BERKELEY LAB

AMReX Microelectronics Documentation for New Users

Teo Lara, Nia Maheshwari

Introduction

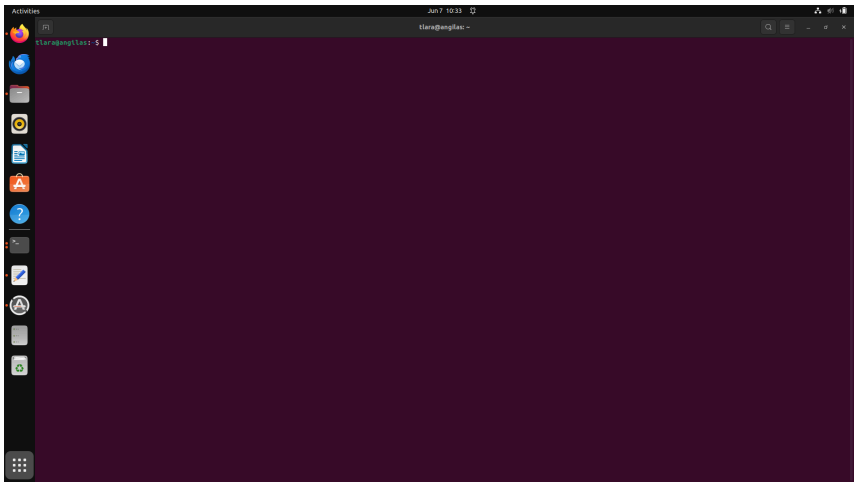
This is a step-by-step installation guide for installing the ELEQTRONeX program used to model carbon nanotubes. This guide will also go through the installation process for a few other programs such as AMReX and HYPRE that are needed to install ELEQTRONeX; a short guide for installing, building, and running FerroX code and the Visit software is also included. This guide is written for a Linux computer running Ubuntu 22, installation on other systems will have a few differences; fortunately, computers at LBNL run Linux by default, so this guide should be applicable.

Every step in this process is written for new users, fully comprehensible for those with very little or no experience using Linux, more experienced users should feel free skipping portions that they already understand.

Basic Console Commands

Introduction to the Terminal

To open a terminal, navigate to the applications search bar and find the “terminal option” (icon is shown to the right). After opening, the page will be similar to the one below. Instead of **`tlara@angilas`**, the local username will be present.



The terminal may look intimidating for new users, but in reality it is just a shortcut for moving between folders and giving commands to the computer. Commands are always expected inside the terminal, to check your current location, use the command `pwd`. **Note that the terminal is case- and space- sensitive. Note also that common text commands like ctrl+c, ctrl+z, and others will not work on a terminal (nor in the Vim text editor).**

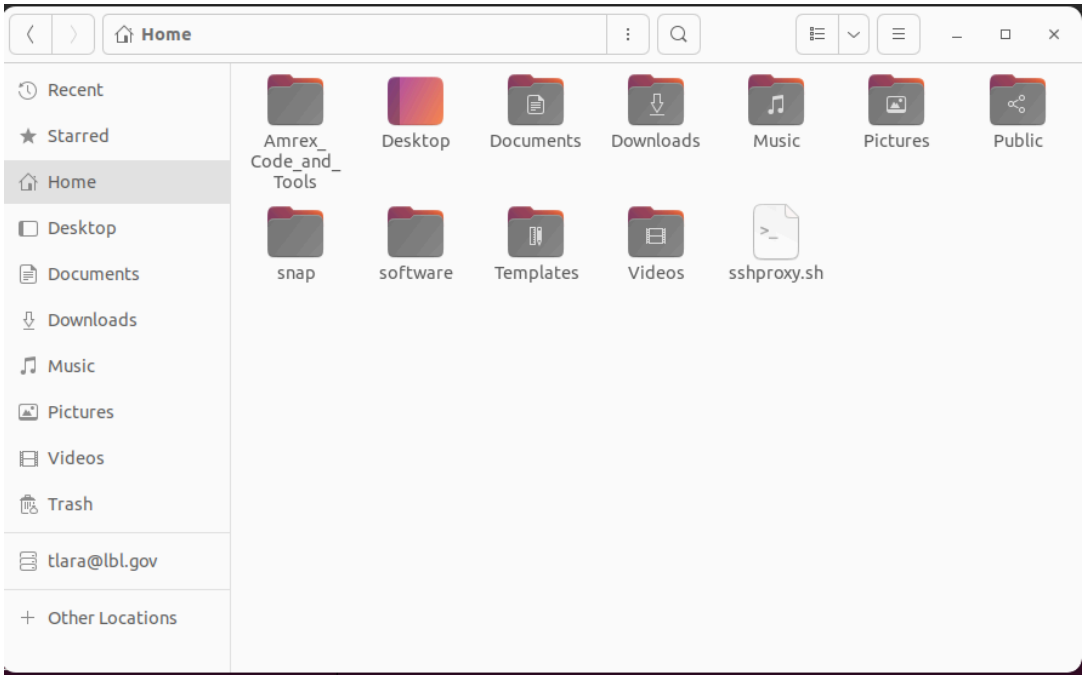
```
Unset
tlara@angilas:~$ pwd
/home/tlara
```

Folder Management

The current location is the home directory, in the `tlara` folder. To list the folders that exist within this directory, type `ls`

```
Unset
tlara@angilas:~$ ls
Amrex_Code_and_Tools  Desktop  Documents  Downloads  Music  Pictures  Public
snap  software  sshproxy.sh  Templates  Videos
```

The listed directories and files are the same as those that can be accessed in the file manager:



In the file manager, a folder can be opened by clicking on it, in the terminal window, the `cd` command is used followed by the terminal name. For example, on this computer, the “software” folder can be opened using `cd software`

```
C/C++
tlara@angilas:~$ cd software/
tlara@angilas:~/software$
```

Notice how commands are now entered into a new prompt: **tlara@angilas:~/software$** indicating that the new location is the `software folder`. To view the contents of this folder, `ls` can once again be used.

```
C/C++
tlara@angilas:~$ cd software/
tlara@angilas:~/software$ ls
3.4.1  current  INSTALL_NOTES_3_4_1.txt  visit3_4_1.linux-x86_64
visit-install3_4_1
bin    data   visit0001.png             visit3_4_1.linux-x86_64-ubuntu22.tar.gz
```

Indeed, the contents of this folder are different from those in the previous directory, indicating that the location has changed.
To move out of this directory to the parent directory, use `cd ..`

```
C/C++
tlara@angilas:~/software$ cd ..
tlara@angilas:~$
```

A few other helpful commands for moving between folders are:

`cd ../..` *Move up 2 folders (add one* `/..` *for every directory farther back)*
`cd ~` *jump from current location to* `home` *directory*
`cd` *return to previous directory after moving up or in*

It is also important to be able to create new directories. This can be done with the `mkdir` command; in the `home` directory, a new folder will be created, named **`guide`**.

```
C/C++
tlara@angilas:~$ mkdir guide
tlara@angilas:~$ ls
Amrex_Code_and_Tools  Desktop  Documents  Downloads  guide  Music  Pictures
Public  snap  software  sshproxy.sh  Templates  Videos
```

Notice the **`guide`** folder has been created. All following commands will be conducted in this folder. To remove a folder, one can use `rm -r` followed by the name of the folder. `rm` is used to remove files, adding `-r` flag removes a folder and everything within it.
Finally, it is also important to mention that the up and down arrows can be used to access previous terminal commands.

ELEQTRONeX Installation Process

Installing AMReX

AMReX is a software developed at LBNL to solve partial differential equations, the ELEQTRONeX program is built using the AMReX library. First, AMReX is cloned from Github and placed in the **`guide`** folder that was previously created using the `git clone` command. `git clone` followed by the url of the Github page will copy the Github files onto the local computer. The url for AMReX is: https://github.com/AMReX-Codes/amrex.git.

```
C/C++
tlara@angilas:~$ cd guide
```

```
tlara@angilas:~/guide$ git clone https://github.com/AMReX-Codes/amrex.git
Cloning into 'amrex'...
remote: Enumerating objects: 196913, done.
remote: Counting objects: 100% (1849/1849), done.
remote: Compressing objects: 100% (1013/1013), done.
remote: Total 196913 (delta 1107), reused 1411 (delta 830), pack-reused 195064
Receiving objects: 100% (196913/196913), 54.69 MiB | 19.75 MiB/s, done.
Resolving deltas: 100% (146683/146683), done.
tlara@angilas:~/guide$ ls
amrex
```

The **amrex** folder is not inside the `guide` directory. From here, the AMReX code will be configured and installed, using the `./configure`, `make`, and `make install` commands from within the **amrex** folder. Note that the "`make`" command will take quite some time.

```
C/C++
      Output of below commands is supressed for clarity
tlara@angilas:~/guide$ cd amrex/
tlara@angilas:~/guide/amrex$ ./configure
tlara@angilas:~/guide/amrex$ make
tlara@angilas:~/guide/amrex$ make install
```

A bit on what is going on: This process is a little like unzipping folders. The folder downloaded from Github does not contain fully formed code, the `make` command runs the `GNUmakefile` within the **amrex** directory, this builds the actual executable files that can be run.

Installing Hypre Library

Note that the Hypre Library is not needed for local installation, only for running with CUDA on an HPC. Skip section if working on local computer

Next, the `git clone` is repeated with the HYPRE library. This is a library that helps solve large, sparse systems of equations. Returning to the **guide** directory, using `cd ..`, the exact same process is repeated, except with the HYPRE url: https://github.com/hypre-space/hypre.git.

```
C/C++
tlara@angilas:~/guide/amrex$ cd ..
tlara@angilas:~/guide$ ls
amrex
tlara@angilas:~/guide$ git clone https://github.com/hypre-space/hypre.git
```

```
Cloning into 'hypre'...
remote: Enumerating objects: 134924, done.
remote: Counting objects: 100% (249/249), done.
remote: Compressing objects: 100% (122/122), done.
remote: Total 134924 (delta 146), reused 202 (delta 127), pack-reused 134675
Receiving objects: 100% (134924/134924), 217.77 MiB | 44.84 MiB/s, done.
Resolving deltas: 100% (111236/111236), done.
tlara@angilas:~/guide$ ls
amrex  hypre
```

From the **hypre/src** directory, the `./configure`, `make`, and `make install` commands are run again.

```
C/C++
      Output of below commands is supressed for clarity
tlara@angilas:~/guide$ cd hypre/src/
tlara@angilas:~/guide/hypre/src$ ./configure
tlara@angilas:~/guide/hypre/src$ make
tlara@angilas:~/guide/hypre/src$ make install
```

To be able to call the HYPRE library from other programs, an environment variable is created. This tells the computer where to find the HYPRE directory

```
C/C++
tlara@angilas:~/guide/hypre/src$ export
HYPRE_DIR=/home/tlara/guide/hypre/src/hypre
```

The **/home/tlara/** directory should be replaced with the local directory containing the **guide** folder. To verify the new variable, the `echo` command can be used:

```
C/C++
tlara@angilas:~/guide/hypre/src$ echo $HYPRE_DIR
/home/tlara/guide/hypre/src/hypre
```

Note that this method will last until the machine is turned off, to permanently add an environment variable, the `baschrc` file should be modified, instructions for this can be found in the appendix.

Installing OpenMPI Library

One final library, the “OpenMPI” library also has to be added to allow for parallelization. This is not done via make commands but rather through “sudo” commands. First, the gcc compiler and libraries are updated using the “sudo apt install build-essential” command. Here, a user’s password needs to be written before proceeding. **By default interns do not have access to sudo commands and this step will not work, a faculty member should contact IT to give access.**

```
C/C++
tlara@angilas:~/guide$ sudo apt install build-essential
[sudo] password for tlara:
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
build-essential is already the newest version (12.9ubuntu3).
The following packages were automatically installed and are no longer required:
  libpython2-stdlib libpython2.7-minimal libpython2.7-stdlib python-six python2
python2-minimal python2.7 python2.7-minimal
Use 'sudo apt autoremove' to remove them.
0 upgraded, 0 newly installed, 0 to remove and 122 not upgraded.
```

From here, the OpenMPI library is installed using “tlara@angilas:~/guide$ sudo apt-get install openmpi-bin openmpi-doc libopenmpi-dev".

```
C/C++
      Output of below commands is supressed for clarity
tlara@angilas:~/guide$ sudo apt-get install openmpi-bin openmpi-doc
libopenmpi-dev
```

Installing ELEQTRONeX

With the required libraries, the ELEQTRONeX program can be installed. From the “guides” directory, the “git clone” command is run again using the ELEQTRONeX url: https://github.com/AMReX-Microelectronics/ELEQTRONeX.git.

```
C/C++
tlara@angilas:~/guide$ git clone
https://github.com/AMReX-Microelectronics/ELEQTRONeX.git
Cloning into 'ELEQTRONeX'...
```

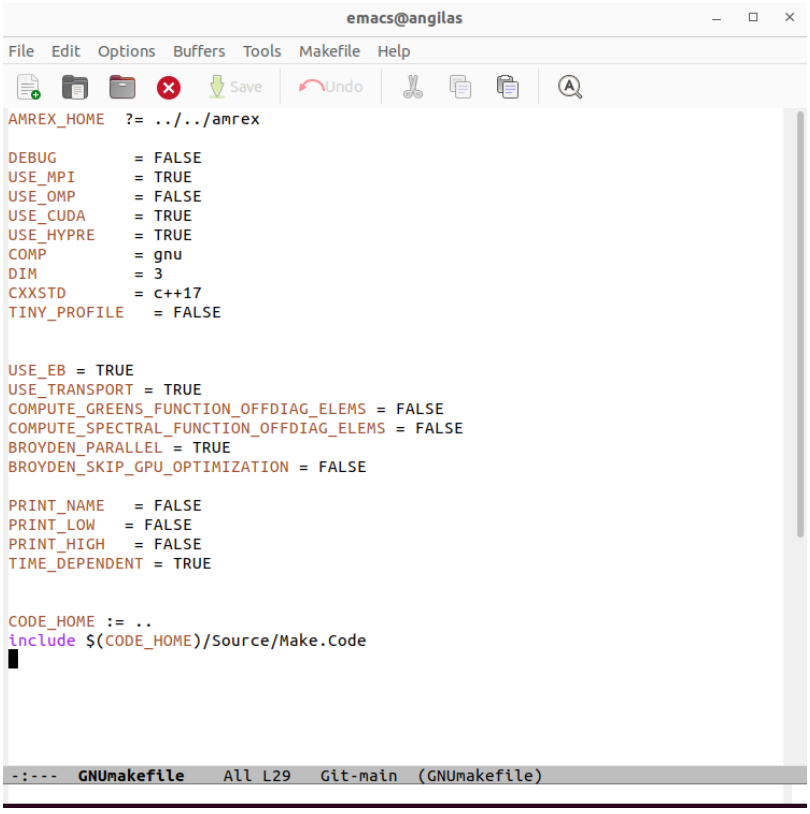
```
remote: Enumerating objects: 4343, done.
remote: Counting objects: 100% (213/213), done.
remote: Compressing objects: 100% (136/136), done.
remote: Total 4343 (delta 98), reused 109 (delta 75), pack-reused 4130
Receiving objects: 100% (4343/4343), 17.19 MiB | 33.78 MiB/s, done.
Resolving deltas: 100% (3058/3058), done.
tlara@angilas:~/guide$ ls
amrex  ELEQTRONeX  hypre
```

Before running the makefile, it should be edited to suit the working conditions. The makefile can be edited using several editors, in this tutorial, the emacs editor is used. Using the “emacs” command, a new window, featuring the text of the makefile, will be displayed.

C/C++

```
tlara@angilas:~/guide$ cd ELEQTRONeX/Exec/
tlara@angilas:~/guide/ELEQTRONeX/Exec$ emacs GNUmakefile
```

The emacs window will look like this:



If working locally, the value of USE_CUDA should be changed from TRUE to FALSE. This will tell the computer not to use a GPU. Locally, USE_HYPRE should also be changed to FALSE. Hypre does not work locally. To save the change, press the ctrl key, followed by the letter “x,” then the ctrl key and the letter “s”. To exit the editor, press the ctrl key and “x” followed by the ctrl key and “c.” Next, make the bash file the source using “source.”

```
C/C++
source ~/.bashrc
```

Now, the ELEQTRONeX program can be compiled using `make -j4`.

```
C/C++
      Output of below commands is supressed for clarity
tlara@angilas:~/guide/ELEQTRONeX/Exec$ make -j4
```

First Simulation Tutorial

Running a Basic Simulation

In this section, a basic test simulation is run and the output is visualized using the visIt software. To run a test simulation, navigate to the “/ELEQTRONex/Exec” folder and run the the executable followed by the input filename, for instance:

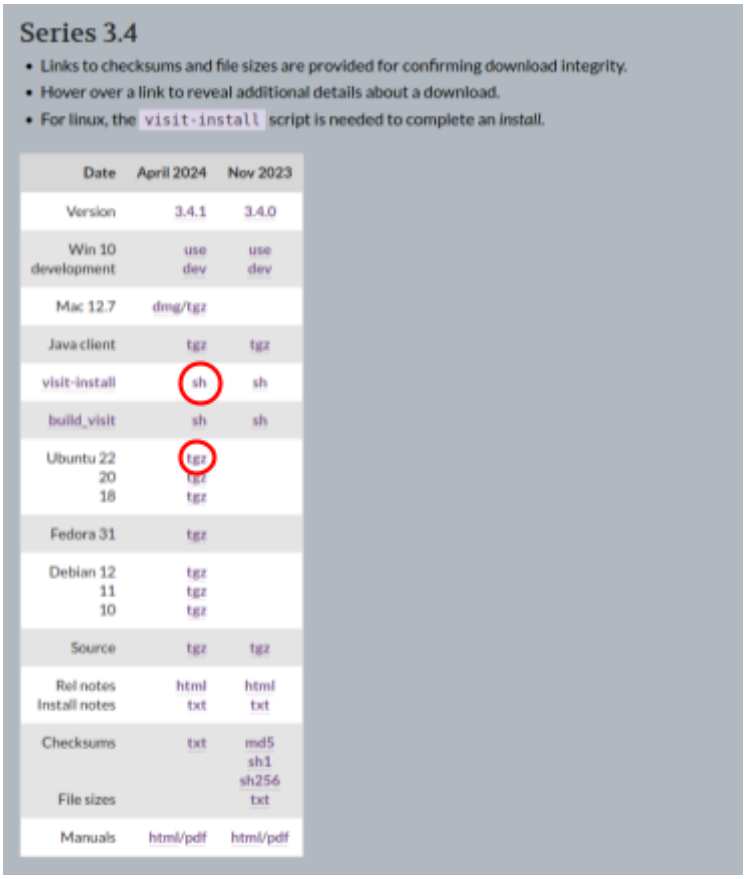
```
C/C++
tlara@angilas:~/guide/ELEQTRONeX/Exec$
./main3d.gnu.MPI.EB.TD.TRAN.BROYPRLL.SKIPGPU.ex ../input/negf/all_around_metal
```

The bolded portion of the above code is the executable, it will be different depending on what options were selected in the makefile. The non-bolded portion on the right is the input file, this case is the “all_around_metal” simulation, modeling a single nanotube. This code will create several “plt” output files in the “/ELEQTRONex/Exec/all_around_metal_test” folder.

```
C/C++
tlara@angilas:~/guide/ELEQTRONeX/Exec$ cd all_around_metal_test/
tlara@angilas:~/guide/ELEQTRONeX/Exec/all_around_metal_test$ ls
negf   plt0001  plt0003  plt0005  plt0007  plt0009  plt_init0
plt0000  plt0002  plt0004  plt0006  plt0008  plt0010
```

Data Visualization with VisIt

To visualize the output, the visIt program will be used. This is one of the two main softwares which can be utilized, the other being Amrvis. Information about Amrvis can be found here. visIt is not downloaded via Github, but from the visIt homepage: https://visit-dav.github.io/visit-website/releases-as-tables/. The necessary files for installation are the Ubuntu 22 tgz file and the visit-install sh file. Install these files and move them to a destination folder, such as the “software” folder for instance.



Series 3.4

- Links to checksums and file sizes are provided for confirming download integrity.
- Hover over a link to reveal additional details about a download.
- For linux, the `visit-install` script is needed to complete an install.

Date	April 2024	Nov 2023
Version	3.4.1	3.4.0
Win 10 development	use dev	use dev
Mac 12.7	dmg/tgz	
Java client	tgz	tgz
visit-install	sh	sh
build_visit	sh	sh
Ubuntu 22 20 18	tgz tgz tgz	
Fedora 31	tgz	
Debian 12 11 10	tgz tgz tgz	
Source	tgz	tgz
Rel notes Install notes	html txt	html txt
Checksums File sizes	txt	md5 sh1 sh256 txt
Manuals	html/pdf	html/pdf

Next, the files need to be extracted from their tar format. This is essentially equivalent to unzipping folders except with a distinct extension name. After navigating to the file location in the command line, the `tar -xvf` command followed by the file name is used to untar the file.

```
C/C++
      Output of below commands is supressed for clarity
tlara@angilas:~/software$ tar -xvf visit3_4_1.linux-x86_64-ubuntu22.tar.gz
tlara@angilas:~/software$ ls
visit3_4_1.linux-x86_64                visit-install3_4_1
```

```
visit3_4_1.linux-x86_64-ubuntu22.tar.gz
```

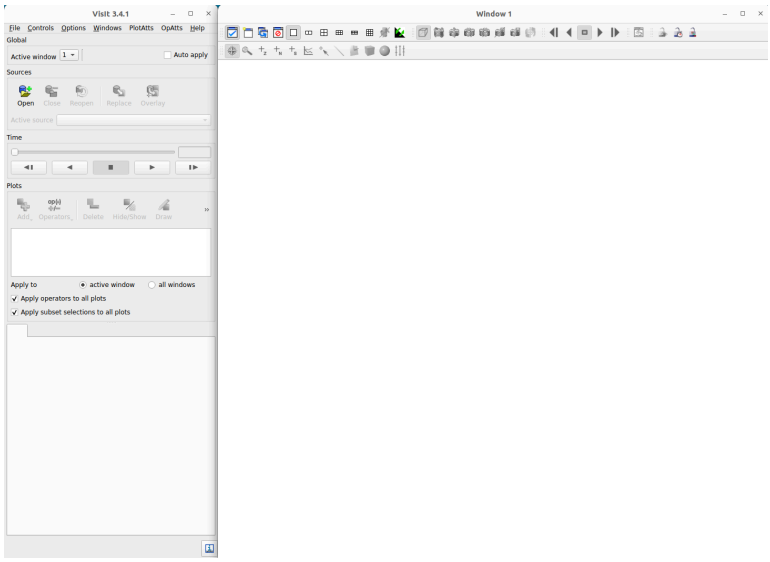
Note that in future updates the tar line will change as new versions of visIT are released. Once opened, the program can be installed, using the `chmod 755` command. **Note that the enter in the below command is important.**

```
C/C++
      Output of below commands is supressed for clarity
chmod 755 visit-install3_4_1
./visit-install3_4_1 3.4.1 linux-x86_64-ubuntu22 /home/tlara/software
tlara@angilas:~/software $ ls
3.4.1  current  visit3_4_1.linux-x86_64              visit-install3_4_1
bin    data   visit3_4_1.linux-x86_64-ubuntu22.tar.gz
```

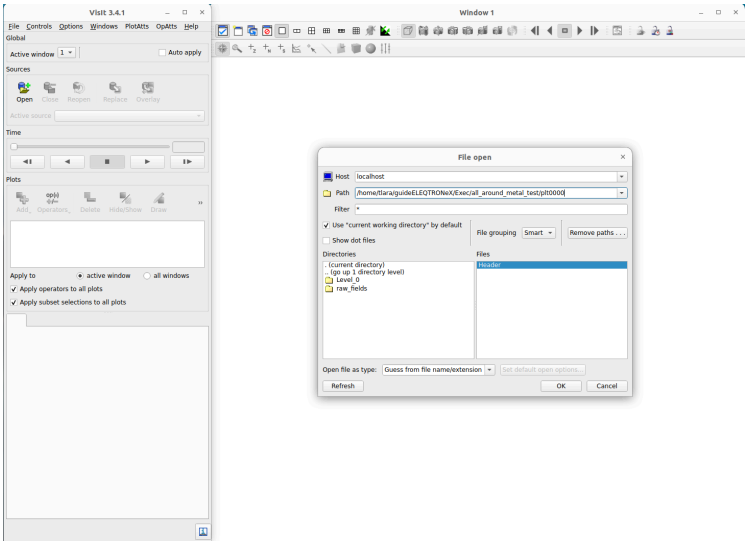
The bolded, right portion indicates the location where the program should be installed. Now the output files can be plotted. The plotting window can be opened with the `visit/bin` command from the `software` folder.

```
C/C++
tlara@angilas:~/software$ bin/visit
Running: gui3.4.1
Running: viewer3.4.1 -geometry 1560x1080+2280+0 -borders 26,4,4,4 -shift 0,0
-preshift 4,26 -defer -host 127.0.0.1 -port 5600
Running: mdserver3.4.1 -host 127.0.0.1 -port 5601
```

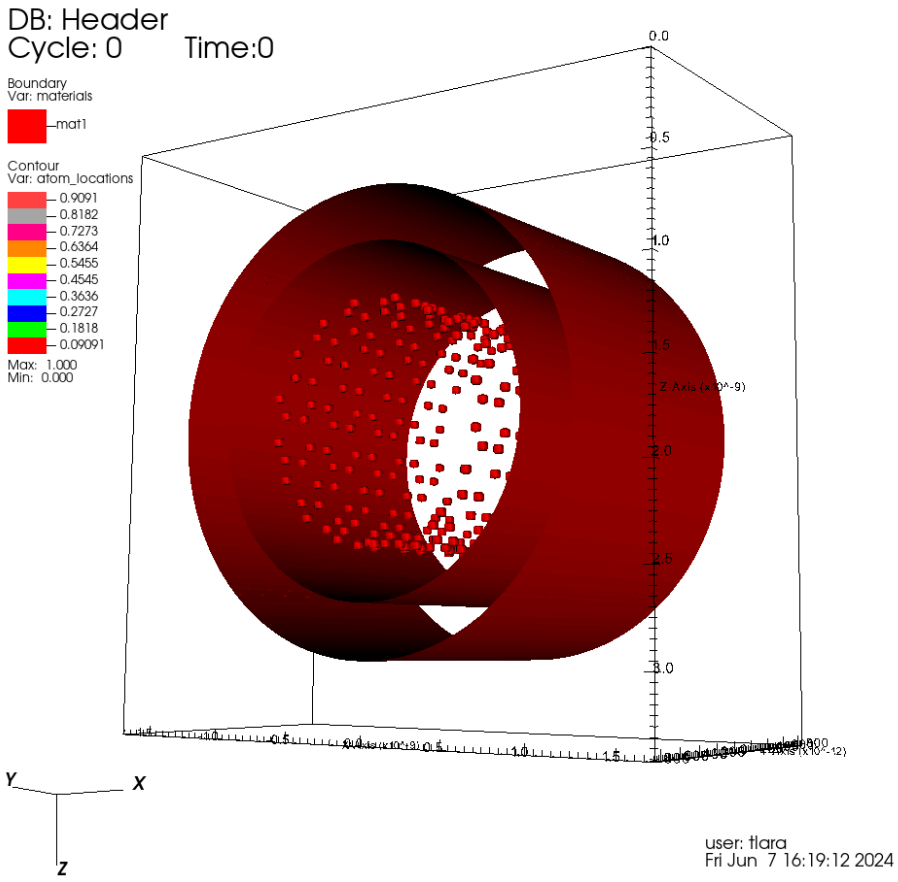
This will open a Visit plotting window that looks like this:



Using the file option in the top left bar, relevant files can be opened. The “Header” of the “plt” file should be opened.



This will not do anything until data is plotted. To do this, navigate to the “Add” option on the left side. By adding boundary-materials and contour-atom_locations, the following plot can be generated using the “draw” button.



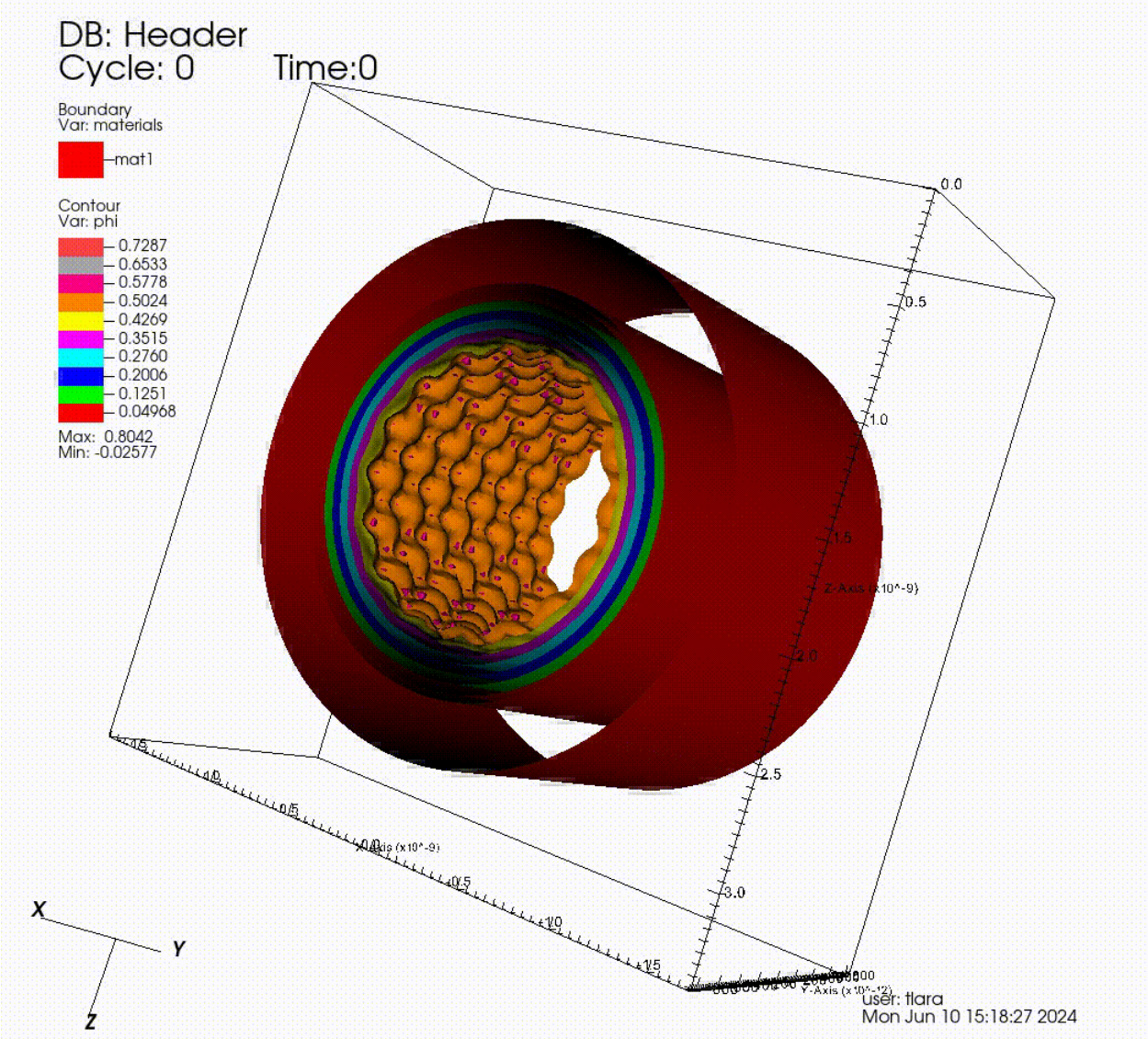
From here, the colors and exact visualization options can be modified, referring to the visIt documentation: https://visit-sphinx-github-user-manual.readthedocs.io/en/v3.2.0/gui_manual/Intro/index.html. Output can be saved using the file->save window selection.

VisIt can also create animations. To do this, create a text file named “move.visit” in the folder containing the “plt” output files, listing the names of the “plt” files.

```
C/C++
movie.visit
plt0000/Header
plt0001/Header
plt0002/Header
plt0003/Header
plt0004/Header
plt0005/Header
plt0006/Header
```

```
plt0007/Header
plt0008/Header
plt0009/Header
plt0010/Header
```

This file can then be opened in visIt the same way as a “plt” file header, and by pressing the play button, a nice animation will be shown. It can be output using the save options with user set specifications. The below animation shows the boundary and the electrostatic potential.



Data Plotting with Jupyter Notebook

Built into the ELEQTRONeX repository is a convenient Jupyter Notebook that can be used to plot potential, charge, and eV through the nanotube. These quantities are quite relevant and are hard to obtain from the visIt visualization. To open a Jupyter Notebook, users need to first install the following sudo installation commands:

```
C/C++
      Output of below commands is supressed for clarity
tlara@angilas:~$ sudo apt install jupyter-core
tlara@angilas:~$ sudo apt install jupyter-notebook
```

This once again requires sudo permissions which are not granted by default to new users. Having done this, a Python module named “seaborn,” needs to be added in order to run the plotting script. This is done via the `pip install` command:

```
C/C++
      Output of below commands is supressed for clarity
tlara@angilas:~$ pip install seaborn
```

With this done, the notebook can be run from the “ElEQTRONeX/scripts/all_around_metal” directory by entering `jupyter notebook`.

```
C/C++
tlara@angilas:~$ cd guide/ELEQTRONeX/scripts/all_around_metal/
tlara@angilas:~/guide/ELEQTRONeX/scripts/all_around_metal$ jupyter notebook
[I 09:44:12.396 NotebookApp] Serving notebooks from local directory:
/home/tlara/guide/ELEQTRONeX/scripts/all_around_metal
[I 09:44:12.396 NotebookApp] Jupyter Notebook 6.4.8 is running at:
[I 09:44:12.396 NotebookApp]
http://localhost:8888/?token=164342c365dd445478a4a54ab6a48f816bf1e3f3493db996
[I 09:44:12.396 NotebookApp]  or
http://127.0.0.1:8888/?token=164342c365dd445478a4a54ab6a48f816bf1e3f3493db996
[I 09:44:12.396 NotebookApp] Use Control-C to stop this server and shut down
all kernels (twice to skip confirmation).
[C 09:44:12.413 NotebookApp]

      To access the notebook, open this file in a browser:
      file:///home/tlara/.local/share/jupyter/runtime/nbserver-315869-open.html
      Or copy and paste one of these URLs:

http://localhost:8888/?token=164342c365dd445478a4a54ab6a48f816bf1e3f3493db996
```

```
      or
http://127.0.0.1:8888/?token=164342c365dd445478a4a54ab6a48f816bf1e3f3493db996
```

This command will actually open a new tab which looks like this:

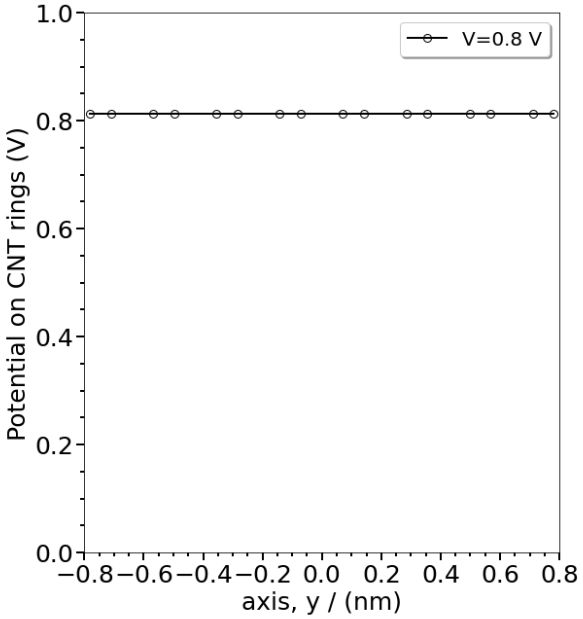
The script can then be opened, and can be executed after just one modification to the path name. The “path[0]” variable in the fifth executable needs to be renamed to add the full directory destination. By default, the script reads:

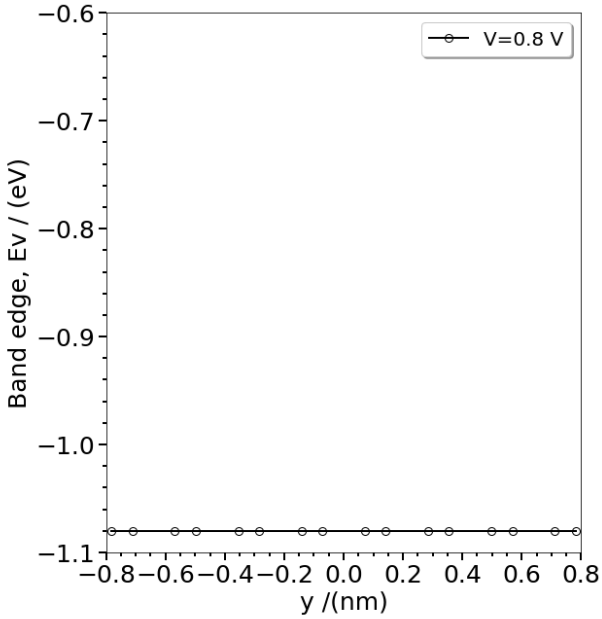
```
Python
path[0] = "all_around_metal_test/negf/cnt/step%04d_"%(step)
```

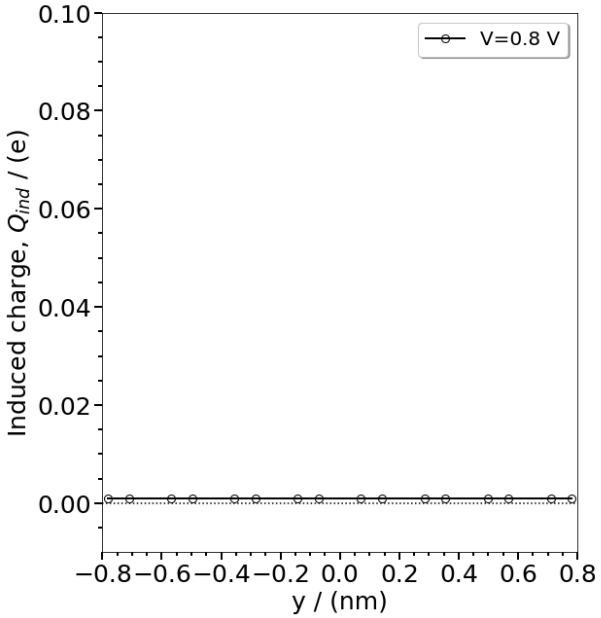
And should be changed to feature the entire directory address:

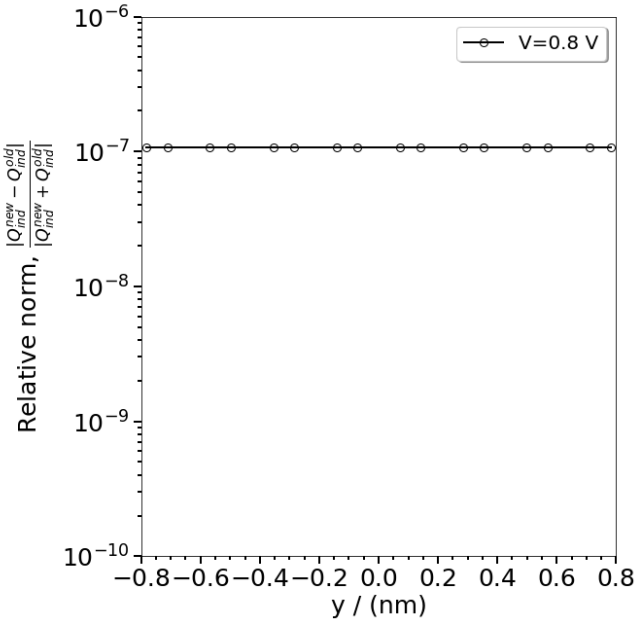
```
Python
path[0] =
"home/tlara/guide/ELEQTRONeX/Exec/all_around_metal_test/negf/cnt/step%04d_"%(st
ep)
```

Having done this, the script can be run using the “Run” button, the output should produce plots looking like this:


V=0.8 V
Potential on CNT rings (V)
axis, y / (nm)


V=0.8 V
Band edge, Ev / (eV)
y /(nm)


V=0.8 V
Induced charge, Q_{ind} / (e)
y / (nm)


V=0.8 V
Relative norm, $\frac{|Q_{ind}^{new} - Q_{ind}^{old}|}{|Q_{ind}^{new} + Q_{ind}^{old}|}$
y / (nm)

FerroX

Installing FerroX

Make sure to clone the FerroX repository within the same directory on your machine as the AMReX repo.

Following a similar process to AMReX, find the FerroX repo at the following git link: https://github.com/AMReX-Microelectronics/FerroX

Find the green '<>Code' button, click the dropdown menu, and copy the link under HTTPS.

Execute the `git clone` command:
`git clone <link>`

For example:

```
Unset
git clone https://github.com/AMReX-Microelectronics/FerroX.git
```

Building and Running FerroX

How you build FerroX depends on the hardware configuration of your machine. To build FerroX, execute `make -j 4` for a computer with a GPU or `make -j 4 USE_CUDA=FALSE` for a computer with a CPU. Make sure that you `export OMP_NUM_THREADS=1` prior to running FerroX, since we are using multiple processes (parallelization) and want to ensure that the number of OMP_NUM_THREADS does not exceed the number of physical CPU cores at your machine. To run FerroX, follow this format:

```
mpirun -n <num_cores> ./<executable>  <input_file>
```

For example:

```
Unset
mpirun -n 4 ./main3d.gnu.TPROF.MPI.OMP.ex inputs_mfim_Noeb
```

Installing Visit: 3D Data Visualization Software

Installing Visit in Terminal
Visit is a software that enables researchers to visualize 3-D data that has been formatted into plot files. Here is a step-by-step guide for installing Visit through terminal on a Linux machine (Ubuntu).

A. Ensuring that the visit-install command executes.

Go to the following website to access all the versions of Visit: https://visit-dav.github.io/visit-website/releases-as-tables/

Look at the tables and locate the release that you are interested in, and find the `visit-install` script link for the appropriate operating system. There may be multiple versions even within this particular release, and it is personal preference which one you download. Once you have decided the version you want to install, right-click and copy that script link.

Go to your terminal and type the `wget` command with the link you copied in the previous step as an argument:

```
wget <link>
```

For example:

```
Unset
wget
https://github.com/visit-dav/visit/releases/download/v3.3.3/visit3_3_3.linux-x8
6_64-ubuntu20.tar.gz
```

Now type the following command in the given format to ensure that the `visit-install` script is executable:

```
chmod +x visit-install<version>
```

For example:

```
Unset
chmod +x visit-install3_3_3
```

B. Downloading + Installing Visit

Once that executes successfully, we can move on to the next step- downloading the Visit software.

`wget https://github.com/visit-dav/visit/releases/download/<version_num>/visit-install<version>`

For example:

```
Unset
wget
https://github.com/visit-dav/visit/releases/download/v3.3.3/visit-install3_3_3
```

And now, we can move onto the final step- installing Visit. To do so, execute the following command (note, this is where the visit-install script comes into play):

`./visit-install<version> <version_num> <platform> <path/to/directory>`

Note that the command pwd reveals the path to current directory location.
Be careful when specifying the platform argument. Check your machine's Ubuntu version.

For example:

```
Unset
./visit-install3_3_3  3.3.3  linux-x86_64-ubuntu20  /home/john_doe/tools
```

C. Running Visit

To run visit, you simply need to go to the terminal and type out the path of where Visit is installed in your machine.

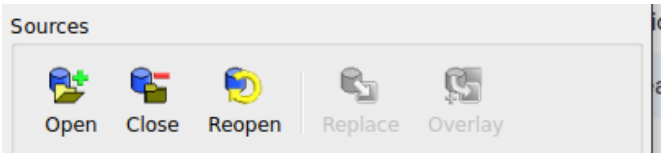
For example:
`/home/john_doe/tools/bin/visit`

Viewing Data in Visit

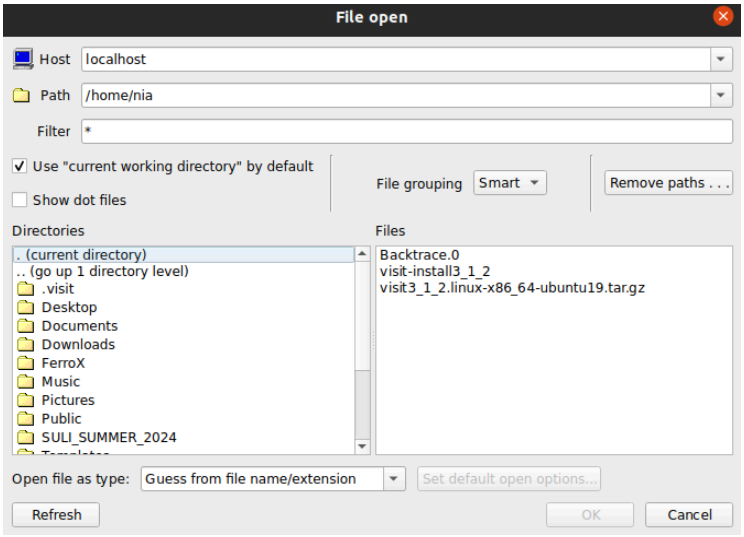
Individual plotfile data

To view individual plot file data in visit, start by launching the software. Once you have done so, find the ‘open’ icon on the Visit toolbar (the smaller rectangular window) and click on it. This should open a window depicting the files on your computer. Navigate into the directory where you ran FerroX (and hence, the directory that contains the plot files you want to view individually).

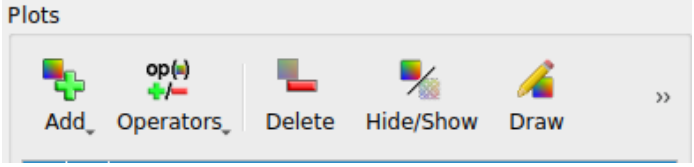
This is where you can open files from.



Go to the plotfile directory that you wish to view and double click on the ‘Header’ file. From here, you can apply various colorings, contour, and slicing operations. After adding your desired features, click ‘draw’ to visualize your data in the larger GUI window. *This is the file path system that pops up from clicking ‘open’.*



Here you can find coloring/contouring/slicing features to customize your data view.



Creating a movie

Similar to AMReX, FerroX creates plot files of data values once it is run. Since we have covered how to view the plot files individually, I will now walk you through how to view them all in a cohesive animation by creating a movie. Open a terminal, traverse to the same directory as all of the plot files that you want to combine, and execute the following command:

```
ls -1v plt*/Header | tee <movie_name>.visit
```

For example:

```
Unset
ls -1v plt*/Header | tee movie5.visit
```

Viewing your movie

Once you have created the movie, you can open Visit, go to the ‘open’ option, and then navigate to the location of the movie in your computer’s file system. From there you can open the movie by double-clicking on the name of the movie file and apply the coloring/operational features before clicking ‘draw’. Then you can use the pause-play-skip features to toggle through the different plot files step-by-step or view all of them as an animation.

BERKELEY LAB

Using Github

Opening Github Repository

Using Github is very important for any computational project. Cloning a file from Github has already been done, but pushing changes is a much more intricate process. Firstly, python has to be installed.

```
C/C++
      Output of below commands is supressed for clarity
tlara@angilas:~$ sudo apt-get install python3-pip
```

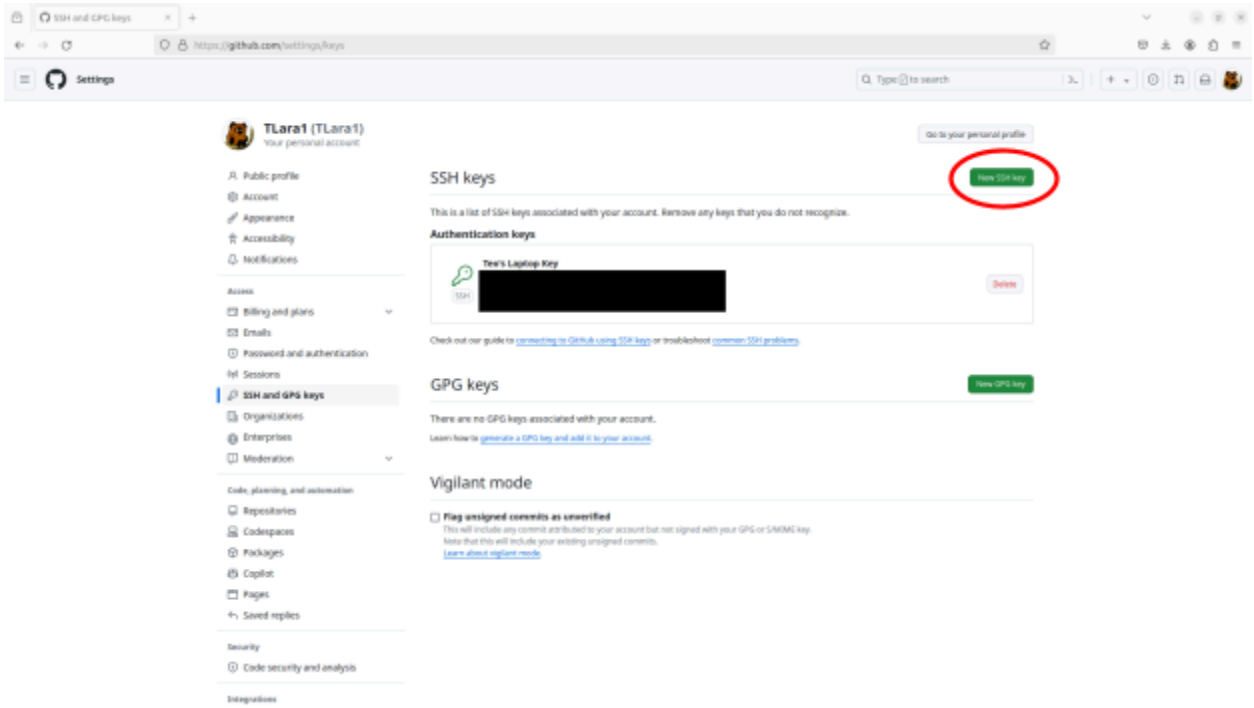
These will be needed for opening and editing files. An environment variable also needs to be added. Add the following to the "/.bashrc" file. (consult the Appendix for more information).

```
C/C++
export PATH=$PATH:/home/tlara/.local/bin
```

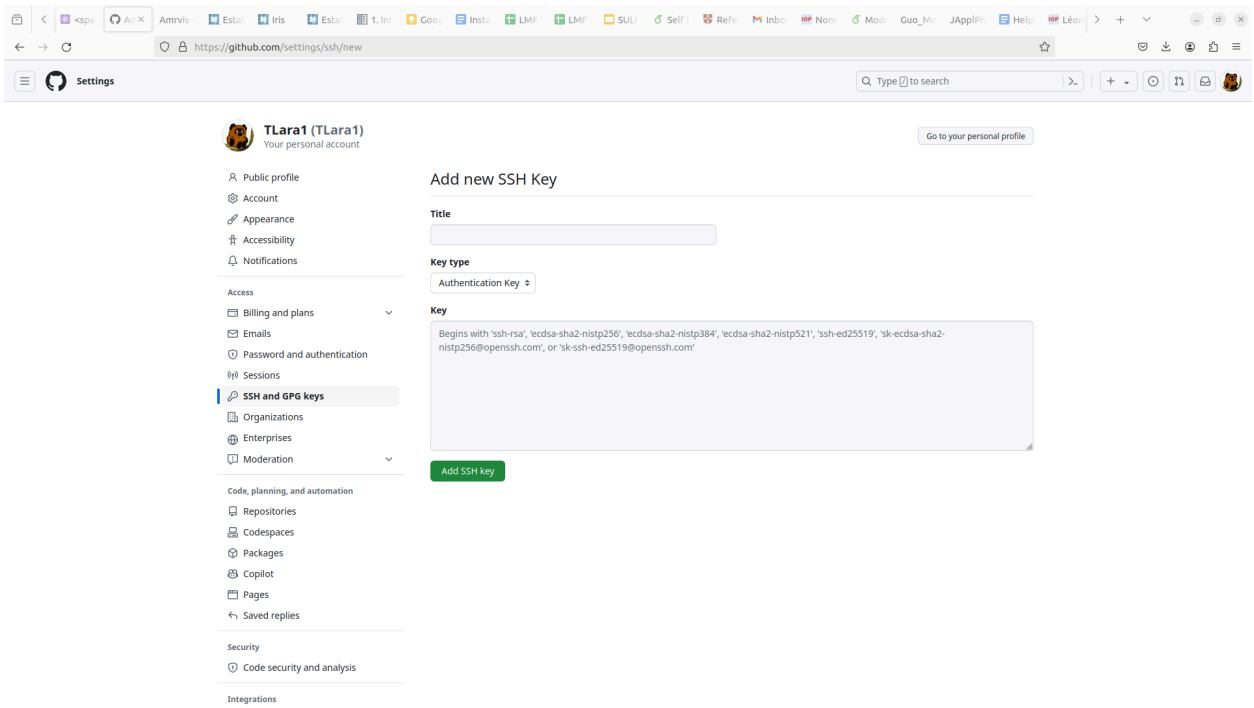
Next, an ssh key can be created to make the login process easier. Navigating to the ".ssh" folder, the "`ssh-keygen`" command will create a key. To view the Key, use `more id_rsa.pub` or `cat id_rsa.pub` (or a similar command that enables you to view the contents of a file).

```
C/C++
tlara@angilas:~$ cd .ssh
tlara@angilas:~/.ssh$ ls
known_hosts  known_hosts.old  nersc  nersc-cert.pub  nersc.pub
tlara@angilas:~/.ssh$ ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/home/tlara/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/tlara/.ssh/id_rsa
Your public key has been saved in /home/tlara/.ssh/id_rsa.pub
The key fingerprint is:
User fingerprint
The key's randomart image is:
User image
tlara@angilas:~/.ssh$ more id_rsa.pub
ssh-rsa user series of random letters and symbols
```

This key must then be linked to one's Github account. This can be done on the Github webpage, in the "SSH and GPG Keys" option in the "settings" menu with the "New Key" button.



Copy and paste the key from the `more id_rsa.pub` command into the shown box:



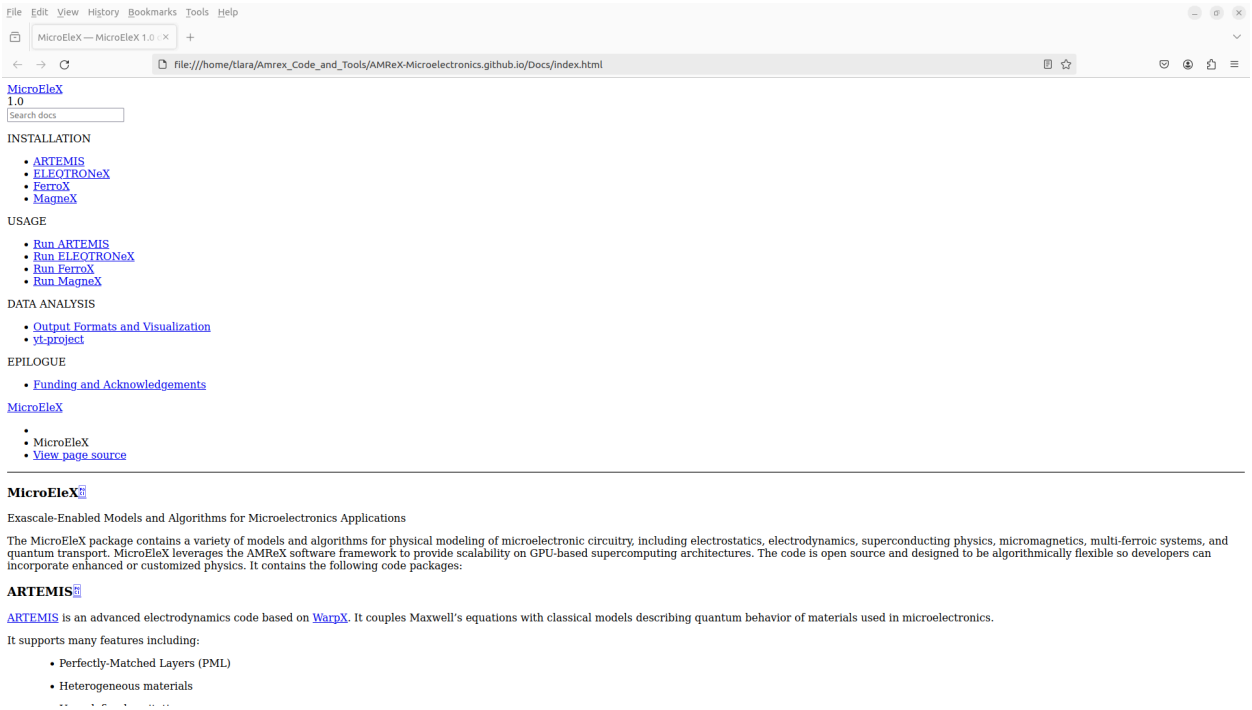
Now changes can be made to “git” directories. To do this, a user has to have their Github account added to a project. In this case, changes will be made to the AMReX-Microelectronics library. First, the file is cloned onto the local device. This is done in a folder called “Amrex_Code_and_Tools”

```
C/C++
tlara@angilas:~$ mkdir Amrex_Code_and_Tools
tlara@angilas:~$ cd Amrex_Code_and_Tools
tlara@angilas:~/Amrex_Code_and_Tools$ git clone
https://github.com/AMReX-Microelectronics/AMReX-Microelectronics.github.io.git
Cloning into 'AMReX-Microelectronics.github.io'...
remote: Enumerating objects: 1881, done.
remote: Counting objects: 100% (1881/1881), done.
remote: Compressing objects: 100% (1072/1072), done.
remote: Total 1881 (delta 979), reused 1617 (delta 757), pack-reused 0
Receiving objects: 100% (1881/1881), 14.82 MiB | 25.76 MiB/s, done.
Resolving deltas: 100% (979/979), done.
tlara@angilas:~/Amrex_Code_and_Tools$ cd AMReX-Microelectronics.github.io/Docs/
```

Here, the python requirements have to be added. This can be done easily using the built-in “requirements.txt” file. Afterwards, the html file can be made using `make html`.

```
C/C++
      Output of below commands is supressed for clarity
tlara@angilas:~/Amrex_Code_and_Tools/AMReX-Microelectronics.github.io/Docs$
pip3 install -r requirements.txt
tlara@angilas:~/Amrex_Code_and_Tools/AMReX-Microelectronics.github.io/Docs$
make html
```

Some warnings may be shown, these can be safely ignored. Now, `firefox index.html` can be run to open the index. html file in firefox.



Making and Editing a Github Branch

A sample branch will be created and edited, from the same Github project. First, to create and move to a *new* branch, the `git checkout -b <branch_name>` command can be used. From here, the source files can be accessed in the "source" folder, and the file to be edited, "index.rst" is opened with emacs (or your preferred text editor, such as *Vim* or *VS Code*).

```
C/C++
tlara@angilas:~/Amrex_Code_and_Tools/AMReX-Microelectronics.github.io/Docs$ git checkout -b Teodocs
Switched to a new branch 'Teodocs'
tlara@angilas:~/Amrex_Code_and_Tools/AMReX-Microelectronics.github.io/Docs$ cd source/
tlara@angilas:~/Amrex_Code_and_Tools/AMReX-Microelectronics.github.io/Docs/source$ emacs index.rst
```

After changes are made and saved, from the "Docs" folder, the "make clean" and "make html" commands can be run again to remake the changes.

```
C/C++
      Output of below commands is supressed for clarity
```

```
tlara@angilas:~/Amrex_Code_and_Tools/AMReX-Microelectronics.github.io/Docs/sour
ce$ cd..
tlara@angilas:~/Amrex_Code_and_Tools/AMReX-Microelectronics.github.io/Docs$
make clean
tlara@angilas:~/Amrex_Code_and_Tools/AMReX-Microelectronics.github.io/Docs$
make html
```

The made changes can be viewed using Firefox. Using the "`firefox`" command within the `build/html` folder, a tab will appear, featuring the newly made changes.

```
C/C++
      Output of below commands is supressed for clarity
tlara@angilas:~/Amrex_Code_and_Tools/AMReX-Microelectronics.github.io/Docs$
firefox build/html/index.html
```

Now, the change can be committed using the `git commit -a` command. This will commit all changes and open a commit text file. In this file, the changes can be appropriately titled. Alternatively, you can also `use the git commit -am "<message>"` command to write the commit message directly on the terminal without opening a text file. Double quotes are needed.

```
C/C++
      Output of below commands is supressed for clarity
tlara@angilas:~/Amrex_Code_and_Tools/AMReX-Microelectronics.github.io/Docs$ git
commit -a
```

Next, the changes can be pushed using the `git push` command. This will push local changes onto the Github branch. One may need to enter their username and password at this step, depending on whether or not an ssh tunnel has been set up.

```
C/C++
      Output of below commands is supressed for clarity
tlara@angilas:~/Amrex_Code_and_Tools/AMReX-Microelectronics.github.io/Docs$ git
push
```

From here, with the appropriate approval, the branch can be merged with a main branch on the Github repository homepage.

BERKELEY LAB

HPC resources - Using Perlmutter and Gigan

Accessing an HPC

Any HPC is accessed from the terminal, using the `ssh` command. The following command will access Perlmutter:

```
C/C++
tlara@angilas:~$ ssh tlara@perlmutter.nersc.gov
********************************************************************************
                          NOTICE TO USERS

Lawrence Berkeley National Laboratory operates this computer system under
contract to the U.S. Department of Energy.  This computer system is the
property of the United States Government and is for authorized use only.
Users (authorized or unauthorized) have no explicit or implicit
expectation of privacy.

Any or all uses of this system and all files on this system may be
intercepted, monitored, recorded, copied, audited, inspected, and disclosed
to authorized site, Department of Energy, and law enforcement personnel,
as well as authorized officials of other agencies, both domestic and foreign.
By using this system, the user consents to such interception, monitoring,
recording, copying, auditing, inspection, and disclosure at the discretion
of authorized site or Department of Energy personnel.

Unauthorized or improper use of this system may result in administrative
disciplinary action and civil and criminal penalties. By continuing to use
this system you indicate your awareness of and consent to these terms and
conditions of use. LOG OFF IMMEDIATELY if you do not agree to the conditions
stated in this warning.

********************************************************************************

Login connection to host x3113c0s19b0n0:

(tlara@perlmutter.nersc.gov) Password + OTP:
```

The bolded text should be replaced with whatever one's username is. After entering their password and authentication code in the case of Perlmutter, access is granted to the HPC, and the command line changes to:

```
C/C++
tlara@perlmutter:login12:~>
```

The username will change based on the user. Furthermore, the address for accessing Gigan is different, using “@gigan.lbl.gov.”

Moving Files

Using High Performance Computing (HPC) resources will be very important to any project. There are two main HPCs available, Gigan and Perlmutter. The latter is much smaller and should only be used for local jobs, and the former is adequate for large simulations. To gain access to these clusters, a mentor can email the HPC managers to give new users accounts.

Once on an HPC, the workflow is quite similar as it is locally, with everything managed from the terminal. There is an important command for moving files back and forth between an HPC and a local computer, using the `scp` command. To move files from an HPC server to a local machine use “scp (-r) 'location on HPC’ ‘location on local machine’. For example:

```
C/C++
      Output of below commands is supressed for clarity
tlara@angilas:~$ scp -r
tlara@gigan:/home/tlara/programs/ELEQTRONeX/Exec/all_around_metal_test
/home/tlara/guide/ELEQTRONeX/Exec
```

This code move the “all_around_metal_test” folder from a profile on the “gigan” cluster to the “Exec” folder in “ELEQTRONex” on the local machine. The “-r” addition moves a folder and not a file. To move files from a local machine to an HPC, the order needs only be inverted.

Setting up ELEQTRONeX on Gigan

Fortunately, the installation process for ELEQTRONeX is identical on Gigan as it is locally, and **neither the hypre nore openmpi libraries are needed, just AMReX and ELEQTRONex**. The previous steps need only be repeated from an ssh terminal. The only difference is in the “GNUmakefile” for the make process and in the addition of two environment variables. For the first change, repeat the process of editing the file as was done locally, except keep “USE_CUDA” as “TRUE” and change “USE_HYPRE” to “FALSE.” The makefile will look like this:

```
AMREX_HOME  ?= ../../amrex

DEBUG        = FALSE
USE_MPI      = TRUE
USE_OMP      = FALSE
USE_CUDA     = TRUE
USE_HYPRE    = FALSE
COMP         = gnu
DIM          = 3
CXXSTD       = c++17
TINY_PROFILE   = FALSE


USE_EB = TRUE
USE_TRANSPORT = TRUE
COMPUTE_GREENS_FUNCTION_OFFDIAG_ELEMS = FALSE
COMPUTE_SPECTRAL_FUNCTION_OFFDIAG_ELEMS = FALSE
BROYDEN_PARALLEL = TRUE
BROYDEN_SKIP_GPU_OPTIMIZATION = FALSE

PRINT_NAME    = FALSE
PRINT_LOW    = FALSE
PRINT_HIGH    = FALSE
TIME_DEPENDENT = TRUE


CODE_HOME := ..
include $(CODE_HOME)/Source/Make.Code
```

The next change is to edit an environment variable to be able to use the MPI library of Gigan. The following additions should be added to the “/.bashrc” file on gigan:

```
C/C++
export CUDA_PATH=/usr/local/cuda
export PATH=$PATH:/usr/local/cuda/MPIbin
```

Consult the appendix for information regarding adding environment variables. From here, everything should run on Gigan and files can be transferred back to the local machine as was previously mentioned.

Simulations are run from the command line as done locally.

Setting up ELEQTRONeX on Perlmutter

The faster and more powerful cousin of Gigan is Perlmutter, which is widely used by LBNL researchers to conduct intricate simulations. Running ELEQTRONeX on Perlmutter is a little more complicated than on Gigan or locally as the input script of the ELEQTRONeX program also has to be modified in order to send outputs to the correct locations.

First of all, it is assumed that a Perlmutter account has been created and associated with a project. AMReX, hypre, and ELEQTRONeX can then be installed normally, noting that no changes should be made to the GNUmakefile. “USE_MPI,” “USE_CUDA,” and “USE_HYPRE,” should all be set to true, the only false option being “USE_OMP.” There is also an environment variable to hypre that needs to be added when working on Perlmutter, just like installing hypre locally. Unlike Gigan and local machines, Perlmutter consoles do not have a build-in “/.bashrc” file, however, this can be created using the emacs command that would be used to normally access the file:

```
C/C++
tlara@perlmutter:login12:~> emacs ~/.bashrc
```

To this file, one needs to write an “export” command, writing:

```
C/C++
export HYPRE_DIR=/global/homes/t/tlara/programs/hypre/src/hypre
```

Afterwards, the file must be set as a source with:

```
C/C++
tlara@perlmutter:login12:~> source ~/.bashrc
```

With these changes, ELEQTRONeX will be makeable on Perlmutter using the same previously written instructions.

Running a Simulation on Perlmutter

On Perlmutter, simulations are mostly not run from the command line, they need to be submitted using a submission script. Submission scripts can be created using the emacs command:

```
C/C++
tlara@perlmutter:login03:~/programs/ELEQTRONeX/perlscripts> emacs
all_around_metal_submission.sh
```

In this case, the submission script is placed in the “perlscripts” directory. The submission script in question is as follows:

```
C/C++
#!/bin/bash
```

```
#SBATCH -N 1
#SBATCH -C gpu
#SBATCH -G 4
#SBATCH -q debug
#SBATCH --mail-user=tlara@lbl.gov
#SBATCH --mail-type=ALL
#SBATCH -A m4540
#SBATCH -t 0:10:0

# OpenMP settings:
export OMP_NUM_THREADS=1
export OMP_PLACES=threads
export OMP_PROC_BIND=spread

# Run Executable
srun -n 4 -c 32 --cpu_bind=cores -G 4 --gpu-bind=single:1
../Exec/main3d.gnu.MPI.CUDA.EB.TD.TRAN.BROYPRLL.HYPRE.ex
../input/negf/all_around_metal
```

There are many options in submission scripts, and much more information as well as a script generator can be found here. The important feature in this case is the last line, the executable. The location of files written here have to be in relation to the submission location of the script. So, since the the “perlscripts” directory is inside the “ELEQTRONeX” directory, the script tells the machine to move to the “Exec” folder to find the executable and then to the “input” folder to find the input file. It is important to check the correct filenames on executables, or else code will not compile correctly.

Changing Output Destination Within Input File

When running on Perlmutter, it is inadvisable to store files in the “home” directory, as this directory is fairly small and will quickly become overpopulated. Instead, users should save files in the “cfs” directory. To do this, **all input files must be modified before being submitted to Perlmutter**. Below, is a modification to the “all_around_metal” input file to place outputs in the “cfs” directory.

First, the input file is opened:

```C/C++
tlara@perlmutter:login12:~> emacs
programs/ELEQTRONeX/input/negf/all_around_metal
```

Then, locating the “plot.folder_name” field, this is changed to the following:

```
C/C++
plot.folder_name = /global/cfs/projectdirs/m4540/teo/all_around_metal
```

After saving, output files will be placed in a new folder created within the “cfs” directories. For any simulation, the input file must be modified in this way.

Note, if performing a “git pull” to update code, it is crucial to stash changes before pulling in order to avoid merge conflicts. This can be done with the “git stash” command.

```
C/C++
      Output of below commands is supressed for clarity
tlara@perlmutter:login03:~/programs/ELEQTRONeX/perlscripts> git stash
tlara@perlmutter:login03:~/programs/ELEQTRONeX/perlscripts> git pull
tlara@perlmutter:login03:~/programs/ELEQTRONeX/perlscripts> git stash pop
```

The first command saves any changes made to files. The second pulls new files from github, and the third “pops” the stash, replacing the newly imported files with the changes. It is very important to perform this action, or else all of the modified files will not be saved as pulling a new code version from Github.

Issue With all_around_metal Test Case When Using Hypre

There is an issue when running the basic “all_around_metal” test case on Perlmutter. Essentially, since Perlmutter uses Hypre, the input of the file has to be changed to reduce the intersection of the embedded and neumann boundaries. To do this, navigate to the location of the “all_around_metal_test” input in the “input/negf.” folder:

```
C/C++
tlara@perlmutter:login12:~> emacs
programs/ELEQTRONeX/input/negf/all_around_metal
```

Then, locate “Gate.height” value. This can be using the search command ctrl s, which functions the same way as ctrl f. Once here, change the value from “Ly + 2*dy/5” to “Ly - 16*dy - dy/5.” The Gate is being made a little less tall, farther away from the boundaries. For this case, that is the only change that needs to be made.

Appendix

Adding Environment Variables and Aliases by modifying the bash file

To permanently add environment variables, rather than changing them temporarily, the local bash file can be modified. This can be done by opening the hidden file “./bashrc” using either the vim or emacs editor.

```
C/C++
tlara@angilas:~$ emacs ~/.bashrc
```

Anything written in this file should not be changed, at the bottom, one can write environment variables as they would ordinarily. The bottom of the file and added text are shown below.

```
alias l='ls -CF'

# Add an "alert" alias for long running commands.  Use like so:
#   sleep 10; alert
alias alert='notify-send --urgency=low -i "$([ $? = 0 ] && echo terminal || echo
 error)" "$(history|tail -n1|sed -e '\''s/^\s*[0-9]\+\s*//;s/[;&|]\s*alert$//'\'
')"'

# Alias definitions.
# You may want to put all your additions into a separate file like
# ~/.bash_aliases, instead of adding them here directly.
# See /usr/share/doc/bash-doc/examples in the bash-doc package.

if [ -f ~/.bash_aliases ]; then
    . ~/.bash_aliases
fi

# enable programmable completion features (you don't need to enable
# this, if it's already enabled in /etc/bash.bashrc and /etc/profile
# sources /etc/bash.bashrc).
if ! shopt -oq posix; then
  if [ -f /usr/share/bash-completion/bash_completion ]; then
    . /usr/share/bash-completion/bash_completion
  elif [ -f /etc/bash_completion ]; then
    . /etc/bash_completion
  fi
fi

export HYPRE_DIR="/guide/hypre/src/hypre"
```

After this change is saved, the emacs window can be closed, and the bash file should be made the source using:

```C/C++
tlara@angilas:~$ source ~/.bashrc
```

Aliases can also be helpful, allowing a user to shorten commands in the same way as the “export” command can be used, the “alias” command can set shortcuts. For example, by adding the following line to the “.bashrc” file:

```C/C++
alias NERSC="ssh tlara@perlmutter.nersc.gov"
```

Logging into NERSC will only require entering “NERSC” into the terminal, as opposed to the entire ssh command.

Command Dictionary

Analogous
up and down arrows - access previous terminal entries
ctrl c - quit execution of current command
tab - used to finish commands rather than writing commands entirely
top - see currently running CPU processes
nvidia-smi - see currently running GPU processes

Moving between directories
pwd - lists current directory
ls - displays contents of current directory
cd “foldername” - moves to the provided directory, replace “foldername” with the directory name
cd .. - moves to parent directory
cd ../.. Move out 2 folders, add /.. To move farther back
cd ~ move back to home directory
cd - return to previous directory after moving out

Managing directories
mkdir “foldername” - makes a new directory, replace “foldername with directory name
rm (-r) “filename” - removes file named “filename,” or a folder, if “-r” is added
mv (-r) “filename” “destination path” - moves the indicated file to the folder along the destination path. Add “-r” to move a folder.
(s)cp (-r) “filename” “destination path” - copies the indicated file to the folder along the destination path. Add “-r” to copy a folder. Add (s) if transferring between an ssh host and a local machine.
tar “foldername” - open a tar format folder

cat “filename” - shows the contents of a file

Installation from source
./configure
make - used to make packages
make install - used to make packages
make clean - undos “make” command
make distclean - undos both “make” and “./configure” commands

Github related commands
git clone “url” - clones Github files from provided url onto computer
git branch -a - shows all branches as well as current branch status
git checkout -b “branchname” - switches and moves to a new branch
git status - shows current branch and update states of Github branch
git checkout “branchname” - switches to branch
git commit -a - opens commit text file to commit changes.
git stash - creates a “stash” of changes made to git files to be saved
git stash pop - “pops” the stashed filed, replacing newly imported files with the stashed ones.
git pull - pulls latest version of current branch. Note that files may need to be remade and re configured after performing this action.

Setting variables
export “variable name”=”value” - this command sets the text written in “variable name” to the text written in “value”
alias “variable name”=”value” - this command sets the text written in “variable name” to the text written in “value” when it is entered in the terminal
echo $”variable name” - returns the value of the provided variable
set - shows lots of output, near the top is a list of all set variables

Sudo commands
sudo apt-get update - checks if repository is up-to-date
sudo apt install “library name” - installs the provided library
sudo apt remove “library name” - removes the provided library

emacs commands - More commands can be found here
emacs “filename” - opens emacs text editor to modify given “filename”
ctrl x ctrl s - saves file
ctrl x ctrl c - exists file
ctrl s “string” - searches the file for the string

vim commands - More commands can be found here

BERKELEY LAB

vim “filename” - opens vim text editor to modify given “filename”
shift g - moves to end of file
1, shift g - moves to top of file
a - activates edit mode
esc - leaves edit mode
:w! - saves file
:q! - closes file
:wq! - closes and saves file