27
2-4-81
J. Miller
24th

①

MASTER

R-1689

UCID- 18809

IDEAL BALLOONING MODES IN AXISYMMETRIC
MIRROR MACHINES

D. E. Baldwin
B. McNamara
P. Willmann

December 15, 1980

This is an informal report intended primarily for internal or limited external distribution. The opinions and conclusions stated are those of the author and may or may not be those of the Laboratory.

Work performed under the auspices of the U.S. Department of Energy by the Lawrence Livermore Laboratory under Contract W-7405-Eng-44.

DISTRIBUTION STATEMENT

IDEAL BALLOONING MODES IN AXISYMMETRIC MIRROR MACHINES

INTRODUCTION

A simple code is described that finds marginally stable ($\omega^2 = 0$) ballooning-type MHD modes, localized about a field line in an axisymmetric, open-ended, plasma confinement device. The equations are based on a lower bound for the perturbed energy δW , derived by W. Newcomb¹ from the ideal MHD energy principle, and are cast in the form of a Ricatti equation for the first derivative of the eigenfunction, with the open boundary conditions that this derivative vanish at the plasma boundary down each field line. The input to the code is the two-dimensional shape of a field line, the field strength $B(s)$, and parameters to define pressure profiles throughout the system. The objective is to find the highest plasma pressures for which the given line is MHD-stable.

THE MHD EQUATIONS

It is assumed that the plasma-pressure profiles can be given as separable functions of the magnetic flux ψ and field strength B at every point along the line between $s = 0$ and $s = s_{\text{end}} \approx L$. The equations to be solved are then

$$\frac{dY}{ds} + \frac{1}{f} Y^2 = g, \quad Y(0) = 0, \quad Y(L) = 0, \quad (1)$$

where

$$\frac{1}{f} = \frac{r^2 B^3}{Q}, \quad g = - \frac{k}{r B^2 H} \left(H \hat{p}_{\parallel} + Q \hat{p}_{\perp} \right) \frac{\partial \psi}{\partial s}, \quad (2)$$

the pressure tensor is (when $\hat{b} \equiv \vec{B}/B$)

$$P = \alpha(\psi) \left\{ \hat{p}_{\perp}(B) I + \left[\hat{p}_{\parallel}(B) - \hat{p}_{\perp} \right] \hat{b} \hat{b} \right\}, \quad (3)$$

and

$$H = \frac{B^2}{4\pi} + B r \frac{\partial p_{\perp}}{\partial B}, \quad (4)$$

DISCLAIMER

This document contains information which is proprietary to the Lockheed Martin Corporation. It is the property of Lockheed Martin Corporation and is loaned to your agency. It and its contents are not to be distributed outside your agency without the express written approval of Lockheed Martin Corporation.

$$Q = \frac{B^2}{4\pi} - B\alpha \frac{\partial \hat{P}_1}{\partial B} . \quad (5)$$

The radius of the field line, $r = r(s)$, can be calculated from the vacuum field coils by EFFI² or by a finite- β equilibrium code such as that used for reversed-field studies.³ The curvature,

$$k = - \hat{h} \cdot \nabla \hat{h} \equiv \frac{1}{B^2} \left(B_z \frac{\partial B}{\partial s} - B_x \frac{\partial B}{\partial s} \right) , \quad (6)$$

is positive when the center of curvature is on the larger flux side of the field line, making g negative in bad curvature regions.

The eigenfunction X is found from

$$\frac{dX}{ds} = \frac{X}{f} Y , \quad X(0) = 1 . \quad (7)$$

The GEAR package⁴ is used to integrate Eq. (1) from 0 to L , the end of the plasma, and a simple zero finder is used to adjust a chosen parameter β_0 in the pressure profiles until $Y(L, \beta) = 0$. Equation (7) is then integrated for the eigenfunction. Typically, one looks for the marginal β in one call for given β 's in the others.

The pressure profiles must be chosen to represent the expected variations in an experiment. It is usually found that field lines near the edge of the plasma, where $\alpha \rightarrow 0$, are most unstable for a given pressure gradient, or $\partial\alpha/\partial s$. A given field line may have several magnetic wells along its length, each capable of confining its own mirror-trapped tensor-pressure plasma dropping to zero at some field maximum. Regions in which plasma is electrostatically trapped should be described by a constant scalar pressure up to the edge of the electrostatic plug, and by a simple model of how this scalar pressure drops to zero at the peak of the plug. Careful coding is required to identify each region where a different profile is to be used. Typical profiles used in the code listed in Appendix A are obtained from

$$\alpha = 1 - \psi/\psi_{\text{edge}} \quad (8)$$

and either (a) or (b):

(a) for tensor-pressure plugs--

$$\frac{\partial P_1}{\partial B} = - \frac{P_0}{B_M} , \quad P_1 = P_0 \left(1 - \frac{B_M}{B} \right) \quad (9)$$

and

$$\frac{\partial p_{\parallel}}{\partial B} = \frac{p_0}{B_0} \ln\left(\frac{B_M}{B}\right), \quad p_{\parallel} = p_0 \left[1 - \frac{B}{B_M} - \frac{B}{B_M} \ln\left(\frac{B_M}{B}\right) \right], \quad (10)$$

where B_M is the field where $p \rightarrow 0$. This should be the lower of the two nearest maxima in the locality.

(b) for scalar-pressure-overlapping plasma in a plug--

$$p = p_0 \left(\frac{B}{B_{\min}} \right)^{1.2}, \quad (11)$$

where $B_{\min}$ is the minimum B in the plug and p_0 the scalar pressure in the neighboring electrostatic well. The power 1.2 has been chosen to model the decrease of solenoid pressure with B when the mirror is subject to charge-exchange pumping.

The unstable side of the marginal line in parameter space can be deduced by studying the sign of the growth rate determined by

$$\gamma^2 = \int ds \left(fX'^2 + gX^2 \right) = \int ds X^2 \left(\frac{\gamma^2}{f} + g \right). \quad (12)$$

It is clear that g supplies the drive for a perturbation X and that f controls the bending energy which must be supplied by the plasma perturbation. The Euler equations were solved to minimize γ^2 to zero.

The GEAR package is used to integrate Eq. (1) from 0 to L , the end of the plasma. A chosen parameter, p_0 , in the pressure profiles is adjusted until $\gamma(L, p_0) = 0$. This is a tricky calculation, because sometimes values of p_0 cannot be handled by GEAR. A heuristic algorithm was developed to search for an interval of p_0 containing a root. The IMSL routine ZBRENT is then used to evaluate the root to the prescribed accuracy.

THE FLUTE-STABILITY CONDITION

When a given field line is unstable to a simple flute mode, it is a waste of time to evaluate the ballooning criterion, as it will yield negative pressures. The following integral is evaluated as a low- β flute-stability condition:

$$F = \int_{r_1}^{r_2} \frac{k ds}{XB^2} = \int ds \left(\frac{1}{XB^2} \right)' \sin^{-1}(X') > 0, \quad (13)$$

where $k(s)$ is the radius of the field line between points where

$$\frac{dk}{ds} \left(s = \frac{1}{m} \right) = 0.$$

AN EXAMPLE

The code listed in Appendix A is designed for axisymmetric mirror cusp designs as shown in Fig. 1. There are two different kinds of field lines, one running through the central cell and the ring plug, and one in the cusp region connecting the ring and spindle plugs. These field lines are shown in Figs. 2 and 3, with various boundaries marked for the several pressure profiles.

We choose to describe the field lines in the cusp region (field line 2 from Fig. 1) in more detail. Their curvature is shown in Fig. 4, where the positive spikes represent bad curvature, and their field strength is shown in Fig. 5. The BALLOON code plots the coefficients and eigenfunction at the half-way point in the search of β_c , the β in the central cell. Figure 6 shows the driving function g , which is worst in the spindle mirror, has a large line-tying effect in the cusp region where B is lowest, and is apparently small but negative in the ring.

The parallel-pressure profile (Fig. 7) shows the overlap of the central-cell plasma with the mirror plasmas. This overlap contributes to the instability of the mirror at large β_c but also provides the line-tying effects at lower β_c . The eigenfunction at these β 's (Fig. 8) shows that the marginal mode is flute-like in the ring and bends over to its smallest values in the cusp and to a small value in the spindle. Figure 9, a plot of the bending-energy term, Y^2/f of Eq. (12), shows why the mode dips close to zero as it approaches the minimum of B in the cusp. The required bending energy between the spindle and the cusp is also quite large because $1/f \approx r^2 B$, the flux through the spindle, which is constant. Since f does not vary much here, Y must vary to avoid the large stabilizing effect of the cusp. Thus, although the worst g values are in the spindle, the marginal mode is largest in the ring. The variation of $1/f$ with B in the ring (r being almost constant) allows the mode to bend.

The net result (Fig. 10) shows the ring β approaching asymptotically to about 12% for central-cell β 's of 50% or more. Similar results are obtained for field line 1, as shown in Fig. 11.

The ring β in this design can be improved by moving the bad ring curvature closer to the mirrors, thus reducing the β at which the drive is bad. Field line 1 actually has some bad curvature in the central cell, but the bending energy term still confines the marginal mode to the ring plasma.

MODIFYING THE CODE

Different configurations, based on Elmo Bumpy Torus (EBT) rings, SURMACS, or other choices of field-design principle, require some code modification. Essentially, subroutine SEARCH must be completely rewritten to pick off the right maxima and minima of B and their locations from the species of field line to be tested for stability. The results are to be passed to DIFFUN by a suitable COMMON block. This routine evaluates $p_{\perp}, p_{\parallel}$ and hence g, f as functions of s .

All the other required routines are in the source file, available on FILEM by typing "read 374 .ETMR PB." The graphics routines are a soft-option set taken from the "do" system on the Cray-1, available by typing "do."

RUNNING THE CODES

EFFI

The EFFI code, described in UCID-17621, will output $B(s)X(s)$ for any desired field to the PLOTFILE, which should be used as input to BALLOON. The input field to EFFI for the example described above is given in Appendix B.

BALLOON

The BALLOON code requires a short input file, BALLIN, to specify the field line characteristics, and the output file from EFFI, which must have its name changed to EGRPH. The input file, in namelist format, is also listed in Appendix B. The following 13 variables can be specified, although for many the default value is generally adequate:

SEND = The length of the field line to process.
 ALPHA = The profile factor (DEFAULT ALPHA = 0).
 P0 = The pressure parameter we are solving for. We need an
 initial guess. NOTE: The program alters P0.
 BSPINDLE = B on the z axis in the spindle plug.
 BRZO = B for R = 0 and z = 0.
 IEQ = 1 for lower equation, IEQ = 2 for upper equation
 (DEFAULT IEQ = 1).
 IFL = The number of the field line, IFL = -1 indicates a field
 line where z is monotonic with s. IFL = 0 means no more
 data (DEFAULT IFL = 1).
 IPR ≠ 0 means most printing eliminated (DEFAULT IPR = 0).
 IPLOT ≠ 0 means most plots eliminated (DEFAULT IPLOT = 0).
 BETAC1 = Lowest central cell (DEFAULT BETAC1 = 0.04).
 BETAC2 = Highest central cell (DEFAULT BETAC2 = 0.8).
 NBETAC = The number of β_c 's. The NBETAC β_c 's are evenly spaced
 between BETAC1 and BETAC2 (DEFAULT NBETAC = 25).
 TITLE = 8 word title for printing and plotting (DEFAULT is BLANKS).

ACKNOWLEDGMENT

The first version of the code was written by Terry Chu at LLNL and has been substantially modified to support all these features.

REFERENCES

1. W. A. Newcomb, Lawrence Livermore National Laboratory, Livermore, CA, private communication (1980).
2. S. J. Sackett, EFFI: A Code for Calculating the Electromagnetic Field, Forces, and Inductance in Coil Systems of Arbitrary Geometry, Lawrence Livermore National Laboratory, Livermore, CA, UCID-17621 (1977).
3. J. K. Boyd, S. P. Auerbach, P. A. Williams, H. L. Berk, and B. McNamara, Computational Methods for Reversed-Field Equilibrium, Lawrence Livermore National Laboratory, Livermore, CA, UCRL-84044, Rev. 1 (1980).
4. A. C. Hindmarsh, GEAR: Ordinary Differential Equation System Solver, Lawrence Livermore National Laboratory, Livermore, CA, UCID-30,001 (1974).

FIG. 1. Coil set and typical lines for Mirror-Cusp Tandem. Field line 1 is the upper line, field line 2 is the lower line.

FIG. 2. Field line 1 from Fig. 1. Tensor-pressure plasma is confined between D-B, scalar pressure between A-D, dropping to zero at the minimum field in D-B.

FIG. 3. Field line 2 from Fig. 1. Tensor-pressure confined in spindle, A-B, and ring mirrors, C-D. Scalar pressure plasma is confined in B-C with drop-off to zero at minimum field points in spindle and ring.

FIG. 4. Curvature of the second field line. Bad regions in the spindle and ring mirrors are similar.

FIG. 5. B on the second field line shows mirror ratios of 4:5 in spindle and ring.

FIG. 6. The balloon drive coefficient g . The central spike is stabilizing and disconnects the spindle and ring parts of the eigenfunction.

FIG. 7. The parallel-pressure profile indicates high β in the central cell.

FIG. 8. Solution of Riccati equation Y , and the balloon eigenfunction X .

FIG. 9. The bending-energy, Y_2/f , excludes the mode from the spindle.

FIG. 10. Ring β (and spindle β) vs central-cell β . The plasma is MHD unstable above this line.

FIG. 11. Ring β vs central-cell β for field line 1 of Fig. 1. Result is similar to Fig. 10.

APPENDIX A
BALLOON Code Listing

(Graphics and other library routines are not included in this listing.)

```

program balloon
  common/block1/s(500),x(500),z(500),bx(500),bz(500)
  * ,b(500),xkappa(500),xf(500),yf(500)
  common/block2/index,p0,alpha,pc,dcdpsi,ndat,neon,lk,brz0,betac1
  * ,ca,sac,cc,cd,cbd,cb,scond,csail,pcsr,ppar,bzpindlc,cfun,gfun
  * ,ch,hac,bac,bcc,bcbd,bcb,r.knt1,betac2,nbetac,ieq,ifl
  * ,iplot,ipr
  common /fastfp/ lgraph,icase,lond,iquit,istep,lin,lout,lty
  dimension dbxds(500),dbzds(500),y0(1),cpper(500),appcr(500)
  * ,betac(500),beta0(500),agfun(500),afgun(500),ay2dgr(500)
  external read
  neq = 1

c  readat reads in and prints the input data.
  1 call readat
  ndat1=scond/ds+1
  call flute
  tndatnds=2.

c  calculate dbx/ds, dbz/ds, kappa at each point.
  do 5 i=2,ndat-1
    dbxds(i)=(bx(i+1)-bx(i-1))/tnds
    dbzds(i)=(bz(i+1)-bz(i-1))/tnds
    xkappa(i)=(bz(i)*dbxds(i)-bx(i)*dbzds(i))/(b(i)*b(i))
  5 continue
  dbxds(1)=(bx(2)-bx(1))/ds
  dbzds(1)=(bz(2)-bz(1))/ds
  xkappa(1)=(bz(1)*dbxds(1)-bx(1)*dbzds(1))/(b(1)*b(1))
  dbxds(ndat)=(bx(ndat)-bx(ndat-1))/ds
  dbzds(ndat)=(bz(ndat)-bz(ndat-1))/ds
  xkappa(ndat)=(bz(ndat)*dbxds(ndat)-bx(ndat)*dbzds(ndat))/(
    b(ndat)*b(ndat))

c  search finds the locations and amounts of maxima and minima of L.
  call search
  dcdpsi = -2./(b(1)*b(1))*2
  if(ifl.lt.0) go to 8

c  field line non-monotonic.
  isa = (sac+.00001)/ds + 1
  ic = (cc+.00001)/ds + 1

c  r0 = the largest value of x between a and c.
  call eninmx(x(isa),1,ic-isa+1,1,ra,r0)
  dcdpsi = 2./(bzpindlc*r0)*2
  conc = 3.1415926535.e-7/brz0*2
  conb = 3.1415926535.e-7/bzpindlc*2
  if(nbetac.gt.1) betac = (betac2-betac1)/(nbetac-1)
  icase = icase + 1
  p0sav = p0

c  cycle on the betac's.
  do 200 ibetac = 1 , nbetac
    istep = ibetac
    neon=0

```

```

c      betac = ratio of plasma pressure to magnetic field pressure
c      in the central cell.
      betac(ibetac) = betac1 + dbetac*(ibetac-1)
      icount = -8
      pc = betac(ibetac)/conc
c      search for correct p0; start with p1 = guessed p0 for first
c      betac. set p1(2nd betac) = p0(first betac). for later betac's
c      use p1 = linear extrapolated value from two previous betac's.
      p2 = 1.01*p1. try new points depending on results. when find
c      interval that contains the zero use abrent. if all fails go to 230
c      print results and look for more data.
      p1 = p0 + p0 - p0sav
      if(ibetac.eq.2) p1 = p0
      p0sav = p0
32  p2 = 1.01*p1
      jcount = 0
      y1 = resid(p1)
      ay1 = abs(y1)
      if(index.lt.0) go to 43
31  y2 = resid(p2)
33  if(index.lt.0) go to 43
      if(jcount.gt.100) go to 230
      jcount = jcount + 1
      if(y1*y2.lt.0.) go to 40
      if(abs(y2).lt.abs(y1)) go to 33
      y2 = y1
      p2 = p1
      p1 = (.99+.005*rand(x))*p2
      if(ibetac.eq.1) p1 = .90*p2
      y1 = resid(p2)
      go to 33
30  p1 = p2
      p2 = (1.00+.005*rand(x))*p1
      if(ibetac.eq.1.and.abs(y2).gt.0.6*ay1) p2 = 1.23*p1
      y1 = y2
      go to 34
40  ifail = 0
      nsig = 4
      ftol = .03
c      abrent finds that p0 which makes y(send) = 0.
      itmax = 25
      dconv = abrent(resid,ftol,nsig,p1,p2,itmax,ifail)
      p0 = p2
c      beta0 = ratio of plasma pressure to magnetic field pressure
c      on the separatrix in the central cell.
      beta0(ibetac) = conc*p0
c      check for error in gear.
      if(ifail.eq.0) go to 50
49  if(ipr.eq.0) write(lout,1009) index,p0,pc
      icount = icount + 1
c      try a different p0
      p0 = p0sav*(1.5)**icount
      p1 = p0
      if(icount.lt.5) go to 32
      go to 230

```

```

50 if(ipr.eq.0) write(1out,1010) p0,pc,beta0(ibetac),betac(ibetac)
   if(ibetac.ne.(nbetac+1)/2) go to 200
   if(ipr.ne.0.and.iplot.ne.0) go to 200
c   evaluate y at each mesh point for printing and plotting.
   s0=s(1)
   y0(1)=0.
   yf(1)=0.
   index=1
   h0=1.e-6
   do 50 i=2,ndat1
     cout = s(i)
     call drive(1,c0,h0,y0,cout,epsil,10,index)
     index=2
     yf(i)=y0(1)
50   continue
c.. now integrate second equation
   ncon = 1
   y0(1)=1.
   xf(1)=1.
   c0=s(1)
   index=1
   h0=1.e-6
   do 150 i = 2 , ndat1
     cout = s(i)
     call drive(1,c0,h0,y0,cout,epsil,10,index)
     index=2
     xf(i)=yf(i)
150  continue
c   plot on the middle step yf, xf, ppar, pper, g, 1/g, and yf2/g.
   is = 10
   do 170 i = 1 , ndat1
     call diffun(1,s(i),0.,ydot)
     cpper(i) = pper
     cpper(i) = pper
     cffun(i) = cfun
     cffun(i) = 1./ffun
     cy2df(i) = yf(i)**2/ffun
170  continue
   if(ipr.eq.0) write(1out,1013)
   .   (s(i),cpper(i),cpper(i),cffffun(i),-ffun(i),cy2df(i)
   .   ,yf(i),xf(i),i=1,ndat1)
   if(.not.gt.0) go to 200
   call plotfx(yf,s,ndat1,12,'yf vs. s')
   call plotfx(xf,s,ndat1,34,'xf vs. s')
   call plotfx(cpper,s,ndat1,1234,'pper vs. s')
   call plotfx(cpper,s,ndat1,1234,'pper vs. s')
   call plotfx(cffun,s,ndat1,1234,'g vs. s')
   call plotfx(cffun,s,ndat1,1234,'1/g vs. s')
   call plotfx(cy2df,s,ndat1,1234,'y**2/g vs. s')
200  continue
250 if(iplot.gt.0) go to 260
   call plotfx(b,s,ndat1,1234,'b vs. s')
   call plotfx(xkappa,s,ndat1,1234,'k vs. s')
   call plotfx(x,z,ndat1,1234,'x vs. z')

```

```

250 jbetae = min(1,betae,nbetae)
    call plotfx(beta0,betae,jbetae,1254,'beta0 vs betae')
    write(lout,1015) (betae(i),beta0(i),i=1,jbetae)
    if(1,betae.lt.nbetae) write(lout,1016) betae(1,betae)
c   return for new data case.
600 go to 1
1009 format(' errors: index = 'i3,' p0 = 'e13.5,' pc = 'e13.5)
1010 format(' integration completed')
    ' p0 = 'e13.5,' pc = 'e13.5,' beta0 = 'e13.5,' betae = 'e13.5/'
1013 format(5x,'s',9x,'ppar'8x,'pper'10x,'s',10x,'1/3'7x,'p'2/4'
    6x,'y'10x,'x'/'(1p0e12.5))
1015 format('///' betae beta0/(2*12.5))
1016 format('/// program failed to converge for betae = 'f7.5)
end
subroutine readat
common/block1/a(500),x(500),z(500),bx(500),bz(500)
    ,b(500),xkappa(500),xf(500),yf(500)
common/block2/index,p0,alpha,pc,dcdpsi,ndat,ncon,lt,brn0,betae1
    ,sa,sac,sc,cd,eb0,cb,cond,epsil,ppar,pper,bspindle,sgun,g,mn
    ,ds,bac,bcae,buc,bad,babd,bab,ndat1,betae2,nbetae,icq,ifl
    ,iplot,ipr
    common /fastfp/ lgraph,icase,iend,icuit,istop,lin,lout,ltag
    dimension dummy(9,1000),title(8)
    data ifl/-1/,epsil/1.e-5/,alpha/0./,nbetae/25/,icq/1/
    ,betae1/.04/,betae2/.0/,title/8M'/'
    namelist/nml/ cond,alpha,p0,bspindle,brn0
    ,icq,ifl,ipr,iplot,betae1,betae2,nbetae,title
c   cond = length of field line to process.
c   alpha = profile factor. (default alpha=0.).
c   p0 = the parameter solving for - we need an initial guess.
c   bspindle = b on the z axis in the spindle plug.
c   brn0 = b for r=0. and z=0.
c   icq=1 if lower equation, icq=2 if upper equation. (default icq=1)
c   ifl = 0 of field line; ichs = starting address of field line.
c   on ifl < 0 indicates a field line where z is non-linear with r.
c   ifl = 0 means no more data. (default ifl=-1).
c   ipr not = 0 eliminates most printouts. (default ipr=0).
c   iplot not = 0 eliminates most plots. (default iplot=0).
c   betae1 = lowest center coll betae. (default betae1=.01).
c   betae2 = highest center coll betae. (default betae2=.0).
c   nbetae = 0 of betae's; betae1 <= betae <= betae2. (default 25).
c   title = 8 word title for printing and plotting. (default blank).
c
c   initialize plotting.
c
c   if(icase.gt.0) go to 10
c   initialize files first time through.
    call keep90(1,3)
    call fr90id
    call keep90(4rball,3)
c   open output file.
    call link('unit3=(bo,create,nc),unit5=(ballout,create,nc)')
c   open input(namelist) file.
    call open(lin,"ballin",0,len)
c   open effi data file.
    call open(20,"e9rph",4,len)

```

```

c   read in namelist variables.
10 call ddi(nml,lin,lout,1)
   if(ifl.eq.0) call exit(1)
   nread = min0(9000,len)
c   read up to 1000 eeffi produced points.
   call rdcb(20,dummy,nread,0)
21 if(iocait(20)) 21,22
22 do = dummy(1,2) - dummy(1,1)
c   do = step size in s.
   ndat=send/do*2
   j = 1
c   check if first field line.
   if(iabc(ifl).eq.1) go to 50
c   where does field line ifl start?
   jfl = 1
   do 25 i = 2 , 1000
     if(dummy(1,i).gt.0) go to 25
     jfl = jfl + 1
   if(jfl.eq.iabc(ifl)) go to 26
25 continue
c   start of field line can't be found.
   call exit(1)
26 j = i
c   is all data needed already in core?
   if(jndat.le.1000) go to 50
   jabc = SM(j-1)
   call rdcb(20,dummy,5*ndat,jabc)
35 if(iocait(20)) 35,36
c   put eeffi data in appropriate vectors.
40 do 60 i=1,ndat
   k = i + j - 1
   s(i)=dummy(1,k)
   x(i)=dummy(2,k)
   z(i)=dummy(4,k)
   bx(i)=dummy(5,k)
   bz(i)=dummy(7,k)
   b(i)=dummy(8,k)
   x2(i)=0.
c   have we come to a new field line?
   if(dummy(1,i+1).eq.0.) go to 70
60 continue
   go to 60
70 ndat = i
c   write namelist data on output file and first plot.
62 write(5,1005) title,send,alpha,ifl,ioc,bapindic,bra0
   call setch(0.,22.,1,0,2,0)
   write(1graph,1005) title,send,alpha,ifl,ioc,bapindic,bra0
   call frame
c   if ipr = 0 print eeffi data.
   if(ipr.eq.0)
     . write(5,1000) (s(i),x(i),z(i),bx(i),bz(i),b(i),i=1,ndat)
     return
1000 format(6x,'s'11x,'x'11x,'z'11x,'bx'10x,'bz'10x,'b'/(6f12.6))

```



```

1005 format('0c0//') send = 'F3.5/
.   ' alpha = 'F7.5/' if1 = 'i2/' ieq = 'i2/
.   ' bspindle = 'e13.5/' brz0 = 'e13.5//')
    chu
    subroutine diffun(n,s1,y,ydot)
    dimension y(n),ydot(n)
    common/block1/e(500),x(500),z(500),bx(700),bz(500)
    = .b(500),xkappa(500),xf(500),yf(500)
    common/block2/index,p0,alpha,pc,dcapoi,ndat,neon,lk,brz0,betac1
    .   ,cd,cc,ce,cd,cbd,cb,ccnd,epsil,ppar,pper,bp,indlo,fdun,cfun
    .   ,ds,hda,bacc,bac,bad,babd,bab,ndat1,betac2,nbtac,icq,ici
    .   ,lplot,ipr
    common /fact7p/ lgraph,leaze,iond,iqut,istop,lin,icut,iteg
c... locate cl in array s
    if(s1.gt.s(ndat)) s1=s(ndat)
    fourpi=12.53637e-07
    isg
100 isg=1
    if(s1-s(1)) 110,100,100
110 ind1=1
c interpolate.
    ratio=(s(ind1)-s1)/(s(ind1)-s(ind1-1))
    rx=(x(ind1)-(x(ind1)-x(ind1-1))*ratio
    buad=(b(ind1)-(b(ind1)-b(ind1-1))*ratio
    xkappa=(xkappa(ind1)-(xkappa(ind1)-xkappa(ind1-1))*ratio
    yf=(yf(ind1)-(yf(ind1)-yf(ind1-1))*ratio
    bu = dmin(buad,bu)
    lq1 = lce
    if(s1.lt.cd) go to 150
    bu = dmin(bu,bu)
    bu1 = bu
c find in what region we are; cd ,cc ,ce ,cd ,cbd ,cd and cd
c are calculated in search.
120 if(s1.lt.cd.or.s1.gt.sb) go to 230
    if(s1.lt.cac) go to 190
    if(s1.lt.ce) go to 200
    if(s1.lt.cd) go to 210
    if(s1.ge.cbd) go to 220
190 ppar1=pc*(1.-buad/bu)
    ppar2=pc*(1.-buad/bu1)**(1.2)
    if(buad.gt.bu) go to 205
    if(ppar1.ge.ppar2) go to 212
205 ppar=ppar2
    ppar=pper
    cparcb=1.2*pc/(1.-buad/bu1)**buad1.2
    cparcb=cparcb
    go to 205
c call procedure: ppar = ppar = pc.
210 ppar=pc
    ppar=pc
    cparcb=0.
    cparcb=0.
    go to 205

```

```

c p tensor only.
220 if(bval.gt.bn) go to 230
222 ppar=p0*(1.-bval/bn)
ppar=p0*(1.-bval/bn-bval/bn*aalog(bn/bval))
qparcb=p0/bn*(alog(bn/bval))
dparcb=p0/bn
go to 230
c p = 0 case.
230 ppar = 0.
ppar = 0.
dparcb = 0.
232 h2du0 = bval*bval/4ourpi
hfun=2du0*bval*alpha*dparcb
qfun=2du0*bval*beta*dparcb
qfun=xtapu/(hfun*rad*bval*(1+beta*dparcb))
if(i.eq.2) gfun=gfun+(xtapu/(rad*bval))**2/bval*(qfun*(qfun+hfun)
/hfun**3. Epper**4. Epper*(2. Epper*Epper)**2/(1+du0*2. Epper))
gfun = rad*rad*bval**3/qfun
ydot(1)=gfun*Efun*y(1)*y(1)
if(i.eq.1) ydot(1)=rad*rad*bval**3/qfun*y**2
if(xk.lt.10) write(1out,3000) p(1),ydot(1),q1,gfun,qfun,h2du0,qfun
is = 10
return
3000 format(4,F10.3,1)
end
function resid(p30)
common/block1/c(300),x(300),z(300),lx(300),bz(300)
= ,b(300),xtappa(300),x9(300),y9(300)
common/block2/index,p0,alpha,pc,dcspci,ndet,norm,lt,lum,lt1
,sc,pac,cc,cd,cbd,cb,cond,cpoil,ppar,qpar,bvindl,lt2,lt3
,ds,bca,bcac,bac,bad,babd,bab,ndet1,batad1,batad2,batad3,lt4,lt5
,iplot,ipr
common /fastfp/ lgraph, lcase, lcond, lquit, lcbp, lln, llnf, llt
p0 = p30
alt=10
50 cout=10
lt=10
c3=c(1)
index=1
y0=0.
70 lk = 10
call drive(1,c0,lt,y0,cout,cpoil,10,index)
c if lk < 10 call diffun to get diagnostic printouts.
if(lk.lt.10) call diffun(1,cout,y0,ydot)
if(index.eq.0) go to 80
c error return.
return
80 cout=cout+10
lt=lt+2
if(cout.lt.(cond*.51)) go to 70
if(ipr.eq.0) write(1out,1011) p0,y0
call empty(5)
resid = y0
return

```

```

1001 format(' integration completed: p0 = 'a13.5,' g0 = 'c13.5)
    end
    subroutine flute
    common/block1/a(500),x(500),z(500),bx(500),bz(500)
    = ,b(500),xkappa(500),xf(500),yf(500)
    common/block2/index,p0,alpha,pc,dadpsi,ndat,neon,lk,braz0,betac0
    ,ca,cae,ce,cd,sbd,cb,scnd,epsil,ppcr,pper,bapindlc,ffun,gf'n
    ,da,bca,bce,bac,bad,babd,bab,ndat1,betac2,nbetac,icq,icf
    ,iplot,ipr
    common /fastfp/ agraph,leace,icnd,icuit,istop,i'n,icnt,itty
    flut = 0.
    twoda = 2.0da
    do 10 i = 2 , ndat1
    dxda = (x(i+1)-x(i-1))/twoda
    if(dxda.le.0.) go to 20
    flut = (1./(x(i+1)exp(i+1)exp2)-1./(x(i-1)exp(i-1)exp2))/twoda
    flut = flut + flut*asin(dxda)
10 continue
20 flut = da*flut
write(1out,100) flut
100 format(' flut = c13.5)
    return
    end
    subroutine search
    common/block1/a(500),x(500),z(500),bx(500),bz(500)
    ,b(500),xkappa(500),xf(500),yf(500)
    common/block2/index,p0,alpha,pc,dadpsi,ndat,neon,lk,braz0,betac0
    ,ca,cae,ce,cd,sbd,cb,scnd,epsil,ppcr,pper,bapindlc,ffun,gf'n
    ,da,bca,bce,bac,bad,babd,bab,ndat1,betac2,nbetac,icq,icf
    ,iplot,ipr
    common /fastfp/ agraph,leace,icnd,icuit,istop,i'n,icnt,itty
    ca = 0.
    ce = 0.
    cd = 0.
    if(191.11.0) go to 10
    lca = b(1)
    do 10 i = 2 , ndat1
    if(191.11.0.000000) go to 12
    lca = b(i)
10 continue
12 ii = i
    ca = c(ii-1)
    lca = b(ii)
    do 14 i = ii+1 , ndat1
    if(b(i).gt.1.01*ca) go to 16
    lca = b(i)
14 continue
16 ii = i
    cae = c(ii-1)
    lca = b(ii)
    do 17 i = ii+1 , ndat1
    if(b(i).11.0.99*cae) go to 18
    lca = b(i)
17 continue

```


APPENDIX E
EFFI Input and BALLIN (BALLOON Input)

EFFI Input

```

effi ar12
$
coil= a $
loop 0. 0. 0. 4.5 0. 0. 36. .049 25.06 $
loop 0. 0. 0. 22.3 5.75 0. 0. 1.2 .95 25.06 $
loop 0. 0. 0. 29.75 5.75 0. 0. 1.2 .95 25.06 $
loop 0. 0. 0. 22.3 2.67 0. 0. 1.7 1.25 -25.06 $
loop 0. 0. 0. 29.75 1.5 0. 0. 1.5 .46 -2.375e7 $
loop 0. 0. 0. 29.75 3.02 0. 0. 1.5 .95 -2.5e7 $
loop 0. 0. 0. 25.25 3.3 0. 0. .65 .2 -2.5e7 $
loop 0. 0. 0. 25.025 3.3 0. 0. .65 .2 -15.06 $
loop 0. 0. 0. 26.8 3.3 0. 0. .65 .2 -2.5e7 $
loop 0. 0. 0. 32.5 2.6 0. 0. .6 .5 -2.5e7 $
loop 0. 0. 0. 35.5 3.9 0. 0. .4 .4 -2.5e7 $
loop 0. 0. 0. 37.5 3.9 0. 0. .3 .3 -2.5e7 $
loop 0. 0. 0. 39.5 4.75 0. 0. .2 .2 -2.5e7 $
loop 0. 0. 0. 42.5 6.75 0. 0. .2 .2 -2.5e7 $
loop 0. 0. 0. 45.5 9. 0. 0. .1 .1 -2.5e7 $
loop 0. 0. 0. 24.5 3.6 0. 0. 1. .2 -1.06 $
loop 0. 0. 0. 27.5 3.6 0. 0. 1. .2 -1.06 $
loop 0. 0. 0. 26.5 5. 0. 0. 1. .2 3.06 $
$
b-line 2.5 0. 0. .2 48. $
b-line .45 0. 29.75 .2 45. $
z= 10. 25. 35. 0. 6. 6. 0. 0 1 $
$

```

BALLIN Input File

```

FILE=EFFI.DAT
FILE=1 EFFI DETAC1=0.04 DETAC2=0.8 NDETAC=51 SEND=30.
FILE=1, COILNAME=2.57 BRZG=1.53 PG=1.025
FILE=1 EFFI DETAC1=0.04 DETAC2=0.8 NDETAC=51
FILE=1, COILNAME=2.57 BRZG=1.53 PG=1.025
FILE=1

```