Installing and Running 21cmFAST and 21CMMC on Compute Canada

Jade Ducharme

July 2021

1 Package Introduction

21cmFAST is a Python package which allows us to simulate various fields in the early Universe (it is especially useful for studying the epoch of reionization, a period for which we don't have much real data). It generates density, velocity, halo, ionization, and spin temperature fields among many other useful applications. The user can specify a set of initial conditions (parameters) and easily study the effect of varying these parameters.

21CMMC is an extension of **21cmFAST** which allows us to perform MCMCs on the various fields presented above. The MCMC sampler is designed to fit the set of parameters used by **21cmFAST**. Additionally, **21CMMC** lets the user write their own likelihood functions, making it an incredibly extensible package allowing for a variety of useful applications.

However, the caveat is that both **21cmFAST** and **21CMMC** are incredibly memory-greedy and the simplest code can run for hours on a desktop computer. This is why it is of interest to install and run these packages in a Compute Canada server, since their systems are much more powerful than most ordinary computers. I use the Cedar server in particular, although the installation steps should be similar for other Compute Canada systems.

2 21cmFAST: Installation Step-by-step

First, you will want to log in to a Cedar session.

```
$ ssh <username>@cedar.computecanada.ca
```

Then, you will need to clone the official **21cmFAST** GitHub repo in your home folder. This is done by first copying the git address of the **21cmFAST** repo and typing the following command:

```
$ git clone https://github.com/21cmfast/21cmFAST.git
```

This will create a folder named **21cmFAST** containing all the source code for this package. Next, you will need to load the following modules:

```
$ module load python
$ module load gsl
$ module load fftw-mpi
```

At this point, you can save these modules, otherwise they will need to be manually re-loaded every time you start a new session.

```
$ module save
```

By default, the modules are saved under the name "default", and accessible in different sessions through the use of the "module restore default" command.

The next step will be to create a virtual environment housing all the Python modules that **21cmFAST** and **21CMMC** are dependent on. You can start by choosing a name for the environment. I will simply name mine "myenv", but feel free to adopt a more inspired name:

```
$ virtualenv --no-download myenv
```

This command created a new hidden folder called **myenv** where the necessary packages are stored. However, since it's a hidden folder, you won't be able to see it using the standard "ls" command. You'll need to use "ls -a", which lists *all* folders and files. Now go ahead and activate your new virtual environment:

```
$ source myenv/bin/activate
```

At this point you need to install all the dependencies. Since you've activated the virtual environment, these are automatically installed in the myenv folder and forever accessible through the "source myenv/bin/activate" command. Now for the specifics:

```
$ pip install numpy scipy jupyter matplotlib astropy h5py cffi pyyaml click --no-index
$ pip install cached_property
```

Finally, the last step is to navigate into the **21cmFAST** folder and launch the installation process!

```
$ cd 21cmFAST
$ pip install -e .  --no-deps
```

3 21CMMC: Installation Step-by-step

After following the steps in Section 2, you are ready to move on to **21CMMC**. The process will be very similar. First, you'll need to navigate back into your home folder.

```
$ cd ~
```

Then, you'll need to clone the official **21CMMC** GitHub repo into your home folder.

```
$ git clone https://github.com/21cmfast/21CMMC.git
```

If you aren't picking up exactly where we left off in Section 2, now would be a good time to restore your modules and reactivate your virtual environment (if you've been following along, ignore the following commands):

```
$ module restore default
$ source myenv/bin/activate
```

It is important to restore the modules *before* activating the virtual environment, since many of the Python packages are dependent on those modules. Then, you'll need to install a few more Python packages to get **21CMMC** running smoothly.

```
$ pip install corner cosmoHammer powerbox pymultinest
$ pip install emcee==2.2.1
```

It is important to specify the version for emcee because, as of the time of writing this text (July 17, 2021), 21CMMC does *not* support emcee version 3 or higher. If this gets fixed, please let me know so I can edit the document! Next, let's navigate to our **21CMMC** folder and launch the installation!

```
$ cd 21CMMC
$ pip install -e .  --no-deps
```

4 Final Notes

Remember that the whole point of installing **21cmFAST** and **21CMMC** on Compute Canada is because they use up a LOT of RAM (and storage too due to the cache). So when you write your batch scripts before submitting jobs, don't be afraid to ask for a LOT of memory (upwards of 100GB is not unusual). Cedar has a few nodes with up to 3TB of RAM. The number of nodes and their corresponding memory allocations on Cedar can be found here. Of course, the more memory you ask for, the longer you'll be stuck in the queue!

Lastly, don't forget to load the modules, *then* reactivate the virtual environment every time you fire up a new Cedar session. Otherwise, your jobs won't run!