Computational Chemistry: Molecular Dynamics Simulation Module

Name:_____

Introduction

1. What is computational chemistry?

2. What are the advantages of simulating a molecule or a reaction over doing the experiment in a lab?

Programming

Match the following terms to their definitions:

1. Programming language_____

2. Algorithm_____

3. Program_____

4. Input_____

5. Output_____

6. Binary_____

7. Compiler_____

8. Variable_____

9. Iteration (Loop)_____

10. Conditional Statement_____

11. Debugging_____

12. Function_____

13. Parameter_____

14. Comment_____

15. Initialization_____

16. Hardware_____

17. Software_____

18. Declare_____

Definition
A. A method of encoding data using two symbols, 1 and 0.
B. Repetition of a particular process either infinitely or until a test condition is met.
C. The data generated as a result of running a function or program.
D. A named procedure or routine than can be called upon to perform a specific task.
E. A set of instructions that the computer executes in order to achieve a particular objective.
F. An annotation in the source code of a computer program added with the purpose of making the source code easier to understand, and are generally ignored by the computer.
G. Formal language used to give a computer instructions.
H. A program that converts a high level programming language into a low level language that the computer can execute.
I. Assigning a value to a variable prior to its use in a calculation in which its initial value will affect the result of the calculation.
J. Data entered into a computer program for processing.
K. A symbol or name used to store data in a program.
L. Creating a variable in memory by naming it and determining its type.
M. A set of unambiguous rules or instructions to achieve a particular objective
N. A variable that has a value that remains constant throughout a program.
O. The process of finding and correcting errors in programs.
P. A statement that determines whether specific code will be executed by evaluating a test case.
Q. The physical components that make up a computer.
R. The programs that run on the hardware/computer system.

Average Program

In order to practice using code to simulate the motion of molecules, we will first write a practice program that will use many of the commands that you will need later. The programming language that we will use is Fortran-95. You will now be guided through the process of writing a program. (Note*****All highlighted terms must be replaced with the variable names, file names, and numbers that you chose)

Describe the process of calculating and average of a list of numbers in your own words (you will use this description to help design your program)?

Goal: You will write a program that reads a list of data, calculates the average of the data, and writes the output in a new file.

1. Go to http://www.tutorialspoint.com/compile_fortran_online.php Highlight and delete the program that is already written in the top window. Write a new program name, such as "Averageprogram" in this window.

2. Make sure that you have downloaded the file numbers.txt from the website You will need to upload a file for your program to read. On the left side of the screen, click once on the folder that says "root". Now click "File" then "Upload" on the right side and upload the the file numbers.txt.

3. You now need to create the file that the program will write the output to. Click the "+" on the left side of the screen and name your file. Make sure the file name ends in .txt

4. It's now time to start programming! The first step is to make sure that the program does not automatically declare variables we will not use. To do this, below your program title in the top window, type:

```
implicit none
```

5. The next several lines of code will be used to declare the variables that we will use. There are two types of variables that we will need, integers and real numbers. To declare a variable, write the type of variable, a space, two colons, and the symbol or name of the variable. For example,

```
real :: run_sum
integer :: x
```

In your code, define two real variables and one integer variable. Do not use spaces in the names. Each variable should be declared on a new line.

The integer will be used to keep track of which line in the read file the program is on.
One real will be used to keep track of the total sum of the numbers in the input file as each line is read.
One real will be used to store the next number in the input file that will be added to the total sum.

6. You will need to declare one parameter (a value that will not change throughout the program). To do this type:

```
integer, parameter :: i = 10
```

You can name the variable anything that you want. This must be set to equal 10 as this will be the number of lines the program will read later on in the numbers.txt file.

7. Now we must tell the program to open the file that contains the numbers that we want to average. To do this, we must assign the file a number between 11 and 19. To do this type:

```
open(unit = the number you chose, file = 'your input file name.txt')
```

8. The computer must be told at what value to start the sum variable. To do this type the name of the real value you are using for the sum, a space, equal sign, space, and 0.0. This is called initializing the variable.

```
run_sum = 0.0
```

9. As you described above, the first step in calculating an average is to add all of the numbers in the list. We will be using a "do loop" to carry out this process. The structure of this loop is as follows:

1. Define how many times to run the loop
2. Read a line of the input file and store this value as a real variable
3. Add this value to the running sum
4. Repeat until all lines of the file have been read. (Defined by the parameter)
5. Close the input file

The code should be entered as follows:

```
do integer variable = 1, parameter_name
       read (file number, *) real variable
       sum variable = sum variable + real variable
       end do
close(number of input file)
```

10. The last step of calculating an average is to divide by the number of inputs. We have already stored this value as the parameter. We can use the sum variable to store the average. See if you can figure out how to write this line of the code.

11. We must also tell the computer to display the average. To do this type:

```
print*, sum variable
```

12. Lastly, we want to store this average in a new file to save for later. To do this, open the file that you created earlier to write to, following the procedure in step 7 and choosing a number between 90 and 99. Below this type:

```
write(the number you chose, *) sum variable
```

13. To end your program type:

```
end program your program name
```

14. Click "Compile": in the upper left hand corner to covert your code into a binary executable. If no error messages appear at the bottom half of the screen, click "Execute." The program should run and display the average in the bottom half of the screen. If you double click the file you wrote to, the number should appear there as well. If there is an error, use the information that the compiler gives you to debug your code.

15. Once your program works and you have shown your teacher, upload the new file "nameandgrades.txt" onto the website. Use what you know, as well as some Internet research to modify your program. Your new program should do the following.

   a. Read the new file line by line
   b. Print the name that is in the file

c. Next to the name, print "Yay!!!" if the grade is above a 70 and ":(" if the grade is below a 70.
d. Calculate the average of the grades.

**Hint: Conditional statements (such as "if" statements) might be helpful, I recommend that you look up how these statements work with a quick Google search. Try "conditional statements Fortran."

****

Molecular Dynamics Simulation

1. Describe three advantages of using computational chemistry over traditional lab techniques to explore chemical processes.

2. Briefly describe how a computer simulates chemical processes.

3. We will now use the programming that you just learned to predict the motion of 864 argon atoms in the liquid state. To begin, download the four files provided on the outreach website (http://ase.tufts.edu/chemistry/lin/outreach.html). The files are:

• `ft_mdroutines.f` - This file contains subroutines (functions that will be used in the MD simulation). Among these are the calculations of forces and motion of the particles.

• `ft_parameters.f` – This file contains all of the constant value parameters needed for the simulation, such as the size of the box, number of atoms in the simulation, and the mass of the atoms.

• `ft_main.f` - contains the "main" MD program. This program performs the simulation by calling the subroutines from `ft_mdroutines.f` according to parameters specified in `ft_parameters.f`

• `Ar_Lattice.gro` – This file contains the initial coordinates of all 864 argon atoms.

Upload all four of these files to Coding Ground (http://www.tutorialspoint.com/compile_fortran_online.php) as described in #2 of the previous section.

4. Double click the file `ft_mdroutines.f`. Look through the file and list three parts of the code that you recognize from the previous exercise. Describe what each of these subroutines do. Fill in the chart below with this information.

Code	Purpose

5. List any equations in the file `ft_mdroutines` that are familiar to you.

6. Before we run a simulation, we will look at the initial structure. Open VMD and select "File –› New Molecule" in the "VMD Main" window. Click "Browse…" and find the Ar_Lattice.gro file. Click "Open" in the "Choose a molecule file" window and then "Load" to load in the file. Adjust the representation so that you can better visualize the system. First select "Graphics –› Representations" in the VMD Main window, and under the "Drawing Method" dropdown menu select "VDW."

7. Briefly describe the arrangement of atoms that you see in the viewing window. Does this arrangement of atoms look like a liquid? Explain why or why not.

8. Since our goal is to simulate and calculate the properties of liquid argon, we must do something to get our system to actually look like liquid argon. Thus, the first thing we will do is perform an **equilibration simulation**. In this simulation, we will start with the lattice-like structure provided, and simulate it at a constant temperature (94.4 K, which is in argon's liquid state) for a period of time until the structure represents liquid argon.

9. One important calculation in this simulation is the calculation of the Leonard-Jones potential. Go to http://phet.colorado.edu/en/simulation/legacy/atomic-interactions and play with the simulation. The graph shows the Leonard-Jones potential between two atoms. Describe what happens to the potential energy as you move the atoms closer. What happens when you move the atoms farther apart? At what distances do the atoms repel? Describe where on the graph the atoms are most stable.

10. To perform the equilibration, go to the bottom green terminal window and type the following to compile the program in the correct order and then press "Enter":

```
gfortran ft_parameters.f ft_mdroutines.f ft_main.f –o equilibration
```

If no errors are found, type the following into the green window and press "enter" to run the program:

```
./equilibration
```

The program will now calculate the new positions of the Ar atoms.

11. Click the refresh button on the upper-left side of the screen. Four new files have been created. They are:

- `trajectory.xyz` - contains the coordinates of all of the atoms in the system at different time points in the simulation.
- `temperature.log` - contains the temperature of the system at different time points.
- `energy.log` - contains the kinetic, potential, and total energy of the system at different time points in the system.
- `end_structure.gro` - is a file that contains the coordinates and velocities of every atom in the system at the *very end* of the simulation.

The output frequency for each of these files is controlled by the `stride` *parameter.*

12. Download the file `trajectory.xyz` by clicking on the file, then click "File –›Download File".

13. Load `trajectory.xyz` into VMD as you did in step #6. By dragging the slider in the "VMD Main" window, you will be able to watch the movement of the atoms over time. Describe the new positions of the atoms and how they move. How does this structure compare to the unequilibrated system?

14. We will now simulate how the atoms in liquid Ar move during a short period of time (20 ps) in a production run. Use following steps to setup and run the production run in Coding Ground:

- Right click on the file `end_structure.gro` and change its name to `equil_structure.gro`. This way the production run will not overwrite this file.
- Download and save all of the files in the window onto a new folder on your computer. Name this folder "Equilibration"
- Double click the file `ft_parameters.f` and change the following paramenters
    - Find and change inputfile from '`Ar_Lattice.gro`' to the name of your equilibrated structure `equil_structure.gro`.
    - Change `Tcoupl` to `.FALSE.` Note the periods around the word `FALSE.` This will turn off temperature coupling in your simulation. We can do this because we are starting with a structure that already looks liquid like and has velocities that correspond to a reasonable liquid temperature (94.4 K). Since the structure is equilibrated, the temperature should stay close to our target without artificially maintaining it.
- Compile the program by typing the following into the bottom green terminal:

    ```
    gfortran ft_parameters.f ft_mdroutines.f ft_main.f –o production
    ```

    If no errors are found, type the following into the green window and press "enter" to run the program:

    ```
    ./production
    ```

After running the simulation, you will generate the same files as you did before. At this point, if you are using the Coding Ground environment, download all of the generated files to your desktop and place them in a new folder called "Production."

15. Load trajectory.xyz into VMD from the production folder as you did in step #6. By dragging the slider in the "VMD Main" window, you will be able to watch the movement of the atoms over 20 ps. Describe the new positions of the atoms and how they move in detail. Does the motion match what you would expect to see for the particles in a liquid? Explain why or why not.

Analysis

16. While it is certainly cool that we can run MD simulations and just look at the motions of systems on atomic scales, the quantitative information we can calculate from trajectories and compare to experimental data is part of what makes these simulations so useful.

A convenient way to look at the results of the simulation is to plot the data. Nearly any software including Microsoft Excel is capable of doing this.

We will plot four sets of data, each in a separate spreadsheet.

`energy.log` (from equilibration)	`energy.log` (from production run)
`temperature.log` (from equilibration)	`temperature.log` (from production run)

To import a given file into Excel,

- Select "File –› Import…" to open the import window.
- Click on the "Text file" radio button and then click "Import"
- Select your text file. Because the files generated by the simulation have extension ".log" you may need to change the selection in the "Enable" dropdown menu from "Text Files" to "All Files." Click "Get Data" to open the Text Import Wizard.
- Ensure the "Delimited" radio button is selected and click "Next >"
- Under "Delimiters," uncheck "Tab" and check "Space." Ensure that "Treat consecutive delimiters as one" is checked. Click "Next >"
- Use the "General" column data format and click "Finish."
- Select the place in your Excel sheet you want to put the data and click "OK."

Import each file from the equilibration and production runs into a separate spreadsheet and create a plot of each one. Be sure to include all titles and axes labels, including units. In the energy files, the first column is Time, measured in picoseconds. The second, third, and fourth columns are potential, kinetic, and total energy, respectively, each measured in attojoules (10^{-18} J). In the temperature file, the first column is time, and the second column is temperature in kelvin.

17. Describe the similarities and differences between the energy graphs for the equilibration and production runs. Provide an explanation for why these differences might occur. (Hint: think about the relationship between temperature and energy and the `Tcoupl` parameter)

18. Describe the similarities and differences between the temperature graphs for the equilibration and production runs. Provide an explanation for why these differences might occur. (Hint: think about the relationship between temperature and energy and the `Tcoupl` parameter).

Modifying the Parameters

19. Due to the **modular** nature of the MD code, it is rather easy to modify specific simulation parameters and tinker with the potential function or simulate a different system. For example, if we alter the particle mass and Lennard-Jones potential parameters (the parameters for different atoms can be found online), we would expect different properties in the resulting trajectories. Open `ft_parameters.f` and pick three parameters that you would like to change. In the chart below list the parameter that you want to change, how you are changing it, and what you expect will happen in the simulation because of this change.

Parameter to Change	New Parameter Value	Predicted Change in Simulation

20. Go to the Coding Ground website and repeat step #3. Run the equilibration as described in step #10. Once you have refreshed the file list, you can now change one of the parameters in `ft_parameters.f.` Repeat steps #14-16.

21. Describe any differences that you notice between this new simulation and the original production run in detail. Do these observations match your predictions? Why or why not?

22. Refresh Coding Ground and repeat steps #20-21 for the other two parameter changes and describe your observations below.

23. There are many other measurements that can be made using MD. Your teacher can guide you through these measurements if time allows.