

Kickoff assignment

This document provides you with the fundamental tools used in your project. You will learn how to

- use the terminal (command line)
- connect to the computer clusters (Habrok, Nieuwpoort, etc.)
- run your first simulation
- analyse the simulations using python
- interpret the results

At the end of this tutorial, you should be able to

- find your way in the terminal using the fundamental commands
- connect to Habrok either directly from terminal (for mac and linux users) or through Putty (for windows users)
- run your first simulation on the interactive node of Habrok
- understand the Bead-Spring model and the unit system
- call Python functions
- plot your results using matplotlib
- compare your numerical results to known theory

Throughout this document, you will encounter some tasks. Please complete them in 10 days. You are highly encouraged to talk to each other and collaborate. If you are stuck in any of these steps, ask your supervisor for help.

Part 1: Introduction to command line

The command line, or terminal, is a text-based interface to the system. You can navigate through files, run programs, and even connect to other computers. For many computational projects, especially those involving computer clusters, the command line is an essential tool.

Here are some fundamental commands you will use:

- `pwd` to print the directory you're currently in
- `ls` to list the files and directories in the current directory
- `cd` to change the directory
- `mkdir` to create a new directory
- `rm` to remove files or directories
- `cp` to copy files or directories

DANGER: Never type `rm -rf *` in your home directory. This will delete everything in your system and it's not reversible.

Remote connection to Habrok

To connect to computer clusters like Habrok and Nieuwpoort, you'll use SSH (Secure Shell). SSH allows you to securely connect to another computer over the internet. Click [here](#) for more information on how to connect

to Habrok

For Mac and Linux users, open the terminal and type:

```
ssh s123456@interactive1.hb.hpc.rug.nl
```

Replace **s123456** by your student number.

There are 4 possible ways of connecting to Habrok. If the above node is irresponsive, you can try the following:

```
ssh s123456@login1.hb.hpc.rug.nl  
ssh s123456@login2.hb.hpc.rug.nl  
ssh s123456@interactive2.hb.hpc.rug.nl
```

For Windows users, you'll need to download and install Putty, a free SSH client. Once installed, open Putty, enter the hostname (e.g., **s123456@interactive1.hb.hpc.rug.nl**), and click 'Open' to connect. More information for Windows users [here](#)

Task 1: Complete [this tutorial](#) on the command line

Optional task: Watch [this video](#)

Task 2: Connect to Habrok and type the following command

```
echo $USER
```

What is the output on the screen?

Part 2: Running your first simulation

After connecting to Habrok, you'll navigate to a suitable directory to run your simulation. We will run this simple simulation on the interactive node, but please note that the purpose of the interactive node is to test your simulations. Never run an actual simulation on the login/interactive nodes once you start working your own project.

First, navigate to the scratch partition on Habrok by typing the following.

```
cd /scratch/$USER
```

Task 3: Create a new directory named **tutorial** in this partition.

Before running our simulations, we need to load our simulation software LAMMPS. This is already installed on Habrok. Type the following:

```
module load LAMMPS/23Jun2022-foss-2021b-kokkos  
lmp
```

in the given order. `lmp` is the name of the executable that launches the LAMMPS interface. You should see something like the following after typing this command.

```
LAMMPS (23 Jun 2022 - Update 1)  
OMP_NUM_THREADS environment is not set. Defaulting to 1 thread.  
(src/comm.cpp:98)  
using 1 OpenMP thread(s) per MPI task
```

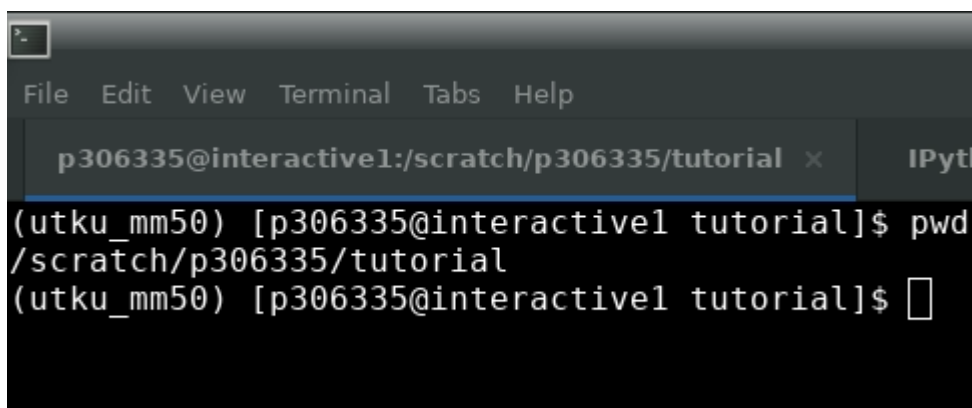
Once you verify that you have access to `lmp` command, hit `ctrl+c` to exit. Now, we are ready to run our first simulation.

Task 4: Navigate to `tutorial` and copy the scripts and the data files from your computer to here.

Type the following and make sure that you are in the correct directory and you have the necessary files.

```
pwd  
ls
```

Make sure that your output is the same as the ones below

A screenshot of a terminal window with a dark background. The terminal title bar shows 'p306335@interactive1:/scratch/p306335/tutorial'. The terminal content shows a prompt '(utku_mm50) [p306335@interactive1 tutorial]\$' followed by the command 'pwd' and its output '/scratch/p306335/tutorial'. The prompt is followed by a cursor.

```
(utku_mm50) [p306335@interactive1 tutorial]$ pwd  
/scratch/p306335/tutorial  
(utku_mm50) [p306335@interactive1 tutorial]$
```

Let's generate a polymer chain that we will simulate. We use Python to generate our polymer architecture. This is a simple example of an architecture creation, but you will use the same principle in your project too.

Task 5: Generate a polymer chain of length 100 using the command below

```
python create_chain.py 100
```

Here, 100 is an input to the Python script. The script is initially written to generate star like architectures that you might work with. You don't need to know what the script does at this point.

Task 6: Run the simulation using the command below

Now, we are ready to run our first simulation. Type the following

```
lmp -in in.single_chain
```

This should take approximately two minutes and you will see that information on your screen at the end of the simulation.

```
...
Loop time of 106.342 on 1 procs for 1000000 steps with 100 atoms
Performance: 4062348.663 tau/day, 9403.585 timesteps/s
99.9% CPU use with 1 MPI tasks x 1 OpenMP threads

MPI task timing breakdown:
Section | min time | avg time | max time | %varavg | %total
-----|-----|-----|-----|-----|-----
Pair    | 0.46523  | 0.46523  | 0.46523  | 0.0      | 0.44
Bond    | 1.3908   | 1.3908   | 1.3908   | 0.0      | 1.31
Neigh   | 98.799   | 98.799   | 98.799   | 0.0      | 92.91
Comm    | 3.015    | 3.015    | 3.015    | 0.0      | 2.84
Output  | 0.052687 | 0.052687 | 0.052687 | 0.0      | 0.05
Modify  | 2.2586   | 2.2586   | 2.2586   | 0.0      | 2.12
Other   | 0.3607   | 0.3607   | 0.3607   | 0.0      | 0.34

Nlocal:      100 ave
Histogram: 1 0 0 0 0 0 0 0 0
Nghost:      0 ave
Histogram: 1 0 0 0 0 0 0 0 0
Neighs:      43 ave
Histogram: 1 0 0 0 0 0 0 0 0

Total # of neighbors = 43
Ave neighs/atom = 0.43
Ave special neighs/atom = 1.98
Neighbor list builds = 98084
Dangerous builds = 81554
System init for write data ...
Generated 0 of 0 mixed pair coeff terms from geometric mixing rule
WARNING: Communication cutoff 1.4200000000000002 is shorter than a bond length based estimate of 1.7500000000000001. This may lead to errors. (src/comm.cpp:727)
Total wall time: 0:01:48
(utku_mm50) [p306335@interactivel tutorial]$
```

Task 7: Search for all the command you see in [in.single_chain](#) on LAMMPS [documentation](#) and learn what they do

Part 3: Introduction to Polymer Physics

Polymer physics is a key area of study in materials science and engineering, focusing on understanding the behaviour of polymeric materials. Polymers are large molecules composed of repeating units, and they play a crucial role in a wide range of applications, from everyday household items to high-performance engineering materials. One of the fundamental models used in polymer physics, especially in coarse-grained molecular dynamics (CGMD) simulations, is the Bead-Spring model. This model represents polymers as a series of beads (monomers) connected by springs (bonds), simplifying the complex molecular structure into a more manageable form. This abstraction is particularly useful for studying the macroscopic properties of polymers, such as their elasticity, viscosity, and thermal behaviour from fundamental physics principles.

Task 8: Read the following chapters of Introduction to Polymer Physics by M. Doi

- Chapter 1
- Chapter 4.1, 4.2

The unit system in CGMD simulations typically involves reduced units. This system simplifies calculations and can be converted to real-world units based on the material being simulated if needed (you will probably not).

The unit system in our LAMMPS simulations is known as Lennard-Jones (LJ) units. You will hear this name quite often.

Task 9: Read the [wikipedia page](#) of Lennard-Jones potential and corresponding LAMMPS units

In the next section, you will learn about how to use python to calculate certain polymer properties. Here, we give a brief introduction to those.

1. End-to-end distance: The measure of the linear size of the polymer molecule, important for understanding the spatial configuration of the polymer chain.

$$\vec{R}_{ee} = \sum_{i=1}^n \vec{r}_i$$

2. Radius of gyration: A key metric for the size of the polymer, reflecting how its mass is distributed around the center of mass.

$$R_g^2 = \frac{1}{N} \sum_{i=1}^N (\vec{R}_i - \vec{R}_{CM})^2$$

3. Mean squared displacement: Offers insights into the motion of polymer segments over time, critical for studying diffusion processes.

$$\langle \Delta \vec{r}^2 \rangle = \langle (\vec{r}(t) - \vec{r}(0))^2 \rangle = \frac{1}{N} \sum_{i=1}^N (\vec{r}_i(t) - \vec{r}_i(0))^2$$

Part 4: Introduction to Python

Python is a powerful programming language that's widely used in scientific computing for its simplicity and the extensive library ecosystem. For analysing simulations, you'll often use libraries such as Numpy for numerical calculations and matplotlib for plotting. On top of the most non-standard standard libraries of Python (Numpy, scipy, pandas, etc.), we heavily rely on the following libraries: MDAnalysis, freud, atooms, and signac. It is very likely that you're not familiar with these libraries. Luckily you can always ask us or read their documentations. The latter is highly encouraged and [here](#) is a guideline on how to read the documentation. Note that this applies to terminal commands as well. If you don't remember how to use a certain command, type `man` in front of the command. For python functions, type `help(<your_function_here>)` in your interpreter. To be able to use these libraries you need to install them in your Python environment. We will use conda environments. Conda is a package and environment manager.

Optional task: Read more about what conda is and what it does [here](#).

Task 10: Create a conda environment on Habrok and install the required packages.

You don't need to install conda on Habrok. Use the existing module similar to how we loaded LAMMPS. Run

```
module load Anaconda3/2023.03-1
conda --version
```

The output should be similar to this `conda 23.7.4`. We've made a list of libraries you need in the `environment.yml` file.

==NOTE: Make sure to modify the line `prefix: /home4/s123456/` in `environment.yml` file with the correct path before running the below command.== Type `cd /scratch/$USER` and replace **prefix:** `/home4/s123456/` by the output of this command. You can use text editors such as `vim` or `nano` on Habrok. These are pre-installed in all Linux distributions, so you can simply type `vim` or `nano` to get started.

Once conda is loaded and the environment file is updated accordingly, you can run the below command to install the libraries. This process might take a couple of minutes, be patient.

```
conda env create -f environment.yml
```

Task 11: Run the post processing script and plot the results.

We can now analyse our first simulation. The following script computes the mean-squared displacement of the beads over the simulation trajectory. It also computes the end-to-end distance of our polymer chain at each time frame. Your task is to run the `post_processing.py` script and plot the results.

```
python post_processing.py
```

Plot the analysis results:

```
import numpy as np
import matplotlib.pyplot as plt

# Load the files
times_log = np.loadtxt('times_log.txt')
#####
# YOUR CODE GOES HERE #

#####

# What is the average end-to-end distance?
# Write a code to compute this average and the standard deviation.
# You can use numpy.

#####
# YOUR CODE GOES HERE #

#####

# Make two separate figures using matplotlib
# Plot the radius of gyration as a function of time (linearly)
# Plot the mean-squared displacement as a function of time
(logarithmically)
```

```
#####  
# YOUR CODE GOES HERE #  
  
#####
```

Part 5: Interpreting your results

After running your simulation and analysing the data with Python, the next step is to interpret your results. This involves understanding what the data tell you about the polymer's behaviour under certain conditions and comparing your findings with theoretical predictions. Answer the following questions based on the theory you've learnt and on your simulation results.

1. What is the expected end-to-end distance of a polymer chain with 100 beads?
2. What is the end-to-end distance you measured? How does it compare to the scaling law that you've learnt about?
 - Explain the difference between the theoretical and the numerical values of R_{ee}