Instruction Manual for running ILSimulations package

Adrian L. Kiratidis[a)]

(Dated: 28 November 2019)

I. INTRODUCTION

In this manual we outline the steps required in order to run the Ionic Liquid Simulations package that was the basis for our paper (REFERENCE). Throughout this document we note the following useful things

- All commands that are to be written will be enclosed in single quotes. i.e. to type the letter l followed by an s, I write type 'ls'. Do not type the single quotes.
- Generic names will be enclosed by angle brackets. For example, if there are lots of script names that could possibly be run, let's say, plot_density.py, plot_potential.py etc, then <script_name.py> will refer to a generic script of that type, and you can replace script_name.py with plot_density.py or plot_potential.py etc. Do not include the angle brackets in anything.
- My email is at the bottom of this document. If you're experiencing problems that are taking an inordinate amount of time to solve, you can email me with a full description and I can help out (my ability to help on code specific/model specific questions is expected to exponentially decay at large times. Configuration/running help ability should remain constant).

II. UBUNTU INSTRUCTIONS

In this section we detail the steps required to run the code on a ubuntu operating system.

- Open the terminal. This should be located on the task bar. If it is not, open the search bar and search for 'terminal'. All commands issued in the terminal are run after you press the return key. **NOTE: These are instructions for the first time you run the code. On subsequent runs all steps that start with 'sudo apt-get install...' or 'git clone...' or 'pip install...' are to be skipped, as these commands install things and you only need to install things once.**
- In the terminal type **'sudo apt-get update'** followed by the return key. This will prompt for the system root password. Enter the root password. After entering that the package manager will be updated. This step is neccesary to install the latest version of things and aviod conflicts in subsequent steps.
- In the terminal type **'sudo apt-get install git'** followed by the return key. This will prompt for the system root password. Enter the root password. After entering that, follow any prompts and git will be installed, if it isn't already. This step is neccesary as the source code is on github and we need to obtain the source code in order to run it.
- In the terminal type **'git clone https://github.com/adriankiratidis/ILSimulations.git'** followed by the return key. This obtains the source code from github. This will create a new directory you can access from the terminal called ILSimulations, and it will copy all relevant code and subdirectories to this folder.
- In the terminal type **'cd ILSimulations'** followed by the return key. This changes directories into the new directory we just created that houses all of the source code.
- In the terminal type **'sudo apt-get install build-essential'** followed by the return key. Follow any prompts. This will install "make", which is used to put all our dependencies together.
- In the terminal type **'sudo apt-get install gfortran'** followed by the return key. Follow any prompts. This will install gfortran, the fortran90 compiler.

[a)] Electronic mail: adrian.kiratidis@gmail.com

- In the terminal type **'make'** followed by the return key. This compiles all the source code, performs all linking required and compiles the executables. The source code in located in the src subdirectory while the executable files are located in the bin subdirectory.

- In the terminal type **'sudo apt-get install emacs'** followed by the return key. Follow any prompts. This installs the text editor emacs. This is my preffered text editor to edit Fortran code, but you are welcome to skip this step if you are using some other text editor.

- In the terminal type **'emacs bin/runSimulation.sh &'** followed by the return key. This opens the runSimulation bash script (that is in the bin directory) that is the driver script that runs the code. If this is the first time you open a file with the emacs text editor, then I recommend ticking the checkbox "Never show it again", at the bottom, and clicking "Dismiss this startup screen". If you don't have this checkbox then ignore the aforementioned suggestion. This is the script that is used to run all simulations, and different simulations are run by changing the parameters in this script.

- Make the appropriate changes to this script based on the parameters you wish to use for your simulation. For example, if you see a line "hs_sphere_diameter=2.4" and you want to change the hard sphere diameter to 1.2, simply delete 2.4 and replace it with 1.2. Note that the comment character for bash scripts is the hash character #. You should only ever have to change things of the form <variable_name> = <value>, and you only need to change what is written in <value>.

- In the terminal type **'bash bin/runSimulation.sh'** followed by the return key. This runs the code with the simulation parameters you entered in the previous step, and prints output to the screen that is helpful in debugging and knowing where you are up to in the calculation, i.e. how many separations have we done. The output files are stored in subdirectories within the run_results folder. You can see exactly where the output files are written to by looking at the output directory path that is set inside the runSimulation.sh script. Naturally, you can change this output path if desired.

- If there is ever a failed run, you should type in the terminal type **'touch 1.params && make clean && make'**. This ensures that from one run does not contaminate any adjusted parameters fro subsequent runs.

- In the terminal type **'cd plot'** followed by the return key. This changes directory to the plotting directory which stores all of the plotting scripts.

- In the terminal type **'ls *.py'** followed by the return key. This lists all of the python plotting scripts that are in the plotting directory, i.e. everything that ends in '.py'

- Look through the list and pick a script name that is most similar to what you want to do. There are generally three classes of script names

  - 1. Ones that plot densities/waterfall plots of densities, scaled densities etc. These will have the word density in the file name i.e. plot_multiple_densities.py etc.
  - 2. Ones that plot the interaction potential, waterfall plots of interaction potential etc. These will have the word potential in the script name, i.e. plot_multiple_potentials.py etc
  - 3. Ones that check the internal consistency of the calculation by plotting the pressure from the contact theorem and from the negative derivate of the poential.
  - There are some other plotting scripts that don't fall into either of these categories, but these are more seldomly used. For example plots that integrate the total charge from the wall was used once, but usually you'll have to do one of the first three things.

- In the terminal type **'emacs <your_script_name>.py &'** followed by the return key, where <your_script_name> is the name of the python script you'd like to run. This command opens the script of your choice so that you can edit it. Note that lines followed by a hash # are commented out lines.

- Now edit the file you just opened. First, search for the line containing 'np.loadtext(...'. This is the line that reads in the data for plotting. Alter the path to be wherever you wrote the output files in previous steps (some subdirectory of the ILSimultions/run_results/ folder).

- These files are reasonably small. You may need to comment out or uncomment other lines as neccesary depending on the script. The comments are a guide and there are not many things to do, so changing the script shouldn't cause too many problems.

- Once finished editing, and before running the plotting code we need to install afew more things for the plotting scripts. In the terminal type the following each followed by the return key, and following any prompts.
  - **'sudo apt install python-pip'**
  - **'pip install numpy'**
  - **'pip install matplotlib'**
  - **'sudo apt-get install python-tk'**
  - **'sudo apt-get install texlive-full'**.
- Once finished editing, In the terminal type **'python <your_script_name>.py &'** followed by the return key, where <your_script_name> is the name of the python script you'd like to run. This command runs the script of your choice. This will bring the plot up on your screen for inspection.
- Note: At the end of the file there is a line starting with 'savefig('. This line saves your plot as a pdf/png/whatever suffix you put on the name. Uncomment this line and write the output file name that you would like your plot to be called if you would to save it. This will save your plot in the plot directory. You can verify this by running the command **'ls <filename_of_your_choice.pdf>'**, which will list your plot name.

III. ADDING FUNCTIONALITY TO THE CODE

In this section we provide instructions on how to make the neccesary alterations required in order to add support for calculating an ionic liquid that is not currently supported. This section simply outlines the rough details of code changes and so is relevant irrespective of whether you are running on Windows or Ubunutu. There are a lot of moving parts and things to get right here. It is recommended that looking at the results from the contact theorem in order to ensure internal thermodynamic consistency is done before looking at any results. In my experience the contact theorem comparison is sensitive to a lot of small errors, so as it is very easy to make an error without realised it if the contact theorem check is not inspected. The recipe to follow is as follows:

- In the src directory open parameters.f90, then do the following: (Note that you will need to change the value of the variable in your input script to whatever you've called it below)
  - Add a routine that sets the constituent bead density from the bulk ion density. See the routine "SetC4MIN_BF4BeadDensityFromBulkIonDensity" as an example.
  - Add the function call to the routine writen in part a) to the call within the the if construct in the routine "SetBeadDensityFromBulkIonDensity".
- In the src directory open constructoligomers.f90, then do the following:
  - Write a routine to update the bead densities of your new ionic liquid based on the structure. See the routine "UpdateC4MIMPositiveBeadDensities" for example.
  - In the routine "UpdateDensities", add a call to the routine you wrote in part a) within the if construct that selects based on the value of "ionic_liquid_name". Note that you need to update the densities of positive, negative and neutral beads here. Currently there are done in separate routines to maintain generality for the case of possible mixing/re-use of routines/structures.
- In order to also be able to verify the contact theorem in the src directory open constructoligomers.f90 and write a routine that calculates the ideal chain contribution to the free energy. See for example the routine "calculate_single_neutral_sphere_ideal_chain_term".
- In the src directory open surfaceforces.f90, then in the function "calculate_ideal_chain_term_per_unit_area" add a call to the routine writen in 3a) with the if construct.
- Now change the value of "ionic_liquid_name" in the input script to whatever you have called it in the above routines, cross your fingers, and hope your changes work as advertised.

IV. INSTALLING A VIRTUAL MACHINE

In this section we detail instructions on how to install a virtual machine.

- First we need to download either VirtualBox (`https://www.oracle.com/virtualization/technologies/vm/downloads/virtualbox-downloads.html`) or VMWare (`https://www.vmware.com/products/workstation-play workstation-player-evaluation.html`), ensuring that you download the version that matches your host system. That is, if you're running windows as the main OS, then download the windows version. Once it's downloaded, run the installer script and follow the prompts just like you were install anything else.
- Go to the Ubuntu website (`https://ubuntu.com/download/desktop`) and download the latest version. This should save a '.iso' file to your downloads directory.
- Open up either VirtualBox or VMWare, whichever you have installed, and find New Machine on the top left file dropdown menu. Then follow all the prompts and provide the .iso file when asked for an OS image.

V. WINDOWS INSTRUCTIONS

The recommended way to run this code is on ubuntu, following the instructions above, as all the plotting scripts and associated packages were written in that environment. In the case that you do not want to run on a linux based operating system or can't for whatever reason, you can run the code with the Fortran IDE codeblocks on windows. The download instructions are as follows. I do however think it will be easier to install a virutal machine and run on that. Nevertheless, if you wish to run the code in Windows, get the output and run your own plotting script then here are the instructions to download codeblocks.

- Open an internet browser and go to http://www.codeblocks.org/
- Click on 'Downloads' in the left menu.
- Click on 'Download the binary release' link.
- This should bring up a table. On the row corresponding to the file 'codeblocks-17.12-setup.exe' click 'Sourceforge.net' under the Download form column.
- Once the download is complete, run the executable and follow the prompts to install codeblocks.