Creating ASCOT input files for orbit-plotting

S. Scott (CFS)

July 3, 2023

this file: ascot\_sims\_for\_orbit\_plotting\_01.docx

Sometimes we want to ask ASCOT to simulate just a few alpha orbits so we can visualize them.

There are two python scripts to plot orbits in the ‘myprompts’ directory:

* plot\_orbits\_2023a.py; and
* compare\_orbits\_2023a.py.

The latter superimposes the orbits from two simulations.

A major limitation for these simulations is memory. In the now-obsolete knl cluster at NERSC, the product of the number of markers times the number of orbit points saved per marker had to less than 99,000,000 otherwise the run would run of of memory and die with a ‘segmentation fault’. I don’t know what the corresponding limit is on the Perlmutter cluster, but I would expect it to be about the same within a factor of several.

A major design feature is that ASCOT records the orbit position (and velocity) at user-defined time intervals (variable = my\_orbitwrite\_interval in my Python utilities, see below). And, importantly, the user must specify the maximum number of orbit positions to record (variable = my\_orbitwrite\_npoint in my Python utilities). So only the last tmax = my\_orbitwrite\_interval \* my\_orbitwrite\_npoint seconds of the orbit will be saved. If tmax is shorter than the maximum user-defined simulation time my\_max\_simtime, then the saved orbit positions will not extend back to the marker-birth time.

Generally, when I plot orbits I do want to see the entire orbit, extending back to its birth location, so I choose the values of the variables in magenta above to ensure that tmax is bigger than my\_max\_simtime.

But there may be times when you only want to see the orbit position near its end and you don’t care about its initial orbit. For example, you might want to save the last few thousand orbit positions for a larger number of markers, so you can verify that markers that are flagged as being lost are in fact crossing the LCFS or hitting the wall.

**Marker ensembles**

marker\_sets.py, module define\_prt\_markers\_13 allows the user to create an ensemble of markers on a fixed grid in space and in pitch. The ensemble of markers does NOT mimic a true alpha source profile. The purpose of this module is to create a (usually-small) ensemble of markers for the sole purpose of computing orbits so we can plot them with plot\_orbits\_2023a.py

The ensemble of markers is constructed on a fixed, uniformily-spaced grid in: toroidal angle, Rmajor, Z, and pitch. The user specifies the numbe of grid positions for each coordinate, along with the minimum and maximum values. If the number of grid points is set to 1, then the mimum value is used.

So with this module, you can create an ensemble of markers that is effectively 1D, 2D, 3D, or 4D.

I think the ‘rhomin’ and ‘rhomax’ are defined so that markers that might be created outside the plasma (i.e.beyond the LCFS, or rho=1) will be excluded from the ensemble.

Example below is taken from group\_go\_2367.py. The variables should be self-explanatory with the exception of ‘nloops’. Each starting marker is reproduced ‘nloops’ times, so typically nloops =1.

But you could, for example, set nr = nz = nphi = npitch=1, but say nloops=100, and that would create 100 identical markers. If you turned Coulomb collisions on, you could then examine how much variability there is in the orbits due to the randomness of the Coulomb collisions.

settings={} # new, ab-initio set of marker settings

settings["eq\_index"] = 0

settings["phi\_min"] = 0.

settings["phi\_max"] = 0.

settings["pitch\_min"] = 0.18

settings["pitch\_max"] = 0.36

settings["rmin"] = 1.84

settings["rmax"] = 2.05

settings["zmin"] = 0.25

settings["zmax"] = 0.40

settings["rhomax"] = 0.995

settings["rhomin"] = 0.

settings["nphi"] = 1

settings["nr"] = 8

settings["nz"] = 6

settings["nloops"] = 1

settings["npitch"] = 7

marker\_sets.define\_prt\_markers\_13(fn\_hdf5, fn\_geqdsk, set, settings, fn\_parent, desc=None)

**Options settings** for ASCOT simulations of orbits

The important settings are:

* maximum simulation time (typically short, see discussion below)
* Time step between time points at which the orbit data is recorded
* Maximum number of orbit positions to store for each marker
* Whether the simulation is collisionless or collisional
* Whether the full gyro-orbit is (GO) simulated, or just ‘guiding center’ (we typically use the full gyro-orbit).
* If the simulation is GO, do we record the actual location of the marker on its gyro-orbit, or the corresponding ‘guiding center’ location?

The following was extracted from group\_go\_2367.py

set = 17

options\_settings = {}

options\_settings["my\_sim\_mode"] = 1 # 1 = full gyro-orbit. 2 = guiding-center

options\_settings["my\_max\_simtime"] = 0.0010

options\_settings["my\_max\_cputime"] = 30000. # effectively infinity

options\_settings["my\_orbitwrite\_npoint"] = 350000 # number of orbit-points to save

# in this case only the last 350,000 points

# will be saved

options\_settings["my\_orbitwrite\_interval"] = 3.e-9 # orbit points saved every 3 nanoseconds

options\_settings["my\_no\_orbitwrite"] = 0 # this effectively turns the orbit-write “on”

options\_settings["my\_go\_record\_mode"] = 1 # 1 = save full GO position. 0 = save GC position

* options\_settings["no\_coulomb"] = 1 # collisionless orbit simulation
* options\_settings["my\_disable\_pitch\_ccoll"] = 1 # collisionless orbit simulation
* options\_settings["my\_disable\_energy\_ccoll"] = 1 # collisionless orbit simulation
* options\_settings["my\_enable\_mhd"] = 0 # 1 to enable MHD, 0 to turn it off

aa\_options = options\_sets.options\_sets(fn\_hdf5, set, options\_settings, desc=my\_description)

Facts to consider when choosing times and number of orbit points to save:

* alpha speed at birth = 1.3e7 m/s.
* The banana bounce time is of order 10 microsec.
* The alpha gyroradius at birth is about 2.5 cm, orbit circumference = 16 cm
* So the time to complete one gyro-orbit is about 0.16/1.3e7 = 1.2e-8 sec.
* The time step between orbit-position-saves will depend sensitively on whether you want to see the full gyro-orbit, or just the guiding-center.
* If you want to see the full gyro-orbit, then maybe you want to see 10 points per gyro-orbit, in which case the time step might be of order 1.2e-9 sec.
* If you just want to see the guiding center, assuming a pitch (v\_parallel/v) =0.5, and that you want spatial steps of 10 cm, then the time step might be 0.1 / (0.5\*1.3e7) = 1.5e-8 sec.