The Use of the DCON Stability Code for Determining the MHD Stability of Axisymmetric Toroidal Plasmas

Version 3.60, September 5, 2002

Alan H. Glasser
Los Alamos National Laboratory
Phone: 505-667-7723
Email: ahg@lanl.gov

SECTIONS

Section 1. Introduction

DCON is a code for determining the MHD stability of static axisymmetric toroidal plasma. It uses a very efficient algorithm, originally developed by Newcomb [W. A. Newcomb, Ann. Phys. 10, 232 (1960)] for cylindrical plasmas and generalized by myself to axisymmetric plasmas. A preprint of a manuscript containing the theory of DCON is available in the tex subdirectory of this distribution in both Plain TeX and PostScript formats. The algorithm involves integrating a system of ordinary differential equations (ODE's), the Euler-Lagrange equations for minimizing the potential energy delta_W from the magnetic axis to the plasma edge. A plasma response matrix W_P is constructed from the solutions. If the smallest inverse eigenvalue of W_P changes sign, there is an unstable internal (fixed-boundary) mode. The eigenvalues of W_P are combined with a vacuum response matrix W_V to form a total response matrix W_T = W_P + W_V. If W_T has any negative eigenvalues, there is an unstable external (free-boundary) mode. The name DCON is an acronym for the Direct Criterion of Newcomb. The code makes no effort to determine growth rates of unstable modes or frequencies of the stable spectrum, but only to determine whether the system supports unstable modes. The distance of a zero crossing from the plasma edge can be used to quantify the degree of instability for an internal mode, while the total perturbed potential energy can be used for this purpose for a free-boundary mode.

The present version of DCON evaluates the Mercier criterion D_I > 0 for unstable localized ideal interchange modes; the related criterion D_R > 0 of Glasser, Greene, and Johnson for localized resistive interchange modes; the criterion C_A < 0 for ideal MHD ballooning modes in the limit of large toroidal mode number; the generalized Newcomb criterion for the internal ideal MHD stability of one low toroidal mode number; and the plasma, vacuum, and total response matrices and their eigenvalues for determining free-boundary stability. A future version of DCON will treat resistive instabilities.

In addition to MHD stability, there is a code ORBIT for tracking the full Hamiltonian orbits of charged particles in the same equilibrium fields used by DCON. The full-orbit equations are integrated with a very efficient adaptive ODE solver with high accuracy, both inside and outside the separatrix, and with extensive graphical output. Full orbits and drift orbits have complementary uses. For particles with small Larmor radius, such as thermal ions and electrons, drift orbits are both more efficient and more accurate, especially for long trajectories. For particles with large Larmor radius, such as beam ions, full orbits are more efficient and more accurate. Drift orbits will be added to ORBIT in a later release, allowing the best of both worlds and comparisons between them.

Both DCON and ORBIT are written entirely in high-level Fortran 90. They make extensive use of dynamic allocation and deallocation to allow maximum use of available memory. They use many other advanced features of Fortran 90, such as derived types, array syntax, and modules. There are no fixed dimensions of large arrays, no goto statements, and no common blocks. Extensive use is made of BLAS and LAPACK linear algebra routines for efficiency. There are interfaces to a large number of commonly-used equilibrium data types for specifying equilibria. Adding to these is usually very simple, and I would be glad to help. Graphic output from DCON is viewed with the XDRAW graphics code, which is distributed as part of the package and runs on any X Window system.

This document is a guide to preparing input for DCON and ORBIT and interpreting their output.

Section 2. Straight-Fieldline Coordinates

A major new feature of dcon_3.50 is the use of generalized straight-fieldline (SFL) coordinates. Previous versions have used only one type of these, Hamada coordinates.

For any flux coordinate system, one coordinate, psi, labels the flux surface, and its gradient is orthogonal to the equillibrium magnetic field; the other two, theta and zeta, are angle-like quantities that parameterize the position around the torus the short and long way, respectively. In DCON, psi is linear in the poloidal flux and normalized to go from 0 at the magnetic axis to 1 at the last closed flux surface. Occasionally, rho=sqrt(psi) is used for graphical purposes as a radius-like variable. Theta and zeta are normalized to increase by 1 (not twopi) in going one full turn around the torus.

A flux coordinate system is an SFL coordinate system for which the safety factor

q = (B.del zeta) / (B.del theta)

is a function of psi only. A particular SFL coordinate system is defined by the variation of (B.del theta) (which is proportional to the inverse of the Jacobian of the coordinate system) along the field line. The coordinate systems incorporated into dcon have B.del theta proportional to B_p**power_bp * B**power_b / R**power_r, with B_p the poloidal field strength, B the total field strength, R the major radius, and power_bp, power_b, and power_r input integers. Some commonly-used and named coordinate systems are:

name	power_bp	power_b	power_r
hamada	0	0	0
pest	0	0	2
equal_arc	1	0	0
boozer	0	2	0

The (ignorable) toroidal coordinate is

zeta = phi/twopi + nu(psi,theta),

where nu is a single-valued function of theta. PEST coordinates have the unique property that nu = 0 and the toroidal coordinate is proportional to the conventional polar angle phi. For all other SLF coordinate system, nu /= 0.

PEST coordinates are well-adapted to high-aspect-ratio, circular equilibria. For low-aspect-ratio, strongly-shaped equilibria, detailed numerical comparisons with DCON show that Hamada coordinates generally have the best Fourier convergence properties, and others are worse in proportion to their departure from Hamada. Increasing power_r, as in PEST coordinates, leads to reduced resolution on the outboard side of the torus. Increasing power_b, as in equal_arc coordinates, leads to reduced resolution in the neighborhood of an x-point. The user is advised to use Hamada coordinates, the default. When using others, proceed with caution, increase the number of Fourier components, and monitor the convergence.

Section 3. Installation

The easiest way to unpack dcon_3.50.tar.gz is:

```
gzip -cd dcon_3.50.tar.gz | tar -xvf -
```

It will create a subdirectory ./dcon_3.50 and place its files there.
The contents of the dcon package are as follows:

README:	this file.
equil:	source code for equilibrium data processing
dcon:	source code for DCON, stability code
match:	source code for MATCH, future code for resistive modes
orbit:	source code for ORBIT, full-orbit particle tracking
multi:	source code for MULTI, multiple runs of DCON
sum:	source code for SUM, data extraction from MULTI runs
draw:	xdraw control files draw*.in
equilibria:	sample equilibrium files
input:	dcon control files
lsode:	source code for ODE solver
makefile:	master makefile for all codes on all machines
tex:	preprint on the theory of DCON and equations for ORBIT
vacuum:	source code for Morrell Chance's VACUUM code
xdraw:	source code for XDRAW graphics code

DCON has been set up and tested on the following machines:

SGI Octane running the IRIX 6.4 operating system
Sun SparcStations running the SunOS operating system
DEC Alpha running OSF1 operating system
PC Compatibles running Red Hat Linux 6.0 with Absoft Pro Fortran 6.0
IBM running AIX

If yours is one of these, do "cd dcon" and type make. It will automatically make all its components, create a directory rundir and a subdirectory with the name of your operating system, and copy the executables and a few selected input files to that directory. This is where you should work. You may want to do:

```
nohup make > make.log &
```

to launch a separate process which will keep a log of its actions and any errors.

If your machine is not among these, you should create a makefile in each of the subdirectories with the name makefile_`uname`, where `uname` is the first word of the output when you type the command "uname -a". The makefile should contain definitions of your Fortran 90 compiler and links to existing LAPACK, BLAS, and X11 libraries. Then type "make" while in ./dcon_3.50.

Section 4. EQUIL Input

Both DCON and ORBIT use EQUIL to process the equilibrium data. EQUIL is controlled primarily by equil.in. It contains two Fortran 90 namelists: equil_control and equil_output. The equil_control namelist contains most of the input parameters needed to control the equilibrium operations of the code. The equil_output namelist controls auxiliary output which is mostly used to diagnose problems. For the most part, these may be left false and ignored. In this and other namelist input files, all input parameters have reasonable default values.

Here is an explanation of the input parameters in the first namelist equil_control, along with typical input values and extensive commentary.

eq_type="efit"

This specifies the type of the input file. It allows DCON to determine whether the file is ascii or binary, whether it is direct or inverse, and how to interpret the data. Here is a list of allowed types:

Name	Description	Data Type	Format
"fluxgrid"	NIMROD data file	inverse	ascii
"miller"	Bob Miller's TOQ code	inverse	binary_8
"chease"	Lausanne CHEASE code, 8-byte	inverse	binary_8
"chease4"	Lausanne CHEASE code	inverse	binary_4
"chum"	Modified Miller code	inverse	binary_4
"galkin"	Sergei Galkin's code	inverse	binary_8
"efit"	GA EFIT code	direct	ascii
"rsteq"	ORNL RSTEQ code	direct	binary_8
"ldp_d"	Don Pearlstein's TEQ code	direct	binary_8
"ldp_i"	Don Pearlstein's inverse code	inverse	binary_8
"jsolver"	Steve Jardin's JSOLVER code	inverse	ascii
"lez"	Leonid Zakharov'v code	inverse	binary_4
"transp"	Data from the TRANSP code	inverse	ascii
"sontag"	Aaron Sontag's direct solver	direct	ascii

In addition, you may specify eq_type="lar" or "soloviev" to get the two analytical equilibria. See below for more details.

eq_filename="../equilibria/efit/82205-1.25X"

This gives the path of the data file specifying the equilibrium. The file may be ascii or binary. It may be the output of a direct Grad-Shafranov solver, which specifies poloidal flux, R*B_phi, pressure, and safety factor on a 1D grid of flux surfaces, and the poloidal flux psi on a 2D grid in R and Z; or of an inverse solve, which replaces psi(R,Z) with values of R and Z on a 2D grid of psi and theta on flux surfaces.

The next three input quantities,

```
jac_type="hamada"
power_b=0
power_r=0
```

are used to control the type of straight-fieldline coordinate system used (see discussion above). If jac_type is one of those listed in the table above, the code automatically chooses power_b and power_r accordingly and ignores the input values. To use any other values, choose jac_type="other" and set the integer variables power_b and power_r.

Next come specifications of grid parameters.

grid_type="ldp"	type of radial grid packing
psilow=1e-4	minimum value of psi, normalized from 0 to 1
psihigh=1	maximum value of psi, normalized from 0 to 1
mpsi=128	number of radial grid intervals
mtheta=128	number of poloidal grid intervals

There are several options for radial grid types. The first, grid_type="rho", distributes radial grid points uniformly in rho=sqrt(psi), a generalized radius. The second, grid_type="ldp", uses a method suggested by L. Don Pearlstein (LDP), which packs the grid in the neighborhood of the axis and the edge. I have a slight preference for "ldp", but they both work well. A third type, "original", uses the same radial grid used in the input data. (Sorry, there's no "extra crispy" option.) The poloidal grid is uniformly distributed in the straight-fieldline coordinate theta. Throughout the code, cubic spline and Fourier representations are used to minimize the effects of grid discreteness.

The input parameter psilow is used only for grid_type="ldp". A good value is 1e-4. For the most part, results are insensitive to this value, but substantially smaller values cause the code to run longer, and substantially larger values may cause accuracy problems.

The input parameter psihigh determines how close to the edge the ODE's are integrated. It must be <= 1. For an inverse equilibrium, which never has a separatrix in the computational domain, it may be set to 1. For a direct equilibrium with a separatrix, psihigh determines how close the code approaches the separatrix. Since the safety factor q goes logarithmically to infinity at a separatrix, the required number of poloidal harmonics increases with q (see below), and the run time increases with both the number of harmonics and the number of singular surfaces,it can be expensive to choose psihigh very close to 1.

The next input parameter is

newq0=0

For newq0 = 0, the code uses the original q profile specified in the equilibrium file. For newq0 /= 0, it readjusts the q profile to give the specified value of q at the axis, in such a way as to leave the Grad-Shafranov solution invariant. This can be used to explore the stability of a range of equilibria for each equilibrium file.

The final input parameter in equil_control is

input_only=f

This is occasionally used in the course of debugging interfaces to new equilibrium file types. It causes the code to generate information about the input and then quit.

The equil_output namelist of equil.in determines which auxiliary output files are produced. These are primarily of interest to me, the developer, and of very little interest to users. For most code operation, they should all be left off (f). Turning them on (t) when they are not used wastes cpu time and disk space. The ascii files are readable by any text editor and can also be printed. The binary files are intended for viewing with XDRAW. Here is a brief description of the optional output files.

gse_flag=f	produces diagnostic output for accuracy of solution to Grad-Shafranov equation
out_eq_1d=f	ascii output of 1D equilibrium file data
bin_eq_1d=f	binary output of 1D equilibrium file data
out_eq_2d=f	ascii output of 1D equilibrium file data
bin_eq_2d=f	binary output of 2D equilibrium file data
out_2d=f	ascii output of processed 2D data
bin_2d=f	binary output of processed 2D data

There are two other input files, each containing a single namelist. The first, lar.in, is for large-aspect-ratio, circular cross, low beta equilibrium, using the Shafranov expansion to second order. The second, sol.in, is for the Soloviev equilibrium.

The lar_input namelist of equil.in is used when eq_type="lar" and eq_filename="lar.in". This allows the user to construct a large-aspect-ratio, circular-cross-section equilibrium, using the Shafranov expansion to second order, specifying minor radius a, major radius r0, central pressure ratio beta0, and central safety factor q0. The pressure profile is given by p0*(1-(r/a))**p_pres, and the profile of sigma=J.B/B^2 is given by sigma0*(1-(r/a))**p_pres. Changing these to something more interesting would be very simple. Here are sample input parameters:

ma=64	number of radial grid points
mtau=64	number of azimuthal grid points
lar_r0=10	major radium
lar_a=1	minor radius
beta0=0	value of beta (pressure ratio) on axis
q0=3.2	value of q (safety factor) on axis
p_pres=2	power used in specifying pressure profile
p_sig=2	power used in specifying current profile

The sol_input namelist of equil.in is used when eq_type="soloviev" and eq_filename="sol.in". See Berger et al [D. Berger,L.C. Bernard, R. Gruber,and 6. Troyon, J. Appl. Math. Phys. (ZAMP) 31 (1980) 113] for a published stability analysis of this equilibrium. Here are input parameters:

mr=128	number of radial grid points
mz=128	number of axial grid points
ma=128	number of flux surfaces for surface quantities
e=1	vertical elongation factor
a=1	minor radius
r0=3	major radius
q0=1.26	value of q (safety factor) on axis

Section 5. DCON Input

The stability operations of DCON are controlled by the file dcon.in, containing two namelists, dcon_control and dcon_output. The first begins with a set of flags which determine which features of DCON are exercised:

bal_flag=t	Ideal MHD ballooning criterion for short wavelengths

mat_flag=t	Construct coefficient matrices for diagnostic purposes
ode_flag=t	Integrate ODE's for determining stability of internal long-wavelength mode
vac_flag=t	Compute plasma, vacuum, and total energies for free-boundary modes

Next come values determining mode numbers:

```
nn=1
delta_mlow=4
delta_mhigh=8
delta_mband=0
sing_start=0
```

For each run of the code, there is a single toroidal mode number nn and a range of poloidal mode numbers from mlow to mhigh. The most unstable modes are those which bend the field line lease, for which m-nn*q is as small as possible. The code chooses mlow and mhigh as follows:

```
mlow=MIN(nn*qmin,0._r8)-4-delta_mlow
mhigh=nn*qmax+delta_mhigh
mpert=mhigh-mlow+1
```

where mpert is the total number of perturbed mode numbers. The purpose of delta_mlow and delta_mhigh is to give the user some control over the range of mode numbers. For fixed-boundary modes, I have found empirically that the code works well with delta_mlow = delta_mhigh = 0, and increasing them increases the run time of the code without substantially improving the accuracy (at least for Hamada coordinates). For free-boundary modes, they should be positive, high enough to accommodate the band width of the equilibrium shape. A good way to choose them is to set newq0 = 0 and run the code several times with increasing values of delta_mlow and delta_mhigh to find when convergence is obtained.

In principle, DCON should work for any toroidal mode number nn. In practice, the larger the value of nn, the more singular surfaces there are and the more harmonics it uses, so the longer it runs. Run time increases quite fast with increasing nn. I have mostly used it for nn <= 15. There is an option controlled by the parameter

```
sing_start=0
```

which is intended for operation with higher values of nn. With sing_start=0, integration is initialized near the magnetic axis. For sing_start > 0, it is initialized at singular surface number sing_start. This reduces the work necessary, at the possible expense of accuracy. With sing_start > 0, a different method is used to determine mlow. with mmin the minimum value of nn*q between the starting singular surface and the plasma edge, mlow is chosen as:

```
mlow=mmin-delta_mlow
```

Using sing_start > 0, I have tested DCON up to nn = 15.

DCON is designed to use banded matrices, which couple each poloidal harmonic m to only the nearest harmonics, out to m +/- mband. This was intended to make it efficient to treat nearly cylindrical equilibria. While there is no serious penalty for using banded form, in practice it doesn't help much to reduce the band width mband from full. The limiting factor is that if mband is chosen too small, factorization of a key symmetric-positive-definite matrix F breaks because one or more of its eigenvalues goes negative. The code chooses mband as:

```
mband=mpert-1-MAX(delta_mband,0)
```

using the input variable:

```
delta_mband=0
```

For delta_mband = 0, it uses a full matrix, while for delta_mband > 0 it reduces the band width. The safest choice is delta_mband = 0. If the user chooses a value of delta_mband too large, causing F to become non-positive definite, the code aborts with a message advising the user to reduce delta_mband.

The next two input variables are;

```
mthvac=480
thmax0=1
```

The vacuum energy is computed with the Morrell Chance's VACUUM code. In general, the input to that code is controlled by vac.in. See ./tex/vacuum/vacinput.tex for details. An exception is that one of the parameters in vac.in, mth, is overridden by a variable in dcon.in. This is done because it is needed for dynamic memory allocation, which I added. Mthvac is used by VACUUM as the number of points along the plasma-vacuum interface.

Thmax0 is used for the high-n ideal ballooning stability computation. In principle, this requires integrating a 2nd-order ordinary differential equation in the poloidal angle theta from -infinity to infinity on each flux surface. In practice, it is done from -thmax to +thmax, where thmax is the number of complete cycles around the torus the short way. Thmax is chosen automatically to optimize both accuracy and speed. The value of thmax chosen internally is multiplied by the input variable thmax0 to allow some degree of user control. Its default value is 1, and it is usually best left at 1.

Next come variables controlling numerical accuracy:

tol_nr=1e-5
tol_r=1e-5
crossover=1e-2

Tol_nr and tol_r are relative tolerances used in integrating the main system of Euler-Lagrange equations. The first controls the tolerance far from mode-rational surfaces, while the second controls the tolerance near the rational surfaces. Crossover determines the distance from the singular surface at which control passes from tol_nr to tol_r, expressed in terms of the value of m-nn*q which goes to zero at the rational surface. The ability to impose a tighter tolerance near the singular surface will be of interest primarily for the later implementation of resistive instabilities.

singfac_min=1e-5

This is used to determine the position of closest apprach to each singular surface. The integration terminates when m-nq reaches singfac_min, then restarts on the other side of the singular surface.

ucrit=1e4

This controls a feature of DCON required for preserving accuracy. Starting near the magnetic axis, the code integrates a complex system of ordinary differential equations of order 2*mpert, with mpert = mhigh - mlow + 1. There are mpert regular solutions and mpert singular solutions in the neighborhood of the magnetic axis; DCON follows all of the regular solutions. Each regular solution grows roughly as r^|m| as it comes out of the axis. The high-m solutions therefore grow much more rapidly than the low-m solutions. After a short distance, the large solutions could numerically overwhelm the small solutions, causing the computation to lose all accuracy. To prevent this, the integration is stopped occasionally, and the independent solution vectors are reorganized to eliminate this problem. Ucrit determines how often this is done. It should be left at 1e4 by the casual user.

The final section of input, dcon_output controls stability output options. Many of the optional outputs of interest only to the developer have been removed to avoid confusion. The only remaining ones are:

crit_break=t	determines whether the color of the crit curve (see below) changes after crossing a singular surface
ahb_flag=f	produces output on normal magnetic field eigenvalues and eigenfunctions at plasma-vacuum interface (in honor of

Allen H. Boozer)

mthsurf0	real variable, provides user control over number of boundary points used to display surface eigenfunctions for ahgb_flag=t. Default value 1, number of points increases linearly with mthsurf0

Section 6. VACUUM Input

Computation of perturbed vacuum potential energy is performed with Morrell Chance's VACUUM code. This is controlled by the file vac.in, formerly named modivmc. It is safest to leave the variables in this file unchanged, with one exception. The input variable a allows the use of a conforming conducting wall surrounding the plasma, with a = 0 representing a wall right on the plasma boundary; a <= 10 a wall at a relative distance a from form the plasma boundary, normalized to the plasma radius; and a > 20 a wall at infinity. Beware of very small values of a; the results may be inaccurate because of limited resolution. See M. S. Chance, Phys. Plasma 4, 6, 2161 (1997) for theory and code description. Subdirectory ./tex/vacuum of this directory contains a complete description of the input. Use LaTeX.

Section 7. DCON Output

During the running of DCON, the code writes terse messages to the screen to inform the user of its progress. Here is the output of a typical run:

```
$
$ dcon
 Equilibrium: ../../equilibria/efit/82205-1.25X, Type: efit
 Jac_type = hamada, power_b = 0, power_r = 0
 Diagnosing Grad-Shafranov solution
 Evaluating Mercier criterion
 Evaluating ballooning criterion
 Fourier analysis of metric tensor components
 q0 =  1.087E+00, qmin =  1.082E+00, qmax =  5.426E+00, qa =  6.119E+00
 betat =  2.843E-02, betan =  2.579E+00, betap1 =  1.033E+00
 nn =   1, mlow =  -8, mhigh =  13, mpert =  22, mband =  21
 Computing F, G, and K Matrices
 Starting integration of ODE's
 psi = 1.000E-04, q =  1.087
 psi = 5.845E-01, q =  2.000
 psi = 8.024E-01, q =  3.000
 psi = 9.337E-01, q =  4.000
 psi = 9.813E-01, q =  5.000
 psi = 9.900E-01, q =  5.426
 Computing free boundary energies
 Energies: plasma = -1.773E+00, vacuum =  3.050E+00, total =  1.277E+00
 All modes stable for nn =  1.
 Total cpu time = 2.349E+01 seconds
 PROGRAM_STOP => Normal termination.
$
```

If there are violations of the Newcomb stability criterion, indicating the existence of internal (fixed-boundary) instabilities, the position and q-value of the zero crossing(s) are also reported to the screen as they occur. In that case, the free-boundary energies are not meaningful and are not reported.

The main results of DCON are reported in four files: dcon.out, dcon,bin, crit.out, and crit.bin. Here is a description of their contents.

Dcon.out echoes input parameters; gives various global properties of the equilibrium; lists a table of surfaces quantities, including the ideal and resistive interchange criteria D_I and D_R; lists all singular surfaces for the specified toroidal mode number nn and their properties; and reports zero crossings of the DCON low-n stability criterion. If there are any zero crossings, then the system is unstable to internal, or fixed-boundary, modes, and the computation of free-boundary energies is not meaningful and is skipped. If there are no zero crossings, then dcon.out gives plasma, vacuum, and total energy eigenvalues for free-boundary modes. The lowest total energy reported in the table is the stability criterion for free-boundary modes, also reported to the screen; negative means unstable. Following the table of eigenvalues is a list of the eigenvectors corresponding to each eigenvalue, normalized to unit magnitude. For each eigenvector, the component with the largest absolute value is marked with an asterisk (*). This is useful for identifying the dominant Fourier component of the most unstable mode. It is also useful to note whether the components for mlow and mhigh are sufficiently small, e.g. 1e-3. If not, this is an indication the delta_mlow and/or delta_mhigh should be increased.

Dcon.bin contains data about equilibrium profiles and local stability criteria. It is viewed by doing "xdraw dcon".

Crit.out and crit.bin contain the low-n internal stability results. Do "xdraw crit" to view the graphics file. The main result is in the first frame, showing the stability criterion. If this changes sign, the equilibrium is unstable to an internal ideal mode for the specified toroidal mode number nn. If it is everywhere positive, the system is stable to this mode number. The second frame shows the q profile. The crit curve and the q curve are similarly color coded, changing color as they cross singular surfaces. The color-change can be suppressed by setting crit_break=f in dcon.in, namelist dcon_output.

This is useful for comparing different runs, using a different solid
color for each run. Multiple files can be compared by listing their
paths below the line "filename(s)" in drawcrit.in.

If bin_2d=t in equil.in, then a graphics file 2d.bin is produced
and may be viewed with "xdraw 2d". It produces two windows, one showing
flux surfaces for the equilibrium, the other surfaces of constant theta.
The latter is especially useful for comparing the angular distribution
for different choices of straight-fieldline coordinates.

If gse_flag=t in equil.in, then a graphics file gsei.bin is
prodcued and may be viewed with "xdraw gsei". It shows log_10 of the
Grad-Shafranov error, integrated over flux surfaces. This can be used
to assess the accuracy of the Grad-Shafranov solution. More detailed
local graphs of log10 error can be seen by doing "xdraw gse". A contour
plot can be seen with "xdraw gsec". The latter two require additional
input files, obtainable by "cp ../../draw/drawgse*.in .".

Section 8. ORBIT Input

This section describes input for controlling the ORBIT code for
following guiding-center or full-orbit trajectories of charged
particles, both inside and outside the separatrix.

In order to follow the particle both inside and outside the
separatrix, account must be taken of the fact that the magnetic field
representation is different in these two regions. The Hamiltonian
equations of motion are expressed in terms of the vector potential A,
with the equilibrium magnetic field B = curl A and electric field E =
-grad phi - dA/dt. (The electrostatic potential phi is taken to be
uniform at present; this could easily be changed in the future for any
known configuration of phi.) Inside the separatrix, the flux surfaces
are closed, the vector potential is expressed in terms of the poloidal
coordinate theta, the toroidal field function f = R*B_T is constant on a
flux surface, and it is most convenient to express the equations of
motion in flux coordinates. Outside the separatrix, the flux surfaces
are not closed, the poloidal coordinate theta is not defined, f is
uniform, and it is most convenient to express the equations of motion in
cylindrical coordinates. ORBIT uses flux coordinates inside the
separatrix and cylindrical coordinates outside, and passes the particle
between the two when it crosses the separatrix. Graphs of the orbit
change color to indicate such a crossing.

The equilibrium used by ORBIT is controlled by the same files
equil.in, lar.in, and sol.in, described above in Section 4. For
direct-type equilibria, the orbit is followed both inside and outside
the separatrix, terminating if the orbit crosses a boundary of the
computational domain. For inverse-type equilibria, the orbit is

followed only inside the separatrix, terminating if it crosses the separatrix, this this is the boundary of the compuatational domain for inverse-type data.

The ORBIT code is controlled by orbit.in, containing one namelist, orbit_input. The first two variables:

```
gc_flag=t
orbit_type="inner"
```

determine whether to use guiding-center or full orbits and whether the initial particle position is specified inside or outside the separatrix (see below for more detail).

The next three variables:

```
particle="ion"
ai=2
zi=1
```

determine whether the particle to be followed is an ion or an electron, and if it is an ion, its atomic weight ai and atomic number zi. If the particle is an electon, then ai and zi are ignored.

The next three variables:

```
psi0=.85
theta0=0
zeta0=0
```

specify the particle position in straight-fieldline coordinates (psi,theta,zeta) for orbit_type="inner", with psi linear in the poloidal flux and normalized to go from 0 at the magnetic axis to 1 at the separatrix; and theta and zeta parameterizing position the short and long ways around the torus, increasing by 1 for each full circuit.

The next three variables:

```
r0=.5
z0=0
phi0=0
```

specify the particle position in cylindrical coordinates (r,z,phi) for orbit_type="outer". The initial radial position is set to

```
r=rs2+r0*(rmax-rs2)
```

where rs2 is radius at which the last closed flux surface intersects the horizontal line through the o-point, and rmax is the right edge of the computational domain. The initial axial position is set to

```
z=(zmax+zmin)/2+z0*(zmax-zmin)/2
```

where zmin and zmax are the bottom and top edges of the computational domain. The initial azimuthal position phi0 is specified in radians.

The next three variables:

```
energy0=1e4
alpha0=40
phase0=45
```

specify the initial particle velocity, using spherical coordinates in velocity space. The initial particle energy is specified in eV; alpha0 and phase0 specify the angle of the initial velocity vector from the z-axis and the from the r-axis in the r-phi plane, in degrees.

The next three variables:

```
taumax=10
report=1
stride=5
```

specify the duration of the run and the frequency of output. Tau is time in units of the transit time of the particle with its given initial energy at the o-point, if its velocity were purely toroidal. Taumax specifies the length of the run in these units. Report specifies the interval, in these units, at which the code reports progress to the screen and writes a restart file (see below). Stride specifies the number of internal integrator steps between graphic and tabular output of the orbit. Increasing stride can be used to limit the size of the output files, which decreasing it allows for more detail in the graphs.

The next two variables:

```
tol=1e-9
errmax=1
```

control and monitor numerical error. Tol is the relative tolerance specified to the adaptive ODE solver LSODE. Reducing tol gives tighter tolerance at the expense of computation time. The code monitors the relative change in the particle energy due to accumulated numerical error. If this error exceeds errmax, the code quits.

The next variable:

```
break_flag=f
```

determines whether the graphic output changes color every time tau increments by report. Normally, a color change indicates the crossing of the separatrix. Setting break_flag=t increases the frequency of color change. This can be useful for cross-correlating different orbit segments between graphs.

The next variable:

```
cross=t
```

determines whether the orbit is allowed to cross the separatrix for direct-type equilibrium data. If cross=f, the particle stops at the separatrix. For inversse-type equilibrium data, the orbit can only be followed inside the separatrix.

The next variable:

```
restart_flag=f
```

is used to determine whether to restart the orbit from the last saved restart.out file. The code saves this file each time tau -> tau + report. If restart_flag=t, the code attempts to read this file. If it exists, the code ignores input values for particle type and initial conditions, restarts the trajectory from its last state, and uses taumax to specify an increment on the runtime. If restart_flag=t but restart.out is not found, the code resets restart_flag=f and proceeds as if this were the input value.

The final variables:

```
orbit_out=f
orbit_bin=t
```

determine whether to produce to produce tabular and graphic output for the orbit. Normally the values are set as shown. Setting orbit_out=t a tabular description of the orbit in orbit.out, which is usually not very useful and is time-consuming. Setting orbit_bin=f causes the graphical output to be suppressed.

Section 9. ORBIT Input

This section describes output for the ORBIT code for following full-orbit trajectories of charged particles, both inside and outside the separatrix.

There is an output file orbit.out which starts with the same kind of equilibrium data as dcon.out, described in Section 7 above. Then there are a few lines of information about the orbit that was run. If orbit_out=t, this is followed by a long table of the orbit. This is not recommended: it is not very useful, it wastes time, and the information is better obtained in graphical form.

To view the main graphical output, do "xdraw orbit". This produces 9 graphs:

1. Radial Position, R vs. tau
2. Axial Position, Z vs. tau
3. Orbit in Poloidal Plane, Z vs. R
4. Normalized Poloidal Flux, psi vs. tau
5. Parallel Velocity, v_parallel vs. tau
6. Orbit Viewed Along Axis, R*sin(phi) vs. R*cos(phi)
7. Relative Change in Energy, error vs. tau
8. Relative Change in Magnetic Moment, d_mu/mu_0
9. Larmor Radius / Radius of Curvature, rl/R

The color changes each time the particle crosses the separatrix.

Section 10. Other documentation

There is a subdirectory ./tex containing miscellaneous documentation. Here is a brief description.

1. tex/dcon/manuscript:

An unpublished manuscript on the theory of DCON.

2. tex/dcon/notebook:

Raw equations on the theory of DCON:

dcon.tex	Raw equations on the theory of DCON
bal.tex	High-n ballooning theory
mercier.tex	Expressions for the Mercier criterion
metric.tex	Expressions for the metric tensor

tex/orbit:

orbit.tex	Raw Hamiltonian orbit equations in straight-fieldline coordinates

tex/vacuum:

vacinput.tex	Latex file on preparing input to VACUUM.