Steps. Week 3 Spring 2019. Part B

*Steps to perform the first computation using pre-set run of Fast Fourier Boltzmann (FFB) code.*

**Description of file spaces.**

By their meaning, simulations of scientific codes will have input and output data. Both the input and output data are stored on a some sort of file/disc space. When the software is send into execution, it reads data from the disc space. It also writes intermediate and final results on the disc space. As a rule, there are one or more disc spaces that are dedicated for use during program execution. The HPC User Guide provides detail on this. In the case of HPC Bridges, the User Guide has the following to say:

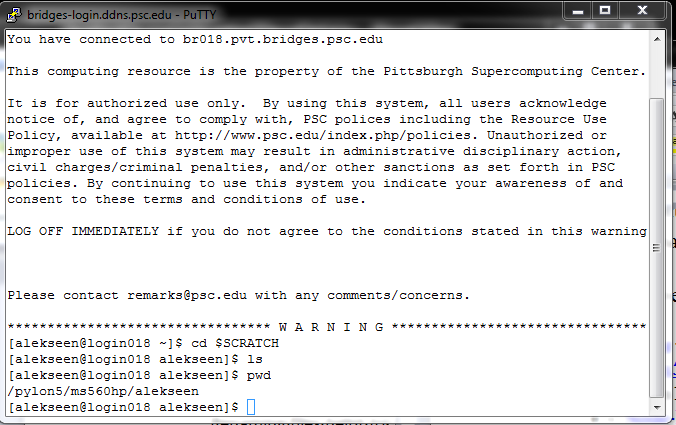
There are several distinct file spaces available on Bridges, each serving a different function.

* [Home](https://www.psc.edu/bridges/user-guide/file-spaces#home) ($HOME), your home directory on Bridges
* [pylon5](https://www.psc.edu/bridges/user-guide/file-spaces#pylon5) ($SCRATCH), a Lustre system for persistent file storage.  Pylon5 has replaced pylon1.
* [Node-local storage](https://www.psc.edu/bridges/user-guide/file-spaces#local) ($LOCAL), scratch storage in the local memory associated with a running job
* [Memory storage](https://www.psc.edu/bridges/user-guide/file-spaces#ramdisk) ($RAMDISK), scratch storage on the local disk associated with a running job

This means that you have four different storage places, each comes with its strong sides and limitations and preferred use. System variables $HOME and $SCRATCH are created for your convenience and point to two of the file spaces. In this course we will use $HOME to store most important files and we use $SCRATCH to place input and output data, executables, and related scripts.

You can access these directories using the command “cd”, e.g.,

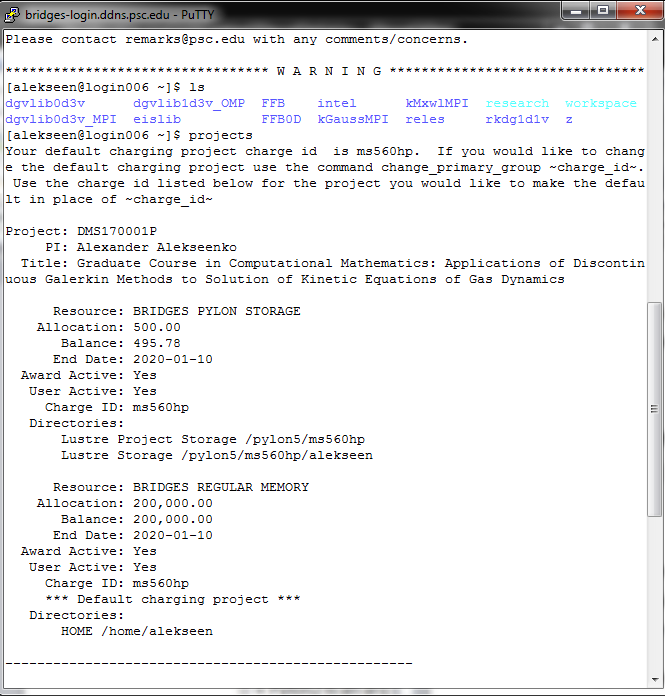
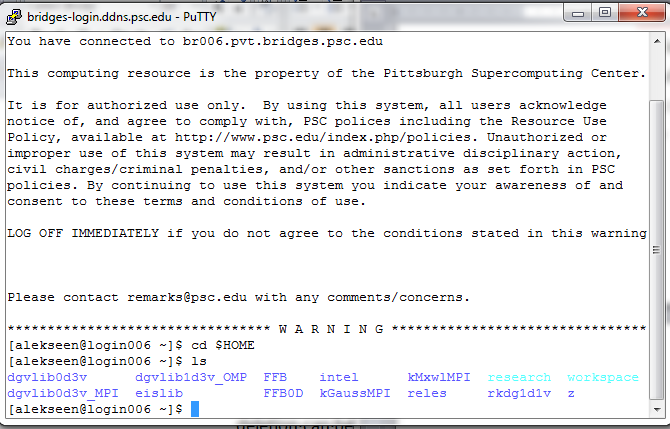
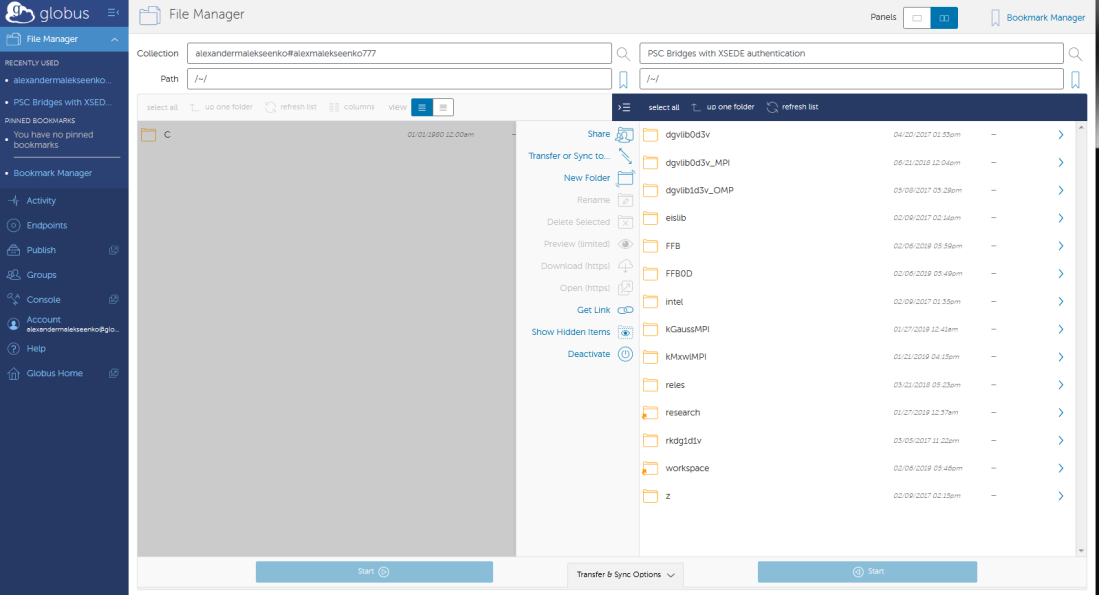
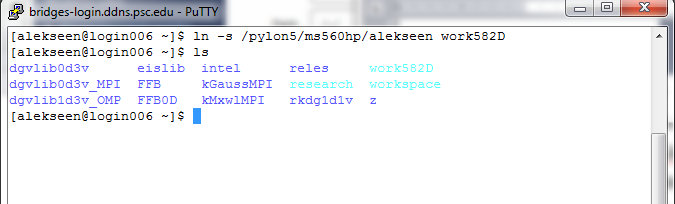
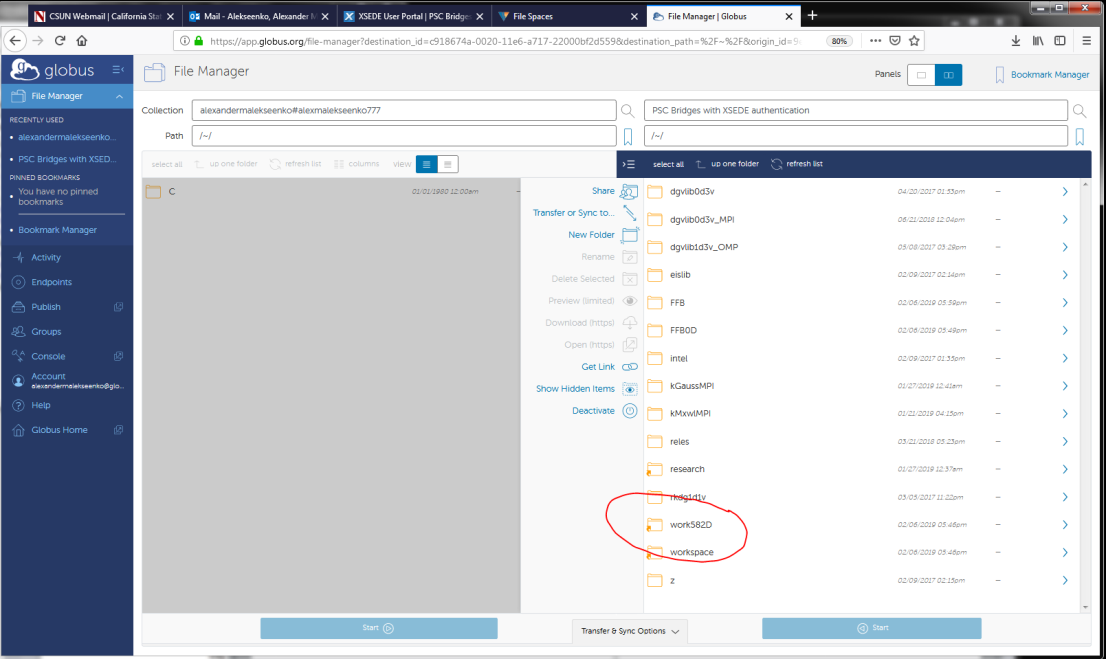
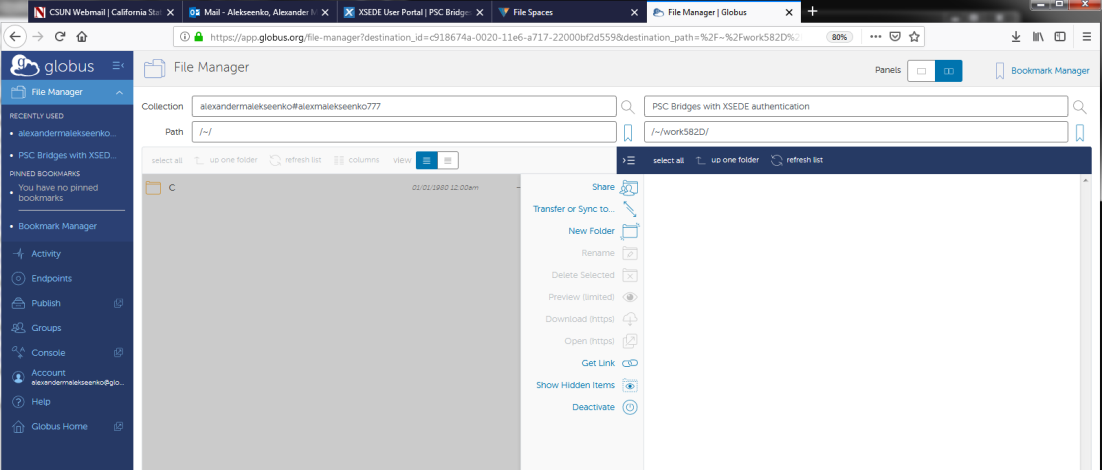
cd $HOME

cd $SCRATCH   
  
Below is an example of logging into bridges and executing cd $SCRATCH and using command **pwd** to find out the path to the directory to which system variable $SCRATCH associated with.   


You can not access $LOCAL and $RAMDISK, but your **job** can. By a job we mean the instance of running software that you sent into execution. We will not use this feature in this course.

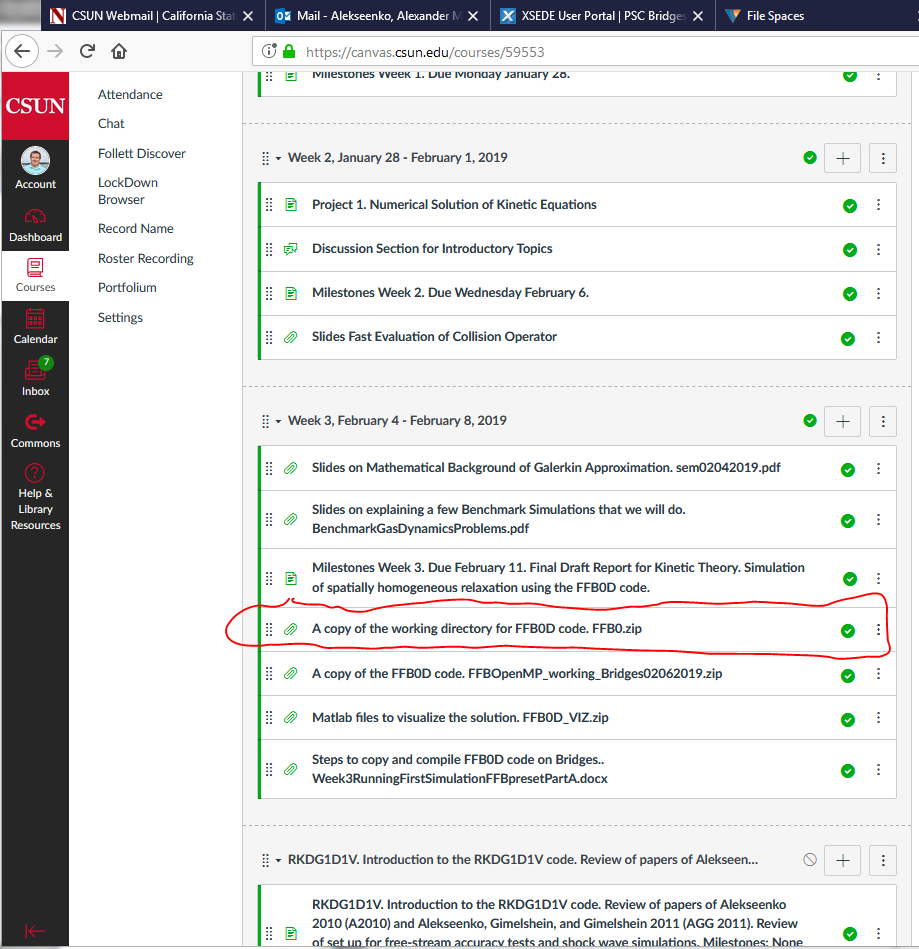
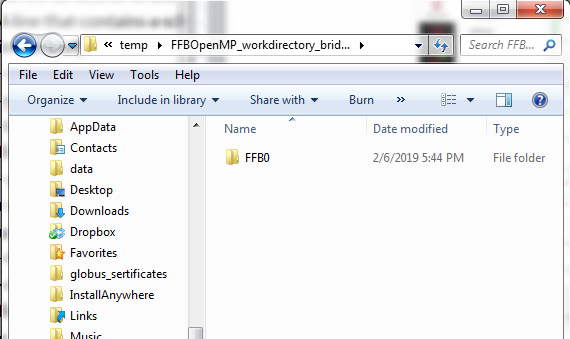
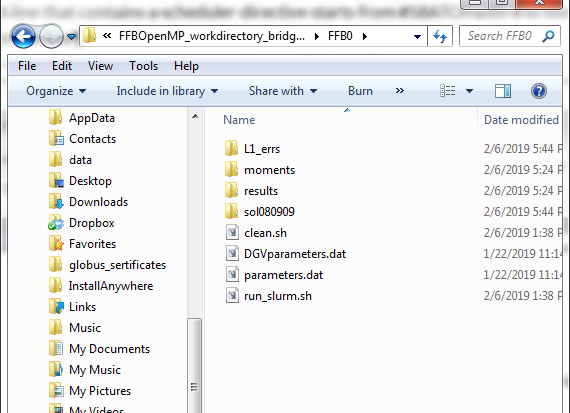
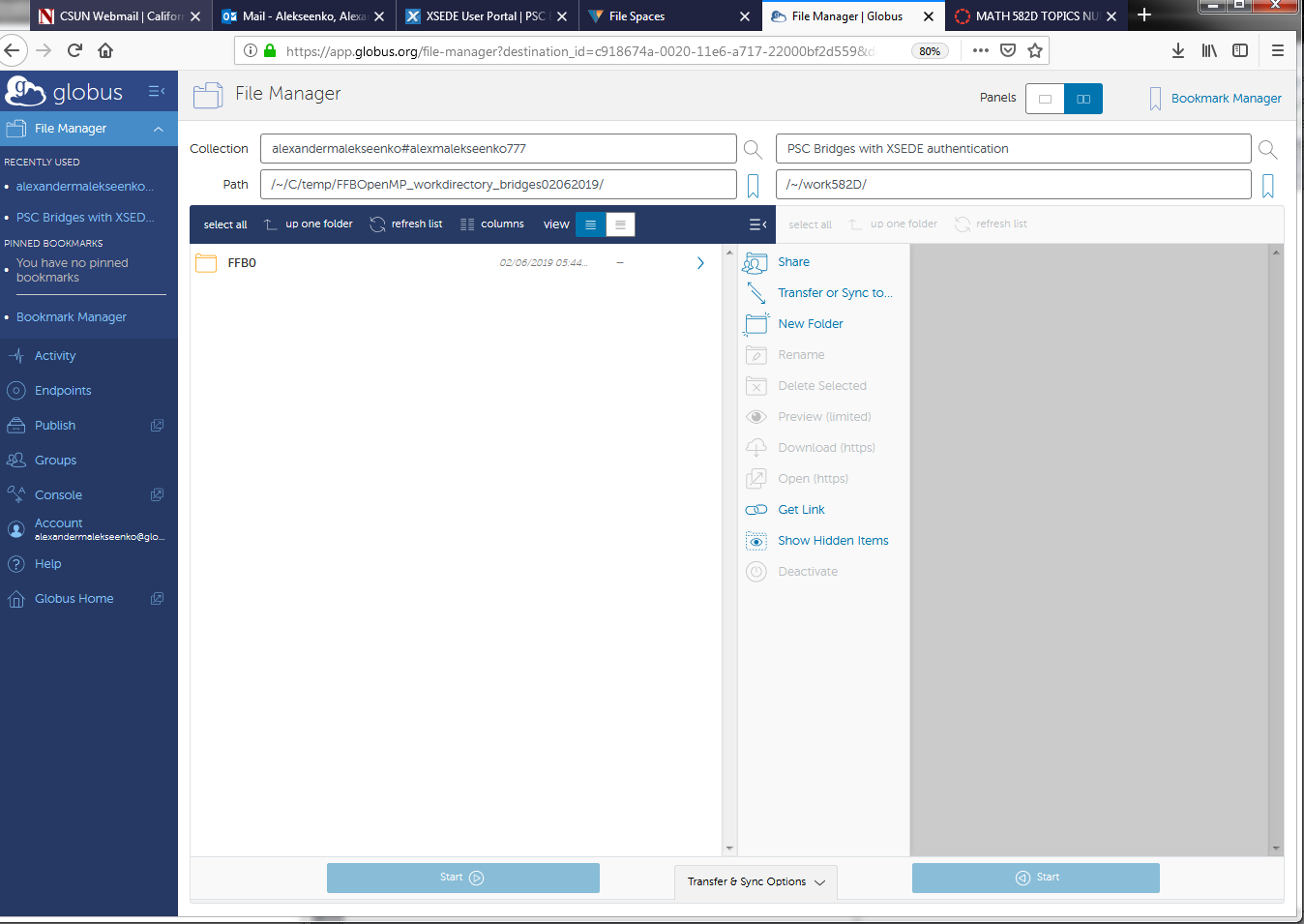
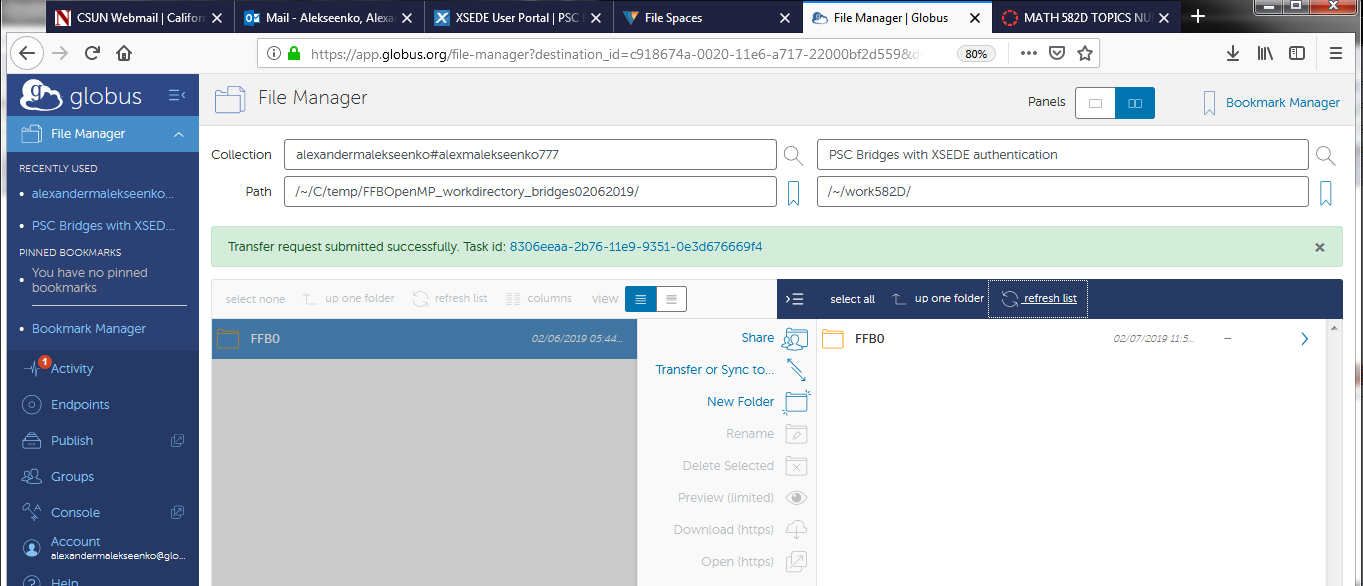
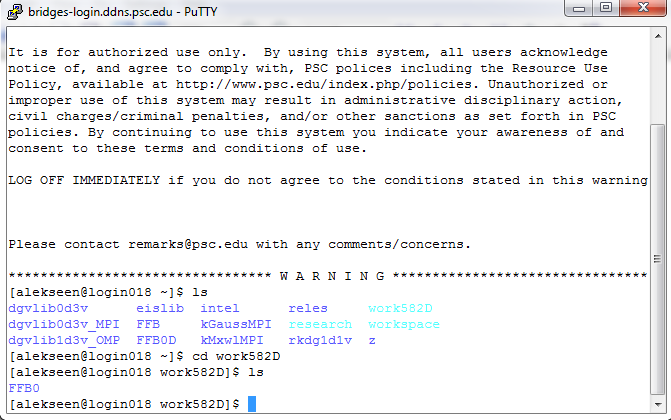
**Steps to set up a shortcut to your $SCRATCH directory accessible from Globus Connect.**

It will be convenient to set up a *symbolic link* from the $HOME directory to your $SCRATCH directory. This symbolic link will be visible/clickable from Globus Connect. Below are the steps to set up the symbolic link

1. There is another way to find out where the **pylon5** storage directory is located. When logged in Bridges, type the command  
     
   **projects**  
     
   you should see something like this:   
     
     
     
   Look at the output of the command and locate the line that looks like this   
     
   Lustre Storage /pylon5/ms560hp/yourusername  
     
   Write down this path. We will need it in the next step. This is the path that your $SCRATCH is pointing to and this is where your will keep your input and output data.  
     
   **EXCEPTION:** if you happened to be assigned to more than one project, then $SCRATCH may will point to a different direction. However, you will need to use the storage directory that is located in   
     
   /pylon5/ms560hp/   
     
   for this project.
2. Now we will use this path to create a symbolic link to the $SCRATCH that is accessible from Globus Connect.
   1. First, we make sure to switch to the $HOME directory, since this is the directory where Globus Connect will be placed initially. We do this using the following command:   
        
      cd $HOME  
        
      followed by   
        
      ls  
        
      to see the content of the $HOME directory. Here is the result in my case:   
        
      
   2. Next we log into Globus Online. And log connect to Bridges.   
        
        
        
      Note that what you see in the output of command **ls** matches what you in Globus online
   3. We will now create a link using command ln. We will name this link *work582D* (or you can use any other name that appeals to you). For this step you will need the path to your $SCRATCH directory that you discovered in step 1 above. Type   
        
      ln –s /pylon5/ms560hp/yourusername work582D  
        
      followed by   
        
      ls  
        
        
        
      The link with the name work582D is now can be seen in this directory
   4. Go to Globus Connect and refresh the $HOME directory look. It should now show the newly created link name in it.  
        
        
        
      clicking on this link will point Globus Connect to the $SCRATCH directory. If this is your first visit to $SCRATCH, the directory should be empty.  
        
      

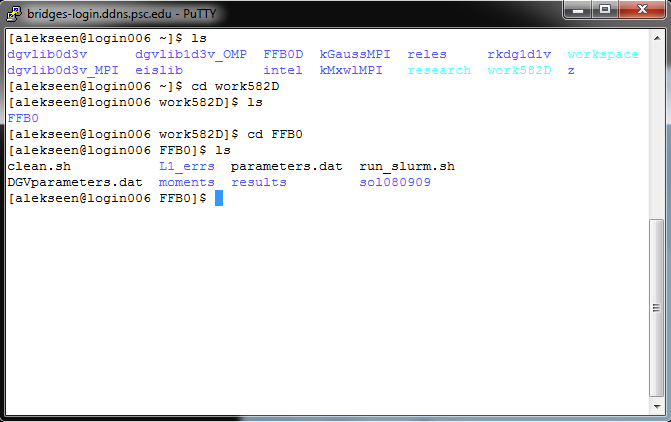
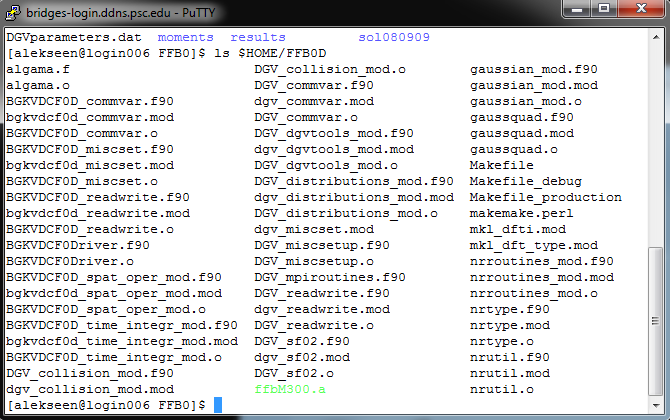
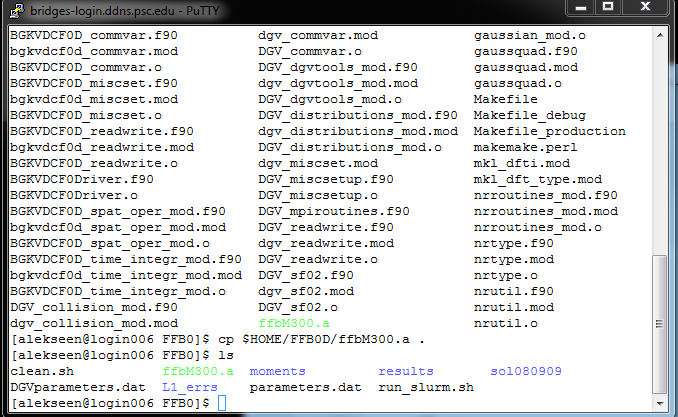
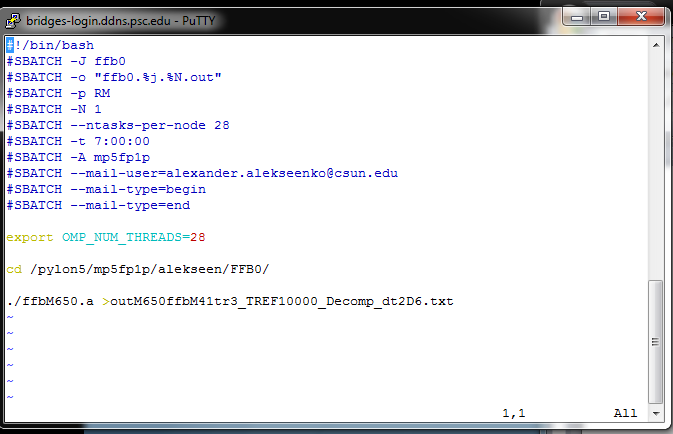
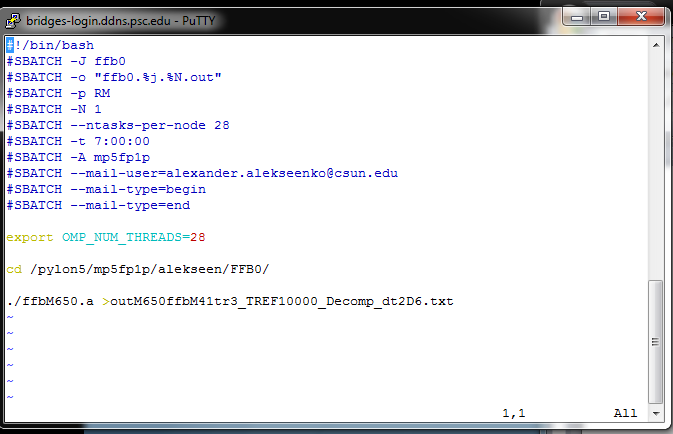
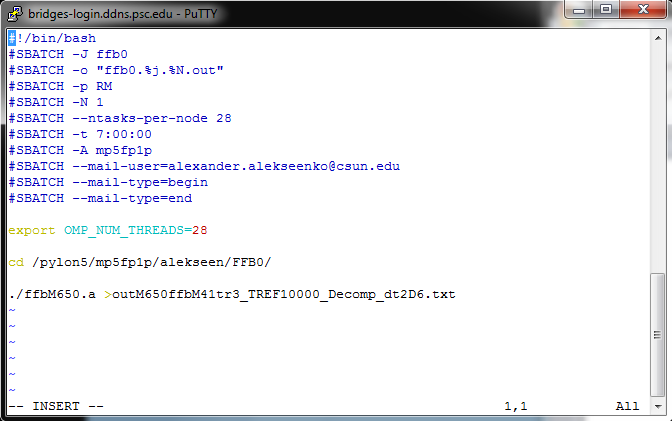
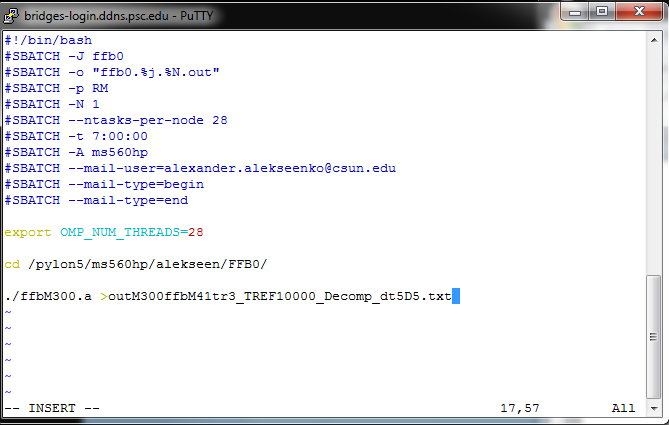
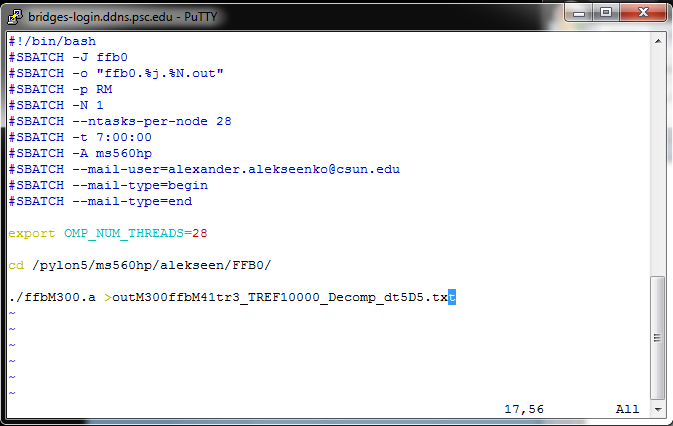
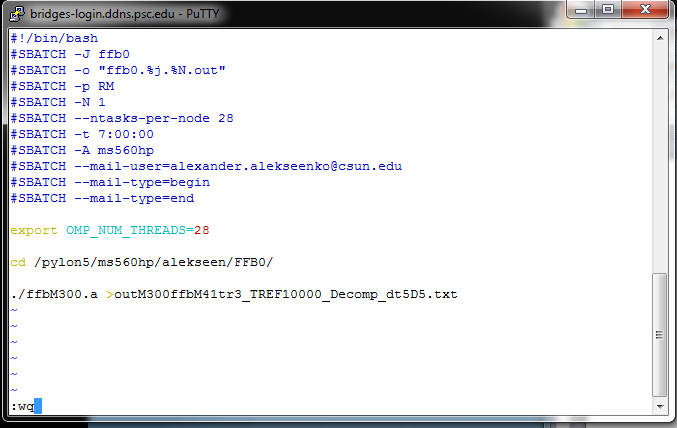
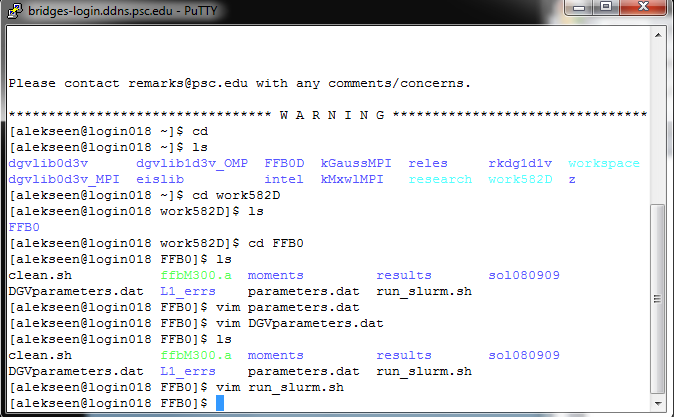
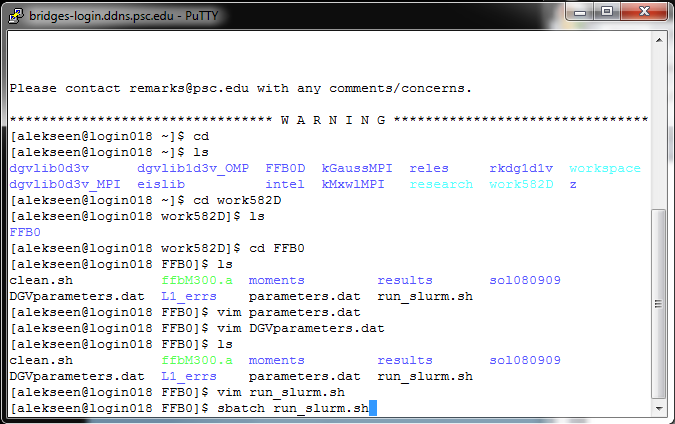
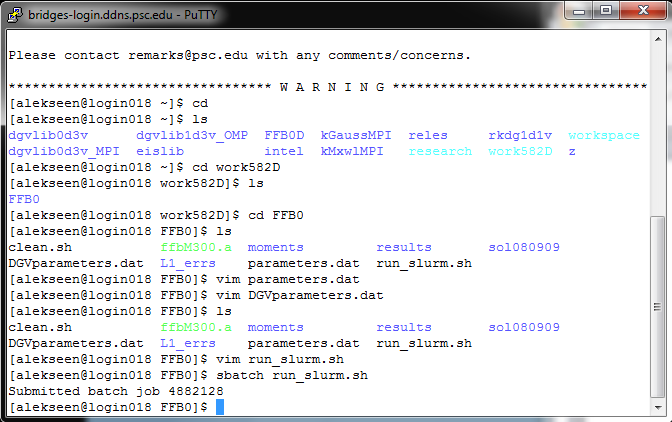
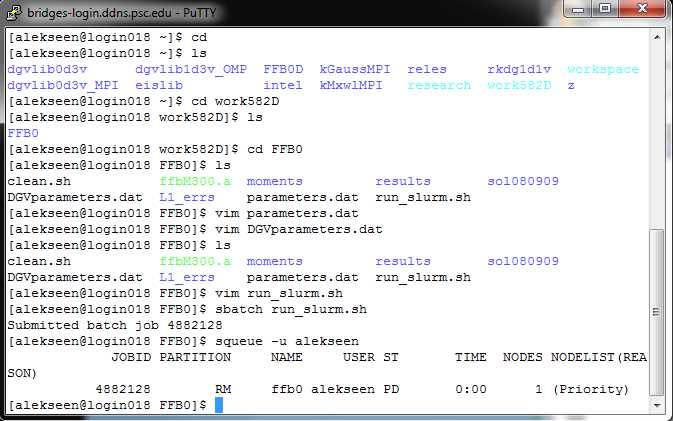
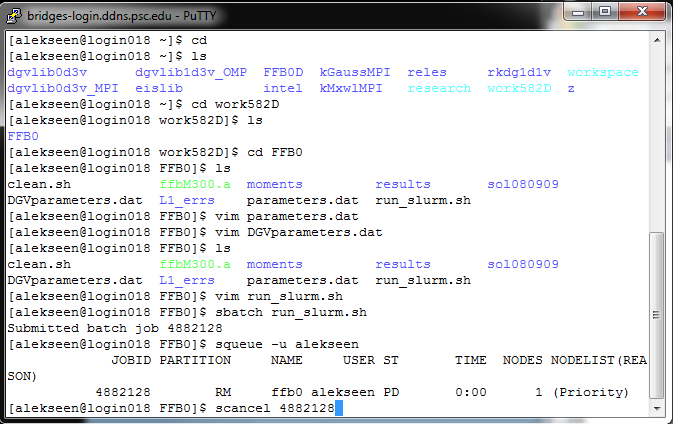
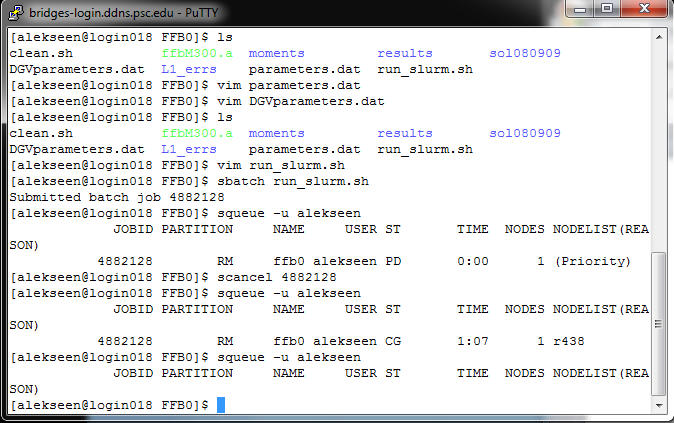
**Steps to set up the directory from which you will run the simulation.**

The code that you compiled in these milestones needs data to run. It also needs certain directory structure to be set up just right. To help you in this task, an archived \*.zip file is provided on the Canvas page that contains a clean copy of the working directory. Your goal is to copy the directory into $SCRATCH files space. Follow the steps below to complete this task.

1. Go to course Canvas page and locate the link to the archive \*.zip file  
     
    [A copy of the working directory for FFB0D code. FFB0.zip](https://canvas.csun.edu/courses/59553/modules/items/1746801)   
     
   You should see something like this:   
     
     
   
2. Use the link to download the file. It is about a half of Gigabyte. When downloaded to your computer, unpack. You should have a directory FFB0D with subdirectories and files in it. This is what is looks on my labtop:  
     
     
     
   Inside FFB0 we will see  
     
     
     
   Please note that all this structure has to be copied exactly to pylon5 to make sure the code will work correctly.
3. Place directory FFB0 on your computer somewhere where Globus Connect has access to it.
4. Next please go to Globus Connect and connect to Bridges.
5. Once you see the contents of $HOME directory, locate the symbolic link **work582D** that you created earlier and click on it. This should take Globus to your $SCRATCH directory.
6. Use the second panel of Globus Connect to navigate to folder that contains working folder **FFB0D.** You should see something like this in your Globus Connect:   
     
   ****
7. Click on **FFB0D** and select it. Then click on the Start button on the same panel to copy the folder as a whole to pylon5. The transfer will be submitted. It may will take a few minutes to begin. And it most likely will take a few minutes to complete. Once you receive the e-mail that the transfer has been completed, you should see the folder in the $SCRATCH directory. E.g.,   
     
     
     
   You can also login into Bridges using Putty and observe that the files have been transferred and a copy of working directory was established. Here is the sequence of commands once you log in:  
    **cd work582D  
   ls  
   **
8. **Now that the working directory has been transferred we have a few steps left to set up our first run of the simulation This will be discussed next.**

**Steps to set up a run of the FFB0D software.**

Now that we have the directory set up, we have just a few steps left before we can submit our first job into execution. We still need to place the executable file with our software in the working directory. And we will have to make a few adjustments to job submission script. Later we will also make changes to the parameters files. However, we will skip this step on the first run.

1. Login into Bridges using Putty or other command shell/terminal.
2. Navigate to Pylon5 and to the work directory you prepared in the previous step. Use command **ls** to see the content of the folder. See, e.g.,   
     
     
     
   Use you will see a bunch of files, including the parameter files parameters.dat and DGVparameters.dat. Also you will see the job submission script run\_slurm.sh. However, you should not have an executable file at this stage.
3. Our next task is to copy the executable file to the work directory. We recall that the complied code and the executable are located in $HOME/FFB0D. We will quickly refresh our memory on what is the name of the executable. An executable can be identified by extension “\*.a”. We use command   
     
   ls $HOME/FFB0D   
     
   to see the files of the compiled code:  
     
     
     
   We can see an executable file ffbM300.a
4. We now will copy this file into our work directory. Note that if you followed the steps precisely, you should be still located in your working directory. We can use the following command to cpy the executable. Please pay attention to the “.” In the end of the command. This “.” means “Copy the file into the current directory”.  
     
   “**cp $HOME/FFB0D/ffbM300.a .**”  
     
   We verify the result of the file copy by checking the contents of the folder one more time using command **ls**:  
     
     
     
   The executable file has been successfully copied!
5. The parameter files parameter.dat and DGVparameters.dat are pre-set for the first simulation. We will not visit them this time
6. The final step is to adjust the job submission script, run\_slum.sh. On all HPCs, users are not allowed to start compute jobs manually. Instead, all jobs are submitted to a scheduler. On bridges it is SLURM scheduler. There are exceptions to this rule, e.g., for some exotic cases of debugging. However, even for debugging, the jobs are typically submitted through the schedule using a special quick turn around debugging queue. The reason the jobs are submitted to the scheduler is to increase effectiveness of the HPC. Once the job is submitted, it will be placed in a queue. The Scheduler software makes assessment of the HPC load and resources and uses the jobs in the queue to make sure the HPC has maximum utilization. So, depending on the day, your job may will have to wait before it will be called into execution. Sometimes the wait is as long as a few days. The more resources your job requires, the more is the wait. Nights and weekends are best times to submit jobs that are resource intensive. Well you can submit a job anytime, but it may will get started overnight.   
     
   In this step we will visit the job submission script that is located in your work directory. Look for the file with the name run\_slurm.sh.   
     
   You can modify this file on your local computer and copy it back to the work folder. However, usually, we modify the script file right in the terminal. We will need a tool for opening files for editing in terminal window. A convenient tool is given by the command line editor **vim**. You are strongly encouraged to go online and search for **survival cheat sheet** of commands for **vim.** Having a cheat sheet nearby will help using vim in the future.
7. We open the file run\_slurm.sh for editing by running command  
     
   **vim run\_slurm.sh**  
     
   
8. There are a few lines that we need to change here. In the following we explain the meaning of each line and tell what change needs to be updated. But before we work on that we need to explain that the **vim** editor has two modes: the editing mode (INSERT or REPLACE) and the command node. As you work more with **vim**, the two modes division will make more sense. In the command node you navigate throw the files, that can be thousands of lines long, you write the file on the drive, you can search and replace, and other cool things. But to be able to do that you will need to learn **vim** commands. Hence the request to obtain a survival cheat sheet of commands that can be found on the Internet. For those of us (which includes myself) who did not have time (a.k.a. have been lazy) to learn the commands, there is an easy way out. The way out is the editing mode. You switch to editing mode by pressing **“i” button** and you quit the editing mode and switch to command mode by pressing **“Esc” button.** When you open a file in **vim**, the editor is in the command mode  
     
   By tapping on the key **“i”** we switch it into the editing mode. A word “INSERT” or “REPLACE” will appear at the bottom of the command window. They mean here the same thing as they mean in any other editor and can be toggles by pressing “Insert” key.   
     
   In the editing mode, you can use arrows to navigate the cursor. The mouse navigation will not work on PCs, and scrolling will give results you did not expect. Once you place the cursor in the desired position, delete and add characters as in any other editor
9. Let us now go over each line and explain its meaning.
   1. #!/bin/bash -- it is to tell what command shell to use. It has nothing to do with our needs. Do not change.
   2. #SBATCH –J ffb0 – this is a directive to the scheduler. It says to assign the name ffb0 to the submitted job. It will show up as ffb0 in the queue. Do not change.
   3. #SBATCH –o “ffb0.%j.%N.out” --- this is a directive to the scheduler telling where to direct output generated by the system. After the code is executed, a new file will be added to the working directory, its name will have ffb0 and the job number and the “.out” at the end. If the job completed with no errors, this file will be zero length. Delete it. If the job crashed with an error, the file will be non-zero length and will contain system messages. It is useful to review it to determine what caused the crash. Do not change this parameter.
   4. #SBATCH –p RM – this parameter tells the scheduler what queue you are submitting your job to. There are several queues available on Bridges. They are referred as partitions. RM stands for Regular Memory not shared. You can read about partitions in the Bridges User Guide (<https://www.psc.edu/bridges/user-guide/running-jobs#RM-partitions>). Partition determined on what matheboards and your software will run. In this course, we will use the Bridges regular memory nodes as opposed to large memory nodes or GPUs. In this case we are requesting all processors on the node, therefore we will not share this node with other jobs. If the job does not need all the processors, we can use RM-shared partition instead. Do not change this parameter.
   5. #SBATCH –N 1 – we are requesting one node for use to run our software. Do not change
   6. #SBATCH –ntasks-per-node 28. Each regular memory nodes has 28 processors on it. The FFB0D is a parallel software using the OpenMP shared memory parallelization. In OpenMP approach, big jobs can be divided between processors by creating parallel threads of the code, the parameter –ntasks-per-nodes tells how many threads will be created. It is typical to set the number of treads to be equal to the number of processors. However, in some hardware configurations and for some codes, one processor can run several threads effectively. In this case, you can use 56 or 84 threads. But this code will loose efficiency after 28 threads. Do not change the parameter.
   7. #SBATCH –t 7:00:00 --- this is a directive to the schedule to tell for how long we need the resources. The format is HH:MM:SS, so we ask the resources be given to us for 7 hours. If the job is not finished after 7 hours, it will be killed by the scheduler. So, it is important to have an accurate estimate on how much time we need. The set up simulation should need about 5 hours of time. YOU CAN CHANGE THIS PARAMETER TO A DIFFERENT TOTAL TIME.
   8. -$SBATCH A XXXXXX – this parameter tells what project will be charged for running the simulation. PLEASE UPDATE THIS LINE. The project for this course is ms560hp. So replace this line with   
        
      #SBATCH –A ms560hp
   9. #SBATCH --mail-user=youremailhere ---- this is a directive to the scheduler to send an e-mail. The next two lines clarify when to send the e-mail. In particular, we are asking to send e-mail when the job starts and when the job finishes. PLEASE MODIFY THIS LINE by pointing it to your e-mail. Otherwise I will be getting messages from your job.
   10. export OMP\_NUM\_THREADS=28 Before an OpenMP job starts, it is convenient to set this system variable to the desired default number of threads. Do not change
   11. cd /path to your working directory – when the job is sent to execution by the scheduler a terminal will be created, like the terminal you use to login to Bridges. We need to change the directory from wherever the terminal points by default to the working directory that you have prepared. CHANGE THIS LINE. Use the path to your working directory on Pylon5, e.g.,   
         
       cd /pylon5/ms560hp/yourusername/FFB0/
   12. The last line calls the executable file into execution. Specifically,   
         
       ./ffbM300.a   
         
       will call executable file ffbM300.a that you prepared and placed in the working directory. The software creates output that can be seen in the terminal. However, this output will be redirected to the text file with a very long name using the command “>”. The long name allows peek detail about the simulation so it will be easier identified later. CHANGE THIS LINE like this  
         
       ./ffbM300.a >outM300ffbM41tr3\_TREF10000\_Decomp\_dt5D5.txt
   13. After all the necessary lines are adjusted, we should see something like this:   
       
   14. Click “Esc” to exit the editing mode:  
       
   15. Type command “:wq” – write the file on disk and quit  
         
       Hit “Enter”  
         
       you will return to the working directory.
10. You are now ready to submit the job to the schedule. This is done using command  
      
    **sbatch run\_slurm.sh**   
      
      
    hit “Enter”. If all is well, the schedule will accept the job:   
      
    If the schedule rejects the job, go back to steps 8 and 9 to make sure all the lines are as they should in run\_slurm.sh.
11. You can check the status of your job by using command  
      
    **squeue –u yourusername**  
      
    
12. Sometimes your submit the job into execution and then realize that your parameters are wrong or something else is wrong. In this case you want to cancel the job. You can do it using command  
      
    scancel #yourJOBID  
      
    The JOBID can be seen in the output of squeue command. For example to cancel the job We just started, we do this   
    hit “Enter”. Then call squeue again a couple of times to see that the job is being cancelled: 
13. Once the job is cancelled, correct the errors and submit the job again. When you see “R” next to your job’s name, in the ST column, this means that the job has started and is running!