**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.