## So you want to try molecular dynamics

So you want to try molecular dynamics (MD) – lets quickly talk about what it’s useful for. MD simulates the forces between atoms and molecules, but does not simulate the motion of electrons. This means it’s useful for figuring out the how things interact and their 3D structures/conformations. If you want to simulate reaction, you need to use quantum mechanics instead. If you want to try a large number of things very quickly (folding, which molecules fit in an active site, etc), you’re best off starting with a webserver which can do molecular docking or folding predictions. The method here has been used for – conformations of reactants, determining whether reactants will co-localize, and estimating interaction preferences of flexible small molecules. It may work for other stuff too, who knows.

There are other methodologies around, and if you can find someone to teach you properly, it’s definitely best to work with them. It’ll save time and they can help you with the inevitable bugs. Everyone recommends gromacs, which does have tutorials online, so definitely consider that. I only came up with my own method because the gromacs tutorials were for large biological molecules which I was not interested in, and I couldn’t figure out how to reapply the methods to my questions.

Anyway, let’s get on with it:

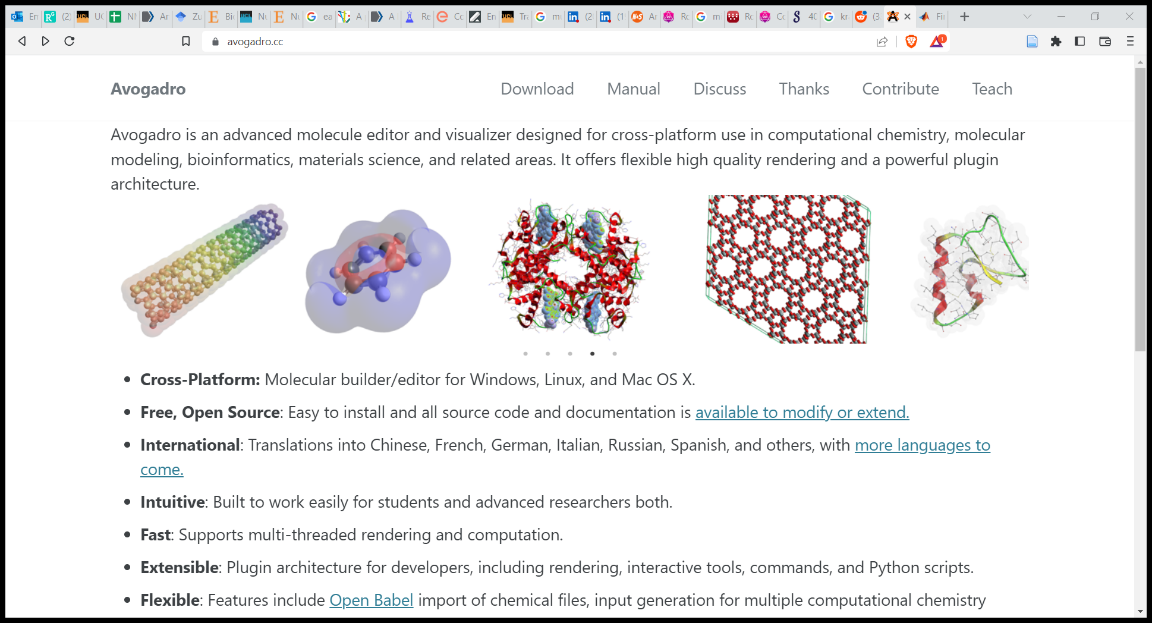
## Level 1: Software

This pipeline uses the following software: Avogadro, matlab (with the MDtoolbox extension), VMD, git bash, and WinSCP. It also uses the CHARMM-GUI website and UCL’s Myriad cluster, both of which need you to set up an account. We’ll go through how to get each of these and what the point of them is. Finding all these was originally a pain in the ass, but downloading and setting these up shouldn’t be too troublesome (hopefully). Just pick the default settings for everything if you can.

### Avogadro

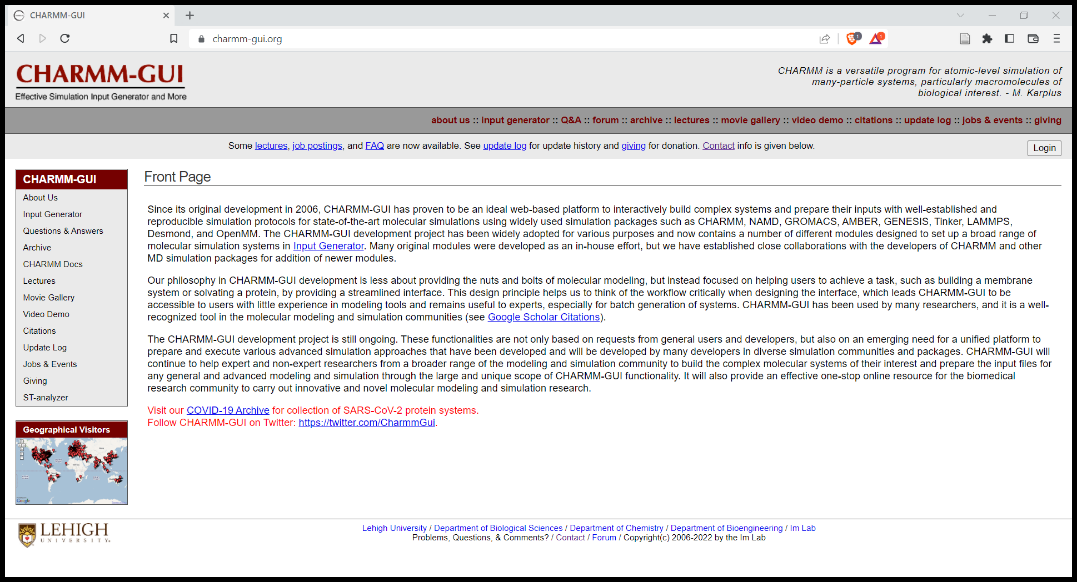
Avogadro is a molecular modelling software that we use for drawing the structure of the molecules you want to simulate. It is free to download and use. It exports structural files we upload to CHARMM-GUI.

<https://avogadro.cc/>



### CHARMM-GUI

https://www.charmm-gui.org/

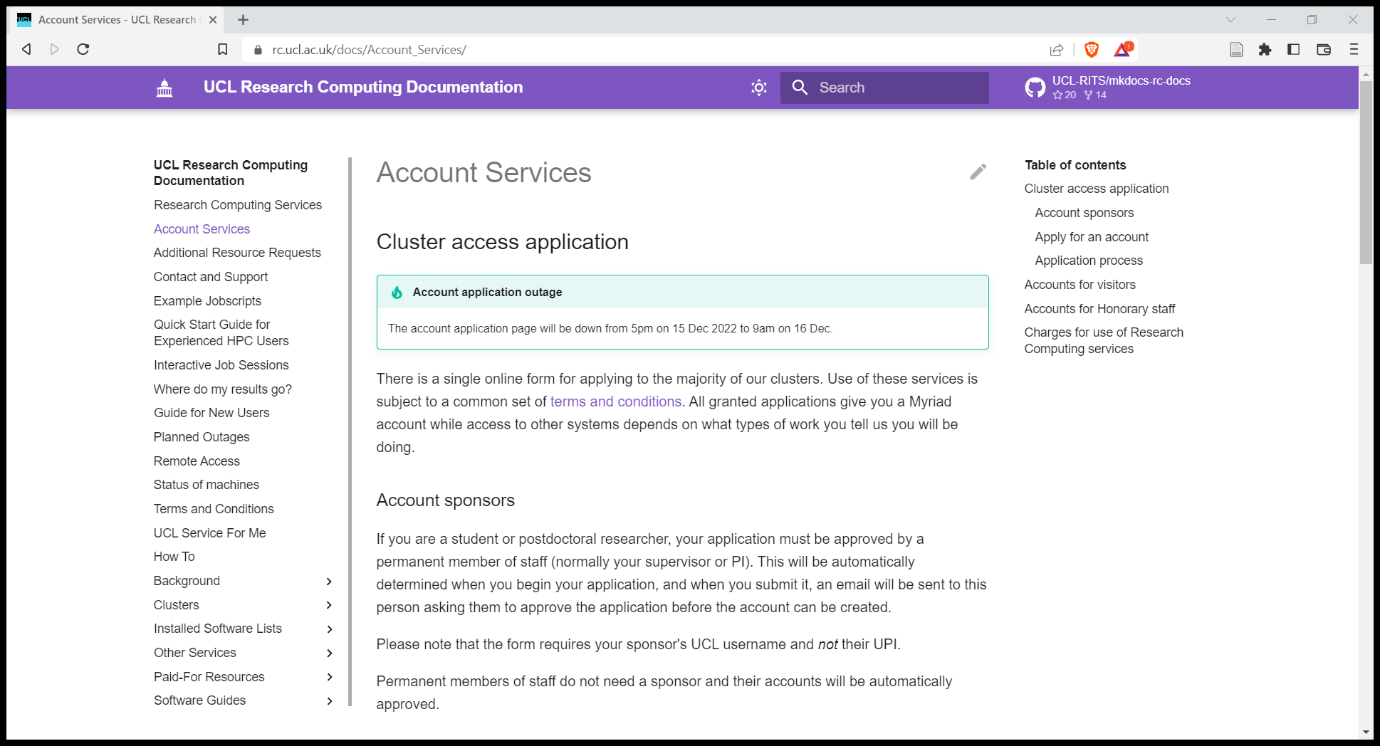


CHARMM-GUI is a webservice that streamlines much of the pipeline. You will need to create an account, which is free for academic use. It has many uses, but for this pipeline we will mainly use it’s “Ligand reader and modeller” and “Multicomponent assembler”, which will accept the files from Avogadro, perform paramaterisation (figuring out how the forcefields will interact with them), add water and ions, and then output files ready for sending to the cluster where NAMD 2 (an MD software package) will perform the simulations.

### UCL Myriad Cluster

<https://www.rc.ucl.ac.uk/docs/Account_Services/>

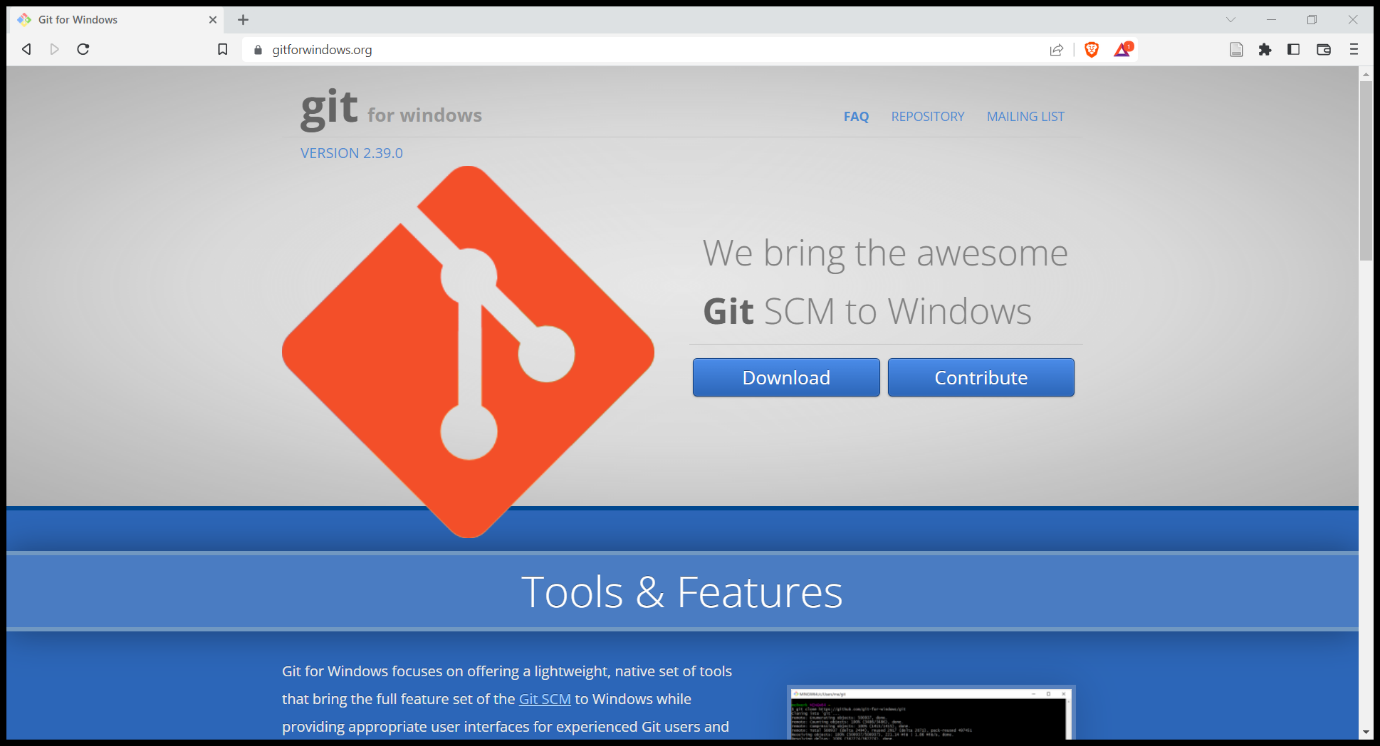
You’ll need to set up an account on the cluster in order to run all the simulations. This requires you to fill in a form about what you want to do, what support/resources you need, and who your boss is. Fill it out as best you can, but don’t stress too much, we never had any issues getting approval. You can just request low amounts of resources and it should be fine. Myriad is the free access cluster, which grants a terabyte of storage space and some good computational resources (with a few restrictions). All the information about this (and much more, including scripts, bug fixing, and other tips) is on the UCL research computing website which is surprisingly helpful given that its UCL.



### Git

https://gitforwindows.org/

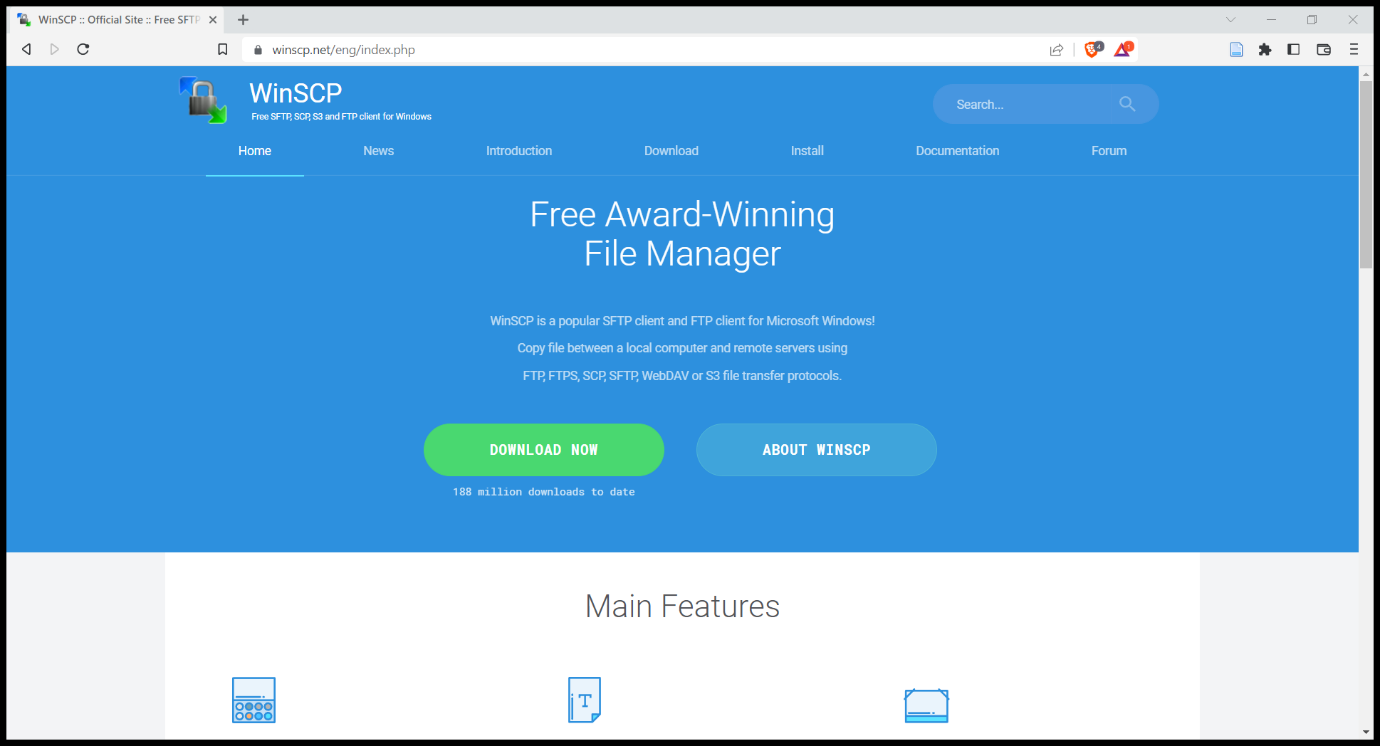
In order to interface with the cluster (on windows at least), you need a bash terminal. Mac has one of these in built I think. There are also other options available for windows (not sure what though), in case this doesn’t work.



### WinSCP

<https://winscp.net/eng/index.php>

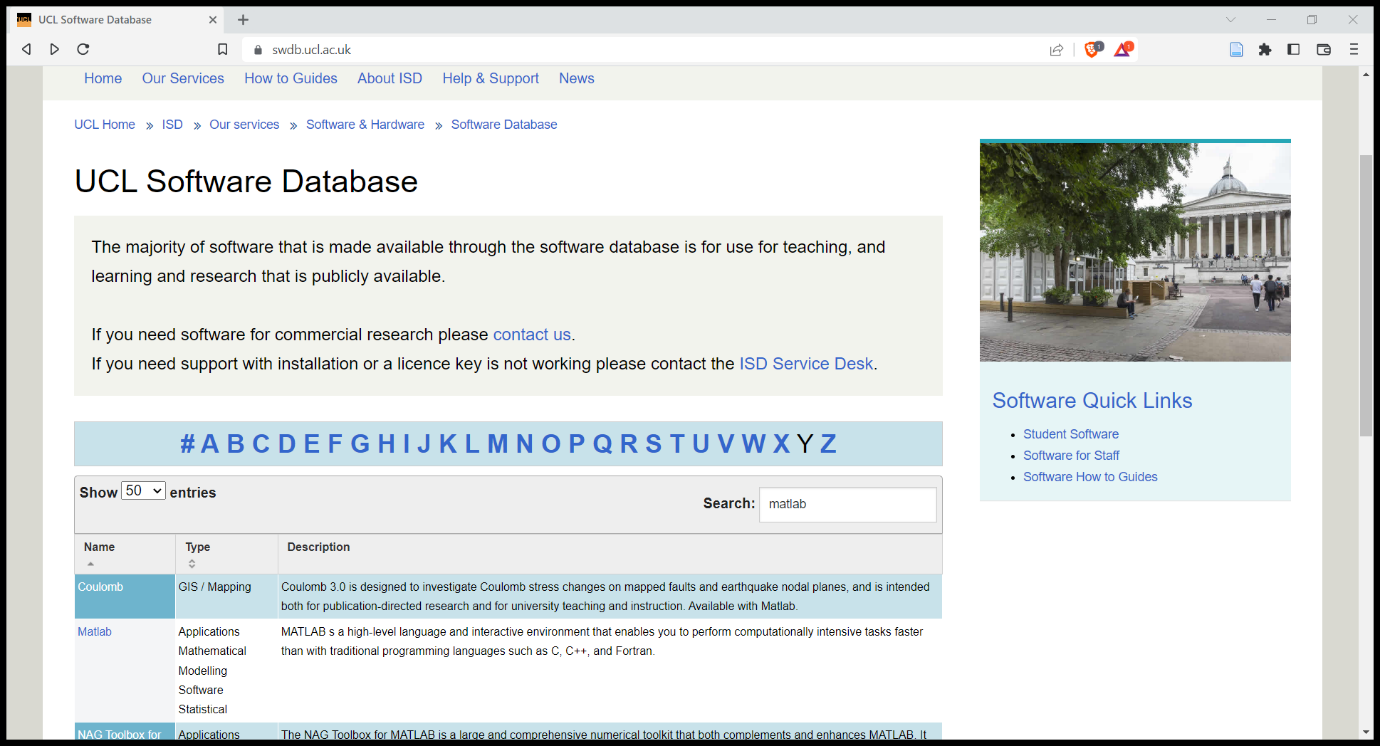
WinSCP allows you to easily transfer files too and from the cluster. There are other programs which work for this in a very similar way, or you can do it manually via the terminal (would not recommend).



### MATLAB

https://swdb.ucl.ac.uk/

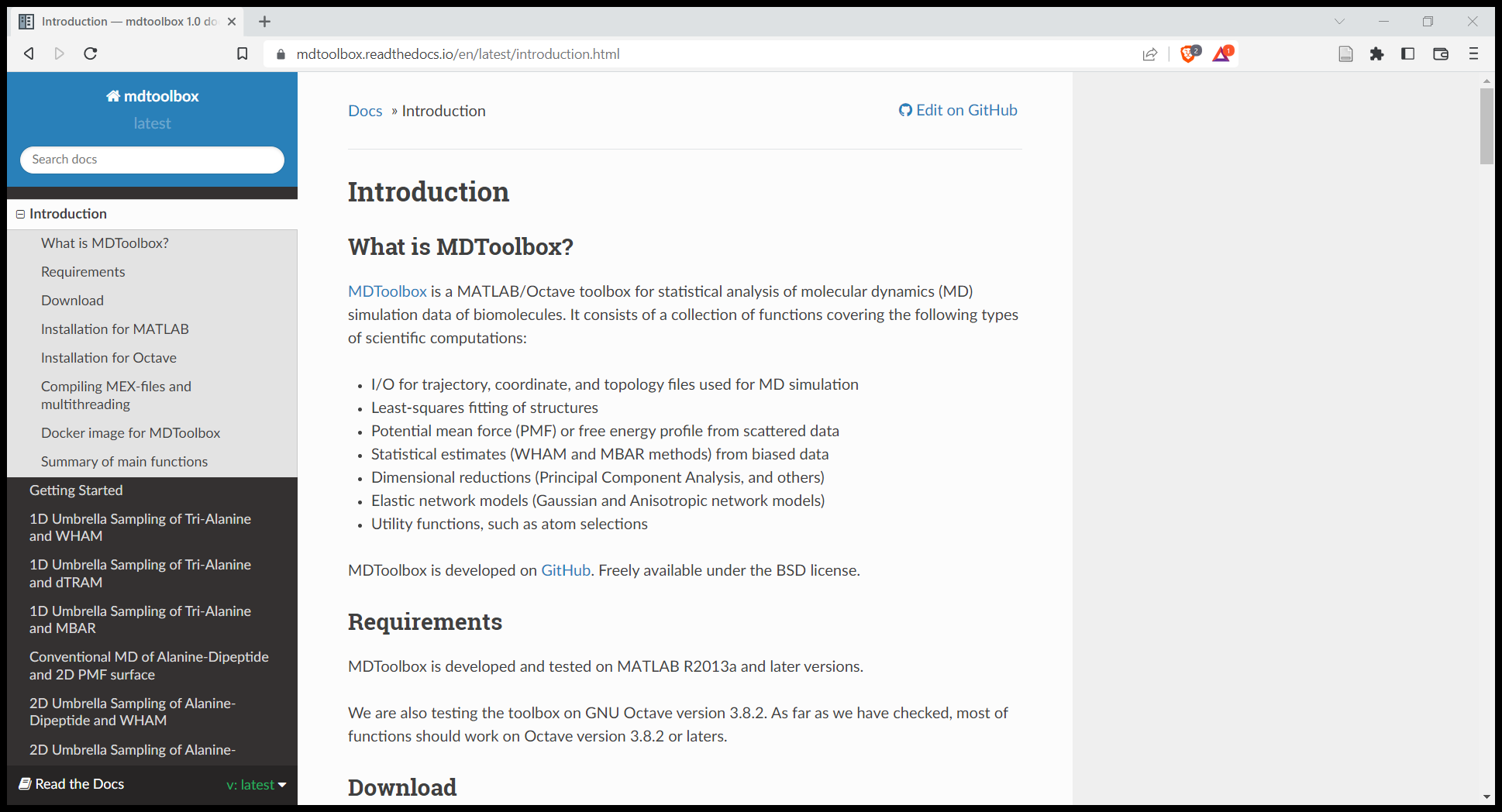
All of the analysis scripts are written in matlab. The only reason for this is that it’s the language I know best. There are similar analysis pipelines for python and there are people around the office who could probably help with that if you really prefer. You’ll need to download matlab via the UCL software database and follow the various steps to get a license from there. This is by far the most time consuming set up step (matlab is big).



### MDToolbox

<https://mdtoolbox.readthedocs.io/en/latest/introduction.html>

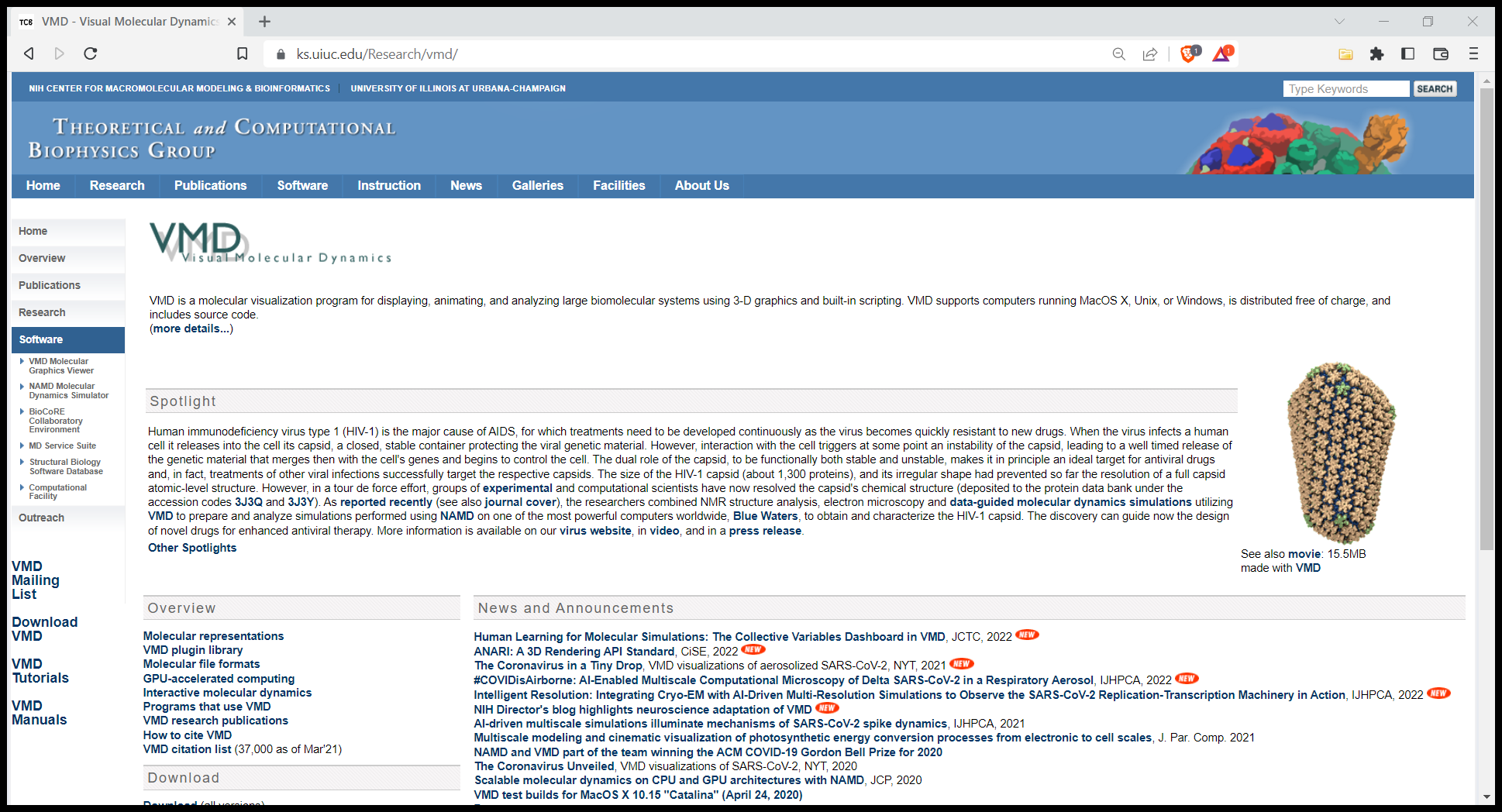
Matlab isn’t able to read the output files from the simulations on it’s own, so you’ll need to add this toolbox. The analysis scripts use a few functions from this (this is noted when it’s the case). Actually adding the toolbox into matlab was not super easy even once it was downloaded – MDtoolbox has a tutorial for this [https://mdtoolbox.readthedocs.io/en/latest/introduction.html#installation-for-matlab]. I think I used the “pathtool” route when I did this.



### VMD

<https://www.ks.uiuc.edu/Research/vmd/>

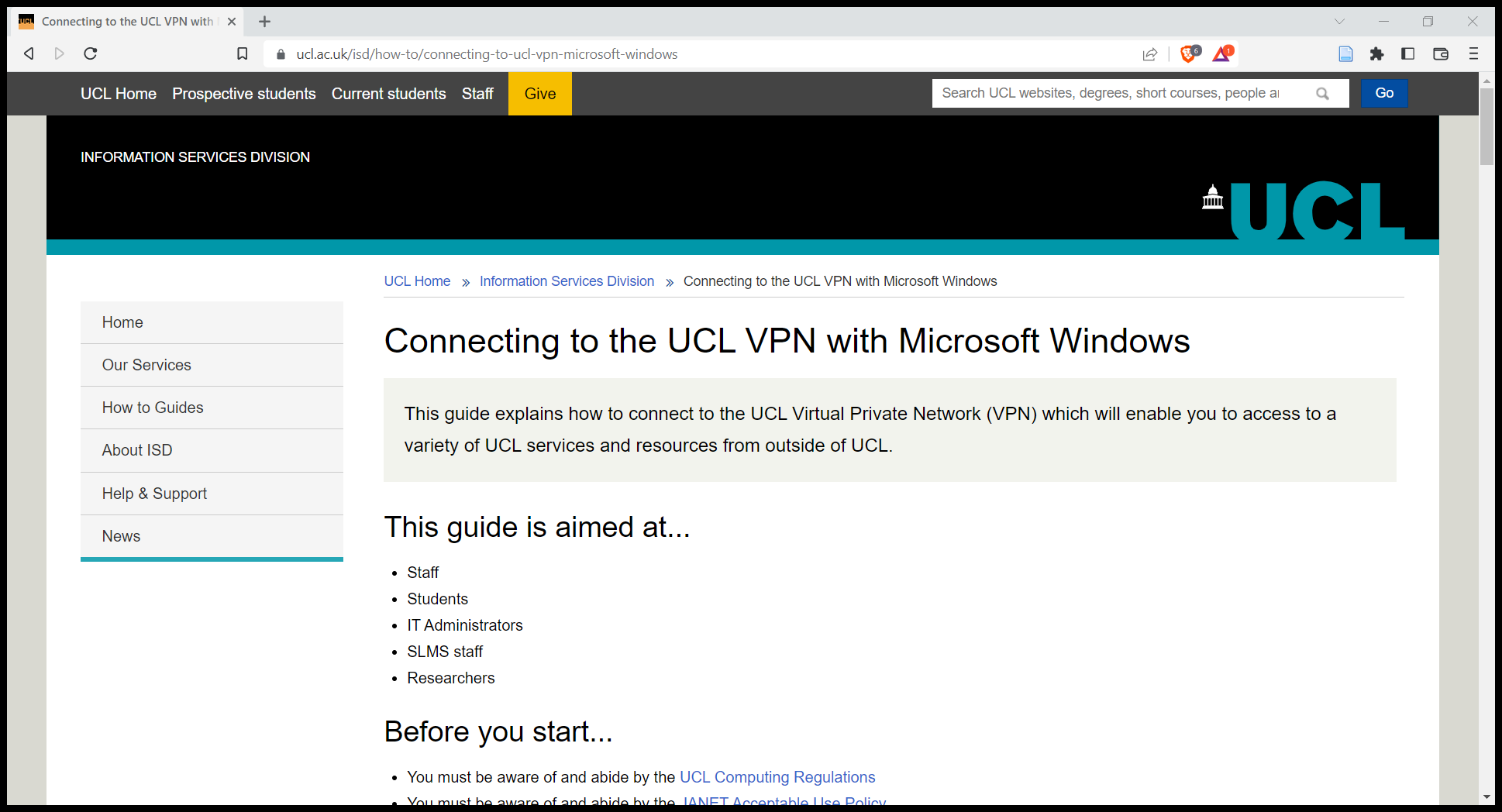
VMD is software for visualising the outputs of MD simulations. This is super helpful for checking whether everything worked, and getting an intuition for the system’s behaviours. It’s also essential for checking how different atoms are named (which you need to know for analysis), because the pipeline labels things very chaotically.



### OPTIONAL: Cisco

https://www.ucl.ac.uk/isd/how-to/connecting-to-ucl-vpn-microsoft-windows

If you ever want to access the cluster to start a simulation or download your data while you’re off campus, you’ll need to use UCL’s VPN. Definitely worth getting this, but be warned, downloading data via the VPN can be veeeery slooow.



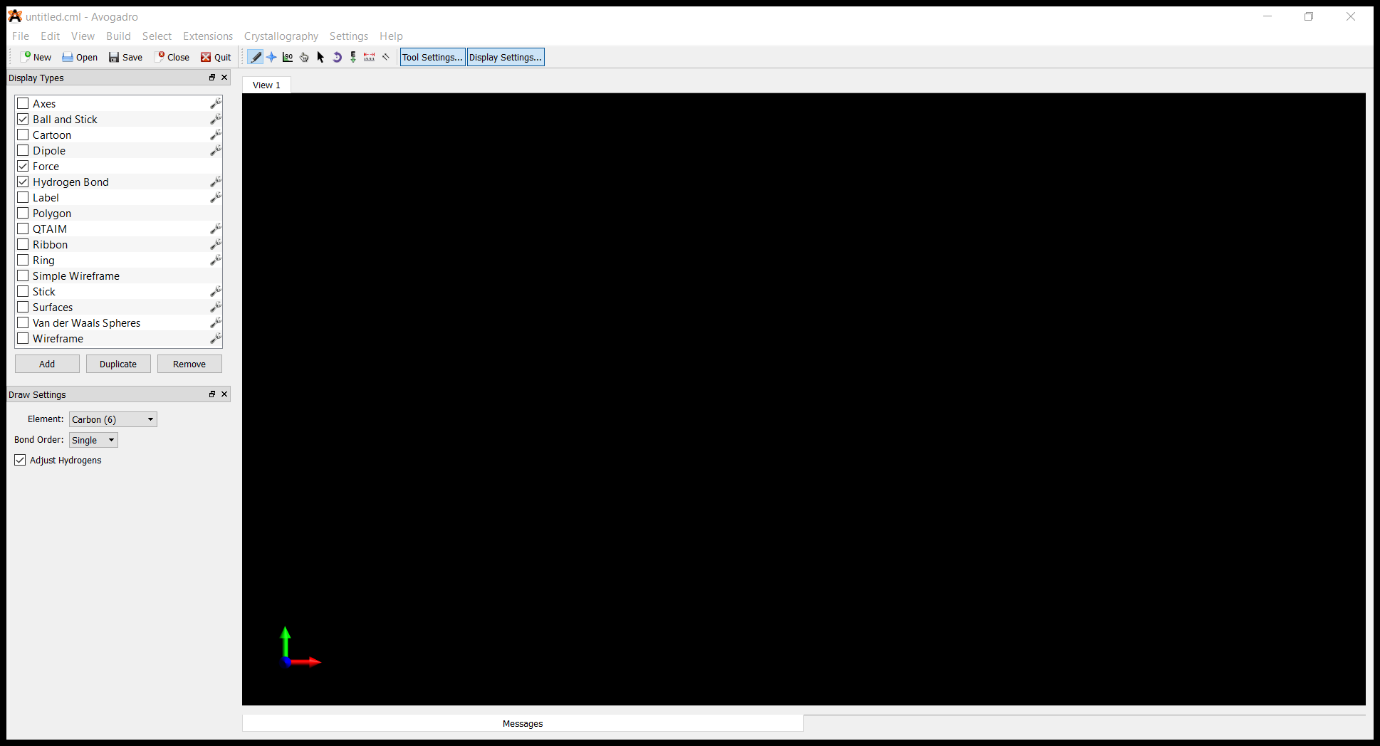
## Level 2: Setting up a simulation

Ok, well done for slogging through all the downloads. We’re gonna try and get a simulation ready to run now – be warned: the pipeline will try to screw up your molecules. At any moment, you may lose or gain random hydrogens, or have a chiral bond switched, or god knows what else. Be on guard at all times.

### Step 1: make a molecule

Fire up Avogadro. From here, you can either draw a molecule by hand using the draw function to add an remove individual atoms and bonds, or you can add whole common biomolecules/polymers using the “build” menu. Avogadro has a full tutorial on their website if you get stuck.

Drawing moleclues:



Or Build; Insert; Peptide:

Graphical user interface, application

Description automatically generated

Also, note the option to add hydrogens for pH! This seems to work reasonably well for amino acids and nucleotides.

Once you have built the molecule and checked it is exactly as you want it, save it as a “.mol2” file:

File; Save as;

Graphical user interface, application

Description automatically generated

### Step 2: Paramaterize the molecule

We need to upload the mol2 file to CHARMM-GUI in order to files with info about how the molecule will behave. Login to the website, then go to Input Generator, then Ligand Reader and Modeler

Input Generator; Ligand Reader and Modeler

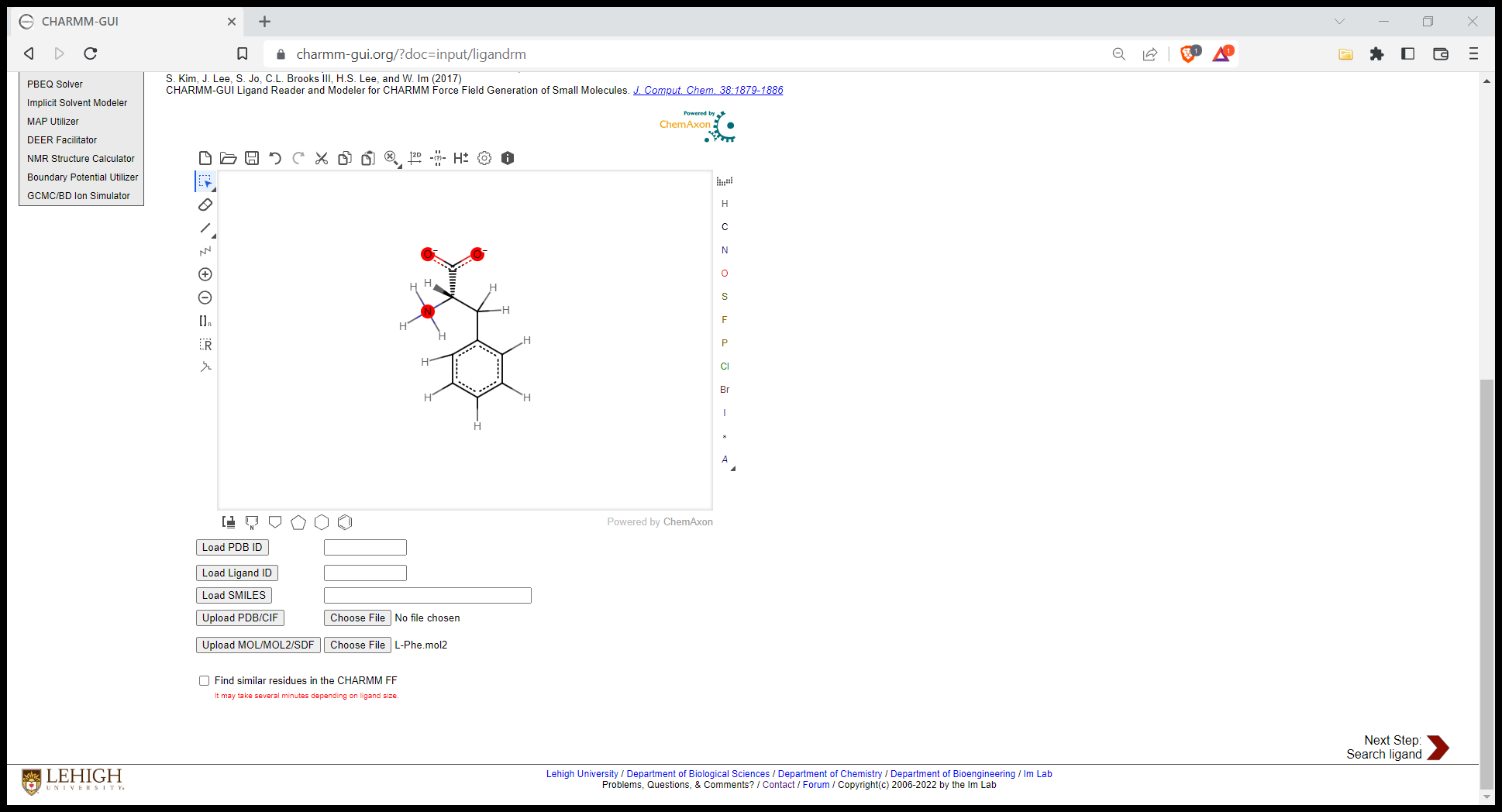
Graphical user interface, text, application

Description automatically generated

Find the “upload MOL2” button and upload the file. Click past the “check all hydrogens!” warning, and then you better check all the damn hydrogens. And all the bonds. And all the charges. And all the stereochemistry. Because computers are stupid and CHARMM-GUI is an evil demon who must be wrangled.

FYI, you can draw the molecule here yourself if you want, but for larger molecules it will be time consuming. You can edit the molecule using the assorted relatively intuitive buttons.

Note any red atoms are ones it already knows are wrong but doesn’t know how to fix, but literally everything still needs to be checked.



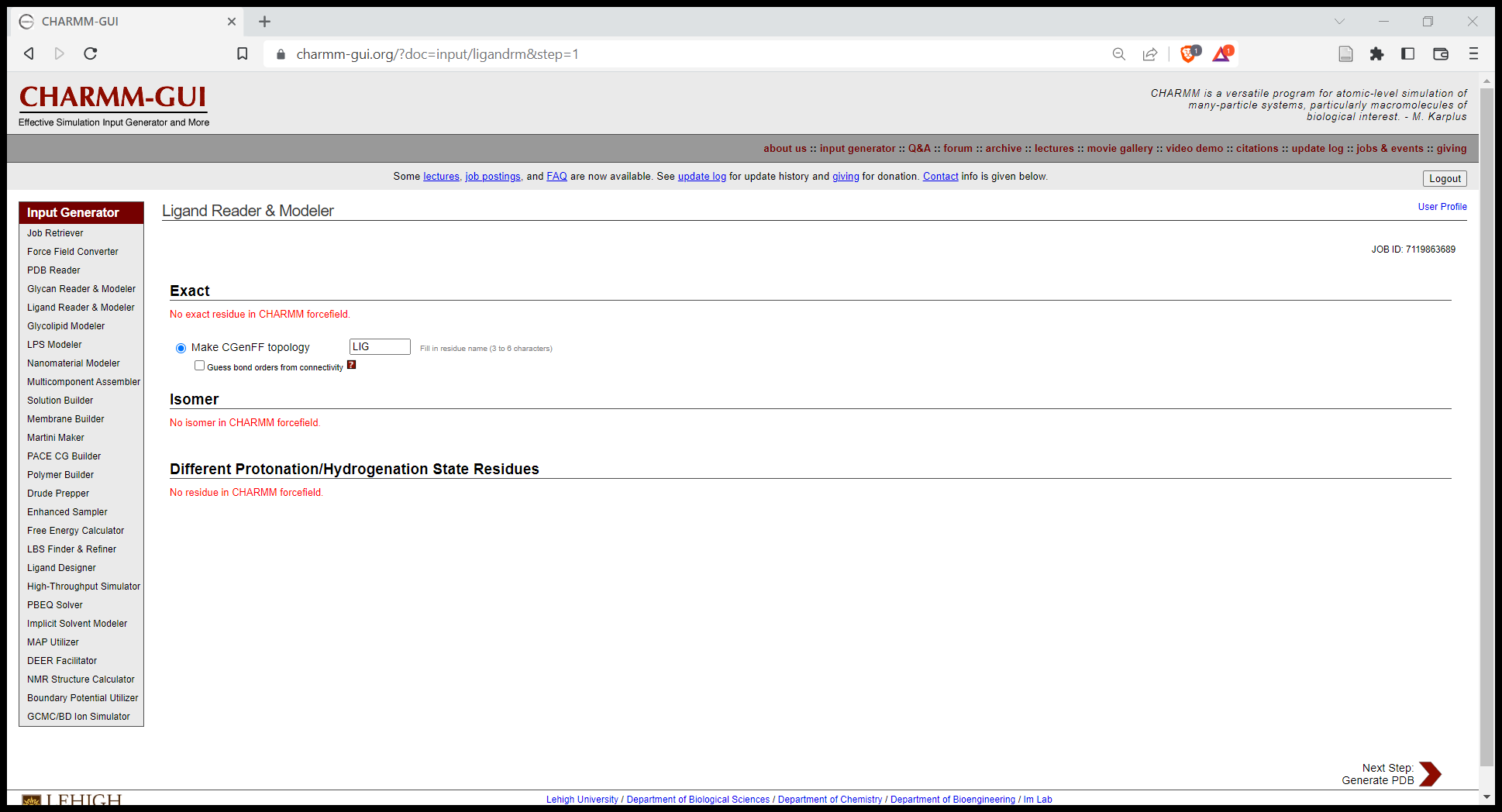
Click next step in the bottom right.

Now you need to name your molecule. You can just leave it as “LIG”, but be careful here. Renaming it will make it much clearer, but if it has a name which isn’t “LIG”, some forcefields (cough cough AMBER) can crash. BEWARE: This also means if the software finds a match for your molecule in the database and auto names it, this can kill the simulation in AMBER.

IF the software does correctly guess your molecule, you’ll have a few options to use the premade version. If you choose to do this, check the suggested molecule!!!! These are not always correct (I’m talking charges, hydrogens, stereochemistry again – CHARMM loves to get things wrong).

Also, note the JOB ID in the top right, this is useful if you ever need to contact the developers.

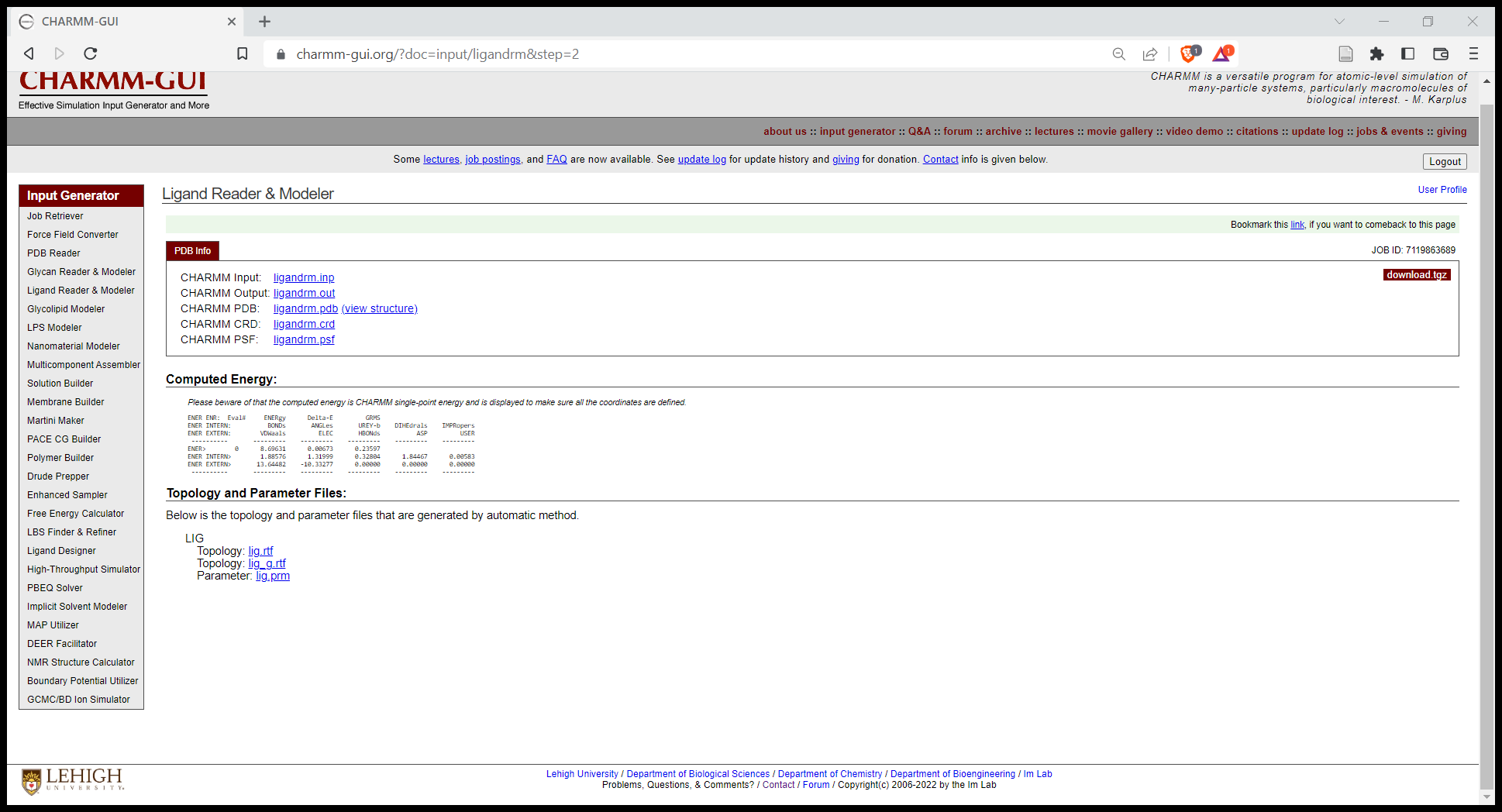
Danger zone – naming the molecule

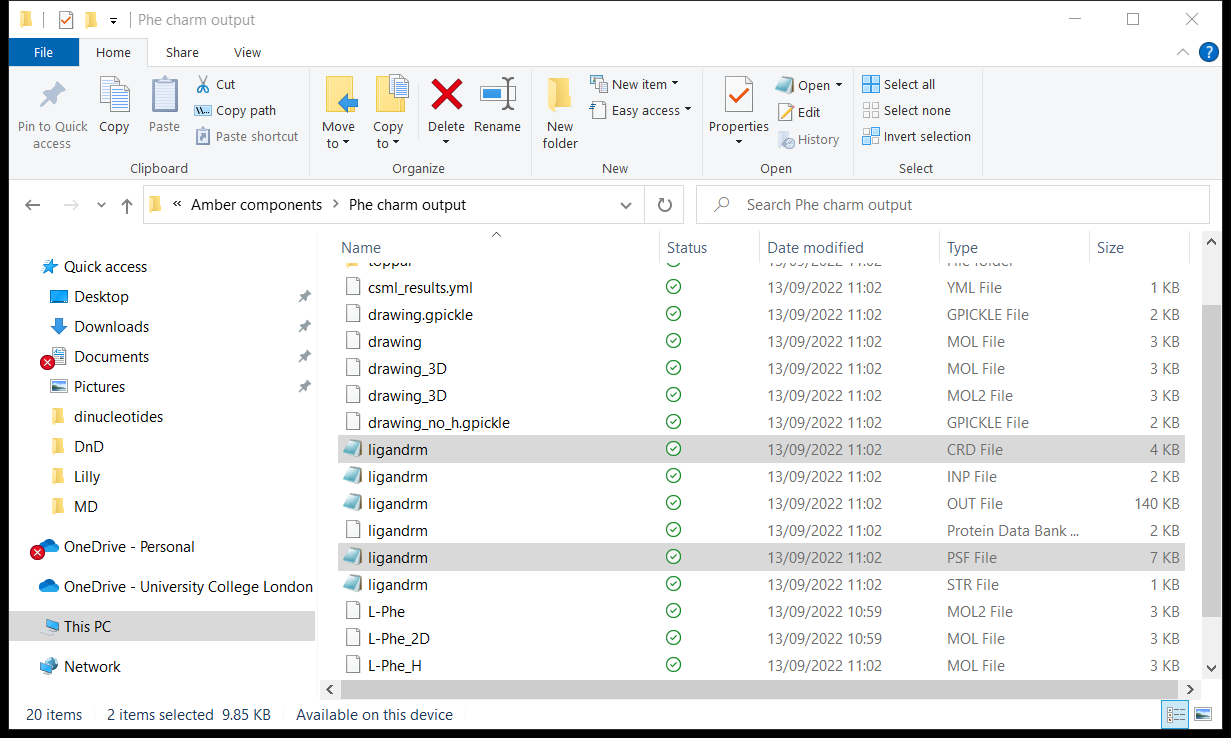


Click next step and wait for the server to do it’s thang.

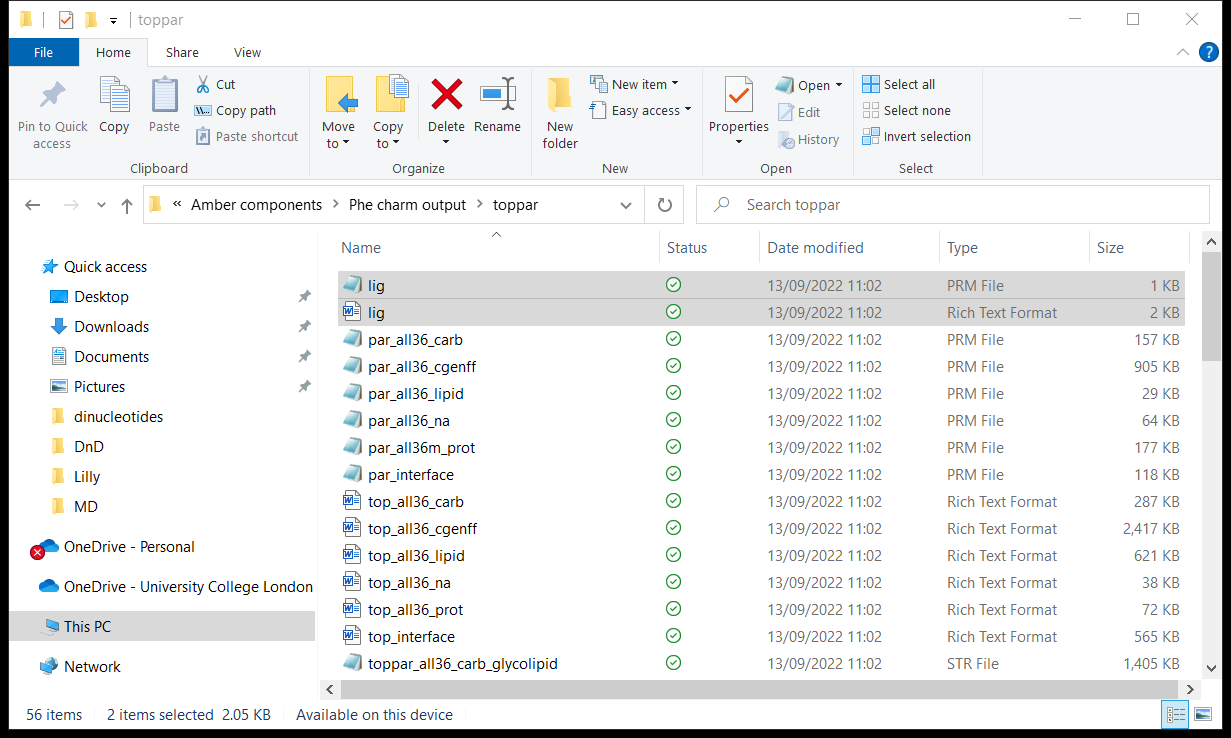
It will take you to the final page where the output files are ready. Double check the molecule structure by clicking “view structure”. Note, for nucleotides, it sometimes uses a DNA style model so it will just look like a line, or sometimes nothing at all. Don’t worry about this, you can double check the PDB file in VMD through the standard VMD method we use later.

Click “download.tgz” to download a compressed folder with the useful files in. This needs to be unzipped *twice*, and it’s may be useful to rename the files to something memorable.



You will need to find several specific files for the next step. Firstly, the PSF and CRD files:

Then in the “toppar” folder, get the PRM and Rich Text Format files (These will only be present if the Ligand Modeler did not identify an exact match for your molecule/you didn’t use the exact match):



Copy these to somewhere accessible/memorable, and consider renaming the files to be more clear as well.

Repeat all of this until you have all the organic molecules you want to simulate.

### Step 3: Prepare the simulation inputs

With the molecules downloaded and unzipped, you can now move onto the next preparing the whole simulation system. This involved immediately reuploading the files.

Go to the Multicomponent Assembler, and drag and drop the PSF, CRD, and PRM/Rich Text Format (if you have them) files from the previous step, then click next step.

Once you’ve gone through this process a few times, you can parralellize it by running it in several tabs simultaneously. This is worth while as each step is repetitive, and there is often a few minutes of waiting time while the server does it’s maths magic.

Graphical user interface, application, Word

Description automatically generated

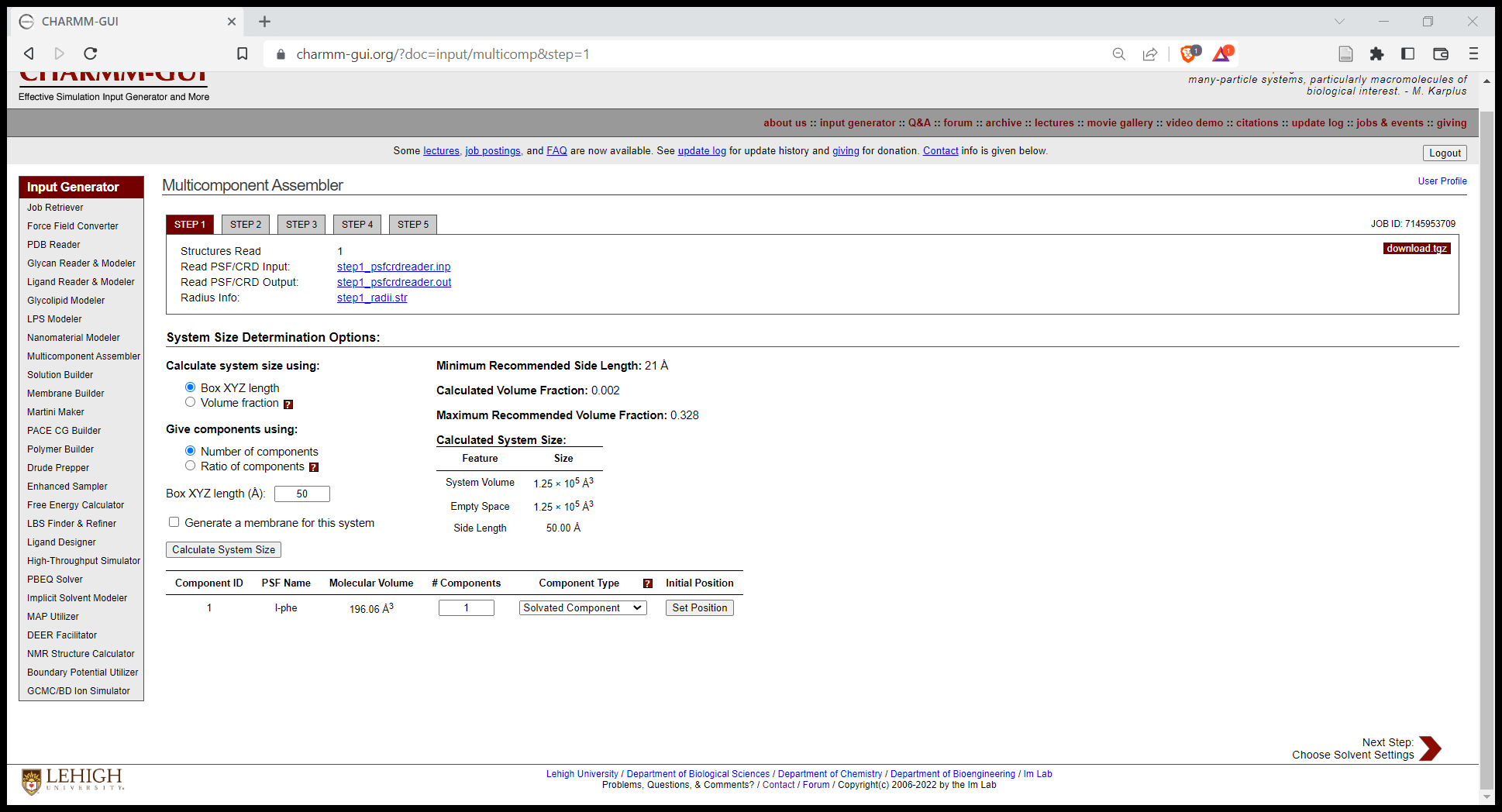
The next step is where we will choose the number of key molecules, and the size of the system.

Set the size of the periodic box\* to what you want - boxes increase in volume exponentially and bigger simulations will take longer to simulate. You will also have fewer collisions between molecules (if that is of interest) in a bigger box. Smaller simulations are much faster, but you need to be careful that the periodic box isn’t so small that things can directly (or indirectly through agglomeration) interact with themselves. The default is 50, but I usually used 40A for my simulations. Other sizes are fine, but you’ll definitely want to check the end results in VMD after the simulation is complete to make sure theres nothing weird going on.

Bare in mind that for consistency, its best to do all simulations you wish to compare in the same size box, so it may be best to leave some extra space for safety.

You will also see that you can adjust the number of molecules (components) in the system at this step (aka within one repeat of the periodic box). Do that, then click “calculate system size” and proceed to the next step.

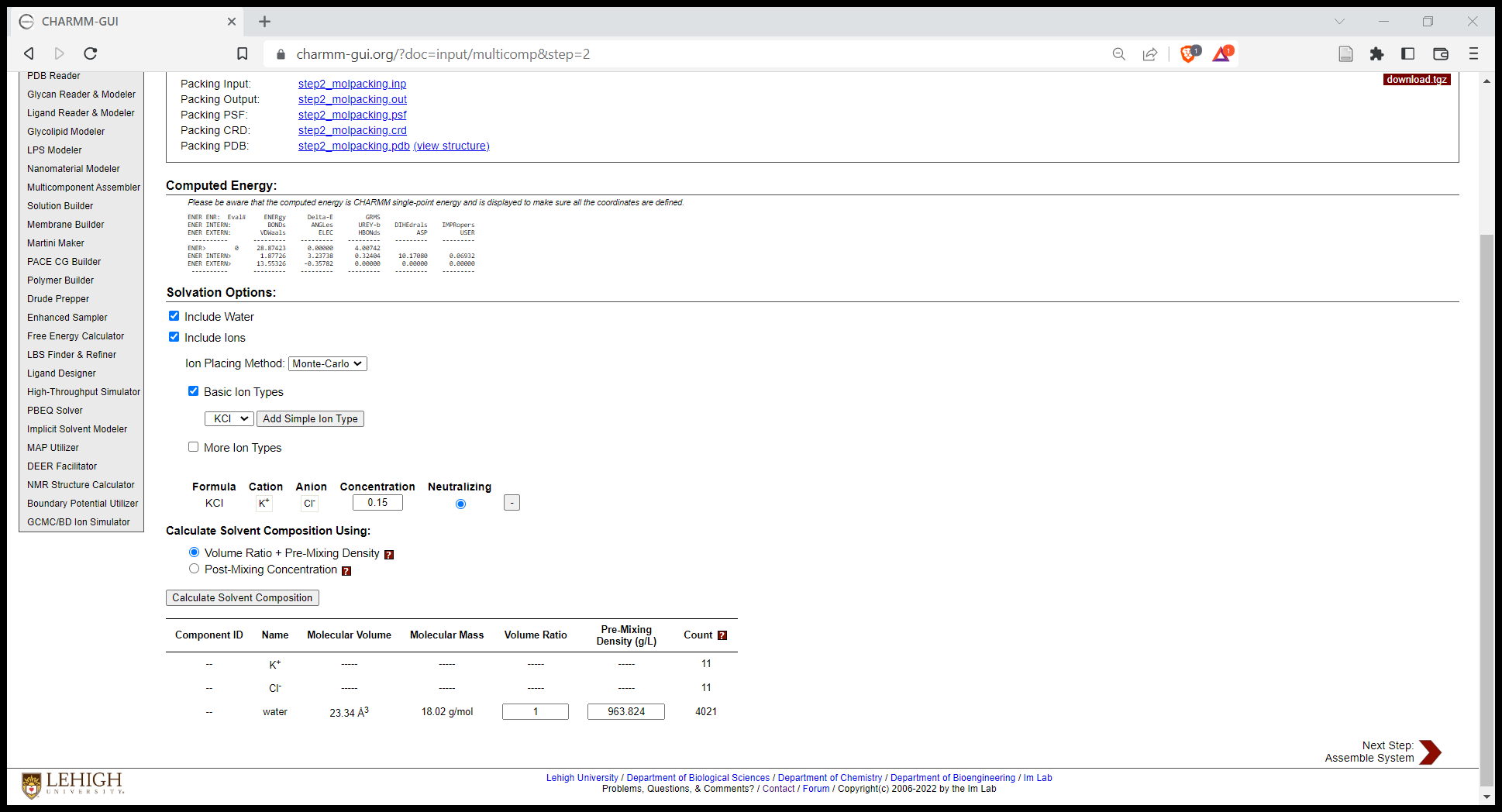
\*periodic box basically means the simulation repeats spatially – if a molecule goes out one side of the box, it comes back in the other side.



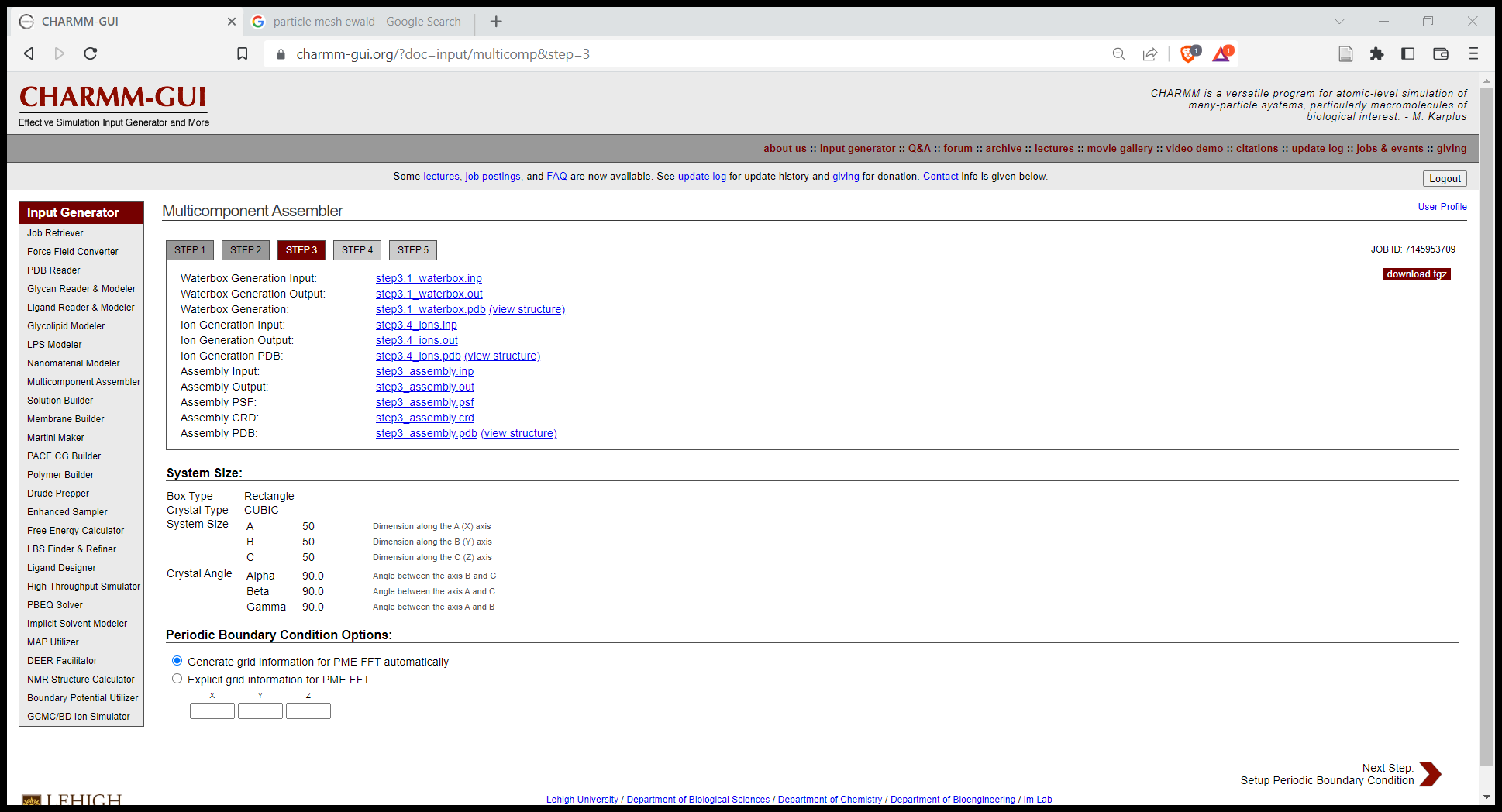
We now need to add ions (and water). These are essential to balance charge in most systems.

1. Change the ion placing method to Monte Carlo (not essential, just speeds up the “equilibration” step down the line)
2. Add any specific ion types you need (this is a balance between keeping the system simple and making it as realistic as possible – ideally try and match it to any experimental systems you have as closely as possible)
3. Adjust their concentrations and pick which one is “neutralizing”. The neutralising one will be used to counter any excess charges in the system. Next click “calculate solvent composition” and check the number of ions in the following table. You will often have to adjust the concentration in the previous step several times to get the number of ions you want in the system.

Click next step.

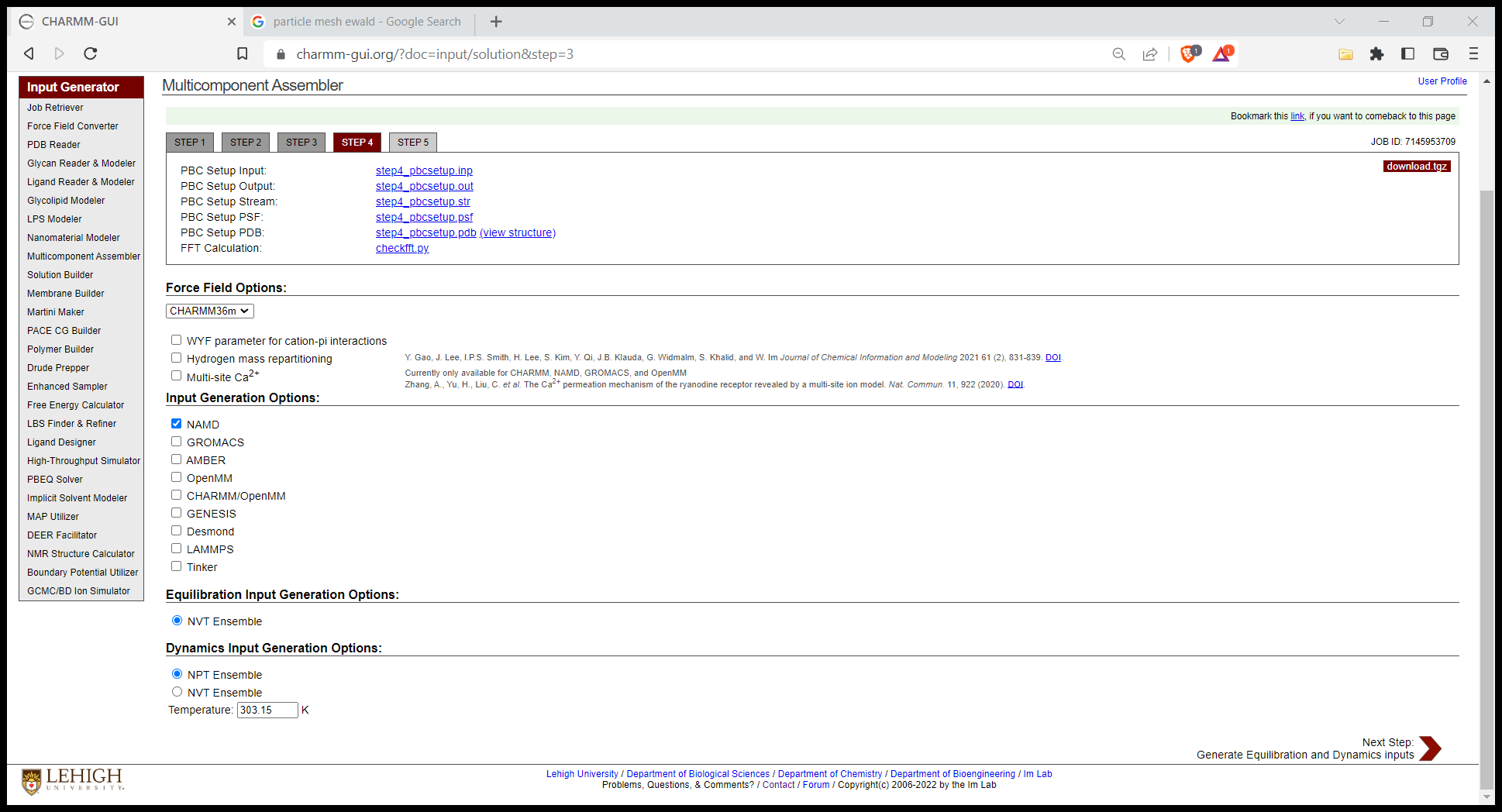


Next step is generating the Particle Mesh Ewald (PME). This the distance at which the system stops calculating long range interactions (I think…). I always leave this on default and click next step.



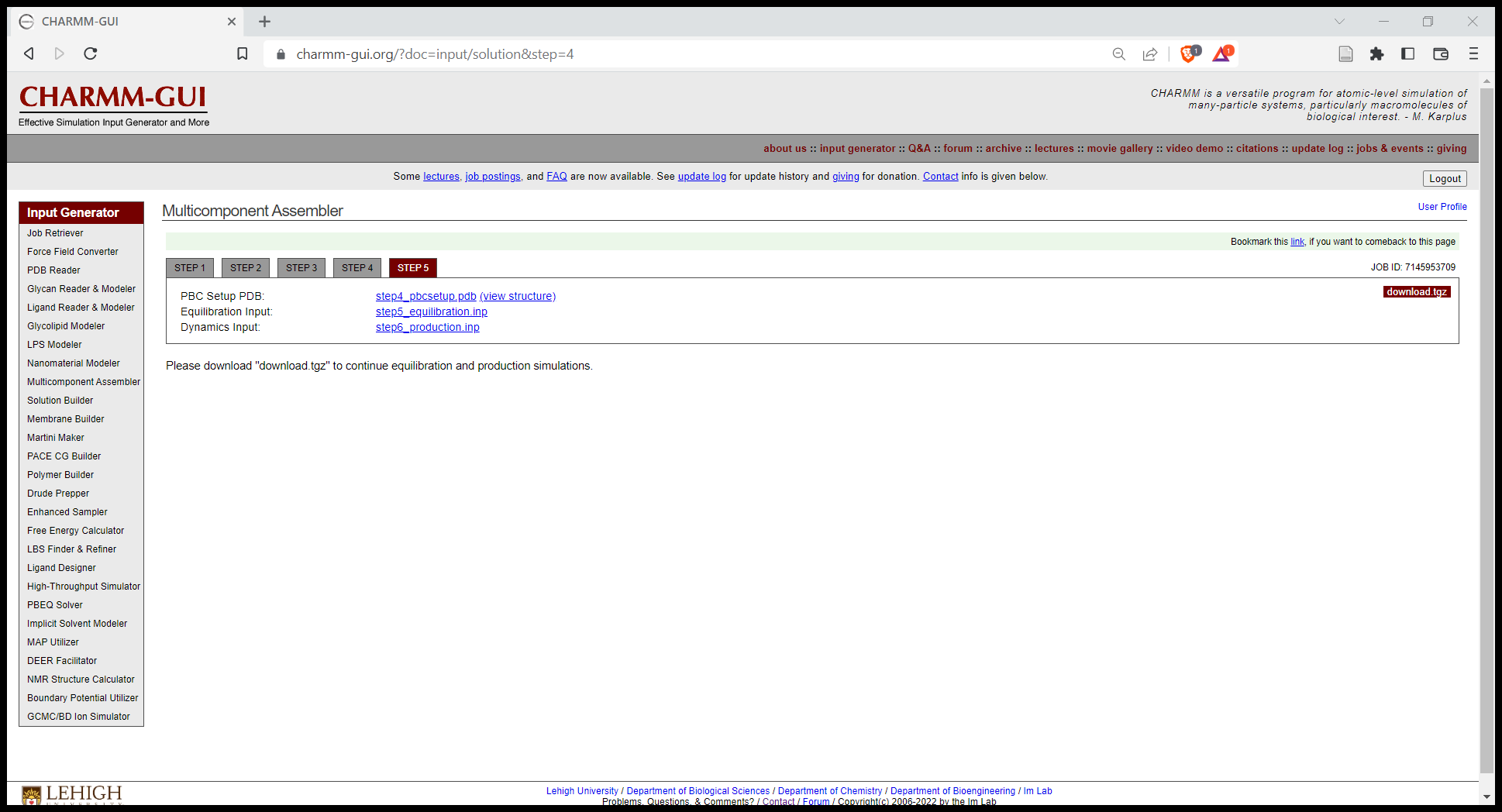
We are onto the penultimate step. This is where we specify which forcefield we are using and which software we want to run the simulation on.

1. Pick which forcefield in the “Force field options”. CHARMM36m is the one I used the majority of the time (most modern and up to date). AMBER is the other option but is much more fiddly! If you choose AMBER, you will see a series of additional drop down menus. I would suggest leaving these as defaults, but these represent additional customization options for the forces used to simulate the system (the internet has explanations for the details of each option) \* There is occasionally an issue with AMBER where NAMD does not show as an option for “Input generation” – if this happens to you, change the various forcefield options around (eg the protein dropdown menu, etc) until NAMD turns up again.
2. Ignore the next checkboxes (probably) – these are options to simplify the calculations to speed up simulations for very large systems.
3. Tick NAMD in the “Input Generation Options” section. This is essential. Otherwise you will not get any files at the end.
4. Leave NPT as default (choice of statistical models – basically refers to the thermodynamics representation of temperature and pressure). Google has info if you’re curious.
5. Choose the temperature you wish to use
6. Remember to write all of these choices down… Then click next.



Congrats! You’ve made it through CHARMM-GUI. Now click “download.tgz”, unzip the file if needs be (probably needs to be unzipped twice), and you’re onto the next step. I recommend renaming the unzipped folder (often called charm-123456, where the number is the job-ID) to something clearer.

You may wish to “view structure” again – be warned, this almost never works in my experience. The web visualization often fails to render the molecules properly and will often not show you what’s actually in the system. If you want to check everything is correct, look in VMD using the method we discuss later.



## Level 3: The cluster

Alright, we’ve got some semi-nonsensical files, we are now ready to start some simulations! Aha, not so fast, first we need to transfer the data to the cluster, and then edit/launch the various scripts which will actually operate the software. If you think learning this next bit is painful, imagine my suffering doing it by trial and error.

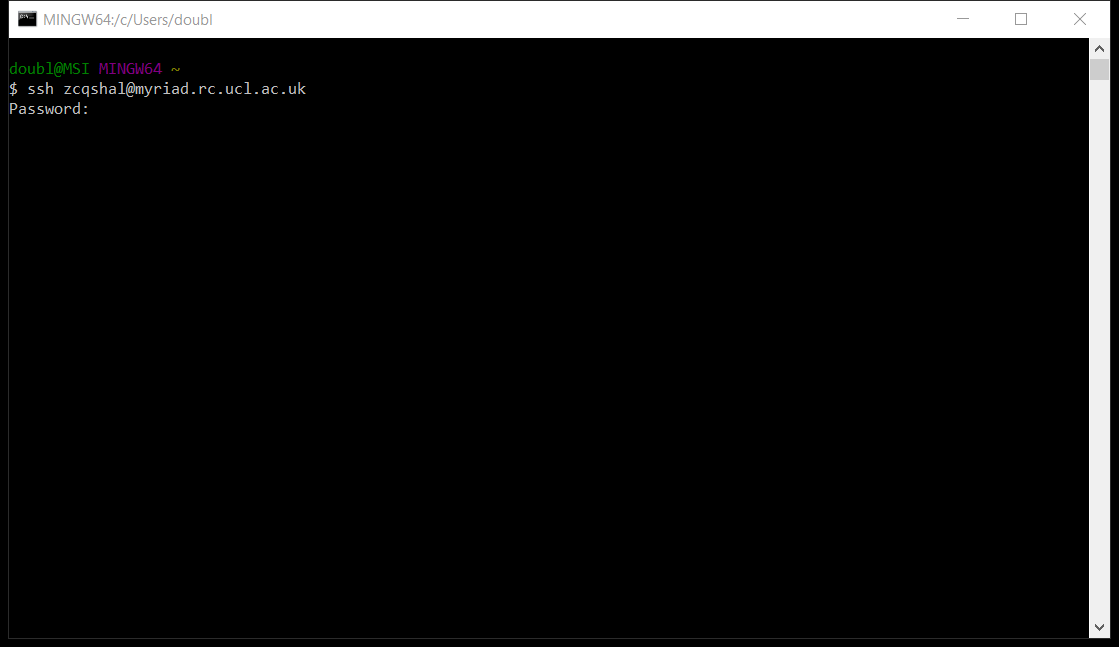
### Step 1: accessing the cluster

Open up Git bash. You’ll also need to turn on cisco (the ucl VPN) if you’re off campus. Type the following command (ignoring quotation marks) and click enter:

“ssh [youruserid@myraid.rc.ucl.ac.uk](mailto:youruserid@myraid.rc.ucl.ac.uk)”

To clarify, your user id will be something like “zcqshal” – whatever you have for your email. Ssh is “secure shell host” or something like that.

Then, when prompted, enter your password and click enter. It won’t show up when you type, which is a little unnerving, but just trust it.



If you successfully access the cluster, you’ll get the feeling of being a hacker in a movie and also be greeted by a wall of text and information about myriad.

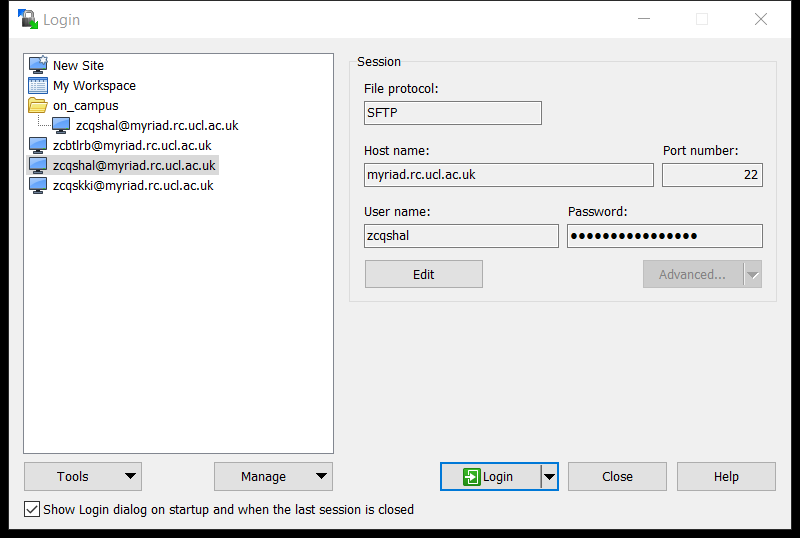
There are some fundamental commands that worth knowing for moving around and manipulating files in the cluster. More extensive lists are available online of course, but this will be a helpful start:

|  |  |
| --- | --- |
| ls | List files in the current directory |
| cd target\_directory | Change directory to target\_directory |
| cd .. | Go back up a directory level (eg return from home/target\_directory to home) |
| lquota | Check how much storage you have left |

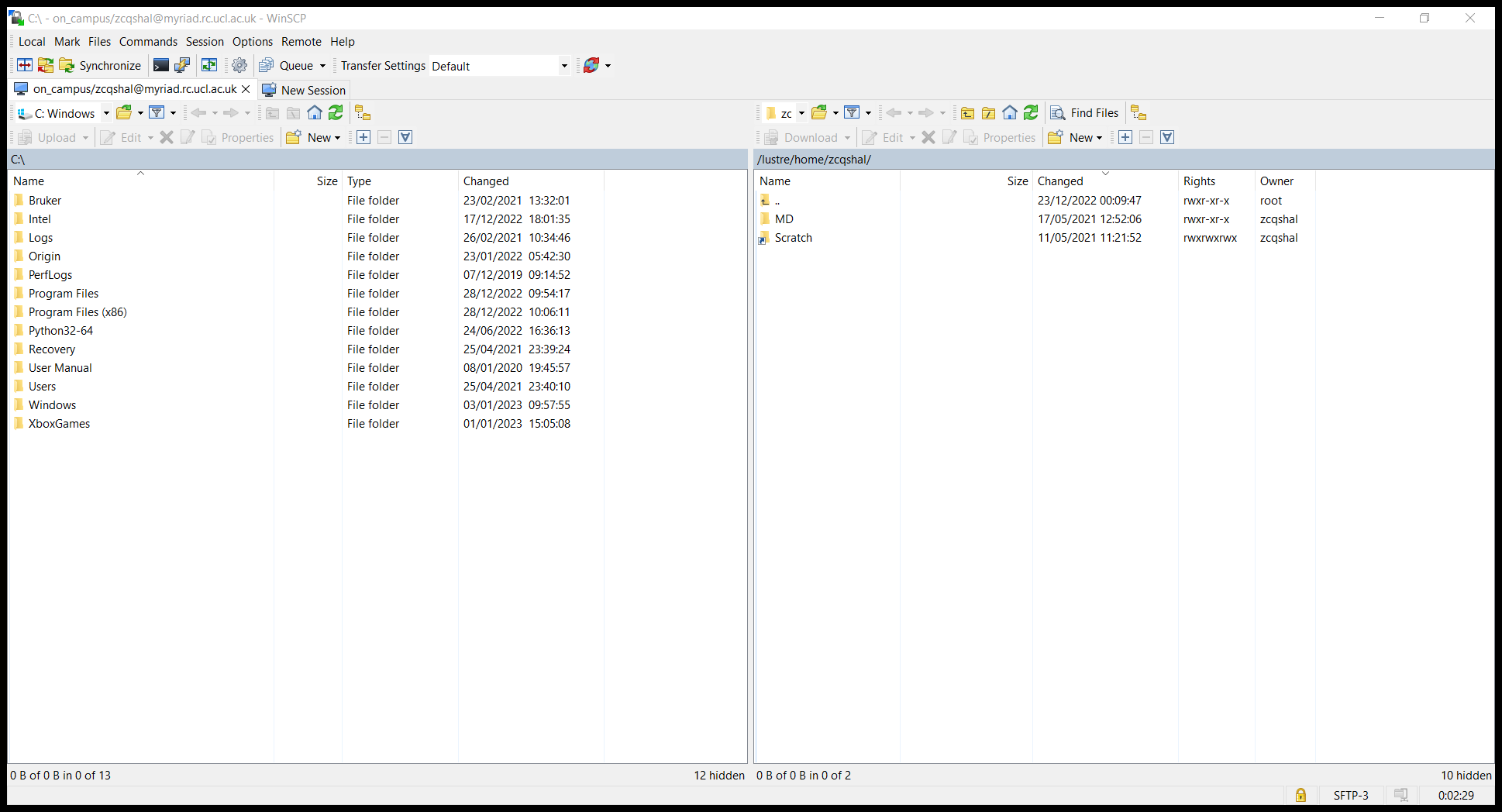
Other helpful tips include pressing “tab” to autocomplete and words you’re typing, using the \* symbol to mean any file (or any file with a given start/ending). There are also file manipulation commands including “mv” (move), “cp” (copy), “rm” (remove), “touch” (create file), and “cat” (show text in files)

For now, use the command “cd Scratch” to move into the “Scratch” directory which is where your free terabyte of storage is. We’ll be back here shortly, but first, we need to fire up WinSCP.

You’ll be greeted by a login window. If it’s your first time, click “New Site” and then fill in the details like this, and then click login. Again, if you’re off campus, you’ll need to have the UCL VPN Cisco turned on.



This should take you to a screen that looks like this:



### Step 2: uploading your files

On the right is the files on the cluster, on the left is your files. You can create, manipulate and delete files on the cluster like you would on your computer by treating the right screen as a standard file browser. You can also transfer files back and forth by drag and dropping.

From this screen, open the Scratch directory and make a folder to do your simulations in. Drag the simulation files from CHARM into this directory.

Graphical user interface, text, application

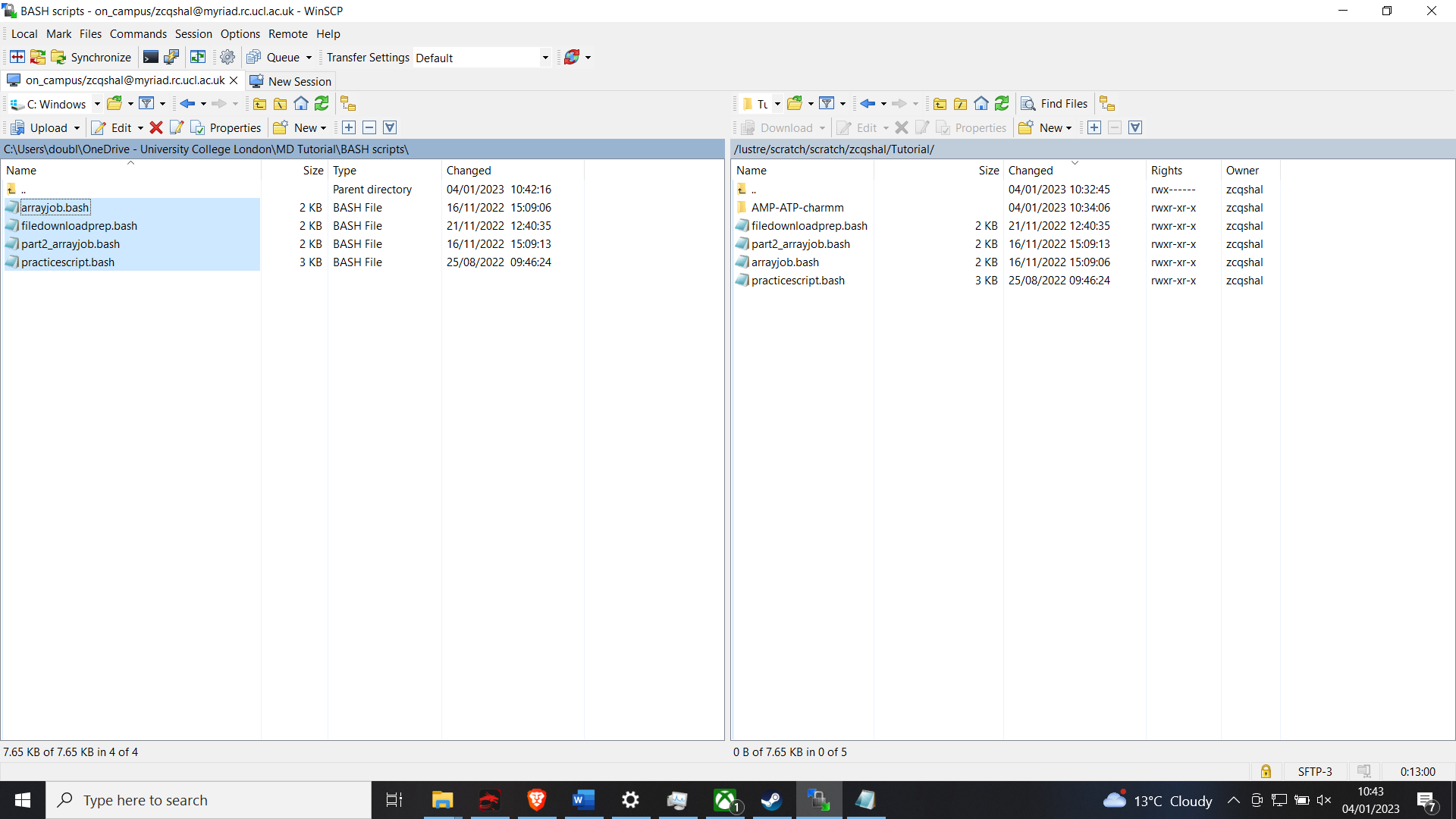
Description automatically generated

Drag and drop the output from CHARMM-GUI into the new folder and wait for it to upload.

Graphical user interface

Description automatically generated with medium confidence

You will also need to upload the four BASH scripts from the MD Tutorial folder:



### Step 3: Scripts

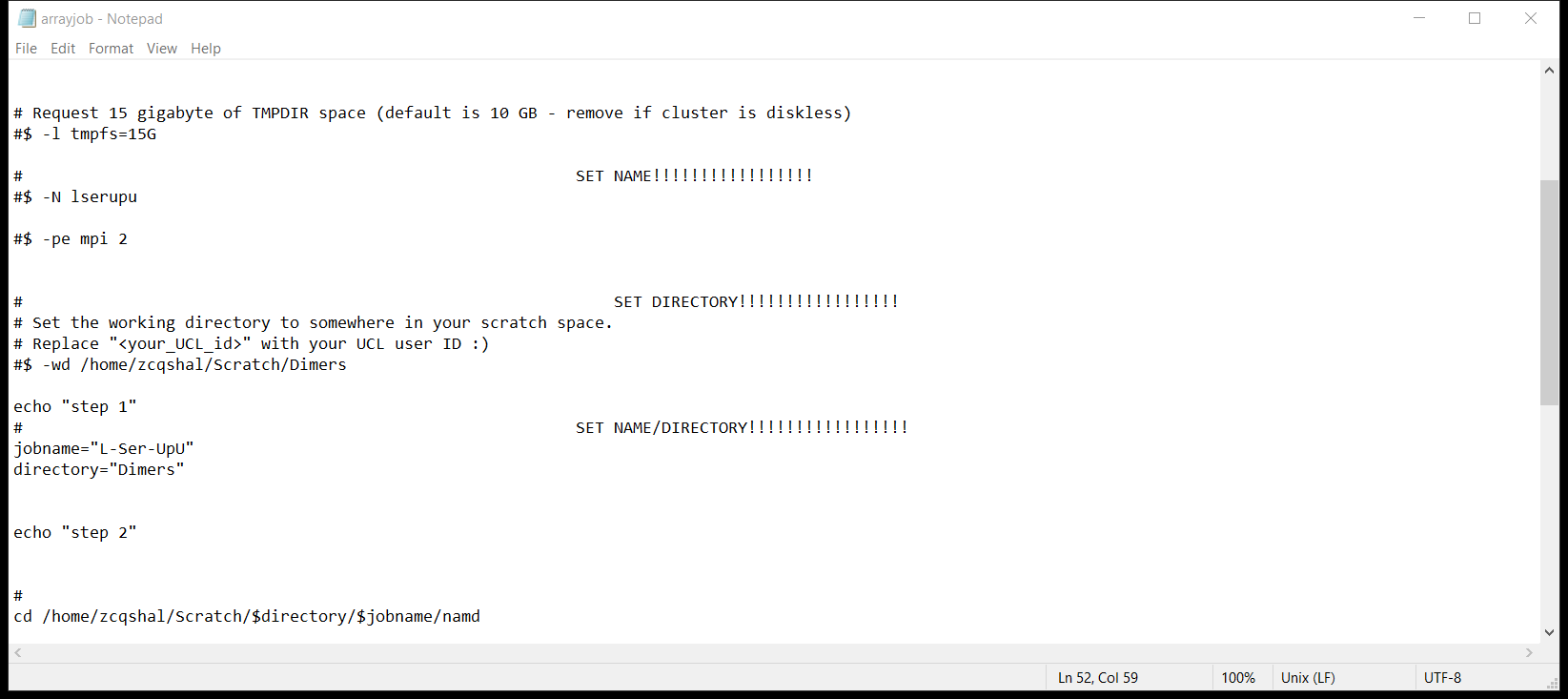
These four scripts are essential, and the thing which are most likely to mess up and cause you pain and suffering. This suffering will mostly be self inflicted because you have forgotten to change a file name or something, but may also be caused by some hidden errors I’ll talk about at the end. I’m sorry it’s a bit janky, if you can be bothered you can try to improve them, but do this at your own peril and keep a backup copy… I’ll give a brief overview of what they all do, then we’ll need to go in and edit them so they work for your experiments.

First is “arrayjob.bash”. This one equilibrates the system and then triggers “practicescript.bash”. practicescript has a weird name because I wrote it as practice early on, but then it turned out to be essential to everything and very complicated so I was so scared of changing anything including the name… practice script makes a series of identical copies of the equilibrated system (for parralelization) and a bunch of important reference files.

Next is “part2\_arrayjob.bash”. This actually runs the simulations. Finally there is “filedownloadprep.bash” which, unsurprisingly, prepares the finished simulations for download.

Lets go in and edit these so they work for your system. To do this, double click on the file on the cluster side of WinSCP. You can then edit and save the file like normal and it will automatically reupload the changes to the cluster.

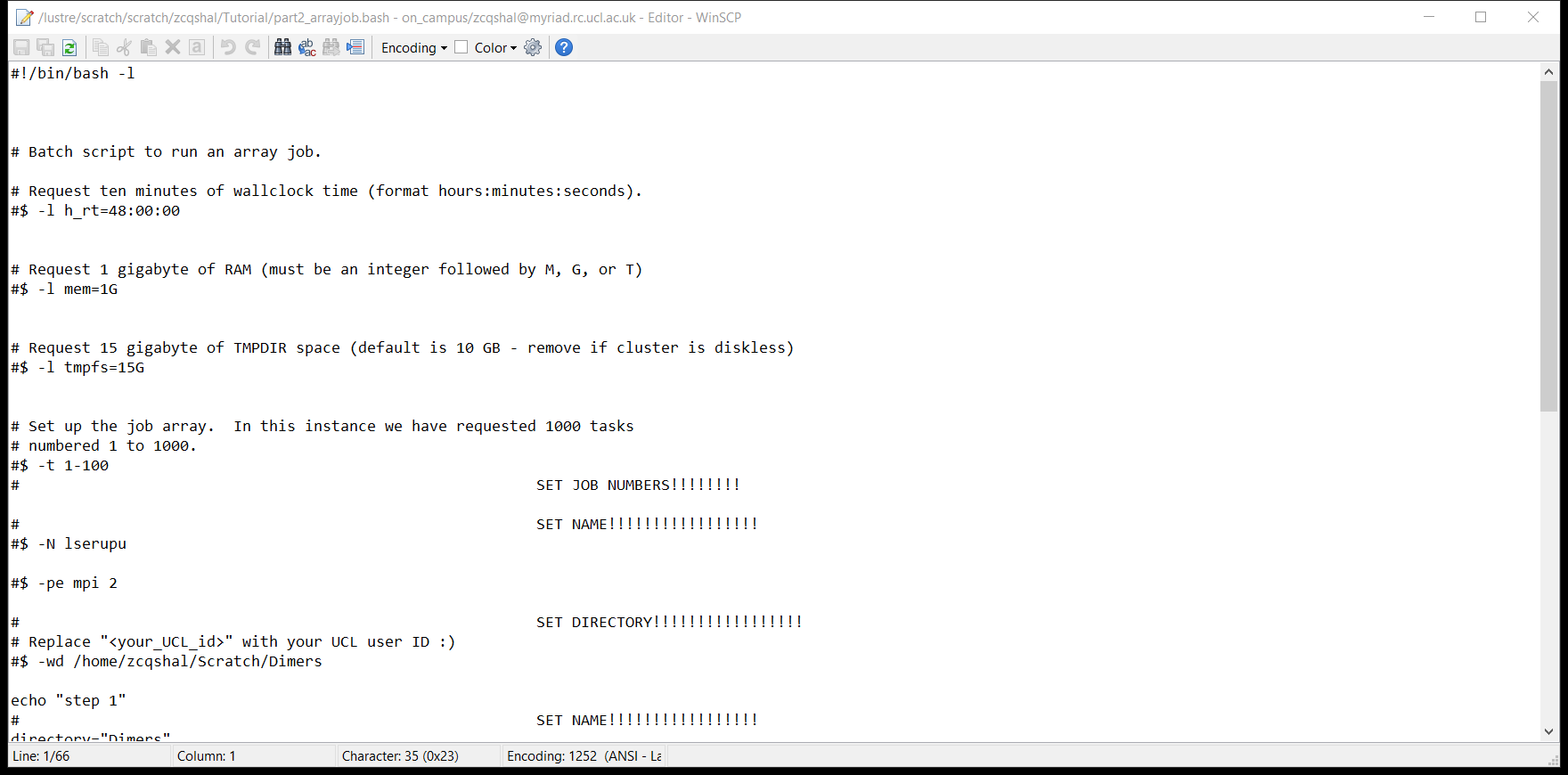
The script will have a bunch of code, and also some comments which shout stuff like “SET NAME!!!!!!!”. The first time you open these, you will need to go through and edit EVERYWHERE which has the userid zcqshal so that it has your user id instead. If any of these is wrong, the job will die. You also need to go to anywhere that says “SET DIRECTORY!!!!” and make sure the directory path is correct, otherwise the job will die. You also need to go to places which say “SET NAME!!!!!!” and make sure these match the name of the folder you got from CHARMM-GUI. If you get this wrong, the job will die (are you seeing the pattern yet?).



At the bottom of array job there is somewhere which says “SET JOB NUMBERS!!!!!”. The three numbers at the end of this line are <how many parallel runs you want> <every x frames, record the molecule coordinates> <run no more than x timesteps>. If I were you, I’d only ever change the first number for runs in parallel, but you may wish to have shorter runs in higher resolution (more recorded coordinates) for certain purposes. Note, there are lots more places where it says “zcqshal” instead of your ID here.



Now save the edited script, and open “part2\_arrayjob.bash”. Repeat the process of editing file names and directories. You also need to make sure the job numbers bit at the top matches the number you put in “arrayjob.bash”.



Once you’ve done that, click save on this file as well. There is no need to edit practicescript. You will need to edit filedownloadprep, but not until after the simulation have run so we will come back to it.

One last point on bugfixing. As I mentioned before, these scripts were the single greatest source of crashes and problems in the pipeline. Most of the time it is due to failing to update all the filenames correctly, but not always.

The second most common issue is permissions – this is where the cluster randomly decides it does not have permission to run the scripts. To fix this, while logged into the cluster on bash, use the command "chmod +x arrayjob.bash". In english, this means: “Change array job to executable”. Repeat this for all the scripts, and they should become green if you type “ls” to list files in the directory.

If other issues arise, you can check for issues several ways. Firstly, you can look at the error and output files generated by the cluster. These will appear as “job\_name.o12345” (for output) and “jobname.e12345” for error. Have a look at these, see how far the scripts have run, and use this to narrow down where something went wrong. Next port of call is googling aggressively. You will eventually find the answer there. You can also reach out the UCL IT support, but only do this if you’re certain the error isn’t on your part. Finally, try asking someone who has done this before – I suggest: Me or Lilly, or Raquel/Aidan/anyone else who works with the cluster regularly.

### Step 4: Submitting the job

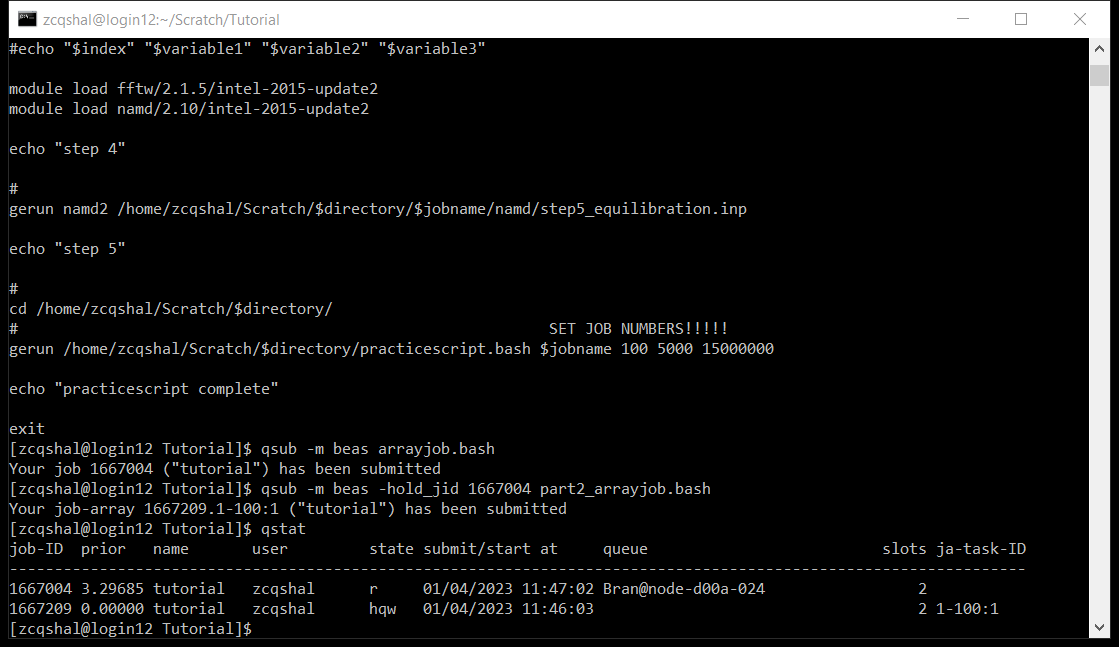
With all your scripts edited and ready for action, we can finally submit the job to the cluster. In the cluster, make sure you’re in the directory where your file and the scripts are (“cd/Scratch/your\_directory”).

Next, type the command “qsub -m beas arrayjob.bash” and press enter. This will submit the job to the cluster queue, and will also send you email updates about your job (the “-m beas” part).

Now look carefully at the job number which your job was assigned. You can copy this by highlighting it and then right clicking your mouse. Now enter the command: “qsub -m beas -hold\_jid 123456 part2\_arrayjob.bash”, where the number is the job number from a second ago. You can paste that in by right clicking the mouse again while you type. (-hold\_jid prevents this second step from triggering until the first step is complete).



Now you can check the progress of your job by entering the command qstat. The job “state” informs you if everything is running smoothly. If it says “r”, that means it’s running. If it says “hqw”, that means it’s waiting. Anything else probably indicates and error, so check what it means on the UCL research computing website. https://www.rc.ucl.ac.uk/docs/howto/#job-states



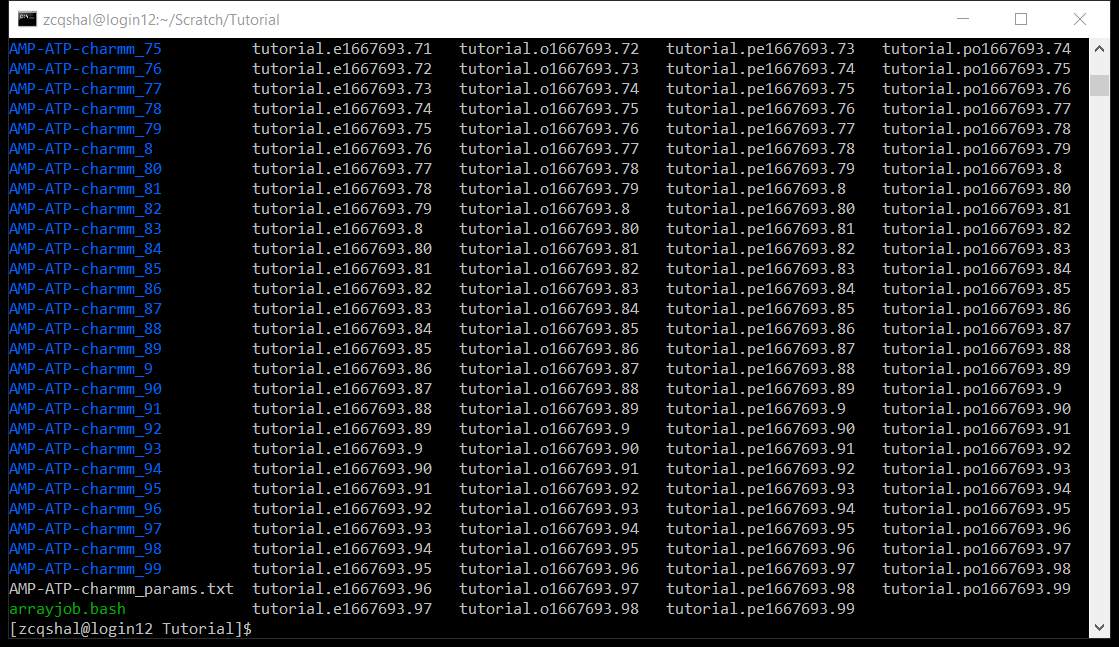
Now, all you need to do is wait for your jobs to finish! Keep an eye on your emails to see if the job ends or if it crashes.

Note, the job often looks like it’s failed if the time limit runs out – it will be considered “Aborted” through “through signal KILL (9)”, but this is fine. If it ends much earlier than expected, it might be worth looking into.

### Step 5: downloading your results

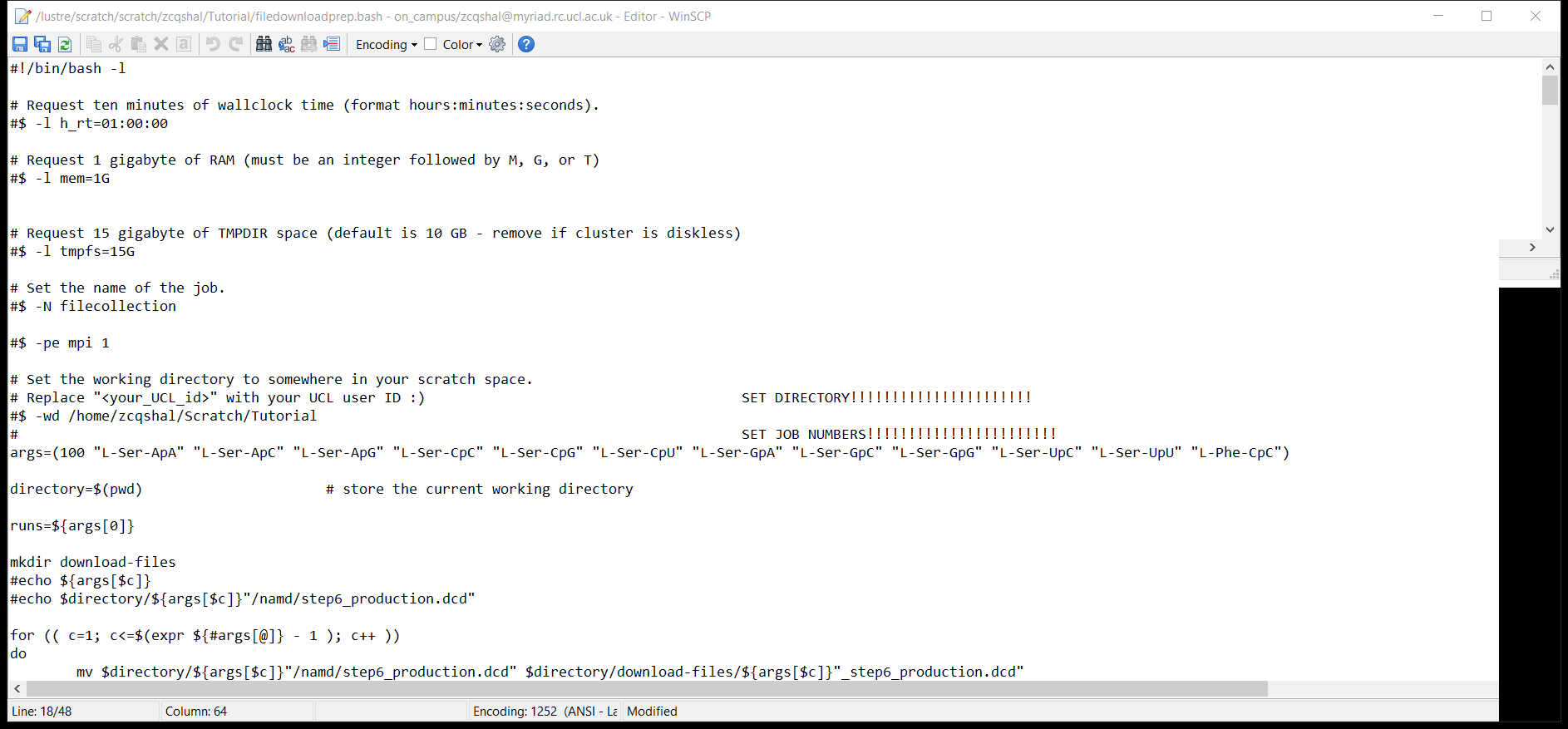
… many hours later….

Welcome back! Now it’s time to download your results. Log back into the cluster and go to the directory where you ran your experiments. If everything has gone according to plan, it should look a little like this:



Absolutely loads of files everywhere. Importantly, there are around 100 copies of the job file. We can tidy this up a bit, but first let’s make sure we get all the data we need. Now is the moment to open and edit “filedownloadprep.bash” in WinSCP using the same method we did in Step 2 and Step 3. This will probably require some scrolling to find it….

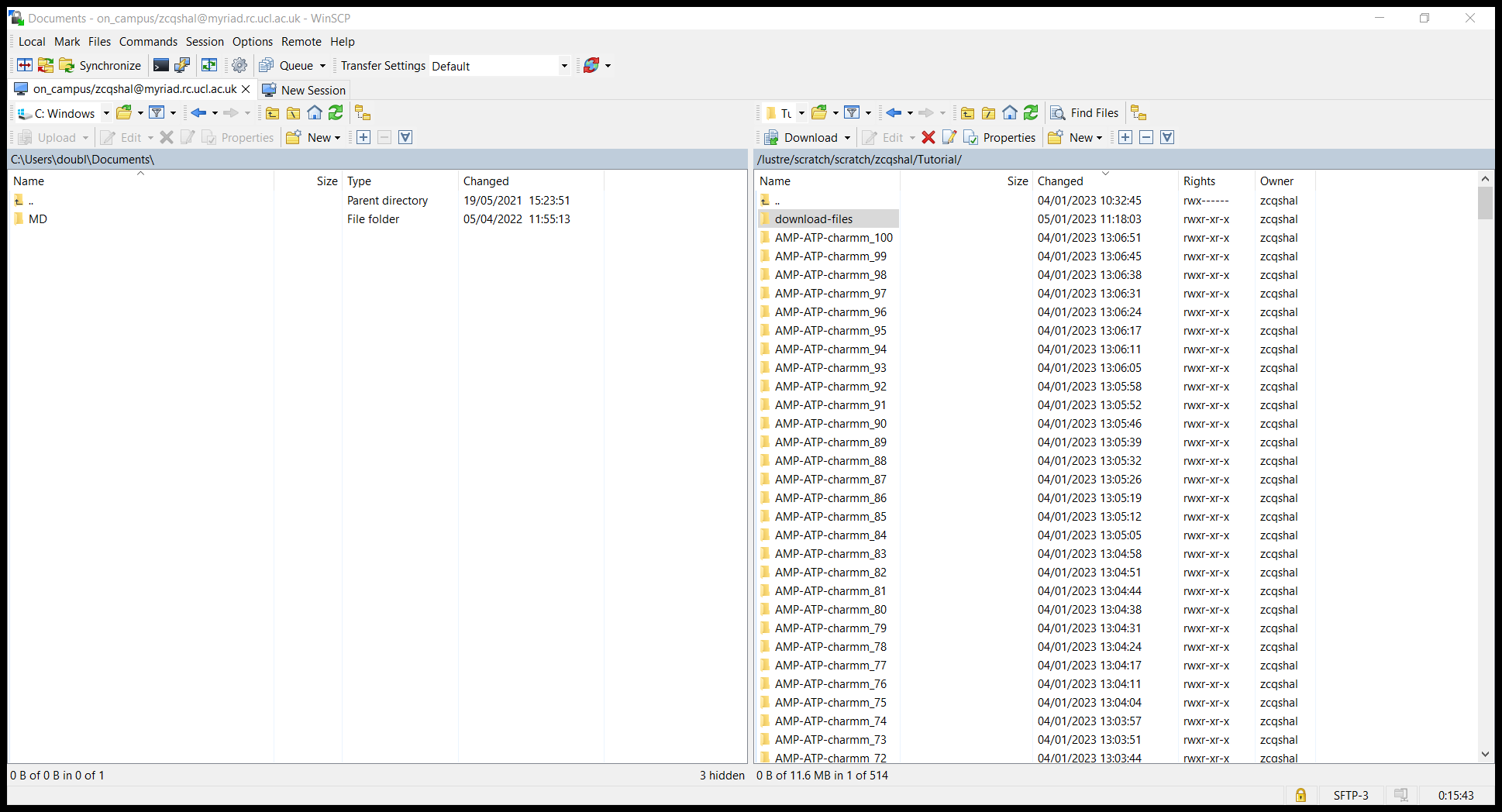
When you do find it, repeat the same process of editing all the directories and number of repeats to be correct. You will also see in the section under “SET JOB NUMBERS!!!” there is also a list of file names. You can write in the names of any experiments you’ve run (make sure the file name is exactly the same as the files above eg. AMP-ATP-charmm [without a number at the end]). In the example in the screenshot you can see that a dozen different experiments are all being prepped for download simultaneously.



Once you’ve edited the file, click save and then go back to the git bash console.

Here, submit the file download prep script using the command “qsub -m beas filedownloadprep.bash”. It normally takes around 5 minutes, but can sometimes be in the cluster queue for a while.

Once this is done, you should be able to find a folder called “download\_files”. This will have all the actual simulation results. You can download this by drag and dropping it onto the left side on WinSCP.



IF you want to clean up your cluster space, you can now. You can do this by typing commands like “rm \*.o\*” to delete all the output files with a “.o” in them. I recommend not doing this for a while though because you may wish to check these files for bugfixing at some point.

The download may take a while (depending on wifi and number of experiments, it may be worth leaving over night). Once it’s done though, you are finally ready to move onto the interesting bit.

## Level 4: Analysis

So, it took a while, but well done for making it here. It’s now time to analyse the data. The matlab script for this is complicated, and mostly automated, but this is where you’ll probably start having to do some real coding.

The main issue is that the outputs from the simulations are encoded very weirdly in “.dcd” files. The data extraction script should work pretty broadly, but it’s not perfectly generic so you will still have to be careful.

The simple analysis functions are also pretty generic, but it will require some effective coding to make them work for your specific system. As a result, I will show some examples, but you will have to adapt them yourself.

### Part 1: VMD

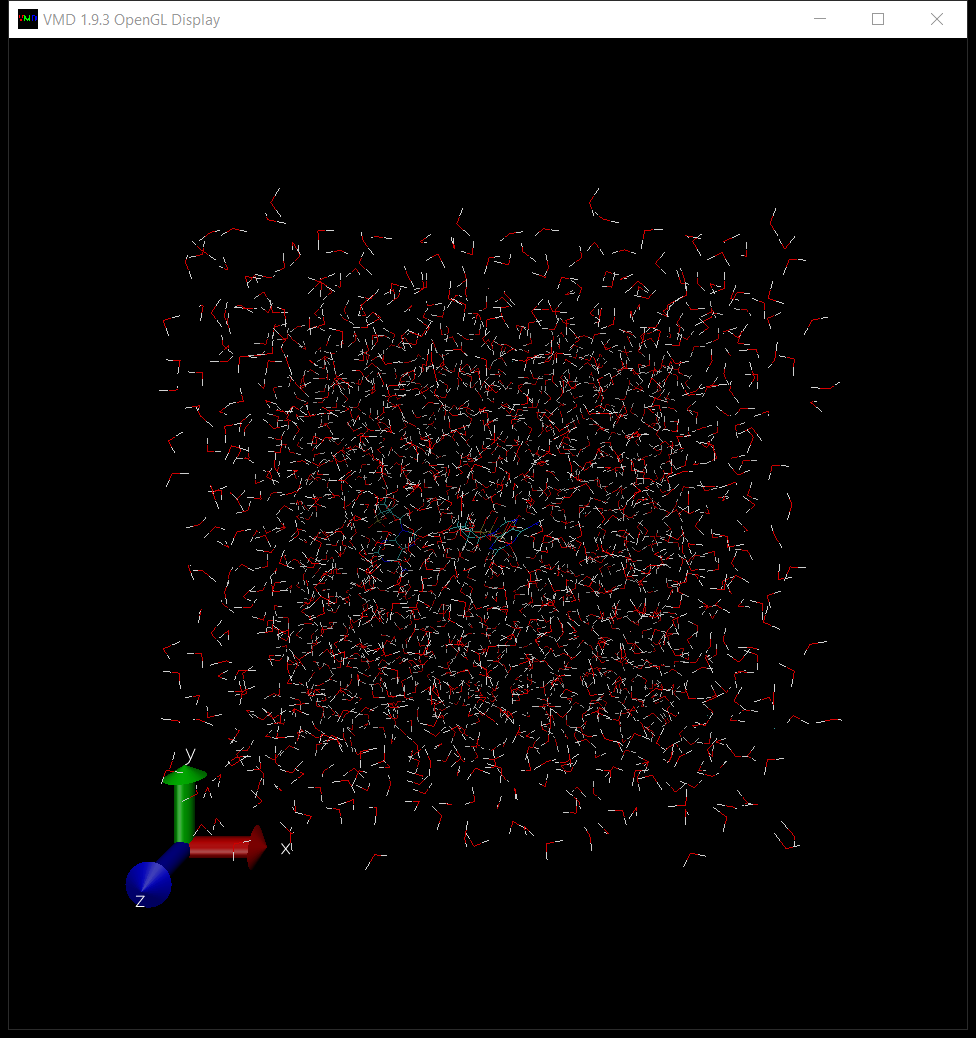
FIRST THOUGH, let’s look check everything worked alright by looking at the simulations in VMD.

Find your downloaded “download-files” folder, and then find a file called “<you\_experiment\_step4\_input.pdb” in it’s name. This is the coordinates of the molecule. Open VMD, then drag and drop the step4 file into the main screen which has a spinning VMD logo in it.

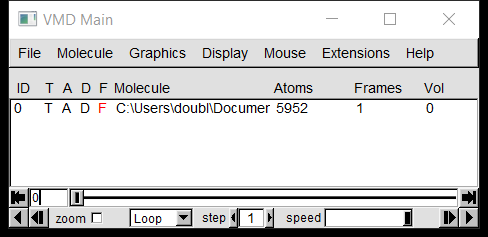
A screenshot of a computer

Description automatically generated

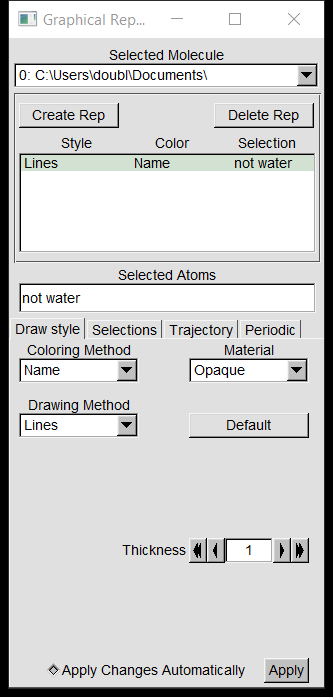
That should produce something that looks like this:



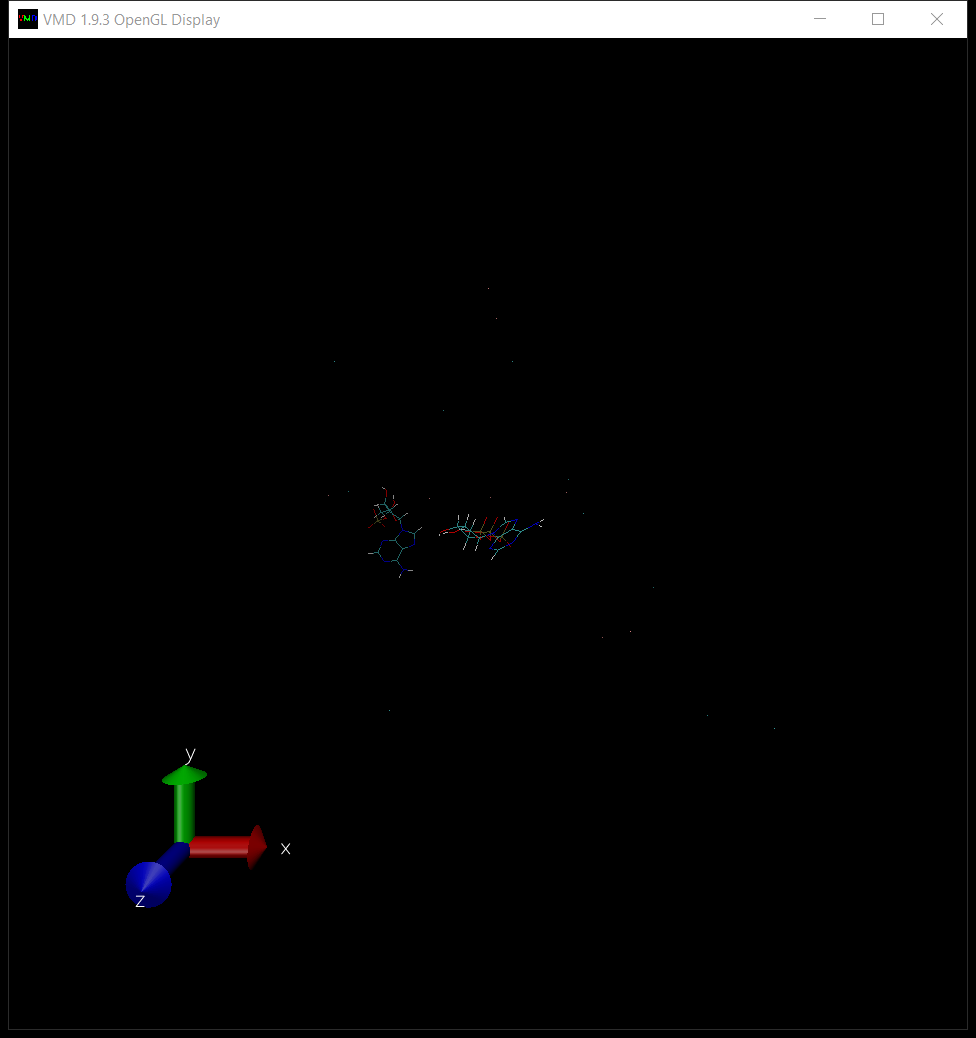
We can’t see anything that’s going on so we need to hide all that water. In the VMD Main window, click “Graphics; Representations”.



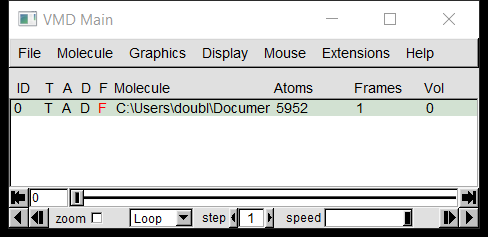
This will open a new window. In the box which says “Selected Atoms”, change “all” to say “not water” and press enter.



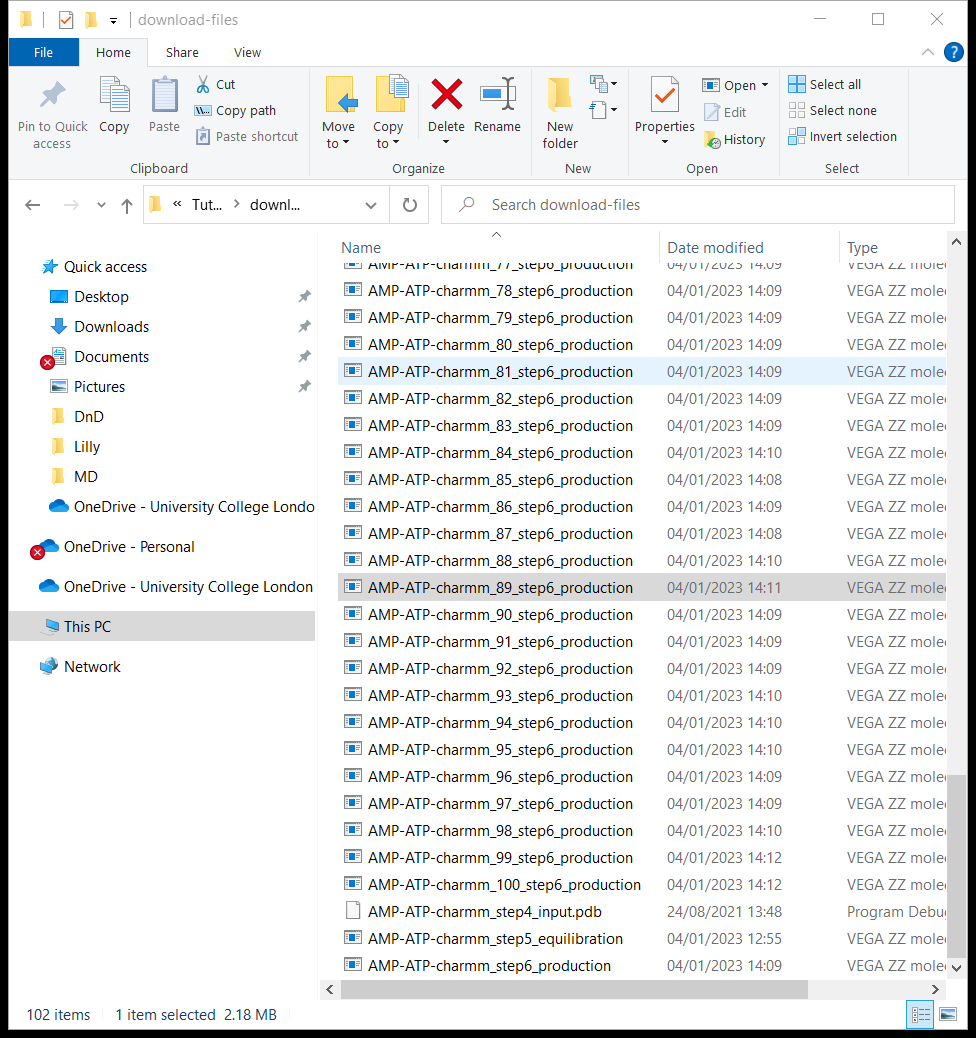
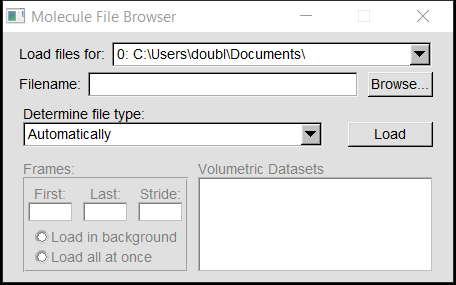
You should now be able to see the molecules of interest in your simulation. You can zoom and spin around and stuff.



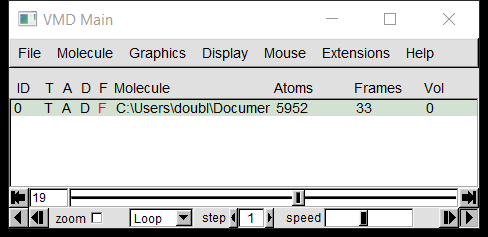
Now we need to load in the actual simulation.

In the VMD Main window, right click on the file name and then click “Load data into molecule…”

Which will open a new window called Molecule File Browser. In the download-files folder, find any file with the name “<your\_job>\_Number\_step6\_production”. Drag and drop this file into the Filename section in the Molecule File Browser, and then click Load



The molecules in the VMD display window should jiggle aggressively. Once this has stopped, you can play your simulation. You may want to adjust the speed.



Now you can check out how your system is behaving qualitively. Make sure it looks sensible. Ions are probably hidden, but you can play around until you get them to appear – there are lots of tutorials online for VMD which are very detailed.

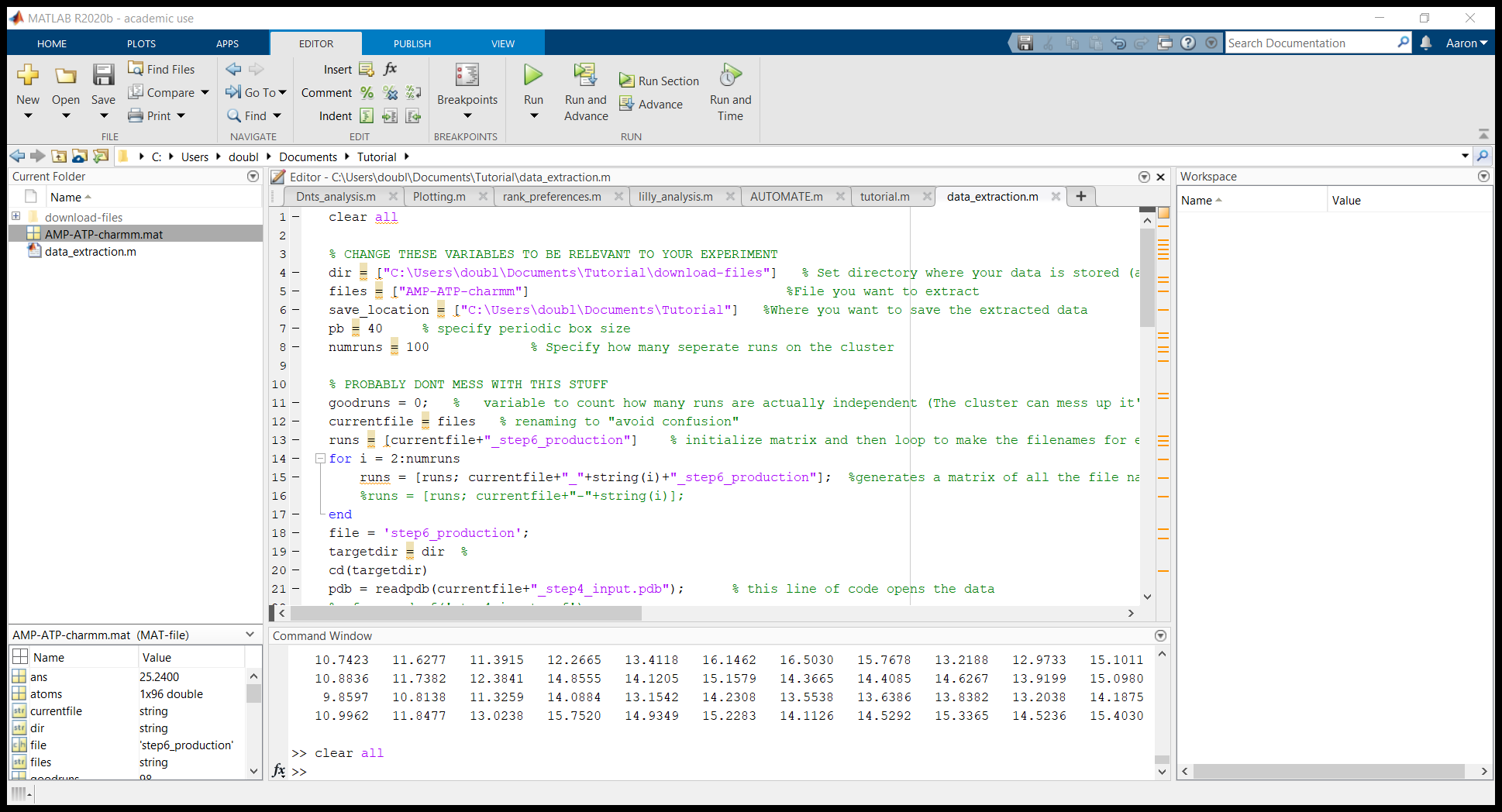
A few other points which are worth noting here. You can load in more runs of the simulation by repeating the “load data into molecule” procedure. You can also see the full periodic box by ticking all the boxes in the Periodic sub menu of the Graphical Representations window. It’s also usually worth smoothing the trajectory in the Trajectory sub menu in the same window. Anything else you’ll have to discover yourself.

### Part 2: Final Boss: Coding ☹

This will be the least helpful part of the tutorial. Apologies.

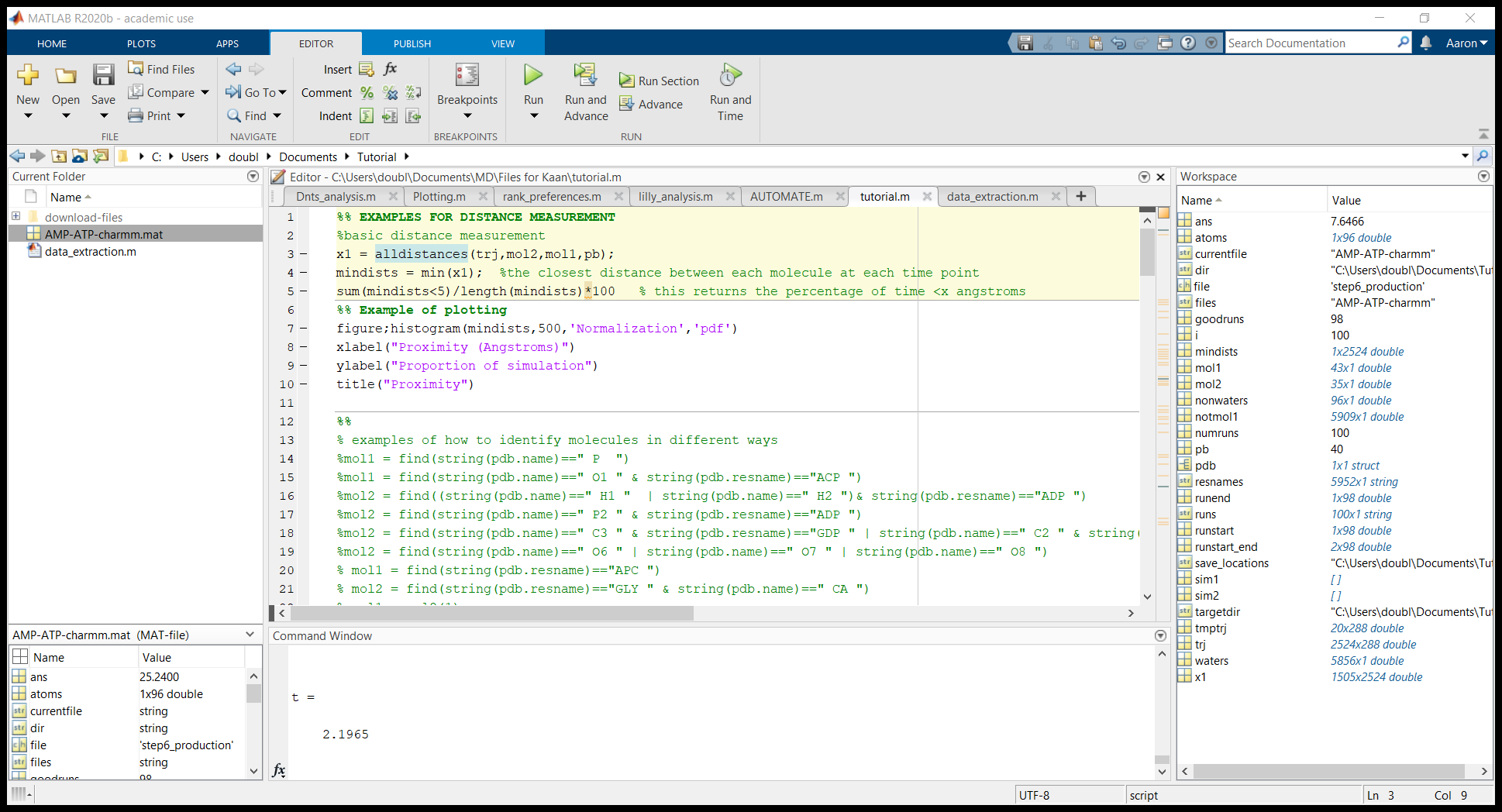
Open the script data\_extraction.m in matlab. There are lots of comments in the file which should clarify some things, but generally reading other people’s code is awful and there’s not much to be done about that. Hopefully it will run mostly smoothly, but you should expect to spend some time reading through it line by line in order to catch bugs when things don’t work inevitably.

Edit the first few lines to match them to your dataset, including file location, periodic box size,etc, then click the big green run button.



Once this has run, open tutorial.m

This has a few basic examples in it, including basic measurement of distance between two molecules, plotting of the distances, calculating error with a bootstrap, and estimating the volume of a molecule. There are also lots of examples of how to specify what your molecule is.



To explain a few key details which may be helpful, the data is extracted and stored in “trj” as a set of coordinates for every atom in 3D (x,y,z). The rows of this matrix are the new coordinates with each tiemstep. Every triplet of columns represents a different atom.

The calculations of distance are just basic Pythagoras, but this has to be done loads of times, so it uses the atomdistance function at the bottom of the page. This also has to be done for lots of atoms, so that is done by the alldistance function also at the bottom.

Information about which atom is what is stored in a data structure called pdb.resnames and pdb.name. Resnames refers to the name of atom, whereas name refers to the name of the molecule. You can check these specific names by printing the variables to the console, or by looking in VMD using Mouse; Label; atoms, and then clicking on atoms of interest. A whole bunch of info will appear in the VMD console, and some info will be labelled on the screen. Avoid using anything which is numbers only because VMD and matlab do not use exactly the same numbering systems.

There’s probably a bunch more things which would be helpful to know, but I can’t think of or write it all down. Hopefully I’ll be able to help you if needs be, but if not, I’m very sorry for your pain, and good luck.