Using Matlab

Overview

There are three ways to run Matlab:

1. Using the Matlab integration with Python in a Jupyter notebook.
2. Graphical mode using the Desktops of OnDemand
3. The command line version using modules. This is the most common as you will typically submit a job using PBS.

Like any other application, Matlab has to be loaded as a module before you can use it. To see all the installed versions of the Matlab use the command `module avail matlab`.

The typical procedure for performing calculations on UA HPC systems is to run your program non-interactively on compute nodes. The easiest way to run MATLAB non-interactively is to use input/output redirection. This method uses Linux operators `<` and `>` to point Matlab to the input file and tell where to write the output (see the example script). The other method is to invoke Matlab from the PBS script and execute specified statement using `-r` option. For details please refer to the manual page of `matlab` command:

https://www.mathworks.com/help/matlab/ref/matlablinux.html

```bash
#!/bin/bash
#PBS -N job_name
#PBS -W group_list=group_name
#PBS -q standard
#PBS -l select=1:ncpus=28:mem=168gb:pcmem=6gb
#PBS -l walltime=01:00:00
#PBS -l cput=28:00:00

cd $PBS_O_WORKDIR

module load matlab

matlab -nodisplay -nosplash < script_name.m > output.txt
```

The options `nodisplay` and `nosplash` in the example prevent Matlab from opening elements of GUI. To view the full list of options for `matlab` command and load the Matlab module and type `matlab -h` in Linux prompt, or use the link above to the manual page on MathWorks website.

Parallel Computing Toolbox

By default, Matlab PCT will dump files to `~/.matlab/MATLAB_VERSION`. This causes problems when multiple Matlab PCT jobs are running simultaneously. Users should always define the environment variable `MATLAB_PREFDIR` so each job uses a unique temporary folder. Files there will be cleaned after the job finishes. For example on Puma:

```bash
export MATLAB_PREFDIR=$(mktemp -d $SLURM_JOBTMP/matlab-XXXX)
```
Matlab on Ocelote

Matlab performs its own hardware discovery and it might try to access all the cores and the memory of the node even if the full node wasn’t allocated. That will result in scheduler killing the job. To prevent that on Ocelote the full node of 28 cores and 168GB of memory should be allocated to run a Matlab job. Puma can keep the job isolated due to “cgroups”.

Matlab on Puma

If you are trying to run Matlab in parallel interactively on Puma, you may encounter the following error:

```
>> Starting parallel pool (parpool) using the 'local' profile ...  
Error using parpool (line 149) 
You requested a minimum of <n> workers, but the cluster "local" has the NumWorkers property set to allow a maximum of 1 workers. To run a communicating job on more workers than this (up to a maximum of 512 for the Local cluster), increase the value of the NumWorkers property for the cluster. The default value of NumWorkers for a Local cluster is the number of physical cores on the local machine.
```

This is due to an interaction between SLURM and Matlab. To resolve this issue, when requesting <n> cores for your interactive job, you will need to set SLURM’s `--ntasks` directive to 1 and `--cpus-per-task` to the number of cores you need. For example:

```
$ srun --nodes=1 --ntasks=1 --cpus-per-task=6 --mem-per-cpu=4GB --time=04:00:00 --job-name=interactive --account=<GROUP> --partition=standard --pty bash -i
```