Hyak MATLAB programming

Below abc is your userid and xyz is your group name.

For interactive use of matlab:

```
ssh -X abc@mox.hyak.uw.edu
```

```
srun -p xyz -A xyz --nodes=1 --ntasks-per-node=28  --time=2:00:00 --mem=100G --pty /bin/bash
```

Below command shows you the available matlab versions on hyak:

```
module avail matlab
```

Use below command to load matlab and start the GUI

export TZ="America/Los_Angeles"

```
module load matlab_2018a
```

```
matlab
```

After the above step, you can continue with your matlab commands at the matlab command line.

For batch use of matlab:

Include below lines in your slurm script. (Do not use -nojvm option of matlab. The -nojvm option prevents usage of parpool for matlab parallel computing.) Here the file name of the matab script is myfunction.m.

export TZ="America/Los_Angeles"

```
module load matlab_2018a
matlab -nosplash -nodisplay -r myfunction
```

Environment variables in Matlab:

Inside your matlab script, you can access environment variables. For example, to get the value of the environment variable SLURM_ARRAY_TASK_ID use below lines:

taskIDstring = getenv('SLURM_ARRAY_TASK_ID');

taskIDnumber = str2num(taskIDstring);

Matlab toolboxes on hyak:

Obtain an interactive node as shown above. Then type below commnds to get a matlab prompt.

export TZ="America/Los_Angeles"

```
module load matlab_2018a
matlab -nosplash -nodisplay
```
At the matlab prompt enter below command to find the list of toolboxes:

```
>>ver
```

We have a large number of toolboxes. Here are some popular toolboxes:

Image Processing Toolbox

Optimization Toolbox

Parallel Computing Toolbox

Signal Processing Toolbox

Statistics and Machine Learning Toolbox

Matlab parallel programming on hyak:

Do not use the build node for matlab parallel programming. The build node is for single core use. Hence, for the build node, the matlab command "feature ('numcores')" will return 1.

For interactive use:

At the matlab prompt enter below command to start the parallel pool

```
>> numCores = feature('numcores');

>> parpool(numCores)
```

Now you can use parfor etc.

(If your run parpool without giving the number of cores, then the default number of cores is 12. It is not the number of cores on the machine.

numCores above is the number of cores allocated to your job by slurm)

For scripting:

Below xyz is your group name and abc is your userid.

Add below lines to your slurm script before the matlab command. This creates a temporary directory which Matlab will use to store its cluster information.

```
mkdir -p /gscratch/xyz/abc/$SLURM_JOB_ID
```

The file trypar.m contains below code. A semicolon after a matlab command means that matlab will not print the output of that command. (For the most common type of nodes on mox, numcores=28. For mox nodes from 2019 or later, numcores=32. For latest mox nodes numcores=40. For ikt nodes, numcores=16.)

```
% create a local cluster object

myCluster = parcluster('local') ;

% Set the JobStorageLocation to the temporary directory that was created in your slurm script

myCluster.JobStorageLocation = strcat('/gscratch/xyz/abc/', getenv('SLURM_JOB_ID')) ;

numCores = feature('numcores');

parpool(numCores);

parfor i=1:1000000
A(i)=i*i;
end
A
quit
```

You can run trypar.m in parallel at the prompt of an hyak interactive node:

```
export TZ="America/Los_Angeles"

module load matlab_2018a
matlab -nodisplay -nosplash -r trypar
```

You can run trypar.m in parallel using below lines in your slurm script. Here xyz is the name of your group, abc is your userid, mydir is the directory which contains trypar.m and trypar.pbs.

```
export TZ="America/Los_Angeles"
module load matlab_2018a
matlab -nodisplay -nosplash -r trypar
```

Matlab STDOUT, STDERR, STDIN

In fprintf( ), fileID=1 is for printing to STDOUT and fileID=2 is for printing to STDERR

See below for more information on fprintf():

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

See below for more information on disp():

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

Use below to get data from STDIN

```
x = input('', 's')
```

See below for more information on input():

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

Matlab Code Performance:

https://www.mathworks.com/help/matlab/code-performance.html

Matlab Memory Usage:

https://www.mathworks.com/help/matlab/memory.html

Matlab Java Memory Usage:

https://www.mathworks.com/help/matlab/matlab_external/java-heap-memory-preferences.html

======== Ignore below text. It is here for historical reasons. ===========

```
myCluster.NumWorkers = 28;
saveProfile(myCluster);
```