# PBS/PBS PRO and Matlab

## Submitting a Matlab Job to the Cluster

By default, when running Matlab, the program is running interactively on the Head or Login node. This should be kept to a minimum when developing programs, as it affects other users of the node. When you wish to execute jobs, especially jobs that run for a reasonable amount of time, the job should be submitted to execute on a queue of the "Cluster" in batch mode or run non-interactively.

If you execute large jobs on the "Head" node, this will slow down usability and will impact other users performance. The following guide provides details on how to submit a Matlab job to the HPC cluster. In order to submit a Matlab job to the cluster, you need to write a script file similar to the ones below. Lines beginning with "##" represents comments. Replace the username in the directory structure with your username (eg. userr) and the email address to your email address. Change the Matlab Script File Name to the name of the Matlab file that you want to be executed on the cluster. Note all "[...]" are required variables or definitions that require defining.

```
#!/bin/bash -l
#
#### job name
#PBS -N [Name of Job]
#
#### select resources
#PBS -l ncpus=[number of cpu's required, most likely 1]
#PBS -l mem=[amount of memory required]
#PBS -l walltime=[how long the job should run for - you may wish to remove
this line to use queue default]
#
#### redirect error & output files
#PBS -e /home/[username]/[location]/[input (standard out) file name]
#PBS -o /home/[username]/[location]/[output (standard out) file name]
#
#### mail Options
#PBS -m abe
#PBS -M [your email address]
#
#### select queue
#PBS -q [default or remove this line to use default]
#

#### load matlab module (setup environment)
module load [matlab or matlab/2016b]

#### change to current working directory
cd /home/[username]/[current working directory]

#### execute program
matlab -nodisplay -nodesktop -nosplash < [Matlab Script FileName.m]
```

## Real Example

```
#!/bin/bash -l
#
#### job name
#PBS -N Matlab-Job1
#
#### select resources
#PBS -l ncpus=1
#PBS -l mem=1g
#PBS -l walltime=100:00:00
#
#### redirect error & output files
#PBS -e /home/eng/userr/matlab/matlab-job1.err
#PBS -o /home/eng/userr/matlab/matlab-job1.out
#
#### select queue
#PBS -q long
#
##### Mail Options
#PBS -m abe
#PBS -M r.user@usq.edu.au
#

#### load matlab module (setup environment)
module load matlab

#### change to current working directory
cd /home/eng/userr/matlab

#### execute program
matlab -nodisplay -nodesktop -nosplash < matlab-job1.m
```

## Real Example 2

This example is based on the Matlab Parallel Computing Toolbox documentation. With Parallel Matlab, a single node is used, however with Distributed Matlab multiple nodes are used in the calculations.

The Matlab file wave.m is used by the PBS script below.

```
matlabpool open 3
parfor i=1:4096
A(i) = sin(i*2*pi/4096);
end
matlabpool close
plot(A)
print ('-r75','-djpeg','plot.jpg')
quit
```

The matlabpool command specifiles a pool of three additional processes to work on the parallel portions of the code. It is not counting the originating process executing wave.m so there are a total of four processes running in parallel for this job. Therefore, the PBS code running this example needs to ask for four processors on a single node.

The PBS code used to run wave.m:

```
#!/bin/csh -l
#PBS -S /bin/csh
#PBS -q standard
#PBS -l nodes=2:ppn=4
#PBS -m be
#PBS -M r.user@usq.edu.au

module load matlab
cd $PBS_O_WORKDIR
unsetenv DISPLAY
setenv MATLAB_PREFDIR /sandisk1/userr/matlabPar
setenv MCR_CACHE_ROOT /sandisk1/userr/my_matlab_mcr_cache
matlab -r wave -logfile matlab.log
```

**<u>Parallel/Dist Computing Toolbox Example</u>**

With distributed Matlab, Matlab takes care of PBS job submission without the need for a separate PBS command file. Though some matlab commands need to be set some variables so that Matlab runs properly.

The example below shows howto setup a distributed Matlab job. It uses two scripts, **parallel.m** used to setup Matlab and PBS environments, and the script **colsum.m** to do the Matlab work.

```
% set default options
% * REPLACE * username with your own loginid
remoteDataLocation = '/sandisk1/username';
clusterHost = 'usqhpc00.usq.edu.au';

% create parallel sched object
sched = findResource('scheduler', 'type', 'torque');
% set sched options
set(sched, 'ClusterMatlabRoot',  '/usr/local/matlab');
set(sched, 'HasSharedFilesystem', true);
set(sched, 'RshCommand', 'ssh');
set(sched, 'RcpCommand', 'scp');

% set PBS options
% Ask for a walltime limit of 4 hours
% * REPLACE * 4 with the number of hours you need
set(sched, 'ResourceTemplate', '-S /bin/sh -l walltime=4:00:00 ');

% select the required PBS queue
set(sched, 'SubmitArguments', '-q matlab');

% The following asks for 2 computers (nodes=2)
% and 2 proccessors per node (ppn=2), a total of 4 processors
% * REPLACE * with the number of nodes and processors per node you need
% In addition, require 2GB of memory for each process
% * REPLACE * with the memory you need
```

```
set(sched, 'ResourceTemplate', '-l nodes=2:ppn=2,pvmem=2gb');

% create a parallel matlab job
pjob = createParallelJob(sched);

% set the number of CPUs to use
% * REPLACE * with the number of CPU's you need
set(pjob, 'MaximumNumberOfWorkers', 4);
set(pjob, 'MinimumNumberOfWorkers', 4);

% sample matlab job follows.
% * REPLACE * with your own job (once you've tested this one)

% tell parallel job which matlab file to run
set(pjob, 'FileDependencies', {'colsum.m'});
t = createTask(pjob, @colsum, 1, {});

% displays variables for parallel job
get(pjob, 'FileDependencies');
get(pjob, 'JobData');
get(pjob, 'PathDependencies');
get(pjob, 'Tasks');

% submit PBS job
submit(pjob);

% get results from parallel job
waitForState(pjob);
results = getAllOutputArguments(pjob)
```

Matlab script **colsum.m** called from the script above.

```
function total_sum = colsum
if labindex == 1
   % Send magic square to other labs
   A = labBroadcast(1,magic(numlabs))
else
   % Receive broadcast on other labs
A = labBroadcast(1)
end

% Calculate sum of column identified by labindex for this lab
column_sum = sum(A(:,labindex))
% Calculate total sum by combining column sum from all labs
total_sum = gplus(column_sum)
```

Once parallel.m has been changed, it can be run as follows:

```
matlab -r parallel
```

The PBS command *qstat* will display the PBS jobs that have been submitted by Matlab.

## Executing script on the cluster

# PBS/PBS PRO and Matlab

The USQ HPC Cluster uses a job scheduler that allows you to schedule and run jobs on the various compute nodes. To submit a job, simply execute the command: *qsub [pbs_script_file]*. A handy command, to check if your job is running, queued or completed is by using the command qstat.

## Multithreading

MATLAB support for multithreaded computation is enabled by default in MATLAB versions available on the HPC. As mentioned in the MATLAB documentation, multithreading speeds up elementwise computations such as those done by the *sin* and *log* functions.

Consequently, using any of the available versions of MATLAB, it is quite likely that your scripts will be multithreaded. This has very important consequences for MATLAB jobs submitted to the HPC cluster. It would be very easy to assume your MATLAB scripts are not multithreaded and request a single cpu for your job, but if that job actually uses multiple threads then it will adversely affect other jobs on the cluster. **Therefore, care must be taken to ensure that your MATLAB script only uses the amount of resources requested in your PBS job script.**

In earlier versions of MATLAB it was possible to constrain the number of threads used by MATLAB with the maxNumCompThreads(N) function. However, this is no longer effective in recent versions of MATLAB. Therefore, the only options are: (a) reserve a whole compute node (ppn=8) and let MATLAB use all available cores; or (b) restrict MATLAB to one thread with the command-line flag -singleCompThread and reserve only one core in the job script (ppn=1).

| **Use Whole Compute Node** | **Use Single Core** |
|---|---|
| #!/bin/bash -l<br><br>#PBS -l nodes=1:ppn=8<br>#PBS -l mem=8gb<br>#PBS -l walltime=12:00:00<br><br>cd $PBS_O_WORKDIR<br><br>matlab -nodisplay -r amatlabscript | #!/bin/bash -l<br><br>#PBS -l nodes=1:ppn=1<br>#PBS -l mem=8gb<br>#PBS -l walltime=12:00:00<br><br>cd $PBS_O_WORKDIR<br><br>matlab -singleCompThread -nodisplay -r amatlabscript |

## References

1. Matlab Parallel Computing Toolbox
2. Getting Started with Parallel Computing Using MATLAB
3. MATLAB Parallel Computing Toolbox Tutorial