflnwidth (2.)
      lxp(1) = 27500
      lxp(2) = 30000
c
      lyt = lyt + 1200
      call cftext (lxt,lyt,'  X')
      lyp(1) = lyt + 180
      lyp(2) = lyp(1)
      call cflntype (5)
      call cflncolor (7)
      call cfpolyline (lxp,lyp,2)
c
      lyt = lyt - 600
      call cftext (lxt,lyt,' dX')
      lyp(1) = lyt + 180
      lyp(2) = lyp(1)
      call cflntype (2)
      call cflncolor (3)
      call cfpolyline (lxp,lyp,2)
c
      lyt = lyt - 600
      call cftext (lxt,lyt,'ddX')
      lyp(1) = lyt + 180
      lyp(2) = lyp(1)
      call cflntype (3)
      call cflncolor (6)
      call cfpolyline (lxp,lyp,2)
c
c     Draw the I-axis
c
      call cflnwidth (1.)
      call cflntype (0)
      call cflncolor (7)
      lxp(1) = lxof
      lxp(2) = lxof + ris * (imax - imin)
      lyp(1) = lyof - 1000
      lyp(2) = lyp(1)
      call cfpolyline (lxp,lyp,2)
```

```
c
      do 200 i = imin, imax
         lxp(1) = lxof + ris * (i - imin)
         lxp(2) = lxp(1)
         lyp(1) = lyof - 1000
         lyp(2) = lyp(1) + 500
         call cfpolyline (lxp,lyp,2)
 200  continue
c
c     Label the I-axis
c
      call cftextcolor (7)
      call cftextfontix (4)
      lyt = 400
c
      lxt = lxof + ris * .5 * (imax - imin) - 100
      call cftext (lxt,lyt,'I')
c
      lxt = lxof - 700
      lyt = 600
      write (ntex$,'(i3)') imin
      call cftext (lxt,lyt,ntex$)
c
      lxt = lxof + ris * (imax - imin) - 700
      write (ntex$,'(i3)') imax
      call cftext (lxt,lyt,ntex$)
c
c     Draw the X-axis
c
      call cflnwidth (1.)
      call cflntype (0)
      call cflncolor (7)
      lxp(1) = lxof - 1500
      lxp(2) = lxp(1)
      lyp(1) = lyof
      lyp(2) = lyp(1) + rxs * (xmax - xmin)
      call cfpolyline (lxp,lyp,2)
c
      do 300 j = 0, 1
         lxp(1) = lxof - 1500
         lxp(2) = lxp(1) + 500
         lyp(1) = lyof + j * rxs * (xmax - xmin)
         lyp(2) = lyp(1)
         call cfpolyline (lxp,lyp,2)
 300  continue
c
      do 310 xt = - 100., 100.
         if ((xt .lt. xmax) .and. (xt .gt. xmin)) then
            lxp(1) = lxof - 1500
            lxp(2) = lxp(1) + 500
            lyp(1) = lyof + rxs * (xt - xmin)
            lyp(2) = lyp(1)
            call cfpolyline (lxp,lyp,2)
         endif
 310  continue
c
```

```
c       Label the X-axis
c
        call cftextcolor (7)
        call cftextfontix (4)
        lxt = 400
c
        lyt = lyof + rxs * .5 * (xmax - xmin) - 200
        call cftext (lxt,lyt,' X')
c
        lyt = lyof - 200
        write (ntex$,'(f9.4)') xmin
        call cftext (lxt,lyt,ntex$)
c
        lyt = lyof + rxs * (xmax - xmin) - 200
        write (ntex$,'(f9.4)') xmax
        call cftext (lxt,lyt,ntex$)
c
        return
        end

        subroutine mapping (jmax,c)
c
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c                                                                      c
c       Compute the mapping from the grid index  I  to the grid        c
c          locations  X.                                               c
c                                                                      c
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c
        common / bound / nreg, xb(0:5), rib(0:5), db(0:5), kb(0:5),
     $                   rir(0:5), irm(0:5), ril(0:5), ilp(0:5),
     $                   xr(0:5), xrm(0:5), xl(0:5), xlp(0:5)
        common / mesh  / x(999), dx(999), ddx(999),
     $                   imin, imax, itot, xmin, xmax, dxmin, dxmax
c
        real*8 x(999), dx(999), ddx(999)
        real*8 c(6)
c
        do 120 n = 1, nreg
c
           call quintic (n,jmax,c)
c
           if (kb(n-1) .eq. 1) then
              i1 = ril(n)
           else
              i1 = ilp(n)
           endif
c
           if (kb(n) .eq. 1) then
              i2 = rir(n)
           else
              i2 = irm(n)
           endif
c
```

```
            do 110 i = i1, i2
               x(i) = 0
               do 100 j = 2, jmax
                  jb = jmax + 2 - j
                  x(i) = (x(i) + c(jb)) * i
  100          continue
               x(i) = x(i) + c(1)
  110       continue
c
  120    continue
c
         return
         end

         subroutine quintic (n,jmax,c)
c
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c                                                                         c
c      Compute the 5th degree polynomial through the end points with      c
c          the specified first differences having zero second             c
c          differences.                                                   c
c                                                                         c
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c
         common / bound / nreg, xb(0:5), rib(0:5), db(0:5), kb(0:5),
     $                    rir(0:5), irm(0:5), ril(0:5), ilp(0:5),
     $                    xr(0:5), xrm(0:5), xl(0:5), xlp(0:5)
         common / mesh  / x(999), dx(999), ddx(999),
     $                    imin, imax, itot, xmin, xmax, dxmin, dxmax
c
         real*8 x(999), dx(999), ddx(999)
         real*8 a(6,7), s(6,7), c(6)
c
         a(1,1) = 1.
         a(2,1) = 1.
         a(3,1) = 1.
         a(4,1) = 1.
c
         do 100 j = 2, jmax
            a(1,j) = a(1,j-1) * ril(n)
            a(2,j) = a(2,j-1) * ilp(n)
            a(3,j) = a(3,j-1) * rir(n)
            a(4,j) = a(4,j-1) * irm(n)
  100    continue
c
         do 200 j = 1, jmax
            a(5,j) = (j - 1) * (j - 2) * ril(n) ** (j - 3)
            a(6,j) = (j - 1) * (j - 2) * rir(n) ** (j - 3)
  200    continue
c
         a(1,7) = xl(n)
         a(2,7) = xlp(n)
         a(3,7) = xr(n)
         a(4,7) = xrm(n)
         a(5,7) = 0.
         a(6,7) = 0.
c
```

```
c       Gaussian elimination Algorithm 6.1 of Burden and Faires
c
        do 320 j1 = 1, jmax - 1
           do 310 j2 = j1 + 1, jmax
              s(j2,j1) = a(j2,j1) / a(j1,j1)
              do 300 j3 = 1, jmax + 1
                 a(j2,j3) = a(j2,j3) - s(j2,j1) * a(j1,j3)
 300          continue
 310       continue
 320    continue
c
c       Backward substitution
c
        c(jmax) = a(jmax,jmax+1) / a(jmax,jmax)
        do 410 j1 = 1, jmax - 1
           jb = jmax - j1
           c(jb) = a(jb,jmax+1)
           do 400 j2 = jb + 1, jmax
              c(jb) = c(jb) - a(jb,j2) * c(j2)
 400       continue
           c(jb) = c(jb) / a(jb,jb)
 410    continue
c
        return
        end

        subroutine writegrid (gridfile$)
c
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c                                                       c
c       Write the grid to a file in  NAMELIST  format   c
c                                                       c
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c
        common / mesh  / x(999), dx(999), ddx(999),
     $                   imin, imax, itot, xmin, xmax, dxmin, dxmax
c
        character ident$*80
        character gridfile$*40
        real*8 x(999), dx(999), ddx(999)
c
        open (3, file = gridfile$)
c
        print '(/)'
        print *, ' You may put an identifying text string on the file.'
        print *, '     Enter the string  '
        read '(a80)', ident$
c
        print '(/,a80)',     ident$
        print '(a)',      ' GRID = '
        print '(5(f11.5,a1))', (x(i), ',', i = imin, imax)
c
        write (3, '(a80)') ident$
        write (3, '(a)') ' GRID = '
        write (3, '(5(f11.5,a1))') (x(i), ',', i = imin, imax)
c
        close (3)
c
        return
        end
```

```
      subroutine outparam
c
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c                                                                 c
c     Write the parameters which define the grid to a file.       c
c                                                                 c
c     The grid is defined by three real numbers.  These are       c
c        the coordinate, spacing, and grid index value at         c
c        each of the grid boundaries.                             c
c                                                                 c
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c
      common / bound / nreg, xb(0:5), rib(0:5), db(0:5), kb(0:5),
     $                 rir(0:5), irm(0:5), ril(0:5), ilp(0:5),
     $                 xr(0:5), xrm(0:5), xl(0:5), xlp(0:5)
      common / mesh  / x(999), dx(999), ddx(999),
     $                 imin, imax, itot, xmin, xmax, dxmin, dxmax
c
      real*8 x(999), dx(999), ddx(999)
c
      open  (2, file = 'param.tem')
c
      write (2, '(//,a,i1,a,$)') ' CAPGRID for ', nreg, ' Regions'
      write (2, '(5x,a,i3)') 'Itotal = ', itot
      write (2, '(/,3a)') '     Xbound', '    dXbound', '    Ibound'
c
      do 100 n = 0, nreg
         write (2, '(f11.5,f11.5,f10.1)') xb(n), db(n), rib(n)
 100  continue
c
      close (2)
c
      return
      end

      subroutine iniparam
c
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c                                                                 c
c     Define default values of the parameters                     c
c                                                                 c
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c
      common / bound / nreg, xb(0:5), rib(0:5), db(0:5), kb(0:5),
     $                 rir(0:5), irm(0:5), ril(0:5), ilp(0:5),
     $                 xr(0:5), xrm(0:5), xl(0:5), xlp(0:5)
c
      if (nreg .eq. 1) then
         xb(0)  =  0.00001
         xb(1)  =  8.
         db(0)  =  0.05
         db(1)  =  1.00
         rib(0) =  1.
         rib(1) = 20.
```

```
 elseif (nreg .eq. 2) then
    xb(0)  =  0.
    xb(1)  =  1.
    xb(2)  =  3.
    db(0)  =  0.18
    db(1)  =  0.04
    db(2)  =  0.40
    rib(0) =  1.0
    rib(1) = 10.5
    rib(2) = 21.0
 elseif (nreg .eq. 3) then
    xb(0)  = -3.
    xb(1)  =  0.
    xb(2)  =  1.
    xb(3)  =  4.
    db(0)  =  0.40
    db(1)  =  0.05
    db(2)  =  0.05
    db(3)  =  0.40
    rib(0) =  1.0
    rib(1) = 13.5
    rib(2) = 28.0
    rib(3) = 41.0
 elseif (nreg .eq. 4) then
    xb(0)  =  0.0
    xb(1)  =  0.1
    xb(2)  =  0.5
    xb(3)  =  1.0
    xb(4)  =  2.0
    db(0)  =  0.04
    db(1)  =  0.04
    db(2)  =  0.03
    db(3)  =  0.03
    db(4)  =  0.20
    rib(0) =  1.0
    rib(1) =  3.5
    rib(2) = 12.5
    rib(3) = 24.5
    rib(4) = 34.0
 elseif (nreg .eq. 5) then
    xb(0)  = -3.0
    xb(1)  =  0.0
    xb(2)  =  1.0
    xb(3)  =  1.5
    xb(4)  =  2.0
    xb(5)  =  5.0
    db(0)  =  0.50
    db(1)  =  0.05
    db(2)  =  0.05
    db(3)  =  0.05
    db(4)  =  0.05
    db(5)  =  0.50
    rib(0) =  1.0
    rib(1) = 11.5
    rib(2) = 24.0
    rib(3) = 29.5
    rib(4) = 37.0
    rib(5) = 47.0
 endif
```

```
c
      return
      end

      subroutine descrip
c
ccccccccccccccccccccccccccccccccccccccccccccccc
c                                               c
c     Print brief instructions on screen        c
c                                               c
ccccccccccccccccccccccccccccccccccccccccccccccc
c
      character zzz$*8
c
      print '(/)'
      print *,'                    CAP-TSD Grid Design'
      print *,' '
      print *,' This program will aid in generating a CAP-TSD grid.'
      print *,' The space is divided into separate regions for which'
      print *,' you specify the region boundaries in  X  units and'
      print *,' give the indices  I  for these boundaries.  These'
      print *,' indices may be either integer or half integer'
      print *,' according to whether the boundaries fall on or between'
      print *,' the grid lines, respectively.  The program is set up'
      print *,' for a maximum of five regions.'
      print *,' '
      print *,' You specify the grid spacing at the boundary of each'
      print *,' region.  A separate fifth-degree polynomial is used'
      print *,' with zero second derivative at each boundary.  Since'
      print *,' the regions are independent you may do a complicated'
      print *,' grid one region at a time as long as you use the same'
      print *,' boundary values at the common interfaces.'
      print *,' '
      print *,' The grid and its first and second differences are'
      print *,' listed and plotted.  You may write the grid values to'
      print *,' a file in a  NAMELIST  format.'
      print *,' '
      print *,' The default values of the parameters shown may be used'
      print *,' by responding to the prompt with  <Return>.  The'
      print *,' program saves your values so that you need not'
      print *,' re-enter parameters that do not change as you improve'
      print *,' your grid.  The latest parameters are also saved on'
      print *,' file  "param.tem".'
c
      print '(/,a,$)', ' Enter  <Return>  to continue  '
      read '(a8)', zzz$
c
      return
      end
```

Report Documentation Page

1. Report No.	2. Government Accession No.	3. Recipient's Catalog No.
NASA TM-102705		

4. Title and Subtitle	5. Report Date
Interactive Grid Generation Program for CAP-TSD	October 1990
	6. Performing Organization Code

7. Author(s)	8. Performing Organization Report No.
Samuel R. Bland	
	10. Work Unit No.
	505-63-21-01

9. Performing Organization Name and Address	11. Contract or Grant No.
NASA Langley Research Center Hampton, Virginia 23665-5225	
	13. Type of Report and Period Covered
12. Sponsoring Agency Name and Address	Technical Memorandum
National Aeronautics and Space Administration Washington, DC 20546-0001	14. Sponsoring Agency Code

15. Supplementary Notes

To obtain the program contact: John B. Malone, Head
Unsteady Aerodynamics Branch
Mail Stop 173
Hampton, Virginia 23665-5225

16. Abstract

This document describes a grid generation program for use with the CAP-TSD transonic small disturbance code. The program runs interactively in FORTRAN on the Sun Workstation. A fifth-degree polynomial is used to map the grid index onto the computational coordinate. The grid is plotted to aid in the assessment of its quality and may be saved on a file in NAMELIST format.

17. Key Words (Suggested by Author(s))	18. Distribution Statement
Grid generation Transonic Small Disturbance Theory	[redacted] (Review for General Release October 31, 1992) Subject Category 02

19. Security Classif. (of this report)	20. Security Classif. (of this page)	21. No. of pages	22. Price
Unclassified	Unclassified	28	

```
   13. STIMS 1X ACC#  9010523        IPS-FILE ADABAS # =  116862
FICHE AVAIL    = OK            HARD COPY AVL = OK            COPYRIGHT      = N
ORIG AGENCY    = NASA          RECEIPT TYPE  = REG           ACQUIS  TYPE   = REG
DOCUMENT CLASS= TRP            ACCESS LEVEL  =   O           ACCESS RESTR   = UNRES
LIMITATION CAT= NONE           DOCUMENT SEC  = NC            TITLE SECURITY= NC
SUBJECT CATGRY= 02             SPECIAL HANDL =               PAGE COUNT     = 00029
INC AUTHOR LST= N              INC CNTRCT LST= N             LANGUAGE       = EN
COUNTRY ORIGIN= US             COUNTRY FINANC= US            ABSTRACT PREP  = AUT
PUB DATE       = 19901000      CORP SOURCE   = ND210491

TITLE          = Interactive grid generation program for CAP-TSD
AUTHOR         = BLAND, SAMUEL R.
CONTRACT NUM   = RTOP 505-63-21-01
REPORT NUM     = NASA-TM-102705
REPORT NUM     = NAS 1.15:102705
MAJOR TERMS    = APPLICATIONS PROGRAMS (COMPUTERS)
MAJOR TERMS    = COMPUTATIONAL FLUID DYNAMICS
MAJOR TERMS    = GRID GENERATION (MATHEMATICS)
MAJOR TERMS    = POLYNOMIALS
MAJOR TERMS    = TRANSONIC FLOW
MAJOR TERMS    = UNSTEADY FLOW
MINOR TERMS    = AIRCRAFT CONFIGURATIONS
MINOR TERMS    = FORTRAN
ABST AUTHOR    = Author
FORM OF INPUT  = HC

        ABSTRACT =  A grid generation program for use with the CAP-TSD
                     transonic small disturbance code is described. Th
                    e program runs interactively in FORTRAN on the Sun
                     Workstation. A fifth-degree polynomial is used to
                     map the grid index onto the computational coordin
                    ate. The grid is plotted to aid in the assessment
                    of its quality and may be saved on file in NAMELIS
                    T format.

***************     END OF ADABAS RECORD #     116862     ***************
```