Common Problems and their solutions

This section deals with problems that have arisen and their solutions

**unyt**

When I invoke the executable Python script a5combine, I get an error message that the package ‘unyt’ is not installed. The underlying problem is that the script invokes Python 3.6 whereas my default Python version, as defined in a ‘module load’ command in my .bashrc.ext file, is version 3.7. The package unyt is indeed installed in version 3.7 but not in 3.6

The solution is to force the system to use my version: **python3 a5combine**

This solution described by Pablo in a ZOOM on 12/7/2020.

**Segmentation fault**

Typically caused by running out of memory. I think the big consumers of memory are typically (1) recording orbits; and (2) recording the 5D and 6D distribution functions.

**Problem with pitch close to -1 or +1**

I forget the details, but I had some problems when markers were created with v\_parallel/v very close to -1 or close to +1. The solution is to limit the allowable v\_parallel/v to the range [-0.999, 0.999] in marker\_sets.py.

**Div-B not sufficiently close to zero**

The 3D magnetic field is provided to ASCOT on a fixed spatial grid. ASCOT then interpolates using splines to compute the field at the marker’s actual location. Unfortunately, the use of splines for interpolation does not ensure (by construction) that the computed magnetic field satisfies div-B = 0. So it is important to provide the 3D magnetic field on a fine spatial grid.

Also, early on I used the IDL parameter !radeg to convert from degrees to radians and vice versa. It turns out that this variable is only single-precision. Things got a lot better when I used a value for pi that is double precision.

[back to top](#back_to_top)

[What’s the work?](#what_is_work)

In this section I describe where ‘the work is’, i.e. what are the major challenges and time-sinks for the ASCOT simulations.

1. Compiling the code: I’m not conversant with build scripts (makefiles) at NERSC. It took Libby Tolman some time with help from NERSC to get the code built the first time, and it seems to take me a while to recompile every time ASCOT releases a new version. But I did write down a set of instructions on how to compile … hope they are complete.
2. Compile switches and run-time switches: these are very mysterious to me, but important because they can have a big effect on CPU speed. John Wright and the experts at NERSC are good resources.
3. Validating that the simulations are ‘correct’: this is an ongoing effort and requires a considerable amount of Python coding, e.g. to generate new plots that address particular concerns.
4. A sub-component of the previous one: choosing the largest time step that still yields accurate orbit calculations.
5. Implementing 3D wall/limiter shapes: ASCOT needs a wall shape that is defined by a large number of triangles. I have Python code to do this (triangulate\_torus.py) but it is written for a particular ‘family’ of limiter shapes. It might be nice to develop a workflow that translates a SPARC CAD model for the PFCs into the format required by ASCOT. But this is not as easy as it sounds, because (1) if the size of individual area elements (aka triangles) is too big then the wall shape is not a good representation of the actual wall shape, leading to over-estimate of the ripple losses, and (2) if the size of the triangles is too small then we get poor statistics on the computed surface power density.
6. I have done nothing yet on computing the alpha positional and energy / pitch-angle distribution function nor on a workflow that would allow it to be used in e.g. MHD stability codes.
7. I have only highly approximate calculations of the loss of RF tail ions. As John Wright has been saying for some time, we need a way to allow the fast ions to be given a `kick’ by the RF wavefield as the ions pass thru the resonance.