MATLAB

MATLAB is a technical computing environment for high performance numeric computation and visualization. MATLAB integrates numerical analysis, matrix computation, signal processing and graphics. MATLAB toolboxes are collections of algorithms that enhance MATLAB's functionality in domains such as signal and image processing, data analysis and statistics, mathematical modeling, etc.

Versions and Availability

You can use the command `module spider matlab` to get a list of available versions. Feel free to contact cluster team if you need other versions. MATLAB is proprietary software and can be used on the cluster for non-commercial, academic purposes only. As a free alternative to MATLAB you might consider GNU Octave, which is mostly compatible with MATLAB and provides most of the features as well.

Running MATLAB

As MATLAB by default utilizes all available CPU cores, excessive use of MATLAB on the login nodes would prevent other logged in users from using resources. The recommended way to work with MATLAB is to submit a job (interactive or batch) and start MATLAB from within the dedicated compute node assigned to you by the batch system.

To submit an interactive job, you can use the command (you may adjust the numbers per your need).

```
  login03~$ salloc --nodes=1 --ntasks=12 --mem-per-cpu=2G --time=02:00:00 --
x11
```

The `--x11` flag enables X11 forwarding on the compute node if you need to use the graphical MATLAB user interface.

Once you are assigned a compute node, you can run MATLAB interactively by loading the MATLAB module. To load the default version of MATLAB module, use `module load MATLAB`. To select a particular MATLAB version, use `module load MATLAB/version`. For example, to load MATLAB version 2017a, use `module load MATLAB/2017a`.

To start MATLAB interacively with graphical user interface (GUI), after having loaded MATLAB module, type the command.

```
  smp-n010~$ matlab
```

This requires the `--x11` flag for the `salloc` command as well as X11 forwarding enabled for your SSH client or using X2GO for logging in onto the cluster system. MATLAB GUI can also be launched from a web browser via Remote Desktop Session. The following command will start an interactive, non-GUI version of MATLAB.

```
smp-n010~$ matlab -nodisplay -nosplash
```

Type `matlab -h` for a complete list of command line options.

In order to run MATLAB non-interactively via the batch system, you will require a batch submission script. Below is an example batch script (`matlab-job-serial.sh`) for a serial run that will execute MATLAB script (`hello.m`) in a single thread.

matlab-job-serial.sh

```
#!/bin/bash -l
#SBATCH --job-name=matlab_serial
#SBATCH --mail-user=user@uni-hannover.de
#SBATCH --mail-type=BEGIN,END,FAIL
#SBATCH --nodes=1
#SBATCH --ntasks=1
#SBATCH --mem-per-cpu=4G
#SBATCH --time=00:20:00
#SBATCH --output matlab_serial-job_%j.out

# Compute node the job ran on
echo "Job ran on:" $HOSTNAME

# Load modules
module load MATLAB/2017a

# Change to work dir:
cd $SLURM_SUBMIT_DIR

# Log file name
LOGFILE=$(echo $SLURM_JOB_ID | cut -d"." -f1).log

# The program to run
matlab -nodesktop -nosplash < hello.m > $LOGFILE 2>&1
```

Example MATLAB script, `hello.m`:

hello.m

```
% Example MATLAB script for Hello World
disp 'Hello World'
exit
% end of example file
```

To run `hello.m` via the batch system, submit the `matlab-job-serial.sh` file with the following command:

```
login03~$ sbatch matlab-job-serial.sh
```

```
Submitted batch job 736583
```

Output from the running of the MATLAB script will be saved in the `$LOGFILE` file, which in this case expands to `matlab_serial-job_736583.out`.

Parallel Processing in MATLAB

Please note: MATLAB parallel computing across multiple compute nodes is not supported by the cluster system at the moment.

MATLAB supports both implicit multithreding as well as explicit parallelism provided by the Parallel Computing Toolbox (PCT) which requires specific commands in your MATLAB code in order to create threads.

Implicit multithreading allows some functions in MATLAB, particularly linear algebra and numerical routines such as `fft`, `eig`, `svd`, etc, to distribute the workload between cores of the node that your job is running on and thus run faster than on a single core. By default, all of the current versions of MATLAB available on the cluster have multithreading enabled. **A single MATLAB session will run as many threads as there are cores on a compute node reserved for your job by the batch system.** For example, if you request `--ntasks=4`, your MATLAB session will spawn four threads.

matlab-job-multithread.sh

```
#!/bin/bash -l
#SBATCH --job-name=matlab_multithread
#SBATCH --mail-user=user@uni-hannover.de
#SBATCH --mail-type=BEGIN,END,FAIL
#SBATCH --nodes=1
#SBATCH --ntasks=4
#SBATCH --mem-per-cpu=4G
#SBATCH --time=00:20:00
#SBATCH --output matlab_multithread-job_%j.out

# Compute node the job ran on
echo "Job ran on:" $HOSTNAME

# Load modules
module load MATLAB/2017a

# Change to work dir:
cd $SLURM_SUBMIT_DIR

# Log file name
LOGFILE=$(echo $SLURM_JOB_ID | cut -d"." -f1).log

# The program to run
matlab -nodesktop -nosplash < hello.m > $LOGFILE 2>&1
```

However, if you want to disable multithreading you may either request `--ntasks=1` (see an example

job script above) or add the option `-singleCompThread` when running MATLAB.

Using the Parallel Computing Toolbox

MATLAB Parallel Computing Toolbox (PCT) (parallel for-loops, special array types, etc.) lets you make explicit use of multicore processors, GPUs and clusters by executing applications on workers (MATLAB computational engines) that run locally. At the moment, the cluster system does not support parallel processing across multiple nodes (MATLAB Distributed Computing Server). As such, parallel MATLAB jobs are limited to a single compute node with the “local” pool through use of the MATLAB PCT.

Specific care must be taken when running multiple MATLAB PCT jobs. When you submit multiple jobs that are all using MATLAB PCT for parallelization, all of the jobs will attempt to use the same defaut location for storing information about the MATLAB pools that are in use, thereby creating a race condition where one job modifies the files that were put in place by another. The solution is to have each of your jobs that will use the PCT set a unique location for storing job information. An example batch script `matlab-job-pct.sh` is shown below.

matlab-job-pct.sh

```
#!/bin/bash -l
#SBATCH --job-name=matlab_multithread_pct
#SBATCH --mail-user=user@uni-hannover.de
#SBATCH --mail-type=BEGIN,END,FAIL
#SBATCH --nodes=1
#SBATCH --ntasks=12
#SBATCH --mem-per-cpu=3G
#SBATCH --time=00:40:00
#SBATCH --output matlab_multithread-job_%j.out

# Compute node the job ran on
echo "Job ran on:" $HOSTNAME

# Load modules
module load MATLAB/2017a

# Change to work dir:
cd $SLURM_SUBMIT_DIR

# Log file name
LOGFILE=$(echo $SLURM_JOB_ID | cut -d"." -f1).log

# The program to run
matlab -nodesktop -nosplash < pi_parallel.m > $LOGFILE 2>&1
```

And the corresponding MATLAB script `pi_parallel.m`, which in addition starts the correct number of parallel MATLAB workers depending on the requested cores.

pi_parallel.m

```
% create a local cluster object
pc = parcluster('local')

% explicitly set the JobStorageLocation to
% the temp directory that is unique to each cluster job
pc.JobStorageLocation = getenv('TMPDIR')

% start the matlabpool with maximum available workers
parpool (pc, str2num(getenv('SLURM_CPUS_ON_NODE')))

R = 1
darts = 1e7
count = 0
tic
parfor i = 1:darts
  x = R*rand(1)
  y = R*rand(1)
  if x^2   y^2 <= R^2
    count = count   1
  end
end

myPI = 4*count/darts
T = toc
fprintf('The computed value of pi is %8.7f\n',myPI)
fprintf('The parallel Monte-Carlo method is executed in %8.2f
seconds\n', T)
delete(gcp)
exit
```

For further details on MATLAB PCT refer to the online documentation.

Build MEX File

See e.g. online documentation for details on how to build mex files. This section is a straight transscript of that website adapted to fit the cluster system's module system. First, get hold of the example file `timestwo.c` which comes with MATLAB. This can be done from within MATLAB.

```
copyfile(fullfile(matlabroot,'extern','examples','refbook','timestwo.c'),'.'
,'f')
```

Each MATLAB version needs a specific version of GCC in order to build mex files. This is the crucial part. See MATLAB documentation for details.

```
$ module load MATLAB/2018a
$ module load GCC/6.3.0-2.27
$ ls
```

```
timestwo.c
$ mex timestwo.c
Building with 'gcc'.
MEX completed successfully.
$ ls
timestwo.c timestwo.mexa64
```

Notice the file `timestwo.mexa64` was created. You can now use it like a function.

```
$ matlab -nodesktop -nosplash>> timestwo(6)

ans =

    12
```

Toolboxes and Features

On the cluster system you can use the university's MATLAB campus licence. Please see for details and a list of available toolboxes and features.

Further Reading

- MATLAB online documentation
- MATLAB Parallel Computing Toolbox