ASCOT handover materials

S. D. Scott

CFS

April 6, 2022 (v21)

This file: ASCOT\_handover\_021.docx

Conventions in this file:

* Text that the user would type in a Linux terminal is displayed in green text.
* Aliases are displayed in cyan font.
* Buttons that a user would click in an application (e.g. Blender, ParaView) are displayed in purple.
* Links to bookmarks in this document are in blue font.

Links

|  |  |  |
| --- | --- | --- |
| [Introduction](#introduction) | [NERSC allotment](#allotments) | [references](#references) |
| [Preparing ASCOT input file](#preparing_input_file) | [Submitting ASCOT runs](#submitting_run) | [Managing runs](#managing_run) |
| [Bookkeeping](#bookkeeping) | [Data archival](#archival) | [Parent/daughter runs](#parent_daugher) |
| [Directories & compilation](#directory_structure) | [Making Ripple file](#generating_ripple_file) | [Making profile file](#generating_profiles) |
| [Making wall file](#generating_wall) | [Options parameters](#options) | [Structure input files](#script_structure) |
| [ASCOT5 utilities](#software_utilities) | [Structure hdf5 file](#structure_hdf5) | [Postprocessing](#postprocessing) |
| [Comparing 2 runs](#comparing_two) | [Plotting orbits](#plotting_orbits) | [Rendering 3D output](#vtp) |
| [Orbit end conditions](#end_conditions) | [Aliases](#aliases) | [Problems + solutions](#common_problems) |
| [What’s the work?](#what_is_work) | [Reports & presentations](#reports_presentations) | [Cookbook\_01](file:///C:\Users\sscott\Documents\ripple\ASCOT_handover\cookbook_recipe_01.docx) |
| Annotated scripts | [Plasma scenarios](#plasma_scenarios) | [Engineering PFCs](ascot_engineering_PFC_01.docx) |
|  |  |  |

**Introduction**

The ASCOT5 code computes the orbits of energetic ions in magnetic fields, including tokamaks. The code is freely available at

<https://wiki.aalto.fi/pages/viewpage.action?spaceKey=ASCOTCo&title=Guide+to+ASCOT5>

Users need to register to download ASCOT5. We run ASCOT5 at the NERSC supercomputer center in California, nersc.gov. John Wright manages our relationship with NERSC and he is a good source of information about NERSC and running jobs at NERSC. Nathan Howard also has extensive experience running big jobs at NERSC. Pablo Fernandez-Rodriguez is a great resource for all things Python and he has some experience helping me out with ASCOT.

Users can manage their NERSC accounts, e.g. learn how much CPU time they have consumed, at the website <https://iris.nersc.gov/login>. when logging onto nersc.gov, users need to append a one-time passcode (that in my case is generated by an app on my cell phone) to the fixed password, but for reasons unknown to me, when logging onto iris, you give just the password, then you will be prompted for the one-time passcord.

In 2019, ASCOT held a user-training program in Finland. Unfortunately, the ASCOT program is funded by the EU and so EU applicants are given priority, and I was not selected to attend. I would recommend that ASCOT users attend this training if possible if it is offered again in the figure. ASCOT does film the training, but I found the audio quality quite poor.

There is another orbit-following code named SPIRAL maintained at PPPL by Gerrit Kramer (gkramer@pppl.gov). We have done extensive benchmarking between SPIRAL and ASCOT, and the codes now agree – the benchmarking process did reveal some problems in both the SPIRAL and ASCOT pre-processors. Gerrit is a good resource to discuss orbit-simulation issues if you can’t find me.

Slack channel: There is a Slack channel for ASCOT (<https://app.slack.com/client/TQ9MKFHR8/CQB1GDFT5> and <https://app.slack.com/client/TQ9MKFHR8/CQB1GDUP5/thread/CQB1GDUP5-1600622309.027900> ) are my links, but I’m not sure whether that link will work for everyone)

Documentation: when you download and install ASCOT5, it generates a set of documentation. My downloaded version of the documentation is available [here](file:///C:\Users\sscott\Documents\ripple\ASCOT_handover\MyAscotHelp\index.html).

There is human help available at <https://version.aalto.fi/gitlab/groups/ascot/-/issues>. Keep in mind that the ASCOT development team is in Finland, which is about 7 hours ahead of EST time.

Weekly meetings: The ASCOT team holds more-or-less weekly meetings, currently on Mondays. If the CFS ASCOT user is willing to get up at typically 4:30 AM, it might be useful to attend these meetings.

[back\_to\_top](#back_to_top)

**NERSC allotments and job queues**

NERSC CPU time is allocated on a yearly basis, termed an ‘allotment’. The owner of the ASCOT work for CFS needs to justify the requested allotment, and John Wright collects all of the proposals and submits it to NERSC. This process typically happens in the Fall, and the allotment decisions are announced in early January; the new allotment-year typically starts circa mid-January. The file [nersc-allotment-2021-sds.docx](file:///C:\Users\sscott\Documents\ripple\ASCOT_handover\nersc-allotment-2021-sds.docx) is my justification for our 2021 allotment.

Note that the job control system at NERSC allows you to define a priority for your runs. Jobs in the ‘regular’ priority queue get charged at the usual rates (described on the NERSC website). ‘Premium’ priority jobs will get processed faster, i.e. they will wait less time in the queue before starting, but the charge for the CPU time is twice that for regular-priority jobs. There are also a number of low-priority queues which wait even longer to get started; I have not used these low-priority queues.

Keep in mind that NERSC keeps track of your ‘average’ queue priority (with 1.0 being regular and 2.0 being premium). They expect use of the premium queue to be limited to rush jobs, e.g. just before a conference; I think they frown on an average queue priority larger than 1.1. The way NERSC expresses its displeasure with high average queue priorities is apparently to give you less time in the next year’s allocation.

[back\_to\_top](#back_to_top)

**ASCOT References**:

The best reference for the ASCOT5 code is VARJE, J., et al., “High-performance orbit-following code ASCOT5 for monte carlo simulations in fusion plasmas”, arXiv:1908.02482v1 (2019).

Some applications of the ASCOT4 and ASCOT5 codes are presented in:

1. S. Scott et al., “Fast-ion physics in SPARC”, JPP 865860508 (2020).
2. Kurki-Suonio, “Effect of the European design of TBMs on ITER wall loads due to fast ions in the baseline, hybrid, steady-state, and half-field scenarios”, Nuclear Fusion **56** (2016) 112024.
3. A. Snicker et al., “Power loads to ITER first wall structures due to fusion alphas in a non-axisymmetric magnetic field including the presence of MHD modes”, Nuclear Fusion 53 (2013) 093028.
4. T. Kurki-Suonio et al., “Fast ion power loads on ITER first wall structures in the presence of NTMs and microturbulence”, Nuclear Fusion 51 (2011) 083041.
5. T. Kurki-Suonio et al., “ASCOT simulations of fast ion power loads to the plasma-facing components in ITER”, Nuclear Fusion 49 (2009) 095001.

[back\_to\_top](#back_to_top)

**Preparing an ASCOT input file**

ASCOT5 input files use the hdf5 file format. I generate input files from Python scripts, e.g.

cd /project/projectdirs/m3195/ascot/ascot5/runs (or issue the ‘toruns’ alias)

python group\_go\_703.py init

where group\_go\_703.py is the Python script, and ‘init’ tells the Python script to generate an input file. Generally, I copy an existing Python script into a new filename, then make some changes to the input file, and then execute the command above to generate the input file, which in this case would be group\_go\_703.h5. So the full sequence is:

cp group\_go\_703.py group\_go\_725.py

<edit the file group\_go\_725.py>

python group\_go\_725.py init 🡪 generates group\_go\_725.h5

**IMPORTANT!** Toward the end of the .py file there is a line that defines the name of the output file, which in this case would be ‘group\_go\_703.h5’. So you **MUST** change this string to properly correspond to the new simulation number … otherwise you will overwrite the .h5 file of the parent simulation.

**RECOMMENDATION:** Before executing the new python script above to generate the new .h5 input file, I typically do a “diff” on the parent and daughter python scripts to make sure that I changed what I thought I changed, no more and no less. I do this because, more than once, I have made some error when editing the new Python script, leading to erroneous output, and it is tedious and time-consuming to determine why the output is wrong.

**IMPORTANT!**  My python codes that generate the .h5 ASCOT input file call ASCOT-provided routines that actually write the various parameters into the .h5 file. For reasons that I don’t fully understand, the ASCOT development team allows multiple sets of input data (and output data) to be contained in a single .h5 file. I don’t like this approach, so I just have a one-to-one relationship between input files and simulations. Unfortunately, the ASCOT utilities are written so that if you ask to write a set of input data to an **existing** .h5 file, they will **not** erase the existing input data, but simply create a second (or third, …) set of input data into the .h5 file.

This can obviously lead to confusion. So suppose you create an .h5 file, run it, and find that there was an error. You go back to the python script that generated the .h5 file, e.g. group\_go\_725.py and make some change. You **MUST** then delete the file group\_go\_725.h5 before re-generating it, otherwise you will get multiple sets of input data in your .h5 file.

[back\_to\_top](#back_to_top)

**Submitting the ASCOT run**

Bash shell scripts are used to submit the batch job at NERSC. So I have a shell script for each ASCOT simulation, and when I copy the python script that generates the input file I also copy the corresponding batch shell script.

cp knl\_703.sh knl\_725.sh

<edit the file knl\_725.sh>

sbatch knl\_725.sh 🡨 this formally submits the batch job

Once submitted, the job will be assigned an 8-digit identifier, e.g. 36254397. The output files will be stored in a directory e.g. ascot/ascot\_run\_output/ascot\_work\_36254397 and the output file will be ascot\_36254397.h5

So note that there are **two** .h5 files associated with a run. The first is generated by the python preprocessor script e.g. group\_go\_725.py and would be called group\_go\_725.h5 and typically lives in the directory e.g. …/ascot/ascot5/runs/. The second is generated by ASCOT, contains both the input and output data, and lives the directory …/ascot\_run\_output/ascot\_36254397.h5.

The reason why I copy the batch script of the parent run into the batch script for the new run is that (1) one can determine how much time it took the parent run to complete using the mycputime alias; and (2) based on changes of input parameters for the new run, it is typically straightforward to estimate how long the new run will take, and one of the parameters specified in the batch script is how much time to allow for the new run.

**IMPORTANT!** … ASCOT does not support the capability to store intermediate results so that a job that does not complete in the allocated time can be restarted. So if you allocate 10 hours for a run but the run actually needs 10.5 hours, you will run out of time and the run will be lost. But you will still be charged for the 10 hours.

The only downside to just specifying a longer run time than you think the run actually needs is that longer jobs get less priority, so it may take a little longer to get the job to start. In practice, I have found that the wait times on the knl cluster are pretty short.

**A small detail**: if you look in a recent ascot work directory, you will see a file e.g. group\_go\_725.**h6** in additional to the actual output file ascot\_36254397.h5. The file group\_go\_725.h6 is a carbon copy of the python script group\_go\_725.h5. The reason for having this copy is a backup in case you inadvertently delete the original group\_go\_725.py. The reason for the suffix .h6 rather than .h5 is that there are some ASCOT utilities that will contatenate all of the .h5 files in a given directory (in the past even a single ASCOT run would generate multiple .h5 output files, and so there was a need for a utility to stitch the multiple output files into a single output file for post-processing). But it would cause fatal errors if both input (.h6) and output (.h5) files were concatenated.

An annotated sample batch shell script is available [here](file:///C:\Users\sscott\Documents\ripple\ASCOT_handover\knl_780_sample.docx). With the switches defined in that batch shell script, each of the requested 10 nodes will process simultaneously 536 markers. I don’t know how this happens since I think each knl node has only 134 processors but maybe there is multithreading going on.

[back\_to\_top](#back_to_top)

**Managing the run**

sqs 🡨 generates a listing of your running jobs and jobs in the queue.

scancel 36254397 🡨 cancels run 36254397

mycputime 36254397 🡨 lists how much cpu time a run has consumed

mymemory 36254397 🡨 lists how much memory a run has consumed

The batch job will create a summary output file, e.g. slurm-36254397.out which is created in the ascot/ascot5/runs directory. It is sometimes helpful to examine this file, particularly if the job failes for any reason.

The batch job will also create a directory for the run, e.g. ascot/ascot\_run\_output/ascot\_work\_36254397, which will store the ASCOT output file, which in this case would be ascot\_36254397.h5.

Finally, there will be a file generated in the same directory such as ascot\_36254397.stdout. In principle, there is information in this file about how a run is proceeding, but be aware that the algorithm that ASCOT uses to estimate the time-to-completion is often misleading.

[back\_to\_top](#back_to_top)

**Bookkeeping**

In order keep track of the runs, I maintain an Excel spreadsheet, [collisional-runs.xlsx](file:///C:\Users\sscott\Documents\ripple\ASCOT_handover\collisional-runs.xlsx). It has evolved over the months and so the format at the top of the file isn’t necessarily the same as the format at the end. But the basic ideas are:

* Number the simulations sequentially.
* When I create a new line in the table for a new run, I copy the line corresponding to the ‘parent’ run. E.g. if I copy file group\_go\_703.py into file group\_go\_725.py in order to genereate an input file for run 725, I would copy the line in the table corresponding to run 703 into the line for 725. Then I highlight in cyan any input values that I change between the ‘parent’ and ‘daughter’ simulations. Note that each line has a listing of the name of the python script for the current run and its parent. So for the line for run 703, I would change the name of the parent script to group\_go\_703.py.
* The table does not list all of the ASCOT input parameters … there are too many. I list only those that get changed regularly.

[back\_to\_top](#back_to_top)

**Data Archival**

Surprisingly, data generated by ASCOT5 at NERSC is not automatically archived. **Long-term data archivel is the responsibility of the user at NERSC**. **Data stored in the production directories, e.g. project/projectdirs/m3195/ascot/ascot\_run\_output is subject to being erased without warning.**

Files are archived to the HPSS long-term storage using the commands ‘his’ and ‘htar’. I have a shell script my\_htar.sh in my NERSC home directory that invokes the htar utility with appropriate switches.

April 2023: need to add the -L switch. see my\_htar\_2023.sh. Otherwise the last argument is regarded as a single file to store, rather than as a list of pathnames to store.

sscott@cori04:~> cat my\_htar.sh

#

# $1 is the name of the tar file, e.g. july23.tar

#

# $2 is a filename containing a list of directories and/or files, one entry per line

#

echo " "

echo " I will htar the following directories and files "

cat $2

echo " "

echo " I will save this data into file: ", $1

echo " "

echo " I will pause 5 seconds before starting the htar ..."

echo " "

sleep 5

htar -cvf -Hnostage $1 -L $2

Invoking that shell script will cause the text generated by the script to be routed to the screen, which is helpful to know whether there are any error messages. But we could add a > my\_tar.log to the last line in the script, which would then provide a log of the htar for all posterity.

Some useful links for long-term data archival:

* <https://docs.nersc.gov/filesystems/archive/>
* <https://www.nersc.gov/assets/pubs_presos/archive-best-practices-JFUF-final.pdf>

NERSC recommends that you try to store between 100 GB and 2 TB each time you invoke htar.

I recently archived output files that were generated between July 28 and December 22. The total data transferred was 528 GB for ~3700 files; the wall clock time to do the archive was ~8 minutes. The typical size of my my hdf5 files is 1-3 GB, with a few extending to 5 GB. Depending on the frequency of running simulations, I would recommend that we do an archive every 1-2 months.

4/23/2023: source my\_htar\_2023.sh sds\_20220430\_A.tar directories\_thru\_20220430\_A.txt

(this was for 200 ascot\_work\_12345678 directories in file: directories\_thru\_20220430\_A.txt

1.3 terabytes in 776 files stored in 20 minutes:

HTAR Create complete for sds\_20220430\_A.tar. 1,287,469,919,744 bytes written for 776 member files, max threads: 6 Transfer time: 1210.823 seconds (1063.301 MB/s) wallclock/user/sys: 1222.269 221.002 1301.912 seconds

HTAR: HTAR SUCCESSFUL

then e.g. source cleanup\_20220430\_G.sh

did A thru G and this seems to have reduced my disk usage by 4 TB (using the du -hmc command … IRIS does not yet report a reduction. Maybe it checks just once per day?)

[back to top](#back_to_top)

**Parent / daughter runs**

Following alpha orbits with SPARC parameters is computationally expensive, e.g. it can take ~8 hours of CPU time to follow an alpha from birth to thermalization. We do three things to minimize the CPU time that is required to compute the alpha losses:

1. One might think that one should launch markers with a density proportional to the alpha source rate. That might be appropriate if we were interested in the alpha distribution function, e.g. for stability calculations. But we’re mostly interested in the losses, and the losses occur mostly for alphas born closer to the edge. So we typically choose to launch markers randomly in space, and then give them a statistical weight proportional to the alpha source rate at their birth location.
2. We do such a run, i.e.truly random birth locations, and then run a simulation that follows the alphas until they thermalize or cross the LCFS. This run identifies a ‘safe’ region, typically rho\_poloidal < 0.8, inside of which the losses are negligible. So why follow alphas in the safe region at all? … they contribute nothing to our calculated surface power density.
3. We then create say 5 ‘parent’ simulations, each with say 20,000 alphas, that launch markers with 0.8 < rho\_poloidal < 1.0, each of which follows the markers until thermalization or until they cross the LCFS. ASCOT records the positions and velocity vectors of the markers as they cross the LCFS.
4. We then use those ending positions / velocity vectors as the starting data for ‘daughter’ simulations, which follow the orbits from the LCFS until they hit an actual wall. I have determined that it is sufficient to follow the orbits for about 10 ms of simulation time.

This system allows one computatationally-expensive ‘parent’ simulation to provide starting data for multiple, computationally cheap ‘daughter’ simulations to evaluate the loss pattern on multiple candidate wall shapes. For typical simulations, the daughter simulations can be a factor 10-20 faster than parent simulations.

This approach is described further in [spd-analysis-history-8.docx](file:///C:\Users\sscott\Documents\ripple\ASCOT_handover\spd-analysis-history.docx).

The file on [NERSC allotments](file:///C:\Users\sscott\Documents\ripple\ASCOT_handover\nersc-allotment-2021-sds.docx) has information about the required CPU time for calculations of the surface power density. The bottom line is that for simulations with 10^5 markers, which seems to be adequate for calculations of the surface power density, we can simulate 20 different scenarios (i.e. 20 ‘parent’ simulations corresponding to 20 different plasma/ripple scenarios) with a yearly allotment of 1.5 x 10^6 CPU hours, which is our allotment for 2021.

We should probably make more use of the ‘low’ priority queue on KNL, which has a charge factor of 0.5, i.e. we would be able to run 40 rather than 20 parent simulations (there is a still-lower ‘flex’ queue with a charge factor of 0.25, but this queue seems to be suitable for code that is check-pointed, and ASCOT5 is not checkpointed). According to Nathan Howard, jobs in the low-priority queue sometimes wait weeks to process, so using it will require some discipline.

The NERSC policies regarding queues (termed ‘quality of service (QOS)) at NERSC is described at <https://docs.nersc.gov/jobs/policy/>

[back\_to\_top](#back_to_top)

**ASCOT software: directory structure and compilation**

The directory structure for the ASCOT5 code at NERSC is described in file

[ascot-directory-structure.docx](file:///C:\Users\sscott\Documents\ripple\ASCOT_handover\ascot-directory-structure.docx) <-- this link does not work and I don’t know why

Instructions on how to install a new version of ASCOT5 are described in file [installing\_new\_ascot5.docx](file:///C:\Users\sscott\Documents\ripple\ASCOT_handover\installing_new_ascot5.docx). I tried to be comprehensive when I wrote those instructions because compiling a new version was always painful. Pablo has some experience with this, he might be helpful if you run into problems.

[back\_to\_top](#back_to_top)

**Generating a SPARC ripple file**

I have written an IDL script to compute the 3D magnetic field in SPARC. This is described in [computing\_sparc\_ripple\_01.docx](file:///C:\Users\sscott\Documents\ripple\ASCOT_handover\computing_sparc_ripple_01.docx). It is straightforward to run – just a single flat-ascii file that contains the analysis parameters – but pretty slow.

A description of the available ripple files is available in [ripple\_case\_calculations.docx](file:///C:\Users\sscott\Documents\ripple\ASCOT_handover\ripple_case_calculations.docx).

[back\_to\_top](#back_to_top)

**Generating a file with plasma-profile data for use by SPARC from TRANSP output**

You must request that Pablo create a “pickle” file for you, which extracts various profiles out of a TRANSP run at a single time slice. Then on the SPARC Linux workstation, e.g. rws01, in the directory /home/sscott/ascot do the following, for e.g. a pickle file = SPARC\_V1E\_transp\_3mod.pkl.

python

import process\_ascot\_profiles\_mod as proc

proc.process\_profiles("SPARC\_V1E\_transp\_3mod.pkl")

There are a few prompts. Just re-type in the suggested value, except for the last two prompts which want character strings to identify the output – choose something meaningful to identify the new plot files and text file.

A sample output is in file: [test\_profiles.txt](file:///C:\Users\sscott\Documents\ripple\ASCOT_handover\test_profiles.txt).

[back\_to\_top](#back_to_top)

**Generating a candidate wall/limiter shape for use by SPARC**

**To run:**

cd /project/projectdirs/m3195/ascot/mypython (or issue my ‘tomypython’ alias)

python

import triangulate\_torus as tri

tri.construct\_torus(ishape,fn\_stub) 🡨 currently, allowable shapes are 1 thru 19.

You will have to look in the source code to determine what the values of ishape mean. The files [2020\_11\_18\_spd\_003.pptx](file:///C:\Users\sscott\Documents\ripple\ASCOT_handover\2020_11_18_spd_003.pptx) and [spd-analysis-history.docx](file:///C:\Users\sscott\Documents\ripple\ASCOT_handover\spd-analysis-history.docx) describe the ‘family’ of wall/limiter shapes that is currently implemented in triangulate\_torus.py.

The input parameter fn\_stub is a character-string prefix for filenames generated by the triangulate\_torus.py script.

There is also [software](file:///C:\Users\sscott\Documents\ripple\ASCOT_handover\stl_file_generation.docx) to read a fully 3D wall shape provided by the engineers and write it into an ASCOT5 input file (\*.h5).

[back\_to\_top](#back_to_top)

**New capability to generate a candidate wall/limiter shape for use by SPARC**

(2/8/2021): Triangulate\_torus.py has a major limitation: as a function of Z, the height of the limiter is constant, except for a ‘taper region’ near the top and bottom where the height is linearly ramped down to ‘hmin’ at the top and bottom.

In some ASCOT simulations with vertically extended poloidal limiters (going to +/- 1.0 m rather than the usual +/- 0.5 m), and without a toroidal belt limiter, I found excessive surface power density on some limiters. I suspect as we move up say beyond Z = 0.5 m, the wall is ‘pulling away’ from the LCFS, and so the limiter region for -1 < Z < -0.5 and for 0.5 < Z < 1.0 may not get much deposited power.

So I implemented a new capability that allows more flexible control of the limiter shape. In particular, it allows the limiter height to be a function of Z. This will be used to explore whether having the limiter height peak off-midplane, i.e. having a mild depression near the midplane, can more effectively spread out the power over a larger fraction of the limiter surface. The relevant Python script is construct\_profile.py and this is called by triangulate\_torus\_2.py.

A description of the generalized limiter-shape algorithm is given in [optimized\_limiter\_p2.pdf](file:///C:\Users\sscott\Documents\ripple\ASCOT_handover\optimized_limiter_02.pdf). The limiter height-profile is divided into 4 segments, and the shape of each segment can be controlled by various parameters. The control over the height as a function of toroidal angle is unchanged from the original triangulate\_torus.py. A sample limiter surface which has the height peaking off-midplane is illustrated in [shape50\_tflim0.pdf](file:///C:\Users\sscott\Documents\ripple\ASCOT_handover\shape50_tflim0.pdf).

This plot was generated by a Python script in the usual `mypython’ directory:

phthon

import plot\_tflim as my

my.plot\_tflim(‘shape50’, 0) 🡨 reads tflim\_height\_shape50\_0.txt (which is generated by triangulate\_torus\_2.py).

There is a corresponding Python script, plot\_tbllim.py, that plots the height of the toroidal belt limiter.

**ASCOT options parameters**

A sample set of ASCOT options is described in file [sample\_ascot\_input\_file.docx](file:///C:\Users\sscott\Documents\ripple\ASCOT_handover\sample_ascot_input_file.docx). It is partially self-documenting. There are several groups of options parameters:

1. **Simulation mode** (i.e. full Gyro-orbit versus guiding center)
2. Control of **time step** for integration
3. **End conditions**: when to declare that an orbit has been completed, e.g. it hit a wall, it crossed the LCFS, it thermalized, it exceeded some simulation time or CPU time limit, etc.
4. **Physics**: e.g. options to turn off Coulomb collisions
5. **Distributions**: control over the grid resolution for storing the computed alpha distribution function in phase space.
6. **Orbit recording**: control over whether to store the computed orbit, e.g. number of points to store and time step between times when the orbit parameters are store. If you ask to store N points but the orbit history contains more than N points, only the last N points in the orbit’s history will be stored.

You need to think carefully about the specification of orbit recording. First, if you ask to store too many time points (and/or ask for too many markers), the computer will run out of memory and you will get the dreaded segmentation fault. Second, you probably need to decide whether you want to store the entire orbit time history (e.g. from birth to thermalization) with coarse time resolution, or the just end of the orbit with fine time resolution.

The alpha velocity at birth is about 1.e3-7 meters, so if you ask for a recording time step of 3.e-9 sec you will get abou 4 cm between recording times. The circumference of the outside of SPARC is about 15 meters, so if q=4, I think you would need to follow the orbit for about 60 meters to get one poloidal orbit, which would require storing about 1500 time points if you ask for a time resolution of 3.e-9 sec.

The full thermalization time for alphas born at the plasma center is about 0.20 sec (there is IMHO a surprising variability in the thermalization time, even for alphas born at the same radius). So if you want to store the full orbit time history and you have determined that you have memory to store only 2000 time points, then you would need to specify a recording time interval of 0.2/2000 = 10-4 sec.

1. **Transport coefficients**: I have not used these options.

A high-level description of some of the most important options parameters is provided in file [preparing\_ascot\_input.docx](file:///C:\Users\sscott\Documents\ripple\ASCOT_handover\preparing_ascot_input.docx).

[back\_to\_top](#back_to_top)

**Structure of Python scripts that generate the ASCOT5 hdf5 input file**

The ASCOT5 input file, which is written in the hdf5 file format, is generated by a Python script, e.g. group\_go\_728.py. These are stored in the …/ascot5/runs directory. It might be helpful to open up such a file as you read this section, which describes the overall structure of these Python scripts.

The code must provide specifications for ten major sections:

1. Magnetic equilibrium
2. Magnetic field: can be 2D or 3D.
3. Options: specify the options parameters discussed in the previous section.
4. Markers: specify the number, mass, and charge of the markers, and the initial positions and velocity vectors of the markers.
5. Profiles: specify the kinetic profiles of the thermal plasma, e.g. Ne, Ni, Te, Ti.
6. Wall: specify the wall shape. the wall can either be 2D or 3D.

I have not used the following types of data. There is ‘dummy code’ in the Python script to write dummy data into the input file.

1. Boozer: I think this section contains Boozer coordinates. Maybe this is used for orbit simulations for stellarators.
2. Neutral density
3. MHD
4. electric field

Generally, the structure in the Python script (e.g. group\_go\_728.py) for each group of data described above is as follows:

* Some parameter values are written into a Python dictionary.
* The code calls a Python script that I wrote specifically for that group of data (typically in the …/ascot/mypython directory), passing in the disctionary along with some other parameter values. For example, there is a Python script marker\_sets.py that defines the ensemble of markers, and a Python script options\_sets.py that defines the options.
* Inside the Python script e.g. marker\_sets, the code might do some calculations (for example, create the ensemble of birth marker positions) and it might define some default values for some parameter names. Then it calls an official ASCOT-provided Python script which does the actual writing of data into the ASCOT5 hdf5 file.

[back to top](#back_to_top)

**Official ASCOT software utilities**

There are a number of software utilities written and maintained by the ASCOT team. I have used them sparingly, having chosen instead to postprocess the ASCOT output with my home-grown Python scripts, over which I have more control. In retrospect, perhaps I should have spent more time learning how to better use the capabilities of these utilities.

Currently the utilities that I use live in directory /global/homes/s/sscott/.local/cori/3.6-anaconda-4.4/bin/

Confusion: there are also copies of these utilities in directory: /project/projectdirs/m3195/ascot/python/a5py/bin/

Presumably, we should be using these utilities rather than the ones in my personal directory.

I’m a little confused about how they are ‘defined’, in the sense that all I need to do is type e.g. a5gui ascot\_CP.h5 (where ascot\_CP.h5 is the name of an ASCOT output file) from any directory, and the utility is magically invoked. I can’t figure out how the system ‘knows’ that the character string a5gui gets mapped to the directory /global/homes/s/ …

If I type ‘which a5gui’ at the command prompt level, the system responds with /global/homes/s/sscott/.local/cori/3.6-anaconda-4.4/bin/a5gui

My .bashrc.ext file makes an alias for a5gui 🡪 /project/projectdirs/m3195/ascot/python/a5py/bin/a5gui

They seem to be simple Python shells that call other Python scripts. Generally, I think that to invoke these utililties, you typically type <utility name> <h5-fielname>

* **a5gui**: This is an all-purpose utility that allows you to view the input options settings, Bfield, Efield, results (inistate, endstate, distributions), wall, plasma data.
* **a5editoptions**: this allows you to view and edit the options settings. I think by default it uses the VIM editor, but I think that by putting an appropriate line in your .bashrc.ext file, you can force it to use other editors including emacs.
* **a5combine**: this allows you to combine multiple ASCOT hdf5 files into a single file.

IMHO, there is IMHO a small limitation to this utility. As described below, the hdf5 file includes sections for ‘marker’, and ‘results’, along with some other sections of data. It seems that the a5combine utility does combine the ‘results’ from multiple simulations, but it does not combine the ‘marker’ information. This does not seem to be a problem for the various ASCOT software utilities, but my home-grown utility process\_ascot.py has some plots that include both marker and results data (for example, it plots the major radius of the markers versus major radius of the inistate).

So the a5combine utility can be used to combine multiple ASCOT5 hdf5 files if the resulting combined file is to be processed by the ASCOT-supplied software utilities, but it cannot be used to create a combined hdf5 file that can be used by my home-grown Python scripts such as process\_ascot.py.

But note that process\_ascot.py has been upgraded to allow it to read multiple hdf5 files directly, and in doing so it properly copies all of the data from multiple runs, including the marker data.

I opened up issue #245 in the ASCOT web portal about this issue. Maybe it will get fixed in the future.

example:

5combine add ascot\_DV.h5 ascot\_39176023.h5 ascot\_39176024.h5 ascot\_39176025.h5

… then type ‘c’ when prompted

* **a5doxygen, a5vol, a5ls, a5setactive, a5continuerun**: … there are a number of additional utilities that I have not used.

Problem: in the past, I have had problems invoking these utilities because the system reports that it can’t find the module ‘unyt’ (which supports processing of units, e.g. eV, in the utilities). I forget the details, but when I try to install the module (need to remember the syntax for doing that), the system reports no errors during the install, but then I get the same error message when I try to invoke the script. The problem is that when you install a module, you must install it into the version of Python that the script actually uses.

pip install –user unyt 🡨 this worked for Alex (2/17/2021)

[back to top](#back_to_top)

**Structure of ASCOT5 hdf5 files**

The structure of the hdf5 files is:

* + **bfield**: contains all information about the magnetic field
  + **boozer**: presumably contains information about Boozer coordinates
  + **efield**: contains information about the electric field
  + **marker**: contains information about the ‘markers’ which is information about the birth locations, velocitities, mass, charge etc of the particles that are to be followed. This information is provided by the user as input data. Note that the velocities are defined in simple [R, phi, Z] coordinates, not for example in v\_parallel, v\_perp because at the time the marker information is computed, we don’t know anything about the magnetic field.
  + **mhd**: presumably defines the structure of MHD.
  + **neutral**: presumably defines the neutral density profile
  + **options**: contains the user-defined options parameters that controls the simulations.
  + **plasma**: contains all information about the plasma kinetic profiles, e.g. temperature and density.
  + **results**: contains all of the computed results of the simulation. Includes:
    - **dist5d**: (and probably dist6d if you asked the simulation to store the 6-dimensional distribution) contains the distribution function of the fast ions.
    - **endstate**: contains the position, velocity, etc of the markers when their simulation ends. Also includes an integer array ‘ENDCOND’ which defines what event caused the simulation to terminate the marker’s orbit simulation, e.g. the marker thermalized, crossed the LCFS, hit the wall, etc.
    - **inistate**: contains the ‘initial state’ of the markers. Some explanation is needed about how this information differs from the ‘marker’ data: the inistate data is computed by ASCOT given the marker information and the magnetic field data, i.e. if memory serves me correctly, the inistate includes variables such as v\_parallel. Also, depending on the type of simulation (guiding center versus gyro-orbit), there may be a shift of up to a gyro-radius in the position of the inistate versus marker.
    - **orbit**: contains the recorded orbit inforamtion
  + **wall**: contains information about the wall.

[back to top](#back_to_top)

**Postprocessing SPARC output**

**hands-and-feet**

First, we need to copy the output files into the “runs” directory. Reminder: green text is what is typed by the user; cyan text is an alias defined in .bashrc.ext (also typed by the user)

It might be a good idea for someone at some point to add code to the batch script (e.g. knl\_702.sh) that copies the ascot output file to the …/ascot5/runs directory so that we don’t have to manually do this tedious copy.

toruns

copy\_to\_runs 36825352

copy\_to\_runs 36837310

copy\_to\_runs 36837314

copy\_to\_runs 36837322

toruns <-- the last command leaves us in directory … ascot/ascot\_work\_36837322

Now run the home-grown ASCOT postprocessor process-ascot.py

python $dir\_mypython/process\_ascot.py 🡨 launches the Python script

enter comma-separated list (w/o spaces) of ascot output .h5 files( default=) **ascot\_36825352.h5,ascot\_36837310.h5,ascot\_36837314.h5,ascot\_36837422.h5**

enter name of geq file( default=v1e.geq) **<cr>** 🡨 a <cr> accepts default value

enter profiles filename( default=v1e\_profiles\_3.txt) **<cr>**

enter kW of lost alphas( default=0.0) **480.**

enter integer that specifies grid resolution for spd calc( default=0) **6**

enter stub for outout filename (none for outout to screen)( default=)

**ascot\_AR\_pshape7\_ashape6** 🡨 note that the “stub” does not include e.g. .txt

… the code will take a minute to several minutes to process, depending on the number of files and the total number of markers that are being processed

Note that you supply a list of one or more ASCOT output files, in single quotes, **without any spaces between the filenames**. The processor will then combine the outputs of the two files before computing e.g. the fraction of lost alpha power. This allows us to break a big calculation of say 100,000 alphas into more manageable runs of 20,000 alphas each.

The second argument (v1e.geq’) is the EFIT equilibrium for the run.

The parameter “kW of lost alphas” is the number of kilowatts of alpha power that hits the wall. One would think that this would be an ouput parameter rather than an input parameter, but for reasons discussed in the section on parent/daughter runs, this is not the case.

The “integer that specified grid resolution for the spd calc” controls the grid size for the poloidal limiters, toroidal belt limiters, RF antennas etc. As shown in the code snippet below the integer (called ishape internally in the construct\_torus function within process\_ascot.py) is used as an index to select say the number of grid points in the phi direction, nphi\_limiter, and the number of grid points in the Z direction, nz\_limiter. There are similar arrays for the toroidal belt limiter, the RF antennas, and all of the other structures that comprise the plasma-facing wall.

# ishape 1 2 3 4 5 6 7 8

nnphi\_limiter = [5, 5, 20, 10, 5, 5, 4, 5] 🡨 these are hard-wired sets of

available grid resolutions.

nnz\_limiter = [6, 6, 20, 10, 6, 10, 10, 15]

nphi\_limiter = nnphi\_limiter[ishape-1] 🡨 this selects one value from the set

nz\_limiter = nnz\_limiter[ishape-1]

Output: e.g. ascot\_pshape7\_ashape6.txt and ascot\_pshape7 need to finish this

Both the text file and the plot file have accumulated a lot of bloat over the past months, mostly to debug particular problems. The plot file in particular is big – about 200 pages of plots.

Sample output is given in file [ascot\_33081964\_multi.pdf](file:///C:\Users\sscott\Documents\ripple\ASCOT_handover\ascot_33081964_multi.pdf) and annotations for these plots are given in [annotations\_for\_figures.docx](file:///C:\Users\sscott\Documents\ripple\ASCOT_handover\annotations_for_figures.docx).

[back\_to\_top](#back_to_top)

**Comparing two SPARC simulations**

I have written a Python script to generate plots that overlay the results of two ASCOT simulations. This does not have all the bells and whistles of process\_ascot.py, in particular it doesn’t (yet) have the capability to plot a map of the surface heating power density. But it is very helpful if you want to quickly identify how two ASCOT simulations differ.

to run:

python /project/projectdirs/m3195/ascot/mypython/process\_two.py ascot\_12345678.h5 ascot\_87654321.h5 v1e.geq

where ascot\_12345678.h5 and asscot\_87654321.h5 are two ascot output files, and v1e.geq is the EFIT file. Or, if you have defined the aliases in my .bashrc.ext file,

Or, if you have defined the alias process\_two in my bashrc.ext file, you can issue the command

process\_two 12345678 87654321 v1e.geq

There is also a version that makes fewer plots,

python /project/projectdirs/m3195/ascot/mypython/process\_two\_short.py ascot\_12345678.h5 ascot\_87654321.h5 v1e.geq

[back\_to\_top](#back_to_top)

**Plotting orbits**

To generate a plot file containing the orbits computed by ASCOT, do the following:

python /project/projectdirs/m3195/ascot/mypython/plot\_orbits\_new.py 36259049 v1e.geq 20

where 36259049 is the 8-digit identifier of the ASCOT simulation, and 20 is the number of orbits that you want to see. Sample plots are shown in file: [ascot\_36259049\_orbits\_only.pdf](file:///C:\Users\sscott\Documents\ripple\ASCOT_handover\ascot_36539049_orbits_only.pdf).

[back\_to\_top](#back_to_top)

Generating VTP output files for 3D rendering by ParaView

ParaView is a powerful, free 3D rendering program. The ASCOT software suite includes a postprocessor that exports the surface power density computed on the defined 3D PFC surface (of contiguous triangles) in the .vtp format required by ParaView. The Paraview 3D visualization of the surface heating has advantages and disadvantages over the calculations of surface power density as computed by my home-grown ASCOT postprocessor (process\_ascot.py):

* **(Big) advantage:** ParaView “knows” the full 3D shape of the PFC surface, including the “sides” of the poloidal and toroidal belt limiters. These “sides” are present due to the fact that the plasma-facing surfaces of the limiters are, by definition, proud of the wall surface by some distance, typically up to ~10 mm. So at the edges of the limiters, there are “gaps” between the wall surface and the front-facing surface of the limiters – these gaps are the “sides” of the limiters.

By design, these “sides” are recessed by some distance relative to the maximum height of the limiter surfaces and so they are effectively in the “shadow” of the limiters. So one might expect that these sides will receive little of the lost-alpha power. On the other hand, at least in my present design-family of candidate wall shapes, some of the sides are nearly perpendicular to magnetic field lines.

So we must always check the surface power density on these sides, and we must use Paraview to do that.

* **Disadvantage:** the surface power density is computed by ASCOT on a grid of triangles that is defined in the ASCOT input file. So if you want to evaluate a different PFC shape, you need to rerun ASCOT again. This takes about 2 hours of CPU time on NERSC using ~10 nodes, which is not terribly expensive but could add up if a large number of candidate wall shapes are being evaluated.
* **Disadvantage:** my Python script triangulate\_torus generates the ensemble of surface triangles that collectively represent the PFC surface; it generates a file that can be read by the ASCOT preprocessor which constructs the ASCOT input file. Triangular\_torus.py “knows” the [R,Z] geometry of individual structures such as limiters but the knowledge that say triangle #4532 resides in the upper corner of the 14th poloidal limiter is **lost** when the ensemble of triangles is created. So it is not possible, or at least not easy, to average the pattern of surface power density over the 18 poloidal limiters in order to improve statistics.

I have written a [Paraview tutorial](file:///C:\Users\sscott\Documents\ripple\ASCOT_handover\paraview_tutorial.pdf) which explains how to do simple stuff in Paraview, with an emphasis on those functions that are useful for rendering ASCOT5 output.

**Generating the .vtp file**

python $dir\_mypython/export\_vtp\_sds.py ascot\_12345678.h5 ploss\_kW

python $dir\_mypython/export\_vtp\_hits.py ascot\_12345678.h5

python $dir\_mypython/export\_vtp\_norm.py ascot\_12345678.h5 ploss\_MW

The first plots the surface power density on the defined 3D PFC surface; the second plots the number of marker ‘hits’ per triangle on the same surface, and the third plots the normalized surface power density on the same surface (need to remember whether the maximum normalization is 1.0 or 100.0)

Corresponding output files: ascot\_12345678.vtp, ascot\_12345678\_hits.vtp, and ascot\_12345678\_norm.vtp.

**Generating the vtp file: combining two runs**

When we compute the spatial distribution of the ‘prompt’ and ‘nonprompt’ alpha losses in separate ASCOT simulations, we need a utility to combine them into a single .vtp file for use by Paraview and by Matt Reinke’s software that computes the surface heating. To do this:

In the standard ‘runs’ directory (/project/projectdirs/m3195/ascot/ascot5/run):

python $dir\_mypython/export\_vtp\_combine.py

Enter name of hdf5 file with prompt losses: ( default=) ascot\_40890729.h5

Enter name of hdf5 file with nonprompt losses: ( default=) ascot\_40921464.h5

Enter total prompt power loss (kW) ( default=0.0) 476.

Enter total nonprompt power loss (W) ( default=0.0) 170.

This assumes that you already have copied the files e.g. ascot\_40890729.h5 from the output directory into the runs directory with the copy\_to\_runs 40890729 command.

Output: ascot\_40890729\_40921464.vtp

Note: the utility doesn’t do anything different when processing the ‘prompt’ versus ‘nonprompt’ data. It just combines the two simulations with the associated user-defined total alpha power losses. The prompts include ‘prompt’ and ‘nonprompt’ just to remind the user what the utility is intended to do.

**Combining prompt and non-prompt ASCOT output**

python $dir\_mypython/process\_ascot\_pnp.py

enter comma-separated list (w/o spaces) of ascot output .h5 files for PROMPT losses( default=) ascot\_41267512.h5

enter comma-separated list (w/o spaces) of ascot output .h5 files for NONPROMPT losses( default=) ascot\_41268353.h5

enter kW of lost alphas (prompt)( default=0.0) 476.

enter kW of lost alphas (non-prompt)( default=0.0) 170.

enter profiles filename( default=v1e\_profiles\_3.txt) <cr>

enter name of geq file( default=v1e.geq) <cr>

enter stub for outout filename (none for outout to screen)( default=) ascot\_pnp\_41267512\_41268353

enter filename for triangles-parameters( default=) shape81\_parameters.txt

enter True to suppress toroidal belt limiters ( default=False) True

enter maximum marker (0 for all of them) ( default=0) 40000

**Combining ASCOT output files**

The home-grown ASCOT postprocessor process\_ascot.py allows you to combine multiple ASCOT output files and thus the instructions given below are not needed (and in fact would cause a problem for process\_ascot.py).

But the export\_vtp\_sds.py works with just a single ASCOT output file, and so if you want to plot the surface power density from the combination of multiple ASCOT output files, you must first combine them into a single ASCOT output file.

ASCOT has an official utility to do this. I recommend that if you have files ascot\_1.h5, ascot\_2.h5, and ascot\_3.h5 that you want to combine, do the following:

cp ascot\_1.h5 ascot\_123\_combined.h5

a5combine add ascot\_123\_combined.h5 ascot\_2.h5 ascot\_3.h5

In response to the prompt, type ‘c’. The reason for copying ascot\_1.h5 into ascot\_123.h5 is that the a5combine utility copies the contents of the second and third files into the first. So this approach leaves the files ascot\_1.h5, ascot\_2.h5 and ascot\_3.h5 unperturbed.

a5combine seems to be a Python script that currently lives in

/global/homes/s/sscott/.local/cori/3.6-anaconda-4.4/bin/a5combine

**3D rendering .vtp files with ParaView**

Obtain the Paraview download file from paraview.org and install it on your PC

Invoke the ParaView application

Click on File (upper-left corner, as usual) 🡪 Open (or Recent Files)

Migrate to the folder that has your .vtp file and click on it.

For both better and worse, ParaView is a fully-featured application. There can be a considerable learning curve, and I am not very far up that curve.

[back to top](#back_to_top)

**END-CONDITIONS for ASCOT5 simulations**

There is an array ENDCOND in the ‘endstate’ group of output variables which is an integer array that specifies why ASCOT terminated an orbit’s simulation. The possible endconditions and their corresponding integer is given in the table below. I’m pretty sure this table is incomplete, for example I think that there should be an endcondition corresponding to value 128.

aborted 0

max simtime 1

minimum energyemin 2

thermalized 4

emin\_and\_thermalized (?) 6

hit\_wall 8

minimum rho 16

maximum rho 32

unknown = 40 (combination of maximum rho and hit\_wall?)

max cputime 256

hybrid 512

hybrid\_rho 544 (probably combination of hybrid and maximum rho?)

unknown 520 (probably combination of hybrid and hit\_wall?)

[back to top](#back_to_top)

Aliases

Working with ASCOT5 at NERSC involves a lot of moving from one directory to another. Because my typing isn’t so good, and neither is my memory, I define a lot of aliases to take me from directory to directory. These aliases are defined in file /global/homes/s/sscott/.bashrc.ext. A number of these aliases are described in the table below. I would encourage the next owner of ASCOT for SPARC to copy these aliases, or grow his or her own.

In the table below, all directories lie under the /project/projectdirs/m3195/ascot directory.

|  |  |  |
| --- | --- | --- |
| alias | directory | use |
|  |  |  |
| toruns | ascot5/runs | Launch ASSCOT runs from here. Sometimes post-process runs here |
| tooutput | ascot\_run\_output | All ASCOT output files are stored in sub-directories of this directory |
| tomypython | myphthon | All of my home-grown Python scripts, for e.g. postprocessing ASCOT output and generating 3D wall shapes |
| topython | python | Official python scripts written by the ASCOT team |
| toascot5 | ascot5 | /project/projectdirs/m3195/ascot/ascot5 |
| toascot | - | takes you to /project/projectdirs/m3195/ascot |
|  |  |  |

I also have aliases for common commands. In the table below, RUNID is an 8-digit NERSC job number that identifies an individual ASCOT simulation and H5FILE is the name of the ASCOT input or output hdf5 file.

|  |  |
| --- | --- |
| alias | usage |
|  |  |
| mycputime RUNID | lists how much CPU time was charged for a run |
| mymemory RUNID | lists how much memory was used by a run |
| myprocess … | launches my ASCOT postprocessor |
| process\_two … | launches a process to compare two ASCOT simulations. maybe broken? |
| a5gui H5FILE | launches a GUI to examine ASCOT input/otput |
| a5editoptions | launches a GUI to examine ASCOT input switches |
| copy\_to\_runs | Copies an ASCOT5 output run to the …ascot/ascot/runs directory |

[back\_to\_top](#back_to_top)

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.

[back\_to\_top](#back_to_top)

Annotated Python scripts to generate ASCOT input files

The table below has links to Python scripts that generate various types of ASCOT simulations

|  |  |  |
| --- | --- | --- |
| Type | Annotated Python script | ASCOT output file |
|  |  |  |
| grandparent | [group\_go\_637\_annotated.docx](file:///C:\Users\sscott\Documents\ripple\ASCOT_handover\group_go_637_annotated.docx) | ascot\_33081964.h5 |
| Powerloss<p | [group\_go\_1139\_annotated.docx](file:///C:\Users\sscott\Documents\ripple\ASCOT_handover\group_go_1139_annotated.docx) | ascot\_40893244.h5 |
| weights | [group\_go\_1127\_annotated.docx](file:///C:\Users\sscott\Documents\ripple\ASCOT_handover\group_go_1127_annotated.docx) | ascot\_40877296.h5 |
| promptLoss | [group\_go\_1129\_annotated.docx](file:///C:\Users\sscott\Documents\ripple\ASCOT_handover\group_go_1129_annotated.docx) | Ascot\_40879047.h5 |
| nonpromptLoss | [group\_go\_1130\_annotated.docx](file:///C:\Users\sscott\Documents\ripple\ASCOT_handover\group_go_1130_annotated.docx) | ascot\_40880287.h5 |
| promptWall | [group\_go\_1137\_annotated.docx](file:///C:\Users\sscott\Documents\ripple\ASCOT_handover\group_go_1137_annotated.docx) | ascot\_40890729.h5 |
| nonpromptWall | [group\_go\_1141\_annotated.docx](file:///C:\Users\sscott\Documents\ripple\ASCOT_handover\group_go_1141_annotated.docx) | ascot\_40921464.h5 |
|  |  |  |

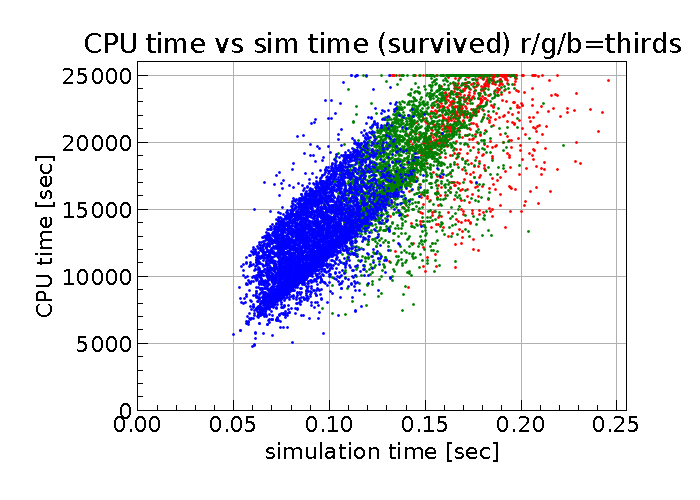
Plasma scenarios

The kinetic profiles needed by ASCOT are computed from predictive time-dependent TRANSP simulations run by Pablo Fernandez-Rodriguez. He has simulated a variety of plasma scenarios including the ‘primary reference discharge’ (PRD): BT = 12.2 Tesla, Ip = 8.7 or 8.9 MA, QDT ~ 9-11.

Other scenarios are described in [ASCOT plasma scenarios](file:///C:\Users\sscott\Documents\ripple\ASCOT_handover\scenario_1_2_notes.docx).

Reports and Presentations (in reverse chronological order)

1. I have a short file [blender\_help.docx](file:///C:\Users\sscott\Documents\ripple\ASCOT_handover\blender_help.docx) that describes how to visualize the 3D wall/limiter geometry that is constructed by triangulate\_torus.py (11/25/2020)
2. A listing of my ASCOT runs is maintained in file [collisional-runs.xlsx](file:///C:\Users\sscott\Documents\ripple\ASCOT_handover\collisional-runs.xlsx).
3. Description of first ASCOT calculations of surface power density for SPARC ([spd-analysis-history.docx](file:///C:\Users\sscott\Documents\ripple\ASCOT_handover\spd-analysis-history.docx)), 11/19/2020. For a standard SPARC V1E ripple field assuming an ensemble of in/out displacements of the 18 TF coils with a normal distribution having a sigma of 6 mm and an imposed maximum displacement of 9 mm, the maximum surface power density is computed to be 260-352 kW/m2, which is less than the 457 kW/m2 computed for the same case by SPIRAL. This was sent to Gerrit Kramer at PPPL to start a discussion about hos to resolve the discrepancy.
4. First ASCOT to-the-wall calculation of the surface power density in SPARC ([2020\_11\_18\_spd\_003.pptx](file:///C:\Users\sscott\Documents\ripple\ASCOT_handover\2020_11_18_spd_003.pptx)), 11/19/2020. This is a powerpoint presentation that covers materials related to the analysis in [spd-analysis-history.docx](file:///C:\Users\sscott\Documents\ripple\ASCOT_handover\spd-analysis-history.docx).
5. Ripple-induced fast-ion loss in SPARC due to misaligned TF coils ([APS-2020-scott-15.pdf](file:///C:\Users\sscott\Documents\ripple\ASCOT_handover\APS-2020-scott-15.pdf) and [APS-2020-scott-15.pptx](file:///C:\Users\sscott\Documents\ripple\ASCOT_handover\APS-2020-scott-15.pptx)), 11/5/2020. Compares calculations of expected alpha loss in SPARC by the ASCOT and SPIRAL codes.
6. A description of the directory structure for the ASCOT code and its utilities is provided in file [ascot-directory-structure.docx](file:///C:\Users\sscott\Documents\ripple\ASCOT_handover\ascot-directory-structure.docx), 10/17/2020.
7. ASCOT PFC shape specification ([ASCOT\_PFC\_shape\_specification.docx](file:///C:\Users\sscott\Documents\ripple\ASCOT_handover\ASCOT_PFC_shape_specification.docx)), 10.16/2020. A family of candidate wall and limiter shapes is generated by the Python script triangulate\_torus.py at NERSC. This file describes the contents of two files, [tflim\_height\_0.txt](file:///C:\Users\sscott\Documents\ripple\ASCOT_handover\tflim_height_0.txt) and [tbllim\_height.txt](file:///C:\Users\sscott\Documents\ripple\ASCOT_handover\tbllim_height.txt). These files were sent to Gerrit Kramer at PPPL so that he could implement the same wall shape in his SPIRAL code.
8. Proposed surface heating calculations ([proposed\_surface\_heating\_calcs.docs](file:///C:\\Users\\sscott\\Documents\\ripple\\ASCOT_handover\\proposed_surface_heating_calcs.docx)), 10/16/2020. This document describes various calculations of surface power and surface power density that will be compared between the ASCOT and SPIRAL codes.
9. Model of candidate wall/limiter shape for SPARC ([PFC\_components\_1.pdf](file:///C:\Users\sscott\Documents\ripple\ASCOT_handover\PFC_components_1.pdf)), 10/16/2020. This is a schematic of the family of candidate wall/limiter shapes that will be simulated by the SPIRAL and ASCOT codes.
10. Early ASCOT calculations of surface power density of lost alphas in SPARC ([gk\_surface\_heating.docx](file:///C:\Users\sscott\Documents\ripple\ASCOT_handover\gk_surface_heating.docx) and [gk\_surface\_heating.pdf](file:///C:\Users\sscott\Documents\ripple\ASCOT_handover\gk_surface_heating.pdf)), includes some 3D renderings of the candidate wall shape for SPARC.
11. I have implemented a simple 2D SPARC wall shape that can also be used by ASCOT. This is illustrated in [group\_go\_667\_wall\_sparc\_simple\_wall.pdf](file:///C:\Users\sscott\Documents\ripple\ASCOT_handover\group_go_667_wall_sparc_simple_wall.pdf), 10/3/2020. This figure was extracted from ASCOT run generated by group\_go\_667.py at NERSC. A flat-ascii text file with [R,Z] coordinates of the wall is given in file [sparc\_simple\_wall.txt](file:///C:\Users\sscott\Documents\ripple\ASCOT_handover\sparc_simple_wall.txt).
12. A number of 3D illustrations of the SPARC candidate wall shape are given in [2020\_10\_01\_fake\_fine\_nowalls\_02.pptx](file:///C:\Users\sscott\Documents\ripple\ASCOT_handover\2020_10_01_fake_fine_nowalls_02.pptx), 10/01/2020. Note that the wall shape is “fake” in the sense that the height of the poloidal and toroidal limiters above the wall surface was artificially increased to allow visual confirmation of the shape.
13. Materials that I sent to Michael Segal for the DOE review of the 2019 CFS/PPPL INFUSE grant are in file [PPPL\_INFUSE\_2019\_mid\_term\_review.docx](file:///C:\Users\sscott\Documents\ripple\ASCOT_handover\PPPL_INFUSE_2019_mid_term_review.docx), 9/24/2020.
14. Instructions on how to install new versions of ASCOT are provided in [installing\_new\_ascot5.docx](file:///C:\Users\sscott\Documents\ripple\ASCOT_handover\installing_new_ascot5.docx), 9/9/2020.
15. I was asked by Martin and/or Alex to think about what fast-ion physics risk for ARC could be retired by experiments in SPARC. But I thought the topic was just fast-ion physics in ARC, so my slides in [2020\_08\_24\_ARC\_fast\_ion\_physics\_01.pptx](file:///C:\Users\sscott\Documents\ripple\ASCOT_handover\2020_08_24_ARC_fast_ion_physics_01.pptx) and [2020\_08\_24\_ARC\_fast\_ion\_physics\_01.pdf](file:///C:\Users\sscott\Documents\ripple\ASCOT_handover\2020_08_24_ARC_fast_ion_physics_01.pdf) are a little off-topic. 8/24/2020
16. My first cut at specifying the alignment tolerance of the TF coils for SPARC is in files [2020\_08\_02\_ripple\_update\_cfs3.pptx](file:///C:\Users\sscott\Documents\ripple\ASCOT_handover\2020_08_02_ripple_update_cfs3.pptx) and [2020\_08\_02\_ripple\_update\_cfs3.pdf](file:///C:\Users\sscott\Documents\ripple\ASCOT_handover\2020_08_02_ripple_update_cfs3.pdf). This is the version of the talk that I gave internally to CFS which includes some discussion of allowable radial leakage current in the TF coils. 8/21/2020
17. My first cut at specifying the alignment tolerance of the TF coils for SPARC is in files [2020\_08\_02\_ripple\_update\_07.pdf](file:///C:\Users\sscott\Documents\ripple\ASCOT_handover\2020_08_02_ripple_update_cfs3.pdf). This is the version of the talk that I gave to the SPARC physics design team and it excludes a discussion of allowable radial leakage current in the TF coils. 8/11/2020
18. I carried out extensive studies with the ASCOT code to determine the maximum allowable time step for ASCOT’s orbit simulations. A longer time step decreases the CPU time needed for a simulation, but if the time step is too big the orbits are not accurately simulated. The results of this study are described in [time-step-report-10.docx](file:///C:\Users\sscott\Documents\ripple\ASCOT_handover\time-step-report-10.docx) and [time-step-report-10.pdx](file:///C:\Users\sscott\Documents\ripple\ASCOT_handover\time-step-report-10.pdf), 7/27/2020.
19. I had an idea for an improved pellet injector. This is described in file [pellet-injector-02.pptx](file:///C:\Users\sscott\Documents\ripple\ASCOT_handover\pellet-injector-02.pptx), 7/22/2020. There is possibly some IP in this file, which I hereby give to CFS.
20. A study of the allowable ‘adaptive’ time step for ASCOT orbit simulations is in [2020\_07\_21\_adaptive\_speed.docx](file:///C:\Users\sscott\Documents\ripple\ASCOT_handover\2020_07_21_adaptive_speed.pdf), 7/21/2020.
21. Up to July 2020, I had a lot of ASCOT simulations using the computed ripple field for the V1E TF design. But somewhere in mid-2020, the new version V2a was released. The file [2020\_07\_20\_v2a\_ripple\_2.pdf](file:///C:\Users\sscott\Documents\ripple\ASCOT_handover\2020_07_20_v2a_ripple_2.pdf) compares the ripple field for the V1E and V2a designs. On the basis of this presentation, we decided not to migrate to the V2a design but rather to continue the orbit simulations based on the V2a design.
22. More ASCOT timing studies, i.e. maximum allowable time step, are in files [2020\_07\_16\_18TF\_10\_15.pdf](file:///C:\Users\sscott\Documents\ripple\ASCOT_handover\2020_07_16_18TF_10_15.pdf), [2020\_07\_16\_18TF\_10\_30.pdf](file:///C:\Users\sscott\Documents\ripple\ASCOT_handover\2020_07_16_18TF_10_30.pdf), [2020\_07\_16\_18TF\_inout\_10\_20.pdf](file:///C:\Users\sscott\Documents\ripple\ASCOT_handover\2020_07_15_18TF_inout_10_20.pdf), [2020\_07\_14\_ascot\_spiral\_GO\_speed.pdf](file:///C:\Users\sscott\Documents\ripple\ASCOT_handover\2020_07_14_ascot_spiral_GO_speed.pdf), [2020\_07\_14\_ascot\_spiral\_GO\_speed.docx](file:///C:\Users\sscott\Documents\ripple\ASCOT_handover\2020_07_14_ascot_spiral_GO_speed.docx), [2020\_07\_15\_12TF\_hybrid\_10\_25.pdf](file:///C:\Users\sscott\Documents\ripple\ASCOT_handover\2020_07_15_12TF_hybrid_10_25.pdf), [time\_step\_report\_07.docx](file:///C:\Users\sscott\Documents\ripple\ASCOT_handover\time-step-report-07.docx), [time\_step\_report\_07.pdf](file:///C:\Users\sscott\Documents\ripple\ASCOT_handover\time-step-report-07.pdf), [time\_step\_report\_08.docx](file:///C:\Users\sscott\Documents\ripple\ASCOT_handover\time-step-report-08.docx), and [time\_step\_report\_08.pdf](file:///C:\Users\sscott\Documents\ripple\ASCOT_handover\time-step-report-08.pdf). these were all created mid-July 2020.
23. I use a ‘parent/daughter’ approach to computing where lost alpha power strikes the SPARC wall and limiters: a ‘parent’ ASCOT simulation computes the orbits of alphas until they either thermalize or cross the last closed flux surface (LCFS). ASCOT records the position and velocity vectors of markers as they cross the LCFS. This data is used as the starting positions/velocities for ‘daughter’ simulations that then follow the alphas until they strike the wall. Multiple daughter runs, which examine multiple candidate wall shapes, can be generated by a single (computationally expensive) parent simulation. This approach is described in [parent\_daughter\_markers\_2.pdf](file:///C:\Users\sscott\Documents\ripple\ASCOT_handover\parent_daughter_markers_2.pdf), 7/12/2020.
24. We performed ASCOT simulations of alpha particle loss for V1E TF coils but for a TF set having 12 rather than 18 TF coils. This is interesting not because SPARC would ever be designed for 12 TF coils – the ripple losses would be much too high – but because it allows us to examine a case where the ripple-induced losses dominate over first-orbit losses. This is described in file [2020\_06\_01\_12TF\_10\_20.pdf](file:///C:\Users\sscott\Documents\ripple\ASCOT_handover\2020_06_01_12TF_10_20.pdf).
25. I started out running ASCOT on the Haswell computer cluster at NERSC, but the wait times were too long, so I migrated to using the KNL cluster. I carried out a number of studies to determine the optimum compile and run-time switches to use on KNL. These are described in files [2020\_01\_17\_haswell\_knl\_timing\_study.docx](file:///C:\Users\sscott\Documents\ripple\ASCOT_handover\2020_01_17_haswell_knl_timing_study.docx), [2020\_01\_17\_haswell\_knl\_timing\_study.pdf](file:///C:\Users\sscott\Documents\ripple\ASCOT_handover\2020_01_17_haswell_knl_timing_study.pdf), [2020\_04\_17\_CPU-SPEED.pdf](file:///C:\Users\sscott\Documents\ripple\ASCOT_handover\2020_04_17_CPU-SPEED.pdf).
26. As shown in the figure below, there is considerable variation in the amount of CPU time needed to simulate an alpha orbit in ASCOT. Originally I would choose the number of KNL compute ‘nodes’ for a simulation such that each processor computed only a single orbit -- this minimizes the wall-clock time needed for a simulation. But it is wasteful in terms of charged CPU time, because the time I am charged for the run is effectively set by the longest orbit simulation. So now I choose the number of nodes such that each processor computes 3-5 orbits. I call this approach ‘stacking’ and it is described in files [2020\_05\_30\_cpu\_stacking\_study.docx](file:///C:\Users\sscott\Documents\ripple\ASCOT_handover\2020_05_30_cpu_stacking_study.docx) and [2020\_05\_30\_cpu\_stacking\_study.pdf](file:///C:\Users\sscott\Documents\ripple\ASCOT_handover\2020_05_30_cpu_stacking_study.pdf).



*Figure nn. Scatter plot of required CPU time (Y-axis) required to complete full gyro-orbit (GO) simulations with nominal ripple in SPARC as a function of the orbit simulation time, for markers that ‘survive’ (defined as either markers that reach the thermalization condition or else hit the maximum defined CPU time, which was 26000 seconds for this run). Blue points are markers born with rho\_poloidal > 0.66; green markers have 0.33 < rho\_pol < 0.66, and red markers have rho\_pol < 0.33. Note that markers born near the center have a longer thermalization time and therefore require longer CPU time, but even within a single birth region there is considerable variation in the simulation time and CPU time; I think this could be due to random Monte Carlo statistics, but I’m a little surprised at how big the variation is.*

1. We held a plan-of-action meeting in June 2020 to discuss issues relating to TF coil fabrication, alignment, and radial current leakage. My presentation is in file [Plan-of-action-alignment-03.pdf](file:///C:\Users\sscott\Documents\ripple\ASCOT_handover\Plan-of-action-alignment-03.pdf), seems to be dated June 23, 2020.
2. yet another study of CPU time requirements for ‘hybrid’ orbit simulations is in file [cpu2.docx](file:///C:\Users\sscott\Documents\ripple\ASCOT_handover\cpu2.docx), dated May 29, 2020.
3. I made a presentation back on May 19, 2020 about how SPARC can retire energetic-particle physics issues for ARC. This presentation is in files [2020\_05\_19\_sparc\_arc\_01.pptx](file:///C:\Users\sscott\Documents\ripple\ASCOT_handover\2020_05_19_sparc_arc_01.pptx) and [2020\_05\_19\_sparc\_arc\_01.pdf](file:///C:\Users\sscott\Documents\ripple\ASCOT_handover\2020_05_19_sparc_arc_01.pdf), dated 5/19/2020.
4. I made a presentation comparing the alpha loss calculations by the ASCOT and SPIRAL codes in April 2020. It is in file [2020\_04\_07\_ripple\_v1e\_02.pdf](file:///C:\Users\sscott\Documents\ripple\ASCOT_handover\2020_04_07_ripple_v1e_02.pdf).
5. Initially, there was poor agreement between the SPIRAL and ASCOT codes regarding the fraction of alpha power that is lost to the wall; SPIRAL’s estimate (6.4%) was much larger than ASCOT’s (2.78%). In the winter of 2020 I realized that SPIRAL was improperly ‘weighting’ the alphas at birth. It took a while to convince Gerrit Kramer that this is so, but eventually he came around. Files [2020\_03\_03\_ripple\_update.pptx](file:///C:\Users\sscott\Documents\ripple\ASCOT_handover\2020_03_03_update.pptx) and [2020\_03\_03\_ripple\_update.pdf](file:///C:\Users\sscott\Documents\ripple\ASCOT_handover\2020_03_03_update.pdf) describe the issue. A detailed memo on the issue can be found in [marker\_weights\_6.pdf](file:///C:\Users\sscott\Documents\ripple\ASCOT_handover\marker_weights_6.pdf). There are two more (earlier) presentations on this issue, [2020\_02\_25\_weight\_discusssion.pptx](file:///C:\Users\sscott\Documents\ripple\ASCOT_handover\2020_02_25_weight_discussion.pptx) and [2020\_02\_25\_weight\_discusssion.pdf.](file:///C:\Users\sscott\Documents\ripple\ASCOT_handover\2020_02_25_weight_discussion.pdf)
6. My ASCOT calculations of alpha loss due to ripple generated by unequal poloidal TF currents (i.e. due to some radial leakage) are discussed in [2020\_02\_20\_curmult.pptx](file:///C:\Users\sscott\Documents\ripple\ASCOT_handover\2020_02_20_curmult.pptx) and [2020\_02\_20\_curmult.pdf](file:///C:\Users\sscott\Documents\ripple\ASCOT_handover\2020_02_20_curmult.pdf).
7. My presentation for the “10%” report is in files [2020\_02\_03\_tenpercent\_04.pptx](file:///C:\Users\sscott\Documents\ripple\ASCOT_handover\2020_02_03_tenpercent_04.pptx) and [2020\_02\_03\_tenpercent\_04.pdf](file:///C:\Users\sscott\Documents\ripple\ASCOT_handover\2020_02_03_tenpercent_04.pdf).
8. Early calculations of alpha loss for V1D, and an early comparison between computed alpha losses between the ASCOT and SPIRAL codes (showing poor agreement) are presented in file [2020\_01\_28\_ripple\_update\_02.pdf](file:///C:\Users\sscott\Documents\ripple\ASCOT_handover\2020_01_28_ripple_update_02.pdf).
9. A physics milestone report for SPARC ripple losses is in file [SPARC-RPT-PHYS-0023-V1.docx](file:///C:\Users\sscott\Documents\ripple\ASCOT_handover\SPARC-RPT-PHYS-0023-V1.docx), dated 1/8/2020. I am not sure whether this is the final version.
10. An early study of CPU time requirements on the NERSC KNL cluster is in file [knl-node-study.docx](file:///C:\Users\sscott\Documents\ripple\ASCOT_handover\knl-node-study.docx), dated 1/7/2020.
11. An early assessment of the power loading on the RF antennas due to lost alphas are in files [ascot\_rf\_antenna\_01.pptx](file:///C:\Users\sscott\Documents\ripple\ASCOT_handover\ascot_rf_antenna_01.pptx) [and ascot\_rf\_antenna\_01.pdf](file:///C:\Users\sscott\Documents\ripple\ASCOT_handover\ascot_rf_antenna_01.pdf), dated 1/7/2020.
12. Back in January 2020 there was work on the design of the V1E shape for the TF coils … there was some consideration of different sizes. File [compare\_V1D\_V1E.pptx](file:///C:\Users\sscott\Documents\ripple\ASCOT_handover\compare_v1D_v1E.pptx) compares the relative ripple for various proposed designs, based on a simple expression for the ripple.
13. ASCOT calculations of alpha loss for the SPARC V1D design are described in file [2019\_12-20\_TF\_ripple.pptx](file:///C:\Users\sscott\Documents\ripple\ASCOT_handover\2019_12_20_TF_ripple.pptx). This was presented at a SPARC physics review in December 2019.
14. In late 2019, engineers raised a concern that the large out-of-plane forces on the SPARC TF coils would cause them to deform, leading to higher ripple. I computed the ripple and found it to be negligible, so long as all the coils deform by the same amount. This work is described in files [2019\_1124\_ripple\_twist\_02.pptx](file:///C:\Users\sscott\Documents\ripple\ASCOT_handover\2019_1124_ripple_twist_02.pptx) and [2019\_1124\_ripple\_twist\_02.pdf](file:///C:\Users\sscott\Documents\ripple\ASCOT_handover\2019_1124_ripple_twist_02.pdf).
15. A memo that describes how I compute the perpendicular and parallel RF-tail and ASCOT marker weights from TRANSP output is discussed in memo [rf\_tail\_04.pdf](file:///C:\Users\sscott\Documents\ripple\ASCOT_handover\rf_tail_04.pdf), dated 11/15/2019.
16. The effect of the time step on alpha orbits (computed with gyro-orbit included) is discussed in [2019\_ascot\_GO\_timing\_study.pptx](file:///C:\Users\sscott\Documents\ripple\ASCOT_handover\2019_ascot_GO_timing_study_01.pptx).
17. Computed alpha losses for the SPARC V1C design are presented in files [2019\_sparc\_ripple\_surface\_heating\_01.pptx](file:///C:\Users\sscott\Documents\ripple\ASCOT_handover\2019_sparc_ripple_surface_heating_01.pptx) and [2019\_sparc\_ripple\_surface\_heating\_01.pdf](file:///C:\Users\sscott\Documents\ripple\ASCOT_handover\2019_sparc_ripple_surface_heating_01.pdf).
18. My 2019 contributed oral talk discussed alpha losses in the SPARC V0 and V1C designs, in file [2019\_aps\_ripple\_final.pptx](file:///C:\Users\sscott\Documents\ripple\ASCOT_handover\2019_aps_scott_ripple_final.pdf).
19. A very early time-step study (single slide) is in file [2019\_time-step-study.pdf](file:///C:\Users\sscott\Documents\ripple\ASCOT_handover\2019_time-step-study.pdf), 9/25/2019.
20. My first (collisionless) alpha-orbit simulations with SPARC ripple are presented in file [2019\_ripple-results-02.pdf](file:///C:\Users\sscott\Documents\ripple\ASCOT_handover\2019_ripple-results-02.pdf), 9/17/2019.
21. Initial calculations of the 2D ripple field in SPARC, and a resolution of a problem in computing the field that arises because the variable !radeg in IDL is off by 5.e-8, are discussed in file 2019\_ripple-presentation\_02.pdf, 9/3/2020.
22. An informal search of the literature on ripple in tokamaks is in file [2019\_07\_29\_sparc\_ripple\_03.pdf](file:///C:\Users\sscott\Documents\ripple\ASCOT_handover\2019_07_29_sparc_ripple_03.pdf).
23. Before I had the capability to run ASCOT, I tried to estimate the ripple losses in SPARC from published work on calculations of alpha loss in ITER. This work is described in file [ascot-ripple-iter-02.pptx](file:///C:\Users\sscott\Documents\ripple\ASCOT_handover\ascot-ripple-iter-02.pptx).

[back\_to\_top](#back_to_top)