# Guiding Center Code Orbit

R.B. White, Jan 2007

The guiding center code Orbit, using guiding center equations derived in *The theory of toroidally confined plasmas*, Imperial College Press 2001, revised second edition 2006, is not restricted to Boozer coordinates. It can use equal arc, PEST, or any other straight field line coordinates $\psi_p, \theta, \zeta$ which along with the parallel velocity and energy completely specify the particle position and velocity. The guiding center equations depend only on the magnitude of the B field, not on components. Orbit uses a 2-D spline representation for B, giving an axisymmetric equilibrium, with capability of adding ripple and MHD modes. The code uses a fourth order Runge Cutta integration routine. It is divided into a main program Orbit.F, which is essentially a heavily commented name list and a set of switches for choosing the type of run, the diagnostics, the data storage, and output. Subroutines are included in initial.f, deposit.f, collisions.f, perturb.f, step.f, record.f, and orbplot.f. The output data is stored as orbit.out, and plot files all of the form file.plt. The Makefile, which will accept compilers pgf90, pathf90, and lf95, deletes all these files so that there are no misleading data lying about from a previous run. So output must be renamed or stored after running if you wish to keep it. **orbit.out contains particle data including velocity, mass, charge, gyro freq, gyro radius, all equilibrium data and all run parameters, including reasons for run abortion.**

There is also a version for following trajectories in stellarator equilibria, Orbit3d which uses a 3d spline.

```
All the codes for Orbit, and Orbit3d are in
 pub/white/Orbit,   pub/white/Orbit3d
 which are all public directories.
        to get file, type
        ftp ftp.pppl.gov
        user: anonymous
        pass:   email address
        cd /pub/white/dir
        get file
```

The steps to performing a simulation are:

- **Read Equilibrium**

- **Load particles, add perturbations and ripple if desired**
- **Advance particles and collect data**
- **Output data**

Since the particles are independent of one another, there is no reason for a parallel version of this code. A poor man's parallelization consists simply of submitting several jobs, identical except for the seed defining the random number generator, producing either a different initial particle distribution or different collisions. Statistics are improved by simply adding the results of the jobs.

To submit a job modify the script batch, modify Orbit.F to choose the run desired, type make, and type qsub -q sque batch

## I. EQUILIBRIUM

The equilibrium is created using mex2eqs, written by Alex Pletzer and maintained by Doug Mccune. Mex2eqs reads either TRANSP or EFIT data, and produces the file map01.cdf, which is read by eqs.f. The program eqs.f reads map01.cdf and produces the spline data needed for Orbit. It can also without map01.cdf produce analytic Shafranov shifted tokamak equilibria. The equilibrium data does not include field ripple. If ripple is desired it must be fit using analytic expressions available in eqs.f. Included are sample equilibrium files.

Data files after each run are read by supermongo files *equilibrium.p*, for the equilibrium. Planes of $P, E$ and $P, \mu$ can be viewed with *mupplane.p* and *peplane.p*. The file *profiles.p* gives radial profiles of $q$, $B$, etc. Particle trajectories can be viewed using *trajxz.p*, *trajt.p*.

## II. MEX2EQS

```
mex2eqs is a 2-d program that produces map01.cdf.
This is the file to feed into eqs to construct the spline data set before
running orbit. Mex2eqs now also computes and saves in the netCDF file
(in addition to flux informations such as q, p, etc... that were
required by eqs):

psixz [Wb/rad]
Bx [T]
Bz [T]
```

```
Bphi [T]
```

as functions of (xcoord, zcoord). The domain is rectangular, with xcoord and zcoord matrices representing the regular coordinates. At present xcoord and zcoord are uniform, though they need not be. The code mex2eqs is part of transp and can be run on any platform by typing /usr/ntcc/bin/mex2eqs (or /usr/ntcc/ffc/bin/mex2eqs on tern)

Here a typical run:

```
[petrel001.pppl.gov|14]
 setenv LD_LIBRARY_PATH /usr/local/lff95/lib:/usr/local/mdsplus/lib
[petrel001.pppl.gov|16] /usr/ntcc/bin/mex2eqs
  MEX2EQS:  retrieve and map equilibrium data for
  postprocessing by EQS, the front end to ORBIT

  Important: EQS is statically linked so in order to
  proceed you need to know the NTHS and NSF parameters
  that have been set in the EQS commonblock o.cln.
  NTHS = max no of poloidal rays + 5 (133)
  NSF  = max no of radial nodes + 1 (194)

Enter NTHS
133, 194
  NTHS * NSF =  133  *  194
  No of poloidal sections =  128
  No of radial nodes      =  193
  Choose your theta variable by specifying the Jacobian dependence
  J ~ X**m/(|grad psi|**n |B|**k). Popular choices are:
  PEST     : m=2, n=0, k=0
  Boozer   : m=0, n=0, k=2
  Equal-arc: m=1, n=1, k=0
 [OLD VALUE:           1]
Enter m
0,0,2
  The equilibrium can be read from various sources and formats
  -1  for CHEASE INP1 format.
   0  for TRANSP data in UFILE format (requires MDSPlus connection)
  +1  for CHEASE inp1.cdf format
  +2  for JSOLVER eqdsk.cdf format
  +3  for EFIT G-EQDSK format
  +4  for EFIT G-EQDSK format, rerunning equilibrium through ESC-Q
  +5  for Menard's psipqgRZ netCDF format
  +6  for Belova's freeqbe netCDF format
Enter number
2
Enter file name or MDSPlus path (use " to prevent uppercase conversion)
"eqdsk.cdf"
  InputFormat =  2  inputFile: eqdsk.cdf
 input file is eqdsk.cdf
 reverse theta orientation
  in call:  eqi_bchk:  BR
  %eqi_alloc:  out of memory
  lfree= 812711  iwant= 264196  neqbuf= 1000000
  %eqi_alloc:  fetching larger buffer...
 *** use direct representation for gpp
  ?f(R,Z) interpolation:  not a function of R,Z: PSI
 --i2mex ERROR-- fraction too high in i2mex_toAxisFraction

 Average Grad-Shafranov error:  -0.20E-01
  --Warning--: large rel GS error > 1%!

   a/R0    Area   Volume   Elong   Beta
   0.772   2.289  10.736   1.947   0.249

  Beta-p  Beta-t  Beta-N    I-MA     li
   0.764   0.249   5.788   2.709   0.360



  Dimensions of X, Z grid.
Enter NX
41
Enter NZ
41
  The size of the computational box encompasses the plasma boundary
  plus some buffer region. Choose the buffer size (dW, dS, dE, dN)
  such that
  XW = (1-dW)*xmin (west boundary of the computational box)
  XE = (1+dE)*xmax (east)
  ZS = (1+dS)*zmin (south)
  ZN = (1+dN)*zmax (north)
  where xmin/xmax/zmin/zmax are min/max values of the plasma boundary.
Enter dW
.1
Enter dE
.1
Enter dS
.1
Enter dN
.1
  %eqm_rzgrid:  creating (R,Z) grid distance map.
  %eqm_rzgrid:  distance map creation completed.
 BLOAT ERROR: BAD BLOATED SURFACE AT XI=  1.0400E+00
 THE JACOBIAN CHANGED SIGNS ON THE SURFACE
 Even on the final iteration (99 loop of BLOATX)
  ... error in BLOATA0!

  (model:  i2mex-EQDSK_netCDF_input_eqdsk) eq_errmsg report:
  %eqm_brz:  user Bvec init failure
 **Error after eqm_brz 1
 The toroidal vacuum magnetic field on magnetic axis Bm =     1.0285
Enter new value for Bm.  Orbit prefers Bm = 1, the value of the field
on axis can be set to any value inside orbit.F.
1.

  Data will now be saved in file map01.cdf
All done.
```

## III. MAIN, ORBIT.F

The main program allows choice of type and length of run, particle distribution, field perturbations, and scattering operator, The selection is made through setting switches in the main.

There are several types of runs available, chosen by **nplot** but if other diagnostics are required the best way to proceed is to start a new value of nplot, nplot = x, and add switches to choose depx in deposit.f to deposit particles, rcrdx in record.f to collect the data you want, and plotx in orbplot.f to produce the plot data files.

nplot = 1 gives single particle orbit,

nplot = 2 local diffusion and loss data, $d\psi^2$ vs t, beginning from initial surface.

nplot = 3 Poincare plot of field,

nplot = 4 distribution modification due to modes

nplot = 5 bootstrap current

nplot = 6 local diffusion determination using steady state tent distribution

nplot = 7 particle flight distributions for chaos analysis

nplot = 8 energy transfer to particles from modes

nplot = 9 improves statistics of a given distribution by time averaging

nplot = 10 kinetic Poincare plots in P, E plane, fixed mu

nplot = 12 phase vector rotation resonance determination

nplot = 14 annealing of stochastic domains

**particle parameters**

nprt = 1000 : number of particles

zprt = 1.D0 : charge in proton units

prot = 1.D0 : mass in proton units

ekev = 20.D0 : energy in KeV

iseed = 0 : seed for random number initiation.

Used for distributions and for scattering operator.

ntor = 6 : simulation run time, in on-axis toroidal transits

bkg = 3.557 : field strength in kGs- the field can be scaled

npert = 1 : field perturbation choice, 0 = none, 1 = analytic, 4 = read data from NOVA

pamp = 1 : potential amplitude, Kev, not used if npert = 0

ndist = 4 : 1=shell distrib, 2=sampledep, read from TRANSP data,
3=poincare, 4=alphas

polo = .5*pw ! initial flux surface for nplot =1, 2, pw = last flux surface

p1 = .6*pw ! minimum surface for Poincare

p2 = .95*pw ! maximum surface for Poincare pchi = .6D0 ! initial pitch for single particle

**dele = 5.D-8 : energy conservation per time step.** Any code modification or inclusion of perturbations should be checked that energy is conserved. The time step is adjusted to keep the energy conservation dE/E less than dele. Note that all particles have their own clocks, they are not at the same time. If time synchronization is desired you must use $dele > 1$ which forces all time steps to be dt0, set in initial.f. The default value is 200 steps per toroidal transit time, and this is sufficient unless you insert perturbations with large n values. One should check a single particle run with dele = 5e-8 to see what time step gives acceptable energy conservation, then set dt0 accordingly. If time dependent perturbations are used energy is no longer conserved, so dele larger than one is necessary. Energy conservation should be controlled using zero frequency.

if(nplot.eq.4) dele = 5. dele larger than 1 gives fixed time step, needed for instantaneous distribution determination.

**Collisions**

ncol = 0 ! no collisions, ncol=1, energy dependence only,

ncol = 2 full profiles

col = 1.D0/(50*tran) : pitch angle scattering frequency, tran is one on-axis transit time.

drag = 0.D0/(200*tran) : slowing down frequency

**For ncol = 2 see scatr in collisions.f and functions denb, deni, tempi, tempe**

massb = 2 ! background plasma mass

chgb = 1 ! background plasma charge

imp = 0 ! imp=1 impurity species, 0=none

massi = 5 ! impurity mass

chgi = 5 ! impurity charge

eion = 1.1 ! ion energy in kev for plots

**Perturbations** Perturbations have the form

$\delta B = \delta \times \alpha \vec{B}$, with $\alpha = \sum_{m,n} \alpha_{mn}(\psi) sin(n\zeta - m\theta - \int \omega(t)dt)$

where $\omega$ is assumed to be slowly varying so treated adiabatically. Each time step the value of the field perturbations are calculated in subroutine *ptrb*1 in perturb.f. Note that for many applications an ideal MHD perturbation must be used, meaning that there is no parallel electric field. Since the perturbation above gives such a field, a potential must be added to cancel it. See my book. This perturbation can be switched on or off, see this subroutine to determine the state of this switch.

Perturbations can be constructed analytically, see the routine *amp*1 in perturb.f, and the spline routine *spln* called by it. They can also be read from the code NOVA, see the routine *readptrb* in perturb.f.

The radial form of the harmonics can be viewed using the supermongo file harmonics.p

### A. Single particle orbit, nplot=1

Setting nplot=1 automatically limits the run to include a single particle, launched at surface polo with pitch = pchi, and calls the diagnostic routine rcrd1. Data is recorded at time intervals of dt1, set in initial.f with default of .01*tran for nplot=1. Data recorded is time, energy change, time step, $\psi_p, \zeta, \theta, \lambda$, B, q. This can easily be augmented or changed. See rcrd1 in record.f. At the end of the run this data is written to files traj1.plt and traj2.plt for postprocessing. The supermongo file trajxz.p shows the poloidal projection of the orbit and trajt.p will give time history of a selected variable.

### B. Local diffusion and particle loss, nplot=2

For nplot=2 monoenergetic particles are initially uniformly distributed on a single flux surface. The routine rcrd2 does a running sum of the mean square displacement from this surface vs time, the mean pitch distribution and the mean energy change. These can easily be changed or augmented. Data is recorded at intervals of nskip steps, set in set1, initial.f, with default to produce 200 plot points in the course of the run. At the end of the run the routine plot2 in orbplot.f writes the plot data set diffusion.plt. The lost particle data is stored in lost.plt, so distributions of lost particles can be plotted using the general supermongo file histogram.p.

### C. Poincare plot, nplot=3

For a Poincare plot the particles are automatically set to have very low energy and pitch of 1, so that they simply follow field lines. They are deposited uniformly with $\psi_p$ between p1 and p2, set in the main. Data is collected in rcrd3 in record.f every time a particle crosses $\zeta = 0$. At the end of the run plot data is written by plot3 in orbplot.f in the file poincare.plt and can be plotted using the supermongo file poin.p.

### D. Modification of distribution by perturbations, nplot=4

To find the modification of a particle distribution by a mode such as a TAE mode, excellent statistics can be generated by running the initial distribution for some time in the presence of the mode and then collecting distribution data every time step for the remainder of the run. This produces time average data for the modified distribution function. The default is that the data is collected for some last fraction of the run time, see the call to rcrd4 in orbit.F.

### E. Local bootstrap current, nplot=5

This is a $\delta f$ calculation of local bootstrap current. See chapter 8 of my book and publication Wu and White (1993) Wu in the attached list. The current is normalized to the theoretical value for small collision frequency in a circular equilibrium and can be plotted vs time using the supermongo file boot.p.

### F. Steady state Diffusion, nplot=6

This routine constructs a local tent distribution, particles exiting at the edges being replaced at the center, which approaches steady state in time, the local diffusion given by the gradient and the flow rate and can be plotted using the supermongo file tent.p.

### G. Particle flight distributions, nplot=7

This routine records flight distances, a flight defined as the period in which the pitch has one sign. See publication with Spizzo in Phys Plasmas 2007, and PPCF in 2009. Distributions can be plotted using the supermongo file flight.p.

### H. Kinetic Poincare plot, nplot=10

Kinetic Poincare plot to find resonances of particles with a time dependent mode. Particles are deposited uniformly in a range of $\psi_p$ determined in *poinkdep*, in deposit.f. Data is collected in rcrd10 in record.f every time a particle crosses $n\zeta - \omega t = 0$. The data is written in the file poink.plt and can be plotted using the supermongo file poinkin.p.

### I. Phase vector rotation for resonance determination, nplot=12

Resonances of particles with a time dependent mode. Particles are deposited uniformly in the E, P plane for fixed $\mu$ with wdep2 in deposit.f. Data is collected in rcrdrot in record.f. The data is written in the file worm.out and slope.out and can be plotted using the supermongo file peplane.p. See papers on Modification of particle distributions by MHD instabilities, R. B. White 2011, listed in attached file.

### J. Annealing, nplot=14

An initial particle distribution is modified by successive annealing due to results of phase vector rotation produced by nplot=12. The routine is anneal2d in chaos.f. The files read are worm.2.sv and slope.2.sv produced by stoch2d, called by nplot=12. The .2 refers to the value of $\mu$ in the $E, P_\zeta$ plane, set for the nplot=12 calculation. Initial and final distributions are written in dist0.plt and distf.plt and can be plotted using the supermongo file histogram2.p. See papers on Modification of particle distributions by MHD instabilities, R. B. White 2011, listed in attached file.

## IV. SET UP, INITIAL.F

These routines calculate initial constants etc for run. Normally nothing has to be changed here

## V. LOAD PARTICLES, DEPOSIT.F

Chosen with ndist, The distributions available in deposit.f are a monoenergetic distribution distributed evenly on one flux surface polo, given by shelldep, a distribution evenly distributed between two bounding surfaces p1 and p2, poindep, the possibility of reading a particle distribution directly from TRANSP, sampledep, and a monoenergetic alpha particle distribution with a choice of radial profiles.

Any distribution in space, energy, or pitch can be constructed using a simple Monte Carlo procedure.

For example to make a distribution of particles in $\psi$ of the form $dn/d\psi = f(\psi)$ simply write

$$\frac{dn}{d\psi} = \frac{dn}{dx}\frac{dx}{d\psi} = f(\psi) \tag{1}$$

and choose $x = ranx()$ giving a uniform distribution of particles in $x$, ie $dn/dx = constant$. Integrate to find $x = \int f(\psi)d\psi$. Invert this to find $\psi(x)$, numerically if necessary. Then this routine will deposit particles with the required distribution. Integration constants are chosen to adjust the range of the deposition.

```
    do 10 k = 1,nprt
5           continue
```

```
        x = ranx()
        ptry = psi(x)
        prob = f(ptry) ! must be normalized to be less than one
        dum = ranx()
        if(prob.lt.dum) go to 5
        psi(k) = ptry
10          continue
```

Similary, distributions can be constructed that are functions of all variables energy, pitch, and position, Since the deposition is done only once before the run begins this is not expensive.

In functions.f there is a function maxwell(tdum), which produces one random value of energy from a Maxwellian distrubution of 'temperature' tdum in KeV. Thus a routine setting particle energy through en(k) = maxwell(tdum) will give a Maxwellian distribution in energy. Attention, if an energy distribution is used the time step must accomodate the fastest particles.

Distributions can be plotted using the supermongo file histogram.p.

## VI. TRANSP PRODUCED BEAM DISTRIBUTIONS

To use beam particle distributions existing in the device ndist = 2 will read beam particle distributions produced by TRANSP. The subroutine is SAMPLEDEP. A sample beam distribution is supplied,

```
fbm_dist.dat
```

It is an ASCI file with simple format. There is also a script file getfh4orbit.scr to produce the beam data from TRANSP.

## VII. PERTURBATION, PERTURB.F

For orbit the perturbations must be of the form $\delta B = \nabla \times \alpha(\psi, \theta, \zeta)B$, normally using harmonics of the form $\alpha = sin(n\zeta - m\theta - \omega t)$. This form produces exactly the component perpendicular to the equilibrium flux surface, responsible for magnetic islands, and is automatically divergence free. It is a good representation for all low beta MHD perturbations. In perturb.f there are the routines called every time step to add the field perturbation to the field, as well as means to construct analytic perturbations of MHD or resistive type. See spln in perturb.f. The routine readptrb will read a perturbation amplitude from an external data set ptrb.dat. A sample 2-harmonic TAE mode is supplied with the data set ptrb.dat, which is a simple asci file. to see the perturbations use the data harmonics.plt, and the supermongo file harmonics.p

## VIII. COLLISIONS, COLLISIONS.F

ncol = 1 gives a simple pitch angle scattering operator

col = 1.D0/(50*tran)

gives the collision frequency as 1/50 of a transit time. Slowing down is given in the same manner.

drag = 1.D0/(200*tran)

using ncol = 2 a much more elaborate collision operator is given, using the Rosenbluth psi function, and profiles for density and temperature, which must be supplied. These routines are all in collisions.f

## IX. DIAGNOSTICS, RECORD.F

rcrd0 - records initial particle data
rcrd1 - records single particle data at intervals of dt1
rcrd2 - records the mean square displacement from the flux surface as well
as the mean pitch and energy change.
rcrd3 - records data for a Poincare plot at crossings of $\zeta = 0$
rcrd4 - performs a time average of the distribution,
starting at a specified time late in the run.
rcrd5 - Bootstrap current data

These routines can be easily modified. To add a new diagnostic, simply copy one of these, call it rcrdx, make the necessary modifications, and introduce nplot=x in the same manner.

## X. OUTPUT, ORBPLOT.F

plot1 - writes single particle data at intervals of dt1 in traj1.plt and traj2.plt
plot2 - writes the mean square displacement from the flux surface as well as the mean pitch and energy change in diffusion.plt
plot3 - writes poincare.plt, the Poincare data
plot4 - writes the time average of the distribution, starting at a specified time into the run in distave.plt.
plot5 - writes the Bootstrap current vs time, boot.plt.

These routines can be easily modified. To add a new diagnostic, simply copy one of these, call it plotx, and introduce nplot=x.

## XI. POSTPROCESSING WITH SUPERMONGO

Super mongo was written by Robert Lupton at Princeton. It is a simple plotting routine that produces postscript files and is written in Pascal. There are copies of routines to handle all of the file.plt output files produced by Orbit available with the code. Those enamored of IDL can read the same output files and use IDL to obtain plots.

An advantage of Super Mongo is that it supports Latex, so that plots are easily produced for publication. All the supermongo files are of the type file.p. Super Mongo is supported at PPPL on the unix devices. Simply type sm, after which one types **input file.p**, and the plot appears. The pound sign at the beginning of a line makes that line inoperative. Thus the first line

device postencap :SY@: :OF@: new.ps

turns the output into the postscript file new.ps, but if preceded by a pound sign the output only goes to the screen for checking. The normal procedure is to play with the plot until it is acceptable, then make this first line operative, and rename new.ps as desired.

**supermongo files available are**

| | |
|---|---|
| diff.p | for diffusion $< d\psi_p^2 >$ vs time |
| dist.p | initial, final or mean particle distributions in space |
| lost.p | particle loss vs time |
| harmonics.p | harmonics of perturbation |
| poin.p | Poincare plot |
| trajxz.p | particle trajectory data in poloidal section |
| trajtop.p | particle trajectory data top view |
| trajt.p | particle trajectory data vs time |
| distave.p | mean distribution |
| equilibrium.p | equilibrium shape |
| profiles.p | equilibrium profiles |
| histogram.p | binned particle distributions in $\psi_p, \theta$, etc |
| scatr.p | profiles for advanced collision operator |
| poinkin.p | Kinetic Poincare plot of resonances in plane of P,$\theta$ or $\psi_p$, $\theta$. |
| boot.p | bootstrap current vs time |
| peplane.p | Plane of P,E showing passing and trapped particle domains |
| muplane.p | Plane of P,$\mu$ showing passing and trapped particle domains |

These files can be easily modified to plot other features.