# DynamO Workshop

## Tutorial Worksheet 1

This tutorial leads you through a repetition of the results of Alder and Wainwright in the discovery of an entropic freezing transition. Its purpose is to familiarise you with the tools of DynamO, and to learn how to automate handling DynamO.

## 1 Simulating a single hard-sphere system with DynamO

In this part of the tutorial, you will explore simulating a single component hard-sphere system using the two main tools of DynamO: **dynamod** and **dynarun**.
The steps are:

- To create an initial ”configuration” file, which is the starting point of the simulation, using **dynamod**.
- Use the initial configuration file as the start of a simulation. Equilibrate or ”relax” this configuration by simulating it for a short time. The output of this stage is hopefully then an ”equilibrated” configuration file.
- Use the equilibrated configuration as the start of a simulation to collect data. You will also generate a final configuration file which can be used as the starting point for other simulations.

The first part of the tutorial (manually running a simulation) is explained in detail in the online documentation:
`http://dynamomd.org/index.php/tutorial2`
Please follow the instructions there. In short, you will run the following commands in a terminal:

```
dynamod -m 0 -d 0.5 -C 7 -o config.start.xml
dynarun config.start.xml -c 1000000 -o config.equilibrated.xml
dynarun config.equilibrated.xml -c 1000000 -o config.end.xml
```

Please remember, you can replace any **dynarun** command with the **dynavis** command to visualise the simulation.

## 2 Automating a study

In this section of the tutorial, we will run several simulations to determine the location of the freezing transition in hard-sphere systems. The idea is to run simulations at a range of densities to plot pressure versus density and test for the appearance of a discontinuity indicating a first-order transition.
To automate simulations, we recommend using Python. This is a popular programming language which has many useful tools built-in, although there are many other choices available (e.g., BASH).
Open your favourite text editor and start a new file called *hardsphere.py*. To create a python script which repeats the manual simulations you've run above, we can use the *os.system* function of Python to call dynamod and dynarun.

Listing 1: hardsphere.py v1.0

```
#!/usr/bin/python
#Load the os functions
import os

os.system("dynamod -m 0 -d 0.5 -C 7 -o config.start.xml")
os.system("dynarun config.start.xml -c 1000000 -o config.equilibrated.xml")
os.system("dynarun config.equilibrated.xml -c 1000000 -o config.end.xml")
```

The first line is the "shebang" marker so that your system knows this is a python script file. We then load the *os* library, so that later on we can run the commands we typed by hand using the *os.system* function.
To run the file, call python on it:

```
python hardsphere.py
```

This script file is not much use; however, we can quickly see the power of the scripting approach by adding a loop:

Listing 2: hardsphere.py v2.0

```
#!/usr/bin/python
#Load the os functions
import os

#Create a list of densities to explore
densities=[0.5, 0.6, 0.7, 0.8, 0.9, 1.0, 1.1, 1.2, 1.3]
for density in densities:
    #Name files uniquely so that they are not overwritten
    startfile="config.start."+str(density)+".xml"
    equilfile="config.equilibrated."+str(density)+".xml"
    endfile="config.equilibrated."+str(density)+".xml"
    outputfile="output."+str(density)+".xml"

    #Run the simulation commands, inserting the computed variables
    os.system("dynamod -m 0 -d "+str(density)+" -C 7 -o "+startfile)
    os.system("dynarun "+startfile+" -c 100000 -o "+equilfile)

    #Ensure that the output data file for the last run has a unique name
    os.system("dynarun "+equilfile+" -c 1000000 -o "+endfile
              +" --out-data-file="+outputfile)
```

**Note:** If you copy/paste code, please check the indentation after the for loop is correct, as many PDF readers automatically remove spaces.
The code above now allows us to run a range of simulations to gather data. Once the simulations have run, we can then collect data from the saved "output.X.xml" files, where X is the density.
To automate data processing, you can use the XML library of Python. Open a new file called *hardsphere-data.py* and type/paste the following code:

Listing 3: hardsphere-data.py v1.0

```
#!/usr/bin/python
#Load the xml library
import xml.etree.ElementTree as ET

#Keep the same densities as before
densities=[0.5, 0.6, 0.7, 0.8, 0.9, 1.0, 1.1, 1.2, 1.3]
for density in densities:
    outputfile="output."+str(density)+".xml"
    #Parse the output file as XML
    xmldoc = ET.parse(outputfile)
    #Grab the Avg pressure attribute from the Pressure tag
    PressureTag=xmldoc.find("./Misc/Pressure")
    print density, PressureTag.attrib["Avg"]
```

This will load each of the output files from before, and print the density and pressure to the screen. You should obtain the following output:

Listing 4: hardsphere-data.py Output

```
0.5 1.6375497518040183
0.6 2.5801186193033998
0.7 4.0117180029763091
0.8 6.2332025739044852
0.9 9.7059148830985915
1.0 10.241418847388031
1.1 14.591946519458203
1.2 23.361056966819294
1.3 47.775997502232883
```

For more examples and information on XML, XPath expressions, or XML support in other languages, see tutorial A: `http://dynamomd.org/index.php/tutorialA`
If you plot the density and pressure values you obtain, a discontinuity in the pressure indicating a phase change should be clearly visible.
One method of plotting is to use the matplotlib of Python. Below is a modified version of hardsphere-data.py which collects the pressures during a loop and plots it to the screen:

Listing 5: hardsphere-data.py v2.0

```
#!/usr/bin/python
#Load the xml library
import xml.etree.ElementTree as ET

densities=[0.5, 0.6, 0.7, 0.8, 0.9, 1.0, 1.1, 1.2, 1.3]
#Create an empty list, ready to collect the pressures within
pressures=[]
for density in densities:
    outputfile="output."+str(density)+".xml"
    xmldoc = ET.parse(outputfile)
    PressureTag=xmldoc.find("./Misc/Pressure")
    #Append the pressure value (after converting it from a string to a float)
    pressures.append(float(PressureTag.attrib["Avg"]))

import matplotlib as mp
mp.plot(densities, pressures)
```

# 3 Tutorial Challenges

- Plot pressure versus density and compare with predictions from the Carnahan-Starling equation of state.

$$\frac{p}{\rho\, k_B\, T} = \frac{1+\eta+\eta^2-\eta^3}{(1-\eta)^3} \tag{1}$$

- Plot the known transition densities of $\rho_{fluid} \approx 0.936$ and $\rho_{solid} \approx 1.036$. Why does the solid phase appear so readily in these simulations (over almost the full range of the coexistance densities)?

- Run a separate simulation to calculate the radial distribution function $g(r)$ on an output file from a system above and below the fluid-solid transition. Compare the two and look for evidence of a regular close-packed arrangement. You will need to load the RadialDistribution output plugin using the following option to dynarun: "-L RadialDistribution:BinWidth=0.01" to collect this data. This is a ticker property, so you will need to specify a sampling time. For more information on output plugins, see:
`http://dynamomd.org/index.php/outputplugins`

- Enable the Mean-Square-Displacement output plugin (add "-L MSD" to the final dynarun command) and collect diffusion data for each density. Confirm that the freezing transition corresponds to a sharp reduction in the diffusion coefficient.

- Run the final "production" dynarun several times (feeding the output of the previous run as the starting point for the next). Use these repeated measurements to estimate the error in the calculation of the pressure/diffusivity.