Table A15 List of the file, **bosdec.equivellipse.**
This is a GENOPT-user-written file that must exist if the GENOPT user's generic class of cases to be optimized makes use of the BIGBOSOR4 software. **See Table a29 for a list of the file, "howto.bosdec", which gives guidelines on how to write a valid bosdec.src file. SUBROUTINE BOSDEC produces a valid input file for BIGBOSOR4 (or for BOSOR4). This particular version of SUBROUTINE BOSDEC produces a valid BIGBOSOR4 input file called "equivellipse.ALL" corresponding to the GENOPT user's generic case called "equivellipse".**

```
======================================================================
C=DECK      BOSDEC
C
C  PURPOSE IS TO SET UP BOSOR4 INPUT FILE FOR "equivellipse"
C
C This program was used in some (uncompleted) research I did in
C 2005 to automate the optimization of ellipsoidal tank heads
C with thickness that varies along the meridian. An ellipsoidal head
C is modelled as a number of shell segments each of which has a
C constant meridional radius of curvature. This is done in order to
C avoid element "locking" that can occur in BOSOR4 shell segments
C which have a meridional curvature that varies within a given
C shell segment.
C
C This technology was used to generate a BIGBOSOR4 input file for
C the ellipsoidal head under uniform internal pressure, studied
C in November, 2006.
C
      SUBROUTINE BOSDEC(INDX,ILOADX,INDIC,IMPERF,IFIL14,IFILE,
     1                  npoint,ainput,binput,LENCYL,nodes,WIMP,
     1                  WMODEX,xinput,xlimit,EMATL,NUMATL,DNMATL,
     1                  THKSKN,HIGHST,SPACNG,THSTIF,THKCYL,
     1                  PRESS,PMAX,N0BX,NMINBX,NMAXBX,INCRBX)
C
C234567890123456789012345678901234567890123456789012345678901234567890
C
C Meaning of INDX:
C  INDX = 1 means linear buckling of perfect shell (INDIC=1).
C           Purpose is to obtain the axisymmetric buckling modal
C           imperfection shape, which is present in all other analyses.
C
C  INDX = 2 means axisymmetric collapse of imperfect shell (INDIC=0).
C                                         (Behavior no. 1: BEHX1)
C  INDX = 3 means non-axisymmetric nonlinear bifurcation buckling
C           of imperfect shell (INDIC=1).    (Behavior no. 2: BEHX2)
C  INDX = 4 means axisymmetric stress analysis at design load (INDIC=0).
C    This branch yields the following behaviors:
C    a. local buckling load factor of shell skin   (BUCSKN). (BEHX3)
```

```
C    b. local buckling load factor of stiffener    (BUCSTF). (BEHX4)
C    c. maximum effective stress in the shell skin (STRMAX). (BEHX5)
C    d. maximum effective stress in stiffener      (STRSTF). (BEHX6)
C    e. normal displacement at shell apex          (ENDUV).  (BEHX7)
C
C definitions of other variables in the argument list...
c   ILOADX = load case number
c   INDIC  = bigbosor4 analysis type (0 or 1 used here)
c   IMPERF = 0 no imperfection; 1 yes imperfection
c   IFIL14 = file where bigbosor4 input "deck" is stored
c   IFILE  = file were list output is accumulated
c   npoint = number of x-coordinates (including x=0 and x at equator)
c            where a segment end is provided by the user: xinput
c   ainput = semi-major axis of ellipse (ainput = xinput(npoint)
c   binput = semi-minor axis of ellipse, x^2/a^2 + y^2/b^2 = 1.0
c   LENCYL = length of the cylindrical segment, if any
c   nodes  = number of nodal points in each segment
c   WIMP   = amplitude of initial buckling modal imperfection shape,
c            WMODEX
c   WMODEX = axisymmetric buckling modal imperfection shape
c            (obtained from bigbosor4)
c   xinput = x-coordinates corresponding to segment ends
c   xlimit = for x < xlimit use x-coordinate for callouts
c            for x > xlimit use y-coordinate for callouts
c   EMATL  = elastic modulus of isotropic material
c   NUMATL = Poisson ratio of isotropic material
c   DNMATL = mass density of isotropic material
c   THKSKN = skin thickness corresponding to xinput
c   HIGHST = stiffener height corresponding to xinput
c   SPACNG = isogrid spacing
c   THSTIF = isogrid member thickness
c   THKCYL = thickness of cylindrical segment, if any
c   PRESS(ILOADX) = applied pressure for load case ILOADX
c   PMAX = maximum pressure to be applied
c   N0BX = starting circ. wavenumber for buckling analysis
c   NMINBX = minimum circ. wavenumber for buckling analysis
c   NMAXBX = maximum circ. wavenumber for buckling analysis
c   INCRBX = increment in circ. wavenumber for buckling analysis
c
      COMMON/NUMPR2/ILAR,ICAR,IOAR,NFLAT,NCASES,NPRINT
      real LENCYL,NUMATL
      double precision x,y,phi,r,rknuck,a1,a2,b1,b2,x03,y03
      double precision x1,y1,x2,y2,x3,y3,a,b,r1,r2
      dimension x(21),y(21),x1(20),y1(20),x2(20),y2(20),x3(20),y3(20)
      dimension r1(20),r2(20)
      dimension THKSKN(21),HIGHST(21)
      dimension PRESS(*),WMODEX(*),xinput(21),NMESH(20)
C
```

```
      REWIND IFIL14
C
      IF (NPRINT.GE.2) WRITE(IFILE,3)
    3 FORMAT(//'  ***************  BOSDEC  ******************'/
     1'  The purpose of BOSDEC is to set up an input file, NAME.ALL,'/
     1'  for equivalent ellipsoidal shell. NAME is your name for'/
     1'  the case. The file NAME.ALL is a BOSOR4 input "deck" used'/
     1'  by SUBROUTINE B4READ.'/
     1'  **********************************************'/)
C
c This version of SUBROUTINE BOSDEC is for an "equivalent" ellipsoidal
head.
c The "equivalent" ellipsoidal head is constructed because BOSOR4
(bigbosor4)
c finite elements tend to "lock up" for shells of revolution in which the
c meridional curvature varies significantly within a single shell segment.
c
c The "equivalent" ellipsoidal head consists of a user-defined number of
c toroidal segments that match as well as possible the contour of the
c ellipsoidal head. The meridional curvature of each toroidal segment
c is constant in that segment. Therefore, there is no problem of finite
c element "lock up" in a segmented model of this type.
c
c For each toroidal segment, bigbosor4 needs three points for input:
c (x1,y1), (x2,y2), and (x3,y3). (x1,y1) and (x2,y2) lie on the
ellipsoidal
c contour and are the (x,y) coordinates at the two ends of the toroidal
c segment. (x3,y3) is the center of meridional curvature of the toroidal
c segment. The trick is to obtain (x3,y3) so that to toroidal segment best
c fits the ellipsoidal contour in that segment.
c
c We use the following procedure to get (x3,y3):
c
c 1. The equation of the ellipse is
c
c     x^2/a^2 + y^2/b^2 = 1.0                                        (1)
c
c 2. The equation for the normal to the ellipse at (x1,y1) is:
c
c     y - y1 = (y1/x1)(a^2/b^2)(x - x1)                              (2)
c
c 3. The equation for the normal to the ellipse at (x2,y2) is:
c
c     y - y2 = (y2/x2)(a^2/b^2)(x - x2)                              (3)
c
c 4. These two straight lines in (x,y) space intersect at (x03,y03),
c    with (x03,y03) are given by:
c    x03 = (b2 - b1)/(a1 - a2);   y03 = (a2*b1 - a1*b2)/(a2 - a1) (4)
```

```
c    in which a1, b1 and a2, b2 are:
c
c     a1 = (y1/x1)(a^2/b^2);      b1 = -a1*x1 + y1                (5)
c     a2 = (y2/x2)(a^2/b^2);      b2 = -a2*x2 + y2                (6)
c
c 5. For an ellipse the distance from the point (x03,y03) to (x1,y1) is
c    different than the distance from the point (x03,y03) to (x2,y2)
c    because the meridional curvature varies along the contour of the
c    ellipse. We wish to find a new point (x3,y3) in the neighborhood
c    of (x03,y03) for which the distance from (x3,y3) to (x1,y1) equals
c    the distance from (x3,y3) to (x2,y2). For such a point the
c    "equivalent" segment will be a toroidal segment in which the
c    meridional curvature is constant along the segment arc.
c
c 6. The square of the distances from (x03,y03) to (x1,y1) and to (x2,y2)
c    are:
c
c     d1sq = (x1 - x03)**2 + (y1 - y03)**2                        (7)
c     d2sq = (x2 - x03)**2 + (y2 - y03)**2                        (8)
c
c    and the difference of these is:
c
c     delsq = d1sq - d2sq                                         (9)
c
c 7. We determine the location of the center of meridional curvature of
c    the "equivalent" torioidal segment by allocating half of delsq to
c    each (distance)**2, d1sq and d2sq. We then have two (distance)^2
c    that are equal:
c
c     (x1 - x03)**2 + (y1 - y03)**2 - delsq/2                    (10)
c     (x2 - x03)**2 + (y2 - y03)**2 + delsq/2                    (11)
c
c 8. Suppose we let
c
c     x3 = x03 + dx  ;             y3 = y03 + dy                 (12)
c
c    Then we have two nonlinear equations for the unknowns (dx,dy):
c
c    [x1 - (x03+dx)]**2 + [y1 - (y03+dy)]**2 =
c                         (x1 - x03)**2 +(y1 - y03)**2 -delsq/2  (13)
c
c    [x2 - (x03+dx)]**2 + [y2 - (y03+dy)]**2 =
c                         (x2 - x03)**2 +(y2 - y03)**2 +delsq/2  (14)
c
c    These two equations say that the square of the distance from
c    (x3,y3) to (x1,y1) Eq.(13) is equal to that from (x3,y3) to (x2,y2)
c    Eq.(14).
c
```

```
c 9. We use Newton's method to solve the two simultaneous nonlinear
c    equations for (dx,dy):
c
c    For the ith Newton iteration, let
c
c     dx(i) = dx(i-1) + u                                          (15)
c     dy(i) = dy(i-1) + v                                          (16)
c
c    Then we develop two linear equations for u and v for the ith
c    Newton iteration:
c
c     u*2.*(x03-x1+dx(i-1)) +v*2.*(y03-y1 +dy(i-1)) = f1pp       (17)
c     u*2.*(x03-x2+dx(i-1)) +v*2.*(y03-y2 +dy(i-1)) = f2pp       (18)
c
c    in which the right-hand sides, f1pp and f2pp, are rather long
c    expressions given in SUBROUTINE x3y3, where the Newton iterations
c    occur.
c
c Now find (x3,y3)...
c
c Get end points (x1,y1), (x2,y2), and center of curvature (x3,y3)
c of each shell segment in the model...
c
c first, given x, get y...
c the y are obtained from the equation for an ellipse: x^2/a^2 + y^2/b^2 =
1
c
      a = ainput
      b = binput
      do 10 i = 1,npoint
         x(i) = xinput(i)
         y(i) = -b*dsqrt(1.-x(i)**2/a**2)
   10 continue
c
c the endpoints of the first segment (bottom of "ellipse") are
c
      r = a**2/b
      x1(1) = 0.
      y1(1) = -b
      x2(1) = x(2)
      phi = dasin(x(2)/r)
      y2(1) = r*(1 - dcos(phi)) - b
      x3(1) = 0.
      y3(1) = r - b
c
c the endpoints of the last segment (nearest the equator) are
c
      nseg = npoint - 1
```

```
      rknuck = b**2/a
      x1(nseg) = x(npoint-1)
      phi = dacos((x(npoint-1) - a + rknuck)/rknuck)
      y1(nseg) = -rknuck*dsin(phi)
      x2(nseg) = a
      y2(nseg) = 0.
      x3(nseg) = a -rknuck
      y3(nseg) = 0.
c
c next, establish the endpoints and centers of curvature of
c shell segments 2 - (nseg-1)
c
C23456789012345678901234567890123456789012345678901234567890123456789012
      if (NPRINT.GE.2) write(ifile,'(/,A,A,I3,A,/,A,A)')
     1'  End points (x1,y1), (x2,y2) and center of curvature, (x3,y3)',
     1'  for',nseg,' toroidal segments',
     1' Seg.    x1           y1          x2          y2           x3',
     1'         y3           r1          r2'
      iseg = 1
c
      r1(iseg) = dsqrt((x1(iseg) - x3(iseg))**2
     1                +(y1(iseg) - y3(iseg))**2)
      r2(iseg) = dsqrt((x2(iseg) - x3(iseg))**2
     1                +(y2(iseg) - y3(iseg))**2)
c
      if (NPRINT.GE.2) write(ifile,'(I3,1P,8E12.4)')
     1 iseg,x1(iseg),y1(iseg),x2(iseg),y2(iseg),x3(iseg),y3(iseg),
     1      r1(iseg),r2(iseg)
      do 1000 iseg = 2,nseg
         iseg1 = iseg - 1
         x1(iseg) = x2(iseg1)
         y1(iseg) = y2(iseg1)
         ipoint = iseg + 1
         x2(iseg) = x(ipoint)
         y2(iseg) = y(ipoint)
c  find point, (x03,y03), where the normals to the ellipse at
c  (x1,y1) and (x2,y2) intersect.
         a1 = y1(iseg)*a**2/(x1(iseg)*b**2)
         a2 = y2(iseg)*a**2/(x2(iseg)*b**2)
         b1 = -a1*x1(iseg) + y1(iseg)
         b2 = -a2*x2(iseg) + y2(iseg)
         x03 = (b2 - b1)/(a1 - a2)
         y03 = (a2*b1 - a1*b2)/(a2 - a1)
c
c  we wish to replace the ellipse with an "equivalent" ellipse.
c  the "equivalent" ellipse consists of a number of torispherical
c  segments with end  points (x1,y1) and (x2,y2) and center of
c  curvature (x3,y3). The purpose of subroutine x3y3 is to
```

```
c  determine (x3,y3) given (x1,y1), (x2,y2), and (x03,y03).
c
         call x3y3(ifile,iseg,x1(iseg),y1(iseg),x2(iseg),y2(iseg),
     1             x03,y03, x3(iseg),y3(iseg))
c
         r1(iseg) = dsqrt((x1(iseg) - x3(iseg))**2
     1                   +(y1(iseg) - y3(iseg))**2)
         r2(iseg) = dsqrt((x2(iseg) - x3(iseg))**2
     1                   +(y2(iseg) - y3(iseg))**2)
c
         if (NPRINT.GE.2) write(ifile,'(I3,1P,8E12.4)')
     1    iseg,x1(iseg),y1(iseg),x2(iseg),y2(iseg),x3(iseg),y3(iseg),
     1         r1(iseg),r2(iseg)
c
 1000 continue
C234567890123456789012345678901234567890123456789012345678901234567890123456789012
c
      IF (INDIC.EQ.0.AND.INDX.EQ.4) WRITE(IFIL14,'(A)')
     1' Nonlinear axisymmetric stress analysis (INDIC=0)'
      IF (INDIC.EQ.0.AND.INDX.EQ.2) WRITE(IFIL14,'(A)')
     1' Nonlinear axisymmetric collapse analysis (INDIC=0)'
      IF (INDIC.EQ.1) WRITE(IFIL14,'(A)')
     1' Bifurcation buckling analysis (INDIC=1)'
C BEG MAR 2008
      IF (INDIC.EQ.-2) WRITE(IFIL14,'(A)')
     1' Bifurcation buckling analysis (INDIC=-2)'
C END MAR 2008
      IF (INDIC.EQ.2) WRITE(IFIL14,'(A)')
     1' Modal vibration of prestressed shell'
      WRITE(IFIL14,'(I3,A)') INDIC, '            $ INDIC'
      WRITE(IFIL14,'(A)')'  1           $ NPRT'
      ISTRES = 0
      IF (INDIC.EQ.0) ISTRES = 1
      WRITE(IFIL14,'(I3,A)') ISTRES, '            $ ISTRES'
      IF (LENCYL.GT.0.001)
     1 WRITE(IFIL14,'(I4,A)') nseg+1,'          $ nseg'
      IF (LENCYL.LE.0.001)
     1 WRITE(IFIL14,'(I4,A)') nseg,'          $ nseg'
C
C Begin loop over Segment data
C
C234567890123456789012345678901234567890123456789012345678901234567890123456789012
      IALL = 0
      Do 2000 iseg = 1,nseg
         NMESH(iseg) = nodes
         WRITE(IFIL14,'(I4,A)') NMESH(iseg),'           $ NMESH'
         WRITE(IFIL14,'(A)')'  3           $ NTYPEH'
         WRITE(IFIL14,'(A)')'  2           $ NSHAPE'
```

```
      WRITE(IFIL14,'(1P,E14.6,A)') x1(iseg), ' $ R1'
      WRITE(IFIL14,'(1P,E14.6,A)') y1(iseg), ' $ Z1'
      WRITE(IFIL14,'(1P,E14.6,A)') x2(iseg), ' $ R2'
      WRITE(IFIL14,'(1P,E14.6,A)') y2(iseg), ' $ Z2'
      WRITE(IFIL14,'(1P,E14.6,A)') x3(iseg), ' $ RC'
      WRITE(IFIL14,'(1P,E14.6,A)') y3(iseg), ' $ ZC'
      WRITE(IFIL14,'(A)')' -1.          $ SROT'
      WRITE(IFIL14,'(I4,A)') IMPERF,'           $ IMP'
      IF (IMPERF.EQ.1) THEN
         WRITE(IFIL14,'(A)')'  4           $ ITYPE'
         WRITE(IFIL14,'(1P,E14.6,A)') WIMP, ' $ WIMP'
         WRITE(IFIL14,'(A)')'  1           $ ISTART'
         NUMB = NMESH(iseg) + 2
         WRITE(IFIL14,'(I4,A)') NUMB,'           $ NUMB'
         DO 5 I = 1,NUMB
         J = I + IALL
         WRITE(IFIL14,'(1P,E14.6,A)') WMODEX(J), ' $ WSHAPE'
5        CONTINUE
         WRITE(IFIL14,'(A)')'  N           $ any more modes?'
      ENDIF
      WRITE(IFIL14,'(A)')'  3           $ NTYPEZ'
      WRITE(IFIL14,'(A)')'  0.          $ ZVAL'
      WRITE(IFIL14,'(A)')'  Y           $ print r(s)...?'
      WRITE(IFIL14,'(A)')'  0           $ NRINGS'
      WRITE(IFIL14,'(A)')'  0           $ K'
      WRITE(IFIL14,'(A)')'  0           $ LINTYP'
      WRITE(IFIL14,'(A)')'  1           $ IDISAB'
      WRITE(IFIL14,'(A)')'  1           $ NLTYPE'
      WRITE(IFIL14,'(A)')'  2           $ NPSTAT'
      WRITE(IFIL14,'(A)')'  0           $ NLOAD(1)'
      WRITE(IFIL14,'(A)')'  0           $ NLOAD(2)'
      WRITE(IFIL14,'(A)')'  1           $ NLOAD(3)'
      WRITE(IFIL14,'(A)')' -1.          $ PN(1)'
      WRITE(IFIL14,'(A)')' -1.          $ PN(2)'
      IF (x1(iseg).le.xlimit) then
         ntype = 3
         call1 = x1(iseg)
         call2 = x2(iseg)
      else
         ntype = 2
         call1 = y1(iseg)
         call2 = y2(iseg)
      endif
      WRITE(IFIL14,'(I4,A)') ntype,'           $ ntype'
      WRITE(IFIL14,'(1P,E14.6,A)')  call1,  ' $ callout1'
      WRITE(IFIL14,'(1P,E14.6,A)')  call2,  ' $ callout2'
      WRITE(IFIL14,'(A)')' 10           $ NWALL'
      WRITE(IFIL14,'(A)')'  2           $ NWALL2'
```

```
         WRITE(IFIL14,'(1P,E14.6,A)') EMATL,   ' $ E'
         WRITE(IFIL14,'(1P,E14.6,A)') NUMATL,  ' $ U'
         WRITE(IFIL14,'(1P,E14.6,A)') DNMATL,  ' $ SM'
         WRITE(IFIL14,'(A)')'  0.          $ ALPHA'
         WRITE(IFIL14,'(A)')'  1           $ NRS'
         WRITE(IFIL14,'(A)')' -1           $ NSUR'
         WRITE(IFIL14,'(A)')'  1           $ NTYPET'
         IRADTH = 2
         WRITE(IFIL14,'(I4,A)') IRADTH,'          $ NTVALU'
         WRITE(IFIL14,'(I4,A)') ntype,'          $ ntype'
         WRITE(IFIL14,'(1P,E14.6,A)') call1,' $ callout1'
         WRITE(IFIL14,'(1P,E14.6,A)') call2,' $ callout2'
         ipoint = iseg + 1
         WRITE(IFIL14,'(1P,E14.6,A)') THKSKN(iseg),' $ THKSKN(iseg)'
         WRITE(IFIL14,'(1P,E14.6,A)') THKSKN(ipoint),' $ THKSKN(ipoint)'
C23456789012345678901234567890123456789012345678901234567890123456789012
         WRITE(IFIL14,'(A)')'  Y           $ print refsurf...?'
         WRITE(IFIL14,'(A)')'  Y   $ are there stringers or isogrid...?'
         WRITE(IFIL14,'(A)')'  0           $ K1 (0 means internal)'
         WRITE(IFIL14,'(1P,E14.6,A)') EMATL,   ' $ E'
         WRITE(IFIL14,'(1P,E14.6,A)') NUMATL,  ' $ U'
         WRITE(IFIL14,'(1P,E14.6,A)') DNMATL,  ' $ SM'
         WRITE(IFIL14,'(1P,E14.6,A)') SPACNG,  ' $ isogrid spacing'
         WRITE(IFIL14,'(A)')'  N           $ constant cross section?'
         WRITE(IFIL14,'(I4,A)') IRADTH,'          $ number of callouts'
         WRITE(IFIL14,'(I4,A)') ntype,'          $ ntype'
         WRITE(IFIL14,'(1P,E14.6,A)') call1,' $ callout1'
         WRITE(IFIL14,'(1P,E14.6,A)') call2,' $ callout2'
         WRITE(IFIL14,'(1P,E14.6,A)') THSTIF,' $ THSTIF'
         WRITE(IFIL14,'(1P,E14.6,A)') THSTIF,' $ THSTIF'
         WRITE(IFIL14,'(1P,E14.6,A)') HIGHST(iseg),' $ HIGHST(iseg)'
         WRITE(IFIL14,'(1P,E14.6,A)') HIGHST(ipoint),' $ HIGHST(ipoint)'
         WRITE(IFIL14,'(A)')'  N           $ are there smeared rings?'
         WRITE(IFIL14,'(A)')'  N           $ print Cij?'
         WRITE(IFIL14,'(A)')'  N           $ print loads?'
C
C end of Segment iseg input data
         IALL = IALL + NMESH(iseg) + 2
 2000 continue
C
C Begin  Segment nseg+1 data       (cylindrical segment)
C
C23456789012345678901234567890123456789012345678901234567890123456789012
      IF (LENCYL.GT.0.001) THEN
      NMESH(nseg+1) = 51
      WRITE(IFIL14,'(I4,A)') NMESH(nseg+1),'         $ NMESH seg.nseg+1'
      WRITE(IFIL14,'(A)')'  1           $ NTYPEH'
      WRITE(IFIL14,'(A)')'  4           $ NHVALU'
```

```
      WRITE(IFIL14,'(A)')'  1            $ IHVALU'
      WRITE(IFIL14,'(A)')' 25            $ IHVALU'
      WRITE(IFIL14,'(A)')' 26            $ IHVALU'
      WRITE(IFIL14,'(A)')' 50            $ IHVALU'
      WRITE(IFIL14,'(A)')' 0.2           $ HVALU'
      WRITE(IFIL14,'(A)')' 0.2           $ HVALU'
      WRITE(IFIL14,'(A)')' 1.0           $ HVALU'
      WRITE(IFIL14,'(A)')' 1.0           $ HVALU'
      WRITE(IFIL14,'(A)')'  1            $ NSHAPE'
      WRITE(IFIL14,'(1P,E14.6,A)') x2(nseg), ' $ R1'
      WRITE(IFIL14,'(1P,E14.6,A)') y2(nseg), ' $ Z1'
      WRITE(IFIL14,'(1P,E14.6,A)') x2(nseg), ' $ R2'
      WRITE(IFIL14,'(1P,E14.6,A)') y2(nseg)+LENCYL, ' $ Z2'
      WRITE(IFIL14,'(I4,A)') IMPERF,'           $ IMP'
      IF (IMPERF.EQ.1) THEN
         WRITE(IFIL14,'(A)')'  4            $ ITYPE'
         WRITE(IFIL14,'(1P,E14.6,A)') WIMP, ' $ WIMP'
         WRITE(IFIL14,'(A)')'  1            $ ISTART'
         NUMB = NMESH(nseg+1) + 2
         WRITE(IFIL14,'(I4,A)') NUMB,'            $ NUMB'
         DO 70 I = 1,NUMB
         J = I + IALL
         WRITE(IFIL14,'(1P,E14.6,A)') WMODEX(J), ' $ WSHAPE'
   70    CONTINUE
         WRITE(IFIL14,'(A)')'  N            $ any more modes?'
      ENDIF
      WRITE(IFIL14,'(A)')'  3            $ NTYPEZ'
      WRITE(IFIL14,'(A)')'  0.           $ ZVAL'
      WRITE(IFIL14,'(A)')'  N            $ print r(s)...?'
      WRITE(IFIL14,'(A)')'  0            $ NRINGS'
      WRITE(IFIL14,'(A)')'  0            $ K'
      WRITE(IFIL14,'(A)')'  0            $ LINTYP'
      WRITE(IFIL14,'(A)')'  1            $ IDISAB'
      WRITE(IFIL14,'(A)')'  1            $ NLTYPE'
      WRITE(IFIL14,'(A)')'  2            $ NPSTAT'
      WRITE(IFIL14,'(A)')'  0            $ NLOAD(1)'
      WRITE(IFIL14,'(A)')'  0            $ NLOAD(2)'
      WRITE(IFIL14,'(A)')'  1            $ NLOAD(3)'
      WRITE(IFIL14,'(A)')'  1.           $ PN(1)'
      WRITE(IFIL14,'(A)')'  1.           $ PN(2)'
      WRITE(IFIL14,'(A)')'  2            $ NTYPE'
      WRITE(IFIL14,'(1P,E14.6,A)')  y2(nseg),  ' $ Z1'
      WRITE(IFIL14,'(1P,E14.6,A)')  y2(nseg)+LENCYL,  ' $ Z2'
      WRITE(IFIL14,'(A)')'  2            $ NWALL'
      WRITE(IFIL14,'(1P,E14.6,A)') EMATL,   ' $ E'
      WRITE(IFIL14,'(1P,E14.6,A)') NUMATL,  ' $ U'
      WRITE(IFIL14,'(A)')'  0.           $ SM'
      WRITE(IFIL14,'(A)')'  0.           $ ALPHA'
```

```
      WRITE(IFIL14,'(A)')'  0            $ NRS'
      WRITE(IFIL14,'(A)')' -1            $ NSUR'
      WRITE(IFIL14,'(A)')'  3            $ NTYPET'
      WRITE(IFIL14,'(1P,E14.6,A)') THKCYL,   ' $ TVAL'
      WRITE(IFIL14,'(A)')'  N            $ print ref. surf?'
      WRITE(IFIL14,'(A)')'  N            $ print Cij?'
      WRITE(IFIL14,'(A)')'  N            $ print loads?'
      ENDIF
C23456789012345678901234567890123456789012345678901234567890123456789012
C     End of (LENCYL.GT.0.001)
C
C End of input for Segment nseg+1 (cylindrical segment)
C
C Start GLOBAL data..
C
      WRITE(IFIL14,'(A)')'  1            $ NLAST'
      WRITE(IFIL14,'(A)')'  N            $ expanded plots?'
C
C Following for linear buckling of perfect shell...
      IF (INDX.EQ.1) THEN
         WRITE(IFIL14,'(A)')'  0            $ N0B'
         WRITE(IFIL14,'(A)')'  0            $ NMINB'
         WRITE(IFIL14,'(A)')'  0            $ NMAXB'
         WRITE(IFIL14,'(A)')'  1            $ INCRB'
         WRITE(IFIL14,'(A)')' 10            $ NVEC'
         WRITE(IFIL14,'(A)')'  0.           $ P'
         WRITE(IFIL14,'(1P,E14.6,A)') PRESS(ILOADX)/1000.0,  ' $ DP'
         WRITE(IFIL14,'(A)')'  0.           $ TEMP'
         WRITE(IFIL14,'(A)')'  0.           $ DTEMP'
         WRITE(IFIL14,'(A)')'  0.           $ OMEGA'
         WRITE(IFIL14,'(A)')'  0.           $ DOMEGA'
      ENDIF
C
C23456789012345678901234567890123456789012345678901234567890123456789012
C Following is for nonlinear axisymmetric collapse...
      IF (INDX.EQ.2) THEN
         WRITE(IFIL14,'(1P,E14.6,A)') PMAX/10.0,  ' $ P'
         WRITE(IFIL14,'(1P,E14.6,A)') PMAX/10.0,  ' $ DP'
         WRITE(IFIL14,'(A)')'  0.           $ TEMP'
         WRITE(IFIL14,'(A)')'  0.           $ DTEMP'
         WRITE(IFIL14,'(A)')'  20           $ NSTEPS'
         WRITE(IFIL14,'(A)')'  0.           $ OMEGA'
         WRITE(IFIL14,'(A)')'  0.           $ DOMEGA'
      ENDIF
C
C Following is for nonlinear non-axisymmetric bifurcation buckling
C of imperfect shell...
      IF (INDX.EQ.3) THEN
```

```
         WRITE(IFIL14,'(I4,A)') N0BX,  '             $ N0B'
         WRITE(IFIL14,'(I4,A)') NMINBX,'             $ NMINB'
         WRITE(IFIL14,'(I4,A)') NMAXBX,'             $ NMAXB'
         WRITE(IFIL14,'(I4,A)') INCRBX,'             $ INCRB'
         WRITE(IFIL14,'(A)')'  1            $ NVEC'
         WRITE(IFIL14,'(1P,E14.6,A)') PMAX       ,  ' $ P'
C BEG MAR 2008
         IF (INDIC.NE.-2)
     1   WRITE(IFIL14,'(1P,E14.6,A)') PMAX/1000.0,  ' $ DP'
         IF (INDIC.EQ.-2)
     1   WRITE(IFIL14,'(1P,E14.6,A)') PMAX/100.0,   ' $ DP'
C END MAR 2000
         WRITE(IFIL14,'(A)')'  0.           $ TEMP'
         WRITE(IFIL14,'(A)')'  0.           $ DTEMP'
C BEG MAR 2008
         IF (INDIC.EQ.-2)
     1   WRITE(IFIL14,'(A)')'  50           $ Number of steps'
C END MAR 2008
         WRITE(IFIL14,'(A)')'  0.           $ OMEGA'
         WRITE(IFIL14,'(A)')'  0.           $ DOMEGA'
      ENDIF
C
C Following is for nonlinear axisymmetric stress analysis...
      IF (INDX.EQ.4) THEN
         WRITE(IFIL14,'(1P,E14.6,A)') PMAX/10.0,  ' $ P'
         WRITE(IFIL14,'(1P,E14.6,A)') PMAX/10.0,  ' $ DP'
         WRITE(IFIL14,'(A)')'  0.           $ TEMP'
         WRITE(IFIL14,'(A)')'  0.           $ DTEMP'
         WRITE(IFIL14,'(A)')' 10            $ NSTEPS'
         WRITE(IFIL14,'(A)')'  0.           $ OMEGA'
         WRITE(IFIL14,'(A)')'  0.           $ DOMEGA'
      ENDIF
C
C Start CONSTRAINTS...
C
      IF (LENCYL.GT.0.001)
     1 WRITE(IFIL14,'(I4,A)') nseg+1,'           $ nseg'
      IF (LENCYL.LE.0.001)
     1 WRITE(IFIL14,'(I4,A)') nseg,'           $ nseg'
C
       Do 3000 iseg = 1,nseg
C
         if (iseg.eq.1) then
C Segment 1 constraint pole condition...
            WRITE(IFIL14,'(A)')'  1            $ number of poles'
            WRITE(IFIL14,'(A)')'  1            $ nodal point at pole'
            WRITE(IFIL14,'(A)')'  0      $ grounded how many stations?'
            WRITE(IFIL14,'(A)')'  N            $ joined to lower segs?'
```

```
         endif
C2345678901234567890123456789012345678901234567890123456789012345678901234567890123
         if (iseg.eq.nseg) then
C Segment nseg constraint conditions...
            WRITE(IFIL14,'(A)')'  0            $ number of poles'
            IF (LENCYL.GT.0.001)
     1      WRITE(IFIL14,'(A)')'  0       $ grounded how many stations?'
            IF (LENCYL.LE.0.001) THEN
               WRITE(IFIL14,'(A)')'  1    $ grounded how many stations?'
               WRITE(IFIL14,'(I4,A)') NMESH(nseg),'     $ INODE = node'
               WRITE(IFIL14,'(A)')'  1            $ IUSTAR constrained'
               WRITE(IFIL14,'(A)')'  1            $ IVSTAR constrained'
               WRITE(IFIL14,'(A)')'  0            $ IWSTAR constrained'
               WRITE(IFIL14,'(A)')'  1            $ ICHI   constrained'
               WRITE(IFIL14,'(A)')'  0.        $ D1=radial eccentricity'
               WRITE(IFIL14,'(A)')'  0.        $ D2=axial  eccentricity'
               WRITE(IFIL14,'(A)')'  N       $ bc same prebuck & buck.?'
               WRITE(IFIL14,'(A)')'  1            $ IUSTARB constrained'
               WRITE(IFIL14,'(A)')'  1            $ IVSTARB constrained'
               WRITE(IFIL14,'(A)')'  0            $ IWSTARB constrained'
               WRITE(IFIL14,'(A)')'  1            $ ICHIB   constrained'
            ENDIF
C           End of (LENCYL.LE.0.001) condition
         endif
C2345678901234567890123456789012345678901234567890123456789012345678901234567890123
         if (iseg.gt.1) then
            if (iseg.lt.nseg) then
               WRITE(IFIL14,'(A)')'  0            $ number of poles'
               WRITE(IFIL14,'(A)')'  0    $ grounded how many stations?'
            endif
            WRITE(IFIL14,'(A)')'  Y            $ joined to lower segs?'
            WRITE(IFIL14,'(A)')'  1      $ at how many stations joined?'
            WRITE(IFIL14,'(A)')'  1      $ INODE= node of current seg.'
            WRITE(IFIL14,'(I4,A)') iseg-1,'     $ JSEG=previous segment'
            WRITE(IFIL14,'(I4,A)') NMESH(iseg-1),'    $ JNODE prev.seg.'
            WRITE(IFIL14,'(A)')'  1            $ IUSTAR constrained'
            WRITE(IFIL14,'(A)')'  1            $ IVSTAR constrained'
            WRITE(IFIL14,'(A)')'  1            $ IWSTAR constrained'
            WRITE(IFIL14,'(A)')'  1            $ ICHI   constrained'
            WRITE(IFIL14,'(A)')'  0.           $ D1=radial eccentricity'
            WRITE(IFIL14,'(A)')'  0.           $ D2=axial  eccentricity'
            WRITE(IFIL14,'(A)')'  Y      $ bc same for prebuck & buck.?'
         endif
c
 3000 continue
c
C2345678901234567890123456789012345678901234567890123456789012345678901234567890123
C
```

```
      IF (LENCYL.GT.0.001) THEN
C Segment nseg+1 constraint conditions...
      WRITE(IFIL14,'(A)')'  0            $ number of poles'
      WRITE(IFIL14,'(A)')'  2          $ grounded at how many stations?'
      WRITE(IFIL14,'(A)')'  1            $ INODE= node of current seg.'
      WRITE(IFIL14,'(A)')'  1            $ IUSTAR constrained'
      WRITE(IFIL14,'(A)')'  0            $ IVSTAR constrained'
      WRITE(IFIL14,'(A)')'  0            $ IWSTAR constrained'
      WRITE(IFIL14,'(A)')'  0            $ ICHI   constrained'
      WRITE(IFIL14,'(A)')'  0.           $ D1=radial eccentricity'
      WRITE(IFIL14,'(A)')'  0.           $ D2=axial  eccentricity'
      WRITE(IFIL14,'(A)')'  N            $ bc same for prebuck & buck.?'
      WRITE(IFIL14,'(A)')'  0            $ IUSTARB constrained'
      WRITE(IFIL14,'(A)')'  0            $ IVSTARB constrained'
      WRITE(IFIL14,'(A)')'  0            $ IWSTARB constrained'
      WRITE(IFIL14,'(A)')'  0            $ ICHIB   constrained'
      WRITE(IFIL14,'(I4,A)') NMESH(nseg+1),'    $ INODE= node of constr'
      WRITE(IFIL14,'(A)')'  1            $ IUSTAR constrained'
      WRITE(IFIL14,'(A)')'  1            $ IVSTAR constrained'
      WRITE(IFIL14,'(A)')'  0            $ IWSTAR constrained'
      WRITE(IFIL14,'(A)')'  1            $ ICHI   constrained'
      WRITE(IFIL14,'(A)')'  0.           $ D1=radial eccentricity'
      WRITE(IFIL14,'(A)')'  0.           $ D2=axial  eccentricity'
      WRITE(IFIL14,'(A)')'  N            $ bc same for prebuck & buck.?'
      WRITE(IFIL14,'(A)')'  1            $ IUSTARB constrained'
      WRITE(IFIL14,'(A)')'  1            $ IVSTARB constrained'
      WRITE(IFIL14,'(A)')'  1            $ IWSTARB constrained'
      WRITE(IFIL14,'(A)')'  1            $ ICHIB   constrained'
      WRITE(IFIL14,'(A)')'  Y            $ joined to lower segs?'
      WRITE(IFIL14,'(A)')'  1            $ at how many stations joined?'
      WRITE(IFIL14,'(A)')'  1            $ INODE= node of current seg.'
      WRITE(IFIL14,'(A)')'  2            $ JSEG = previous segment'
      WRITE(IFIL14,'(I4,A)') NMESH(nseg),'      $ JNODE=node prev. seg.'
      WRITE(IFIL14,'(A)')'  1            $ IUSTAR constrained'
      WRITE(IFIL14,'(A)')'  1            $ IVSTAR constrained'
      WRITE(IFIL14,'(A)')'  1            $ IWSTAR constrained'
      WRITE(IFIL14,'(A)')'  1            $ ICHI   constrained'
      WRITE(IFIL14,'(A)')'  0.           $ D1=radial eccentricity'
      WRITE(IFIL14,'(A)')'  0.           $ D2=axial  eccentricity'
      WRITE(IFIL14,'(A)')'  Y            $ bc same for prebuck & buck.?'
      ENDIF
C     End of (LENCYL.GT.0.001) condition
C
      WRITE(IFIL14,'(A)')'  N            $ rigid body possible?'
C234567890123456789012345678901234567890123456789012345678901234567890123456789012
      IF (INDX.EQ.4) THEN
         do 3010 iseg = 1,nseg
         WRITE(IFIL14,'(A)')'  Y            $ output for seg. i?'
```

```
 3010    continue
         IF (LENCYL.GT.0.001)
     1   WRITE(IFIL14,'(A)')'  N              $ output for seg. nseg+1?'
         WRITE(IFIL14,'(A)')'  Y              $ output for rings?'
      ELSE
         do 3020 iseg = 1,nseg
         WRITE(IFIL14,'(A)')'  Y              $ output for seg. i?'
 3020    continue
         IF (LENCYL.GT.0.001)
     1   WRITE(IFIL14,'(A)')'  Y              $ output for seg. nseg+1?'
         WRITE(IFIL14,'(A)')'  Y              $ output for rings?'
      ENDIF
C
      RETURN
      END
c
c
c
C=DECK      x3y3
      SUBROUTINE x3y3(ifile,iseg,x1,y1,x2,y2,x03,y03,x3,y3)
c  input:
c  (x1,y1), (x2,y2) = end points that lie on the original ellipse
c  (x03,y03) = point where normals to the ellipse at (x1,y1) and
c              (x2,y2) intersect
c  output:
c  (x3,y3) center of curvature of the "equivalent" toroidal segment.
c
c  (x3,y3) are determined by Newton's method from two nonlinear
c  equations in dx,dy, in which dx,dy are the distances between
c  x03,y03 and x3,y3.
c
      double precision x1,y1,x2,y2,x3,y3,x03,y03
      double precision d1sq,d2sq,delsq,a1,a2,b1,b2
      double precision f1,f1p,f1pp, f2,f2p,f2pp
      double precision dx,dy,u,v
c
c  For a toroidal segment, the two distances from (x3,y3) to the two
c  segment end points (x1,y1) and (x2,y2) must be equal. In other
c  words the meridional radius of curvature of the torioidal segment
c  must be constant in that segment.
c
c  However, in the ellipse these two distances are different. The
c  square of the difference is given by delta**2 (delsq):
c
      d1sq = (x1 - x03)**2 + (y1 - y03)**2
      d2sq = (x2 - x03)**2 + (y2 - y03)**2
      delsq = d1sq - d2sq
c
```

```
c   Here we determine the location of the center of meridional
c   curvature of the "equivalent" torioidal segment by allocating
c   half of delsq to each (distance)**2, d1sq and d2sq. We have two
c   (distances)**2 that are equal:
c
c    (x1 - x03)**2 + (y1 - y03)**2 - delsq/2
c    (x2 - x03)**2 + (y2 - y03)**2 + delsq/2
c
c   We must solve the following two nonlinear equations for (dx,dy):
c
c   [x1 - (x03+dx)]**2 + [y1 - (y03+dy)]**2 =
c                        (x1 - x03)**2 +(y1 - y03)**2 -delsq/2   (1)
c
c   [x2 - (x03+dx)]**2 + [y2 - (y03+dy)]**2 =
c                        (x2 - x03)**2 +(y2 - y03)**2 +delsq/2   (2)
c
c   We use Newton's method:
c
c   For the ith Newton iteration, let
c
c   dx(i) = dx(i-1) + u
c   dy(i) = dy(i-1) + v
c
c   Then we develop two linear equations for u and v for the ith iteration:
c
c   u*(x03-x1+dx(i-1)) +v*(y03-y1 +dy(i-1)) = f1pp
c   u*(x03-x2+dx(i-1)) +v*(y03-y2 +dy(i-1)) = f2pp
c
c   solve them, add u and v to dx(i-1) and dy(i-1), respectively, and
c   iterate. We keep iterating until convergence is achieved.
c
      iter = 0
      dx = 0.
      dy = 0.
c
   10 continue
      iter = iter + 1
c
      a1 = 2.*(x03 - x1 + dx)
      a2 = 2.*(x03 - x2 + dx)
      b1 = 2.*(y03 - y1 + dy)
      b2 = 2.*(y03 - y2 + dy)
c
      f1 = (x1 - x03)**2 + (y1 - y03)**2 - delsq/2.
      f2 = (x2 - x03)**2 + (y2 - y03)**2 + delsq/2.
      f1p = f1 - x1**2 + 2.*x1*x03 - x03**2
     1          -y1**2 + 2.*y1*y03 - y03**2
      f2p = f2 - x2**2 + 2.*x2*x03 - x03**2
```

```
     1          -y2**2 + 2.*y2*y03 - y03**2
      f1pp = f1p - dx*2.*(x03-x1) -dy*2.*(y03-y1) -dx**2 -dy**2
      f2pp = f2p - dx*2.*(x03-x2) -dy*2.*(y03-y2) -dx**2 -dy**2
c
      u = (b2*f1pp - b1*f2pp)/(b2*a1 - b1*a2)
      v = (a2*f1pp - a1*f2pp)/(a2*b1 - a1*b2)
      dx = dx + u
      dy = dy + v
c
C23456789012345678901234567890123456789012345678901234567890123456789012
c     if (iter.eq.1) write(ifile,'(/,A,i3,/,A,A)')
c    1' ******** Results from Newton iterations for segment no.',iseg,
c    1' iter     x03       dx       y03         dy          u',
c    1'          v'
c     write(ifile,'(i3,1p,6e12.4)')
c    1 iter, x03, dx, y03, dy, u, v
c
      if (iter.gt.100) then
         write(ifile,'(A)')' No convergence.'
         call exit
      endif
c
      if (iter.lt.3) go to 10
      if (abs(u).gt.0.001*abs(dx)) go to 10
      if (abs(v).gt.0.001*abs(dy)) go to 10
c
c  Convergence has been achieved
c
      x3 = x03 + dx
      y3 = y03 + dy
c
      return
      end
======================================================================
```