A Set of Tutorials for the LAMMPS Simulation Package [Article v1.0]

Simon Gravelle[1*], **Cecilia M. S. Alvares**[2], **Jacob R. Gissinger**[3], **Axel Kohlmeyer**[4]

[1]University Grenoble Alpes, CNRS, LIPhy, Grenoble, 38000, France; [2]Department of Chemistry, University of Warwick, Coventry CV4 7AL, United Kingdom; [3]Stevens Institute of Technology, Hoboken, NJ 07030, USA; [4]Institute for Computational Molecular Science, Temple University, Philadelphia, PA 19122, USA

This LiveCoMS document is maintained online on GitHub at https://github.com/lammpstutorials/lammpstutorials-article; to provide feedback, suggestions, or help improve it, please visit the GitHub repository and participate via the issue tracker.

This version dated September 30, 2025

Abstract The availability of open-source molecular simulation software packages allows scientists and engineers to focus on running and analyzing simulations without having to write, parallelize, and validate their own simulation software. While molecular simulations thus become accessible to a larger audience, the "black box" nature of such software packages and wide array of options and features can make it challenging to use them correctly, particularly for beginners in the topic of simulations. LAMMPS is one such versatile molecular simulation code, designed for modeling particle-based systems across a broad range of materials science and computational chemistry applications, including atomistic, coarse-grained, mesoscale, grid-free continuum, and discrete element models. LAMMPS is capable of efficiently running simulations of varying sizes from small desktop computers to large-scale supercomputing environments. Its flexibility and extensibility make it ideal for complex and extensive simulations of atomic and molecular systems, and beyond. This article introduces a suite of tutorials designed to make learning LAMMPS more accessible to new users. The first four tutorials cover the basics of running molecular simulations in LAMMPS with systems of varying complexities. The second four tutorials address more advanced molecular simulation techniques, specifically the use of a reactive force field, grand canonical Monte Carlo, enhanced sampling, and the REACTER protocol. In addition, we introduce LAMMPS–GUI, an enhanced cross-platform graphical text editor specifically designed for use with LAMMPS and able to run LAMMPS directly on the edited input. LAMMPS–GUI is used as the primary tool in the tutorials to edit inputs, run LAMMPS, extract data, and visualize the simulated systems.

***For correspondence:**
simon.gravelle@cnrs.fr (S.G.)

1 Introduction

Molecular simulations can be used to model a large variety of atomic and coarse-grained systems, including solids, fluids, polymers, and biomolecules, as well as complex interfaces and multi-component systems. While various molecular modeling methods exist, molecular dynamics (MD) and Monte Carlo (MC) are most commonly used. MD is the preferred method for obtaining the accurate dynamics of a system, as it relies on solving Newton's equations of motion. For systems with many degrees of freedom or complex energy landscapes, the MC method can be a better choice than MD because it allows for efficiently exploring a configuration space without being confined by the accessible time scale. MC involves performing random changes to the

Received: 18 March 2025
Accepted: 29 September 2025

https://doi.org/10.33011/livecoms.6.1.3037
Living J. Comp. Mol. Sci. 2025, 6(1), 3037

system configuration that are either accepted or rejected based on energy criteria [1, 2]. Molecular simulations allow for measuring a broad variety of properties, including structural properties (e.g., bond length distribution, coordination numbers, radial distribution functions), thermodynamic properties (e.g., temperature, pressure, volume), dynamic behaviors (e.g., diffusion coefficient, viscosity), and mechanical responses (e.g., elastic constant, Poisson ratio). Some of these quantities can be directly compared with experimental data, enabling the validation of the simulated system, while others, available only through simulations, are often useful for interpreting experimental data [3].

LAMMPS (Large-scale Atomic/Molecular Massively Parallel Simulator) [4] is a highly flexible and parallel open-source molecular simulation tool. Over the years, a broad variety of particle interaction models have been implemented in LAMMPS, enabling it to model a wide range of systems, including atomic, polymeric, biological, metallic, reactive, granular, mesoscale, grid-free continuum, and coarse-grained systems [5]. LAMMPS can be used on a single CPU core, a multi-socket and multi-core server, an HPC cluster, or a high-end supercomputing system. It can efficiently handle complex and large-scale simulations, including hybrid MPI-OpenMP parallelization and MPI + GPU acceleration (for a subset of its functionality).

LAMMPS requires users to write detailed input files, a task that can be particularly challenging for new users. Although its documentation extensively describes all available features [6], navigating it can be challenging. Much of the information may be unnecessary for common use cases, and the detailed manual can often feel overwhelming. Beyond the intrinsic complexity of LAMMPS, performing accurate simulations requires several complex, system-specific decisions regarding the physics to be modeled, such as selecting the thermodynamic ensemble (e.g., microcanonical, grand canonical), determining the simulation duration, and choosing the sets of parameters describing the interactions between atoms (the so-called force field) [7--9]. While these choices are independent of the simulation software, they may occasionally be constrained by the features available in a given package. The tutorials in this article aim to flatten the learning curve and guide users in performing accurate and reliable molecular simulations with LAMMPS.

1.1 Scope

This set of tutorials consists of eight tutorials arranged in order of increasing difficulty. Although each tutorial can be read independently, information introduced in earlier tutorials is generally not repeated in detail in later ones. For this reason, we recommend that beginners follow the tutorials in order. The novelties associated with each tutorial are briefly described below.

In Tutorial 1, the structure of LAMMPS input files is illustrated through the creation of a simple atomic Lennard-Jones fluid system. Basic LAMMPS commands are used to set up interactions between atoms, perform an energy minimization, and finally run a simple MD simulation in the microcanonical (NVE) and canonical (NVT) ensembles.

In Tutorial 2, a more complex system is introduced in which atoms are connected by bonds: a small carbon nanotube. The use of both classical and reactive force fields (here, OPLS-AA [10] and AIREBO [11], respectively) is illustrated. An external deformation is applied to the CNT, and its deformation is measured. This tutorial also demonstrates the use of an external tool to visualize breaking bonds, and show the possibility to import LAMMPS-generated YAML log files into Python.

In Tutorial 3, two components—liquid water (flexible three-point model) and a polymer molecule—are merged and equilibrated. A long-range solver is used to handle the electrostatic interactions accurately, and the system is equilibrated in the isothermal-isobaric (NPT) ensemble; then, a stretching force is applied to the polymer. Through this relatively complex solvated polymer system, the tutorial demonstrates how to use type labels to make molecule files more generic and easier to manage [12].

In Tutorial 4, an electrolyte is confined between two walls, illustrating the specifics of simulating systems with fluid-solid interfaces. With the rigid four-point TIP4P/2005 [13] water model, this tutorial uses a more complex water model than Tutorial 3. A non-equilibrium MD is performed by imposing shear on the fluid through moving the walls, and the fluid velocity profile is extracted.

In Tutorial 5, the ReaxFF reactive force field, which is specifically designed to simulate chemical reactions by dynamically adjusting atomic interactions [14], is used. ReaxFF includes charge equilibration (QEq), a method that allows the partial charges of atoms to adjust according to their local environment.

In Tutorial 6, the adsorption of a fluid in silica pores is modeled. To do so, a Monte Carlo simulation in the grand canonical ensemble is implemented to demonstrate how LAMMPS can be used to simulate an open system that exchanges particles with a reservoir.

In Tutorial 7, an advanced free energy method called umbrella sampling is implemented. By calculating an energy barrier, this tutorial describes a protocol for addressing energy landscapes that are difficult to sample using classical MD or MC methods.

In Tutorial 8, a CNT embedded in nylon-6,6 polymer melt is simulated. The REACTER protocol is used to model the poly-

https://doi.org/10.33011/livecoms.6.1.3037
Living J. Comp. Mol. Sci. 2025, 6(1), 3037

merization of nylon, and the formation of water molecules is tracked over time [15].

2 Prerequisites

2.1 Background knowledge

This set of tutorials assumes no prior knowledge of the LAMMPS software itself. To complete the tutorials, a text editor and a suitable LAMMPS executable are required. We use LAMMPS–GUI [16] here, as it offers features that make it particularly convenient for tutorials, but other console or graphical text editors, such as GNU nano, vi/vim, Emacs, Notepad, Gedit, and Visual Studio Code can also be used. LAMMPS can be executed either directly from LAMMPS–GUI (Appendix A) or from a command prompt (Appendix B), the latter of which requires some familiarity with executing commands from a terminal or command-line prompt.

In addition, prior knowledge of the theoretical basics of molecular simulations and statistical physics is highly beneficial. Users may refer to textbooks such as *Understanding Molecular Simulation* by Daan Frenkel and Berend Smit [1], as well as *Computer Simulation of Liquids* by Michael Allen and Dominic Tildesley [2]. To better understand the fundamental concepts behind the soft matter systems simulated in these tutorials, users can also refer to *Basic Concepts for Simple and Complex Liquids* by Jean-Louis Barrat and Jean-Pierre Hansen [17], as well as *Theory of Simple Liquids: with Applications to Soft Matter* by Jean-Pierre Hansen and Ian Ranald McDonald [18]. For more resources, the SklogWiki platform provides a wide range of information on statistical mechanics and molecular simulations [19].

2.2 Software/system requirements

The LAMMPS stable release version 22Jul2025 [20] and the matching LAMMPS–GUI software version 1.7.0 are required to follow the tutorials, as they include features that were first introduced in these versions. For Linux (x86_64 CPU), macOS (BigSur or later), and Windows (10 and 11) you can download a pre-compiled LAMMPS package from the LAMMPS release page on GitHub [21]. Select a package with 'GUI' in the file name, which includes both, LAMMPS–GUI and the LAMMPS command-line executable. These pre-compiled packages are designed to be portable, and therefore omit support for parallel execution with MPI. Instructions for installing LAMMPS–GUI and using its most relevant features for the tutorials are provided in Appendix A.

LAMMPS versions are generally backward compatible, meaning that older input files typically work the same with newer versions of LAMMPS. However, forward compatibility is not as strong, so input files written for a newer version may not always work with older versions. As a result, it is usually possible to follow this tutorial with more recent releases of LAMMPS–GUI and LAMMPS; older versions may require some (minor) adjustments. These tutorials will be periodically updated to ensure compatibility and benefit from new features in the latest stable version of LAMMPS.

For some tutorials, external tools are required for plotting and visualization, as the corresponding functionality in LAMMPS–GUI is limited. Suitable tools for plotting include Python with Pandas/Matplotlib [22, 23], XmGrace, Gnuplot, Microsoft Excel, or LibreOffice Calc. For visualization, suitable tools include VMD [24, 25] and OVITO [26, 27].

2.3 About LAMMPS–GUI

LAMMPS–GUI is a graphical text editor, enhanced for editing LAMMPS input files and linked to the LAMMPS library, allowing it to run LAMMPS directly. The text editor is similar to other graphical editors, such as Notepad or Gedit, but offers the following enhancements for running LAMMPS:

- Wizard dialogs to set up these tutorials
- Auto-completion of LAMMPS commands and options
- Context-sensitive online help
- Syntax highlighting for LAMMPS input files
- Syntax-aware line indentation
- Editor switches working directory to that of input file
- Visualization using LAMMPS' built-in renderer
- Start and stop simulations via mouse or keyboard
- Monitoring of simulation progress and parallelization
- Dynamic capture of LAMMPS output in a text window
- Automatic plotting of thermodynamic data during runs
- Capture of "dump image" outputs for animations
- Export of thermodynamic data for external plotting
- Inspection of binary restart files

Appendix A contains basic instructions for installation and using LAMMPS–GUI with the tutorials presented here. A complete description of all LAMMPS–GUI features can be found in the LAMMPS manual [16].

3 Content and links

The tutorials described in this article can be accessed at lammpstutorials.github.io, where additional exercises with solutions are also provided. All files and inputs required to follow the tutorials are available from a dedicated GitHub organization account, github.com/lammpstutorials. These files can also be downloaded by clicking «`Start LAMMPS Tutorial X`» (where `X` = 1…8) from the «`Tutorials`» menu of LAMMPS–GUI.

In the following, all LAMMPS input or console commands are formatted with a `colored background`. Keyboard shortcuts and file names are formatted in `monospace`, and

https://doi.org/10.33011/livecoms.6.1.3037
Living J. Comp. Mol. Sci. 2025, 6(1), 3037

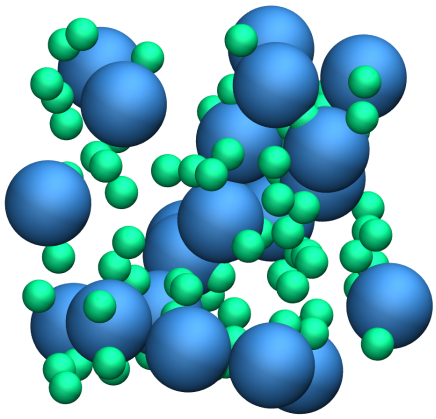

Figure 1. The binary mixture simulated in Tutorial 1, with the atoms of type 1 represented as small green spheres and the atoms of type 2 as large blue spheres.

LAMMPS–GUI options and menus are displayed in «quoted monospace».

3.1 Tutorial 1: Lennard-Jones fluid

The objective of this tutorial is to perform simple MD simulations using LAMMPS. The system consists of a Lennard-Jones fluid composed of neutral particles with two different effective diameters, contained within a cubic box with periodic boundary conditions (Fig. 1). In this tutorial, basic MD simulations in the microcanonical (NVE) and canonical (NVT) ensembles are performed, and basic quantities are calculated, including the potential and kinetic energies.

3.1.1 My first input

To run a simulation using LAMMPS, you need to write an input script containing a series of commands for LAMMPS to execute, similar to Python or Bash scripts. For clarity, the input scripts for this tutorial will be divided into five categories, which will be filled out step by step. To set up this tutorial, select «Start LAMMPS Tutorial 1» from the «Tutorials» menu of LAMMPS–GUI, and follow the instructions. This will select (or create, if needed) a folder, place the initial input file `initial.lmp` in it, and open the file in the LAMMPS–GUI «Editor» window. It should display the following content:

```
# PART A – ENERGY MINIMIZATION
# 1) Initialization
# 2) System definition
# 3) Settings
# 4) Monitoring
# 5) Run
```

Everything that appears after a hash symbol (#) is a comment and ignored by LAMMPS. LAMMPS–GUI will color such comments in red. These five categories are not required in every input script and do not necessarily need to be in that exact order. For instance, the `Settings` and the `Monitoring` categories could be inverted, or the `Monitoring` category could be omitted. However, note that LAMMPS reads input files from top to bottom and processes each command *immediately*. Therefore, the `Initialization` and `System definition` categories must appear at the top of the input, and the `Run` category must appear at the bottom. Also, the specifics of some commands can change after global settings are modified, so the order of commands in the input script is important.

Initialization

In the first section of the script, called `Initialization`, global parameters for the simulation are defined, such as units, boundary conditions (e.g., periodic or non-periodic), and atom types (e.g., uncharged point particles or extended spheres with a radius and angular velocities). These commands must be executed *before* creating the simulation box or they will cause an error. Similarly, many LAMMPS commands may only be entered *after* the simulation box is defined. Only a limited number of commands may be used in both cases. Update the `initial.lmp` file so that the `Initialization` section appears as follows:

```
# 1) Initialization
units lj
dimension 3
atom_style atomic
boundary p p p
```

Strictly speaking, none of the four commands specified in the Initialization section are mandatory, as they correspond to the default settings for their respective global properties. However, explicitly specifying these defaults is considered good practice to avoid confusion when sharing input files with other LAMMPS users.

The first line, `units lj`, specifies the use of *reduced* units, where all quantities are dimensionless. This unit system is a popular choice for simulations that explore general statistical mechanical principles, as it emphasizes relative differences between parameters rather than representing any specific material. The second line, `dimension 3`, specifies that the simulation is conducted in 3D space, as opposed to 2D, where atoms are confined to move only in the xy-plane. The third line, `atom_style atomic`, designates the atomic style for representing simple, individual point particles. In this style, each particle is treated as a point with a mass, making it the most basic atom style. Other atom styles can incorporate additional attributes for atoms, such as charges, bonds, or molecule IDs, depending on the requirements of the simu-

https://doi.org/10.33011/livecoms.6.1.3037
Living J. Comp. Mol. Sci. 2025, 6(1), 3037

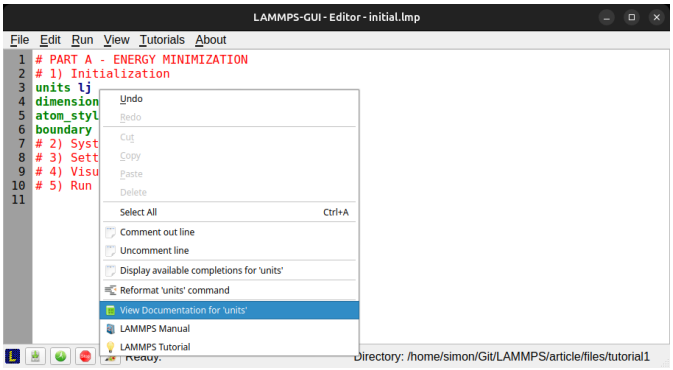

Figure 2. Screenshot of the LAMMPS–GUI «Editor» window during Tutorial 1. The pop-up menu is the context menu for right-clicking on the `units lj` command.

lated model. The last line, `boundary p p p`, indicates that periodic boundary conditions are applied along all three directions of space, where the three p stand for *x*, *y*, and *z*, respectively. Alternatives are fixed non-periodic (f), shrink-wrapped non-periodic (s), and shrink-wrapped non-periodic with minimum (m). For non-periodic boundaries, different options can be assigned to each dimension, making configurations like `boundary p p fm` valid for systems such as slab geometries.

Each LAMMPS command is accompanied by extensive online documentation that lists and discusses the different options for that command. Most LAMMPS commands also have default settings that are applied if no value is explicitly specified. The defaults for each command are listed at the bottom of its documentation page. From the LAMMPS–GUI editor buffer, you can access the documentation by right-clicking on a line containing a command (e.g., `units lj`) and selecting «View Documentation for 'units'». This action should prompt your web browser to open the corresponding URL for the online manual. A screenshot of this context menu is shown in Fig. 2.

System definition

The next step is to create the simulation box and populate it with atoms. Modify the `System definition` category of `initial.lmp` as shown below:

```
# 2) System definition
region simbox block -20 20 -20 20 -20 20
create_box 2 simbox
create_atoms 1 random 1500 34134 simbox overlap 0.3
create_atoms 2 random 100 12756 simbox overlap 0.3
```

The first line, `region simbox (...)`, defines a region named `simbox` that is a block (i.e., a rectangular cuboid) extending from -20 to 20 units along all three spatial dimensions. The second line, `create_box 2 simbox`, initializes a simulation box based on the region `simbox` and reserves space for two types of atoms. In LAMMPS, every atom is assigned an *atom type* property. This property selects which force field parameters (here, the Lennard-Jones parameters, ϵ_{ij} and σ_{ij}) are applied to each pair of atoms via the `pair_coeff` command (see below). We discuss in Tutorial 2 how this applies to many-body pair styles, and in Tutorial 3 how this applies to Coulomb interactions.

From this point on, the number of atom types is "locked in", and any command referencing an atom type larger than 2 will trigger an error. While it is possible to allocate more atom types than needed, you must assign a mass and provide force field parameters for each atom type. Failing to do so will cause LAMMPS to terminate with an error.

The third line, `create_atoms (...)`, generates 1500 atoms of type 1 at random positions within the `simbox` region. The integer 34134 is a seed for the internal random number generator, which can be changed to produce different sequences of random numbers and, consequently, different initial atom positions. The fourth line adds 100 atoms of type 2. Both `create_atoms` commands use the optional argument `overlap 0.3`, which enforces a minimum distance of 0.3 length units between the randomly placed atoms. This constraint helps avoid "close contacts" between atoms, which can lead to excessively large forces and simulation instability. Each created atom in LAMMPS is automatically assigned a unique atom ID, which serves as a numerical identifier to distinguish individual atoms throughout the simulation. Atom IDs by default have the range from 1 to the total number of atoms, but this is not enforced. Deleting atoms, for example, causes "holes" in the list of atom IDs.

Another way to define a system in LAMMPS, besides the `create_atoms` commands, is to import an existing topology file containing atomic coordinates as well as, optionally, other attributes such as atomic velocities and the force field parameters using the `read_data` command, as shown in Tutorial 2.

Settings

Next, we specify the settings for the two atom types. Modify the `Settings` category of `initial.lmp` as follows:

```
# 3) Settings
mass 1 1.0
mass 2 5.0
pair_style lj/cut 4.0
pair_coeff 1 1 1.0 1.0
```


https://doi.org/10.33011/livecoms.6.1.3037
Living J. Comp. Mol. Sci. 2025, 6(1), 3037

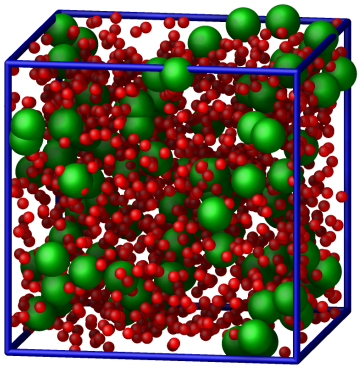

Figure 3. The binary mixture simulated in Tutorial 1. This image was generated directly from the LAMMPS–GUI. Atoms of type 1 are represented as small red spheres, atoms of type 2 as large green spheres, and the edges of the simulation box are represented as blue sticks.

```
pair_coeff 2 2 0.5 3.0
```

The two `mass` commands assign a mass of 1.0 and 5.0 units to the atoms of type 1 and 2, respectively. The third line, `pair_style lj/cut 4.0`, specifies that the atoms will be interacting through a Lennard-Jones (LJ) potential with a cut-off equal to r_c = 4.0 length units [28, 29]:

$$E_{ij}(r) = 4\epsilon_{ij}\left[\left(\frac{\sigma_{ij}}{r}\right)^{12} - \left(\frac{\sigma_{ij}}{r}\right)^{6}\right], \text{ for } r < r_c, \qquad (1)$$

where r is the inter-particle distance, ϵ_{ij} is the depth of the potential well that determines the interaction strength, and σ_{ij} is the distance at which the potential energy equals zero. The indices i and j refer to pairs of atoms with the corresponding atom types. The fourth line, `pair_coeff 1 1 1.0 1.0`, specifies the Lennard-Jones coefficients for interactions between pairs of atoms that both have atom type 1: the energy parameter ϵ_{11} = 1.0 and the distance parameter σ_{11} = 1.0. Similarly, the last line sets the Lennard-Jones coefficients for interactions between atoms of type 2, ϵ_{22} = 0.5, and σ_{22} = 3.0.

By default, LAMMPS calculates the mixed Lennard-Jones coefficients for pairs of atoms having distinct atom types using geometric averages: $\epsilon_{ij} = \sqrt{\epsilon_{ii}\epsilon_{jj}}$, $\sigma_{ij} = \sqrt{\sigma_{ii}\sigma_{jj}}$. In the present case, $\epsilon_{12} = \sqrt{1.0 \times 0.5} = 0.707$, and $\sigma_{12} = \sqrt{1.0 \times 3.0} = 1.732$. Other rules can be selected using the `pair_modify` command.

Single-point energy

The system is now fully parameterized. Let us complete the two remaining categories, `Monitoring` and `Run`, by adding the following lines to `initial.lmp`:

```
# 4) Monitoring
thermo 10
thermo_style custom step etotal press

# 5) Run
run 0 post no
```

The `thermo 10` command instructs LAMMPS to print thermodynamic information to the console every specified number of steps, in this case, every 10 simulation steps. The `thermo_style custom` command defines the specific outputs, which in this case are the step number (`step`), total energy E (`etotal`), and pressure p (`press`). The `run 0 post no` command instructs LAMMPS to initialize forces and energy without actually running the simulation. The `post no` option disables the post-run summary and statistics output.

The 'thermodynamic information' printed by LAMMPS using `thermo_style custom` keywords refers to instantaneous values of the specified thermodynamic properties at each output step, not cumulative averages. However, LAMMPS also allows to reference a wide variety of custom data from compute styles, fix styles, and variables. These can be used for on-the-fly analysis, including cumulative and sliding-window averages.

You can now run LAMMPS (see subsection A.3 for details on running LAMMPS). The simulation should finish quickly, and, with the default settings, LAMMPS–GUI will open two windows: one displaying the console output and another with a chart. The «Output» window will display information from the executed commands, including the total energy and pressure at step 0, as specified by the thermodynamic data request. Since no actual simulation steps were performed, the «Charts» window will be empty.

Snapshot Image

At this point, you can create a snapshot image of the current system using the «Image Viewer» window, which can be accessed by clicking the «Create Image» button in the «Run» menu. The image viewer works by instructing LAMMPS to render an image of the current system using its internal rendering library via the `dump image` command. The resulting image is then displayed, with various buttons available to adjust the view and rendering style. The image shown in Fig. 3 was created this way. This will always capture the current state of the system. Save the image for future comparisons by clicking the «Save as» button in the «File» menu.

Energy minimization

Now, replace the `run 0 post no` command line with the following `minimize` command:

```
# 5) Run
minimize 1.0e-6 1.0e-6 1000 10000
```


https://doi.org/10.33011/livecoms.6.1.3037
Living J. Comp. Mol. Sci. 2025, 6(1), 3037

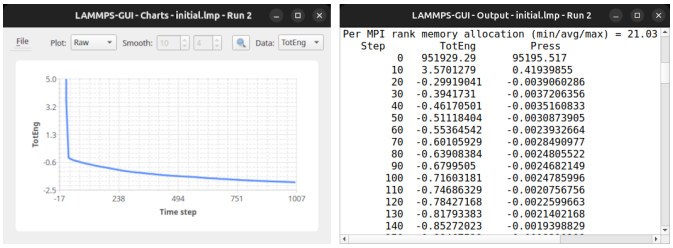

Figure 4. «Charts» (left) and «Output» (right) windows of LAMMPS–GUI after performing the minimization simulation of Tutorial 1.

This tells LAMMPS to perform an iterative energy minimization of the system. Specifically, LAMMPS will compute the forces on all atoms and then update their positions according to a selected algorithm using the computed forces, aiming to reduce the potential energy. By default, LAMMPS uses the conjugate gradient (CG) algorithm [30]. The simulation will stop as soon as one of the four minimizer criteria is met. LAMMPS will then report which stopping criterion was satisfied, along with selected system properties at both the initial and final steps. Note that, except for trivial systems, minimization algorithms will find a local minimum rather than the global minimum.

Run the minimization and observe that LAMMPS–GUI captures the output and update the chart in real time (see Fig. 4). This run executes quickly (depending on your computer's capabilities) and thus LAMMPS–GUI may fail to capture some of the thermodynamic data. In that case, use the «Preferences» dialog to reduce the data update interval and switch to single-threaded, unaccelerated execution in the «Accelerators» tab. You can repeat the run; each new attempt will start fresh, resetting the system and re-executing the script from the beginning.

The potential energy, U, decreases from a positive value to a negative value (Figs. 4 and 5 a). Note that during the energy minimization, the potential energy equals the total energy of the system, $E = U$, since the kinetic energy, K, is zero. The initially positive potential energy is expected, as the atoms are created at random positions within the simulation box, with some in very close proximity to each other. This proximity results in a large initial potential energy due to the repulsive branch of the Lennard-Jones potential [i.e., the term $1/r^{12}$ in Eq. (1)]. As the energy minimization progresses, the energy decreases - first rapidly - then more gradually, before plateauing at a negative value. This indicates that the atoms have moved to reasonable distances from one another.

Since the `thermo_style` command also includes the `press` keyword, you can switch from plotting the total energy to displaying the pressure by selecting «Press» in the «Select data» drop-down menu of the «Charts» window.

Create and save a snapshot image of the simulation state after the minimization, and compare it to the initial image. You should observe that the atoms are "clumping together" as they move toward positions of lower potential energy.

Molecular dynamics

After the energy minimization, any overlapping atoms are displaced, and the system is ready for a molecular dynamics simulation. To continue from the result of the minimization step, append the MD simulation commands to the same input script, `initial.lmp`. Add the following lines immediately after the `minimize` command:

```
# PART B − MOLECULAR DYNAMICS
# 4) Monitoring
thermo 50
thermo_style custom step temp etotal pe ke press
```

Since LAMMPS reads inputs from top to bottom, these lines will be executed *after* the energy minimization. Therefore, there is no need to re-initialize or re-define the system. The `thermo` command is called a second time to update the output frequency from 10 to 50 as soon as `PART B` of the simulation starts. In addition, a new `thermo_style` command is introduced to specify the thermodynamic information LAMMPS should print during during `PART B`. This adjustment is made because, during molecular dynamics, the system exhibits a non-zero temperature T (and consequently a non-zero kinetic energy K, keyword `ke`), which are useful to monitor. The `pe` keyword represents the potential energy of the system, U, such that $U + K = E$.

Then, add a second `Run` category by including the following lines in `PART B` of `initial.lmp`:

```
# 5) Run
fix mynve all nve
timestep 0.005
run 50000
```

The `fix nve` command updates the positions and velocities of the atoms in the group `all` at every step. More specifically, this command integrates Newton's equations of motion using the velocity-Verlet algorithm. The group `all` is a default group that contains all atoms. The last two lines specify the value of the `timestep` and the number of steps for the `run`, respectively, for a total duration of 250 time units.

https://doi.org/10.33011/livecoms.6.1.3037
Living J. Comp. Mol. Sci. 2025, 6(1), 3037

Since the *only* command affecting forces and velocities in the present script is `fix nve`, *and* periodic boundary conditions are applied in all directions, the MD simulation will be performed in the microcanonical (NVE) statistical mechanical ensemble, which maintains a constant number of particles and a fixed box volume. In this ensemble, the system does not exchange energy with anything outside the simulation box.

Run the simulation using LAMMPS. Initially, the system is not equilibrated, as the potential energy decreases while the kinetic energy increases. After approximately 40 000 steps, the values for both kinetic and potential energy plateau, indicating that the system has reached equilibrium, with the total energy fluctuating around a certain constant value.

Now, we change the second `Run` section to (note the smaller number of MD steps):

```
# 5) Run
fix mynve all nve
fix mylgv all langevin 1.0 1.0 0.1 10917
timestep 0.005
run 15000
```

The new command adds a Langevin thermostat to the atoms in the group `all`, with a target temperature of 1.0 temperature units throughout the run (the two numbers represent the target temperature at the beginning and at the end of the run, which results in a temperature ramp if they differ) [31]. A `damping` parameter of 0.1 is used. It determines how rapidly the temperature is relaxed to its desired value. In a Langevin thermostat, the atoms are subject to friction and random noise (in the form of randomly added velocities). Since a constant friction term removes more kinetic energy from fast atoms and less from slow atoms, the system will eventually reach a dynamic equilibrium where the kinetic energy removed and added are about the same. The number 10917 is a seed used to initialize the random number generator used inside of `fix langevin`; you can change it to perform statistically independent simulations. In the presence of a thermostat, the MD simulation will be performed in the canonical or NVT ensemble.

Run the simulation again using LAMMPS–GUI. From the information printed in the «Output» window, one can see that the temperature starts from 0 but rapidly reaches the requested value and stabilizes itself near T = 1 temperature units. One can also observe that the potential energy, U, rapidly decreases during energy minimization (see also Fig. 5 a). After the molecular dynamics simulation starts, U increases until it reaches a plateau value of about -0.25. The kinetic energy, K, is equal to zero during energy minimization and then increases rapidly during molecular dynamics until it reaches a plateau value of about 1.5 (Fig. 5 d).

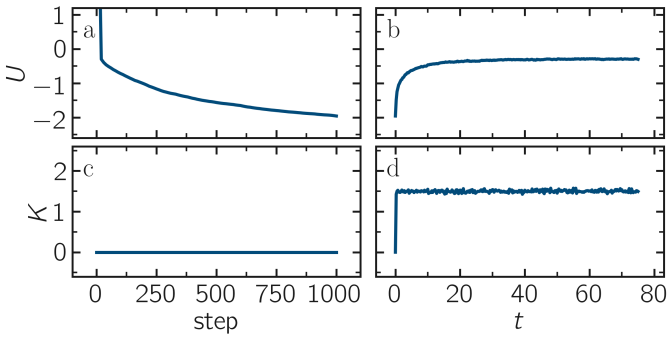

Figure 5. (a) Potential energy, U, of the binary mixture as a function of the step during energy minimization in Tutorial 1. (b) Potential energy, U, as a function of time during molecular dynamics in the NVT ensemble. (c) Kinetic energy, K, during energy minimization. (d) Kinetic energy, K, during molecular dynamics.

All simulations presented in these tutorials are deliberately kept short so they can be executed on a personal computer. These runs are not intended to produce statistically meaningful results, and should not be considered suitable for publication (see for instance Ref. 32).

Trajectory visualization

So far, the simulation has been mostly monitored through the analysis of thermodynamic information. To better follow the evolution of the system and visualize the trajectories of the atoms, let us use the `dump image` command to create snapshot images during the simulation. We have already explored the «Image Viewer» window. Open it again and adjust the visualization to your liking using the available buttons. Now you can copy the commands used to create this visualization to the clipboard by either using the «Ctrl-D» keyboard shortcut or selecting «Copy dump image command» from the «File» menu. This text can be pasted into the into the `Monitoring` section of `PART B` of the `initial.lmp` file. This may look like the following:

```
dump viz all image 100 myimage-*.ppm type type &
  size 800 800 zoom 1.452 shiny 0.7 fsaa yes &
  view 80 10 box yes 0.025 axes no 0.0 0.0 &
  center s 0.483725 0.510373 0.510373
dump_modify viz pad 9 boxcolor royalblue &
  backcolor white adiam 1 1.6 adiam 2 4.8
```

The '&' characters at the end are used to extend the commands across multiple lines. These two commands tell LAMMPS to generate NetPBM format images every 100 steps. The two `type` keywords are for `color` and `diameter`, respectively. Run the `initial.lmp` using LAMMPS again, and a new window named «Slide Show» will pop up. It

https://doi.org/10.33011/livecoms.6.1.3037
Living J. Comp. Mol. Sci. 2025, 6(1), 3037

will show each image created by the `dump image` as it is created. After the simulation is finished (or stopped), the slideshow viewer allows you to animate the trajectory by cycling through the images. The window also allows you to export the animation to a movie (provided the FFMpeg program is installed) and to bulk delete those image files.

The rendering of the system can be further adjusted using the many options of the `dump image` command. For instance, the value for the `shiny` keyword is used to adjust the shininess of the atoms, the `box` keyword adds or removes a representation of the box, and the `view` and `zoom` keywords adjust the camera (and so on).

3.1.2 Improving the script

Let us improve the input script and perform more advanced operations, such as specifying initial positions for the atoms and restarting the simulation from a previously saved configuration.

Control the initial atom positions

Open the `improved.min.lmp`, which was downloaded during the tutorial setup. This file contains the `Part A` of the `initial.lmp` file, but *without* any commands in the `System definition` section:

```
# 1) Initialization
units lj
dimension 3
atom_style atomic
boundary p p p
# 2) System definition
# 3) Settings
mass 1 1.0
mass 2 10.0
pair_style lj/cut 4.0
pair_coeff 1 1 1.0 1.0
pair_coeff 2 2 0.5 3.0
# 4) Monitoring
thermo 10
thermo_style custom step etotal press
# 5) Run
minimize 1.0e−6 1.0e−6 1000 10000
```

We want to create the atoms of types 1 and 2 in two separate regions. To achieve this, we need to add two `region` commands and then reintroduce the `create_atoms` commands, this time using the new regions instead of the simulation box region to place the atoms:

```
# 2) System definition
region simbox block −20 20 −20 20 −20 20
create_box 2 simbox
# for creating atoms
region cyl_in cylinder z 0 0 10 INF INF side in
region cyl_out cylinder z 0 0 10 INF INF side out
create_atoms 1 random 1000 34134 cyl_out
```

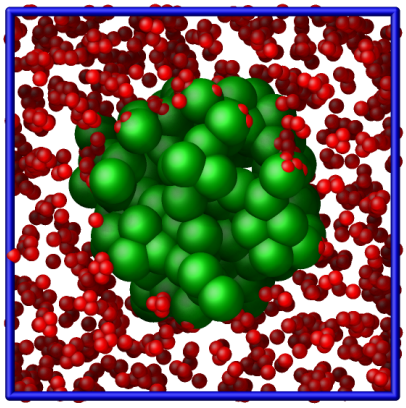

Figure 6. Visualization of the improved binary mixture input after minimization during Tutorial 1. Colors are the same as in Fig. 3.

```
create_atoms 2 random 150 12756 cyl_in
```

The `side in` and `side out` keywords are used to define regions representing the inside and outside of the cylinder of radius 10 length units, respectively. Then, append a sixth section titled `Save system` at the end of the file, ensuring that the `write_data` command is placed *after* the `minimize` command:

```
# 6) Save system
write_data improved.min.data
```

A key improvement to the input is the addition of the `write_data` command. This command writes the state of the system to a text file called `improved.min.data`. This `.data` file will be used later to restart the simulation from the final state of the energy minimization step, eliminating the need to repeat the system creation and minimization.

Run the `improved.min.lmp` file using LAMMPS–GUI. At the end of the simulation, a file called `improved.min.data` is created. You can view the contents of this file from LAMMPS–GUI, by right-clicking on the file name in the editor and selecting the entry «`View file improved.min.data`».

The created `.data` file contains all the information necessary to restart the simulation, such as the number of atoms, the box size, the masses, and the pair coefficients. This `.data` file also contains the final positions of the atoms, along with their IDs and types, within the `Atoms` section. The first five columns of the `Atoms` section correspond (from left to right) to the atom indexes (from 1 to the total number of atoms, 1150), the atom types (1 or 2 here), and the positions of the atoms x, y, z. The last three columns are image flags that keep track of which atoms crossed the periodic boundary. The exact format of each line in the

https://doi.org/10.33011/livecoms.6.1.3037
Living J. Comp. Mol. Sci. 2025, 6(1), 3037

`Atoms` section depends on the choice of `atom_style`, which determines which per-atom data is set and stored internally in LAMMPS.

Instead of the `write_data` command, you can also use the `write_restart` command to save the state of the simulation to a binary restart file. Binary restart files are more compact, faster to write, and contain more information, making them often more convenient to use. For example, the choice of `atom_style` or `pair_style` is recorded, so those commands do not need to be issued before reading the restart. Note however that restart files are not expected to be portable across LAMMPS versions or platforms. Therefore, in these tutorials, and with the exception of Tutorial 3, we primarily use `write_data` to provide you with a reference copy of the data file that works regardless of your LAMMPS version and platform.

Restarting from a saved configuration

To continue a simulation from the saved configuration, open the `improved.md.lmp` file, which was downloaded during the tutorial setup. This file contains the *Initialization* part from `initial.lmp` and `improved.min.lmp`:

```
# 1) Initialization
units lj
dimension 3
atom_style atomic
boundary p p p
# 2) System definition
# 3) Settings
# 4) Monitoring
# 5) Run
```

Since we read most of the information from the data file, we don't need to repeat all the commands from the `System definition` and `Settings` categories. The exception is the `pair_style` command, which now must come *before* the simulation box is defined, meaning before the `read_data` command. Add the following lines to `improved.md.lmp`:

```
# 2) System definition
pair_style lj/cut 4.0
read_data improved.min.data
```

By visualizing the system (see Fig. 6), you may have noticed that some atoms left their original region during minimization. To start the simulation from a clean slate, with only atoms of type 2 inside the cylinder and atoms of type 1 outside the cylinder, let us delete the misplaced atoms by adding the following commands to the `System definition` section of the `improved.md.lmp`:

```
region cyl_in cylinder z 0 0 10 INF INF side in
region cyl_out cylinder z 0 0 10 INF INF side out
group grp_t1 type 1
group grp_t2 type 2
group grp_in region cyl_in
group grp_out region cyl_out
group grp_t1_in intersect grp_t1 grp_in
group grp_t2_out intersect grp_t2 grp_out
delete_atoms group grp_t1_in
delete_atoms group grp_t2_out
```

The first two `region` commands recreate the previously defined regions, which is necessary since regions are not saved by the `write_data` command. The first two `group` commands create groups containing all the atoms of type 1 and all the atoms of type 2, respectively. The next two `group` commands create atom groups based on their positions at the beginning of the simulation, i.e., when the commands are being read by LAMMPS. The last two `group` commands create atom groups based on the intersection between the previously defined groups. Finally, the two `delete_atoms` commands delete the atoms of type 1 located inside the cylinder and the atoms of type 2 located outside the cylinder, respectively.

Since LAMMPS has a limited number of custom groups (30), it is good practice to delete groups that are no longer needed. This can be done by adding the following four commands to `improved.md.lmp`:

```
# delete no longer needed groups
group grp_in delete
group grp_out delete
group grp_t1_in delete
group grp_t2_out delete
```

Let us monitor the number of atoms of each type inside the cylinder as a function of time by creating the following equal-style variables:

```
variable n1_in equal count(grp_t1,cyl_in)
variable n2_in equal count(grp_t2,cyl_in)
```

The equal-style `variables` are expressions evaluated during the run and return a number. Here, they are defined to count the number of atoms of a specific group within the `cyl_in` region.

The `n1_in` and `n2_in` defined above are equal-style variables, which evaluate a numerical expression using the `count()` function. Other LAMMPS variable styles include atom, index, file, loop, string, and vector.

In addition to counting the atoms in each region, we will also extract the coordination number of type 2 atoms around type 1 atoms. The coordination number measures the number of type 2 atoms near type 1 atoms, defined by a cutoff distance. Taking the average provides as a good indicator of

https://doi.org/10.33011/livecoms.6.1.3037
Living J. Comp. Mol. Sci. 2025, 6(1), 3037

the degree of mixing in a binary mixture. This is done using two `compute` commands: the first counts the coordinated atoms, and the second calculates the average over all type 1 atoms. Add the following lines to `improved.md.lmp`:

```
compute coor12 grp_t1 coord/atom cutoff 2 group grp_t2
compute sumcoor12 grp_t1 reduce ave c_coor12
```

The `compute reduce ave` command is used to average the per-atom coordination number calculated by the `compute coord/atom` command. Compute commands do not print or output anything by themselves, nor are they automatically executed; they require a "consumer" command that references the compute. In this case, the first compute is referenced by the second, and we reference the second in a `thermo_style custom` command (see below).

LAMMPS `compute` commands can produce a wide variety of data and one can identify the category from the name of the compute style: global data (no suffix), local data (/local suffix), per-atom data (/atom suffix), per-chunk data (/chunk suffix), per-grid data (/grid suffix). In the example above, the `compute coord/atom` produces per-atom data, which is used as input for `compute reduce` which returns global data. For global data three kinds of data exists: scalars (single values), vectors (one-dimensional arrays), or arrays (two-dimensional tables). When referencing results of a compute, you can use indices: for example, `c_mycompute` refers to the entire scalar, vector, or array, and `c_mycompute[1]` refers to its first element or column (in case of vector or array). In some cases also wildcards like "*" can be used to, for instance, refer to all elements of a vector instead of having specify all elements individually. In general, "consumer" commands (`fix` styles or `dump` styles, `variables`, or other `compute` styles) can only work with certain data types or need to have keywords set to select which data to use. You need to check the documentation of each command to ensure compatibility.

There is no need for a `Settings` section, as the settings are taken from the `.data` file. Finally, let us complete the script by adding the following lines to `improved.md.lmp`:

```
# 4) Monitoring
thermo 1000
thermo_style custom step temp pe ke etotal &
  press v_n1_in v_n2_in c_sumcoor12
dump viz all image 1000 myimage−*.ppm type type &
  shiny 0.1 box no 0.01 view 0 0 zoom 1.8 fsaa yes size 800 800
dump_modify viz adiam 1 1 adiam 2 3 acolor 1 &
  turquoise acolor 2 royalblue backcolor white
```

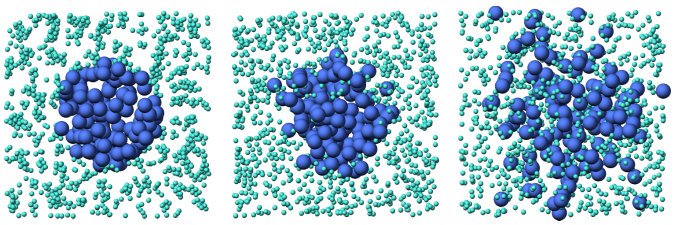

Figure 7. Evolution of the system from Tutorial 1 during mixing. The three snapshots show respectively the system at $t = 0$ (left panel), $t = 75$ (middle panel), and $t = 1500$ (right panel). The atoms of type 1 are represented as small turquoise spheres and the atoms of type 2 as large blue spheres.

The two variables `n1_in`, `n2_in`, along with the compute `sumcoor12`, were added to the list of information printed during the simulation. Additionally, images of the system will be created with slightly less saturated colors than the default ones.

Finally, add the following lines to `improved.md.lmp`:

```
# 5) Run
velocity all create 1.0 49284 mom yes dist gaussian
fix mynve all nve
fix mylgv all langevin 1.0 1.0 0.1 10917 zero yes
timestep 0.005
run 300000
```

Here, there are a few more differences from the previous simulation. First, the `velocity create` command assigns an initial velocity to each atom. The initial velocity is chosen so that the average initial temperature is equal to 1.0 temperature units. The additional keywords ensure that no linear momentum (`mom yes`) is given to the system and that the generated velocities are distributed according to a Gaussian distribution. Another improvement is the `zero yes` keyword in the Langevin thermostat, which ensures that the total random force applied to the atoms is equal to zero. These steps are important to prevent the system from starting to drift or move as a whole.

A bulk system with periodic boundary conditions is expected to remain in place. Accordingly, when computing the temperature from the kinetic energy, we use $3N - 3$ degrees of freedom since there is no global translation. In a drifting system, some of the kinetic energy is due to the drift, which means the system itself cools down. In extreme cases, the system can freeze while its center of mass drifts very quickly. This phenomenon is sometimes referred to as the "flying ice cube syndrome" [7].

Run `improved.md.lmp` and observe the mixing of the two populations over time (see also Fig. 7). From the variables `n1_in` and `n2_in`, you can track the number of

https://doi.org/10.33011/livecoms.6.1.3037
Living J. Comp. Mol. Sci. 2025, 6(1), 3037

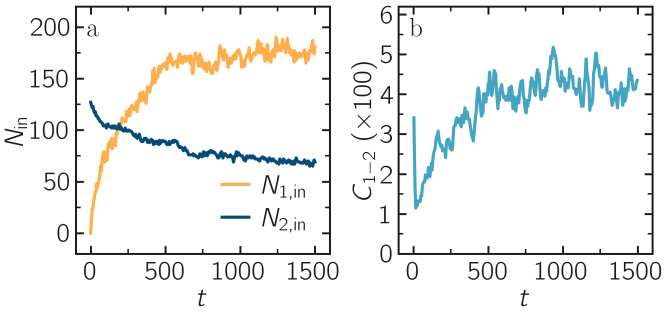

Figure 8. a) Evolution of the numbers $N_{1,\text{in}}$ and $N_{2,\text{in}}$ of atoms of types 1 and 2, respectively, within the `cyl_in` region as functions of time t. b) Evolution of the coordination number C_{1-2} (compute `sumcoor12`) between atoms of types 1 and 2.

atoms in each region as a function of time (Fig. 8 a). To view their evolution, select the entries «v_n1_in» or «v_n2_in» in the «Data» drop-down menu in the «Charts» window of LAMMPS–GUI.

In addition, as the mixing progresses, the average coordination number between atoms of types 1 and 2 increases from about 0.01 to 0.04 (Fig. 8 b). This indicates that, over time, more and more particles of type 1 come into contact with particles of type 2, as expected during mixing. This can be observed using the entry «c_sumcoor12» in the «Charts» drop-down menu.

Experiments

Here are some suggestions for further experiments with this system that may lead to additional insights into how different systems are configured and how various features function:

- Use a Nosé-Hoover thermostat (`fix nvt`) instead of a Langevin thermostat (`fix nve` + `fix langevin`).
- Omit the energy minimization step before starting the MD simulation using either the Nosé-Hoover or the Langevin thermostat.
- Apply a thermostat to only one type of atoms each and observe the temperature for each type separately.
- Append an NVE run (i.e. without any thermostat) and observe the energy levels.

In contrast to the `fix nve` command, which integrates Newton's equations of motion without any thermostatting, the `fix nvt` command adds a Nosé-Hoover thermostat to control the system temperature.

Another useful experiment is coloring the atoms in the «Slide Show» according to an observable, such as their respective coordination numbers. To do this, replace the `dump` and `dump_modify` commands with the following lines:

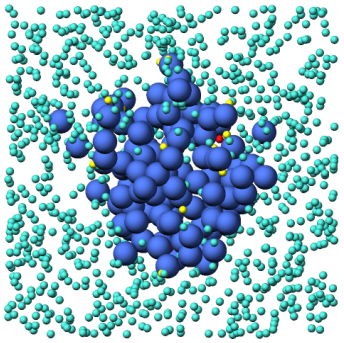

Figure 9. Snapshot of the binary mixture simulated during Tutorial 1 with atoms of type 1 colored according to their computed 1 – 2 coordination number from the compute `coor12` , ranging from turquoise, `c_coor12 = 0` , to yellow, `c_coor12 = 1` , and red, `c_coor12 = 2` .

```
variable coor12 atom (type==1)*(c_coor12)+(type==2)*−1
dump viz all image 1000 myimage−*.ppm v_coor12 type &
  shiny 0.1 box no 0.01 view 0 0 zoom 1.8 fsaa yes size 800 800
dump_modify viz adiam 1 1 adiam 2 3 backcolor white &
  amap −1 2 ca 0.0 4 min royalblue 0 turquoise 1 yellow max red
```

Run LAMMPS again. Atoms of type 1 are now colored based on the value of `c_coor12` , which is mapped continuously from turquoise to yellow and red for atoms with the highest coordination (Fig. 9). In the definition of the variable `v_coor12` , atoms of type 2 are all assigned a value of -1, and will therefore always be colored their default blue.

3.2 Tutorial 2: Pulling on a carbon nanotube

In this tutorial, the system of interest is a small, single-walled carbon nanotube (CNT) in an empty box (Fig. 10). The CNT is strained by imposing a constant velocity on the edge atoms. To illustrate the difference between conventional and reactive force fields, this tutorial is divided into two parts: in the first part, a conventional molecular force field (called OPLS-AA [10]) is used and the functional form of the bonded potential ensures that the bonds between the atoms of the CNT are unbreakable. In the second part, a reactive, many-body force field (called AIREBO [11]) is used, which allows chemical bonds to break under large strain.

To set up this tutorial, select «Start Tutorial 2» from the «Tutorials» menu of LAMMPS–GUI and follow the instructions. This will select a folder, create one if necessary, and place several files into it. The initial input file, set up for a single-point energy calculation, will also be loaded into the editor under the name `unbreakable.lmp`. Additional files are a data file containing the CNT topology and geometry, named `unbreakable.data`, a parameters file named `unbreakable.inc`, as well as the scripts required for the second part of the tutorial.

https://doi.org/10.33011/livecoms.6.1.3037
Living J. Comp. Mol. Sci. 2025, 6(1), 3037

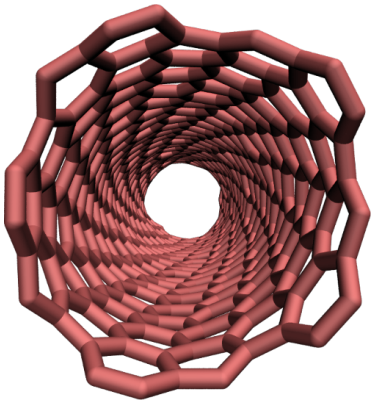

Figure 10. The carbon nanotube (CNT) simulated during Tutorial 2.

3.2.1 Unbreakable bonds

With most conventional molecular force fields, the chemical bonds between atoms are defined at the start of the simulation and remain fixed, regardless of the forces applied to the atoms. In this tutorial, these bonds are explicitly specified in the `.data` file, which is read using the `read_data` command (see below). Bonds are typically modeled as springs following Hooke's law with equilibrium distances r_0, force constants k_b, and bond potential energy $U_b = k_b (r - r_0)^2$. Additionally, angular and dihedral interactions are often imposed to preserve the molecular structure by maintaining the relative orientations of neighboring atoms.

The LAMMPS input

After completing the setup, the editor should display the following content:

```
units real
atom_style molecular
boundary f f f

pair_style lj/cut 14.0
bond_style harmonic
angle_style harmonic
dihedral_style opls
improper_style harmonic
special_bonds lj 0.0 0.0 0.5

read_data unbreakable.data
include unbreakable.inc

run 0 post no
```

The chosen unit system is `real` (therefore distances are in Ångströms (Å), times in femtoseconds (fs), and energies in (kcal/mol)), the `atom_style` is `molecular` (therefore atoms are point particles that can form bonds with each other), and the boundary conditions are fixed. The boundary conditions do not matter here, as the box boundaries were placed far from the CNT. Just like in the previous tutorial, Lennard-Jones fluid, the pair style is `lj/cut` (i.e. a Lennard-Jones potential with cutoff) and its cutoff is set to 14 Å, which means that only the atoms closer than this distance interact through the Lennard-Jones potential.

The `bond_style`, `angle_style`, `dihedral_style`, and `improper_style` commands specify the different potentials used to constrain the relative positions of the atoms. The `special_bonds` command sets the weighting factors for the Lennard-Jones interactions between atoms sitting one, two, or three bonds away from each other, respectively. This is done for convenience when parameterizing the force constants for bonds, angles, and so on. By excluding the non-bonded (Lennard-Jones) interactions for these pairs, those interactions do not need to be considered when determining the force constants.

The `read_data` command imports the `unbreakable.data` file that should have been downloaded next to `unbreakable.lmp` during the tutorial setup. This file contains information about the box size, atom positions, as well as the identity of the atoms that are linked by `bonds`, `angles`, `dihedrals`, and `impropers` interactions. It was created using VMD and TopoTools [33].

> Bonds, angles, dihedrals, and impropers in LAMMPS are assigned types and IDs, just like atoms. The ID uniquely identifies each interaction instance, while the type determines which parameters (from the `bond_coeff`, `angle_coeff`, etc. commands) are applied. In this tutorial, these types and IDs are specified in the `.data` file and read by the `read_data` command.

> The format details of the different sections in a data file change with different settings. In particular, the `Atoms` section may have a different number of columns, or the columns may represent different properties when the `atom_style` is changed. To help users, LAMMPS and tools like VMD and TopoTools will add a comment (here `# molecular`) to the `Atoms` header line in the data files that indicates the intended `atom_style`. LAMMPS will print a warning when the chosen atom style does not match what is written in that comment.

The `.data` file does not contain any sections with potential parameters; thus, we need to specify the parameters of both the bonded and non-bonded potentials. The parameters we use are taken from the OPLS-AA (Optimized Potentials for Liquid Simulations-All-Atom) force field [10], and are given in a separate `unbreakable.inc` file (also downloaded during the tutorial setup). This file - that must be

https://doi.org/10.33011/livecoms.6.1.3037
Living J. Comp. Mol. Sci. 2025, 6(1), 3037

placed next to `unbreakable.lmp` - contains the following lines:

```
pair_coeff 1 1 0.066 3.4
bond_coeff 1 469 1.4
angle_coeff 1 63 120
dihedral_coeff 1 0 7.25 0 0
improper_coeff 1 5 180
```

The `pair_coeff` command sets the parameters for non-bonded Lennard-Jones interactions between atoms type 1 to ϵ_{11} = 0.066 kcal/mol and σ_{11} = 3.4 Å. The `bond_coeff` provides the equilibrium distance r_0 = 1.4 Å and the spring constant k_b = 469 kcal/mol/Å^2 for the harmonic potential imposed between two bonded carbon atoms. The potential is given by $U_b = k_b(r - r_0)^2$. The `angle_coeff` gives the equilibrium angle θ_0 and constant for the potential between atoms forming an angle: $U_\theta = k_\theta(\theta - \theta_0)^2$. The `dihedral_coeff` and `improper_coeff` define the potentials for the constraints between 4 atoms.

Rather than copying the contents of the file into the input, we incorporate it using the `include` command. Using `include` allows us to conveniently reuse the parameter settings in other inputs or switch them with others. This will become more general when using type labels [12], which is shown in the next tutorial.

Prepare the initial state

In this tutorial, a deformation will be applied to the CNT by displacing the atoms located at its edges. To achieve this, we will first isolate the atoms at the two edges and place them into groups named `rtop` and `rbot`. Add the following lines to `unbreakable.lmp`, just before the `run 0` command:

```
group carbon_atoms type 1
variable xmax equal bound(carbon_atoms,xmax)-0.5
variable xmin equal bound(carbon_atoms,xmin)+0.5
region rtop block ${xmax} INF INF INF INF INF
region rbot block INF ${xmin} INF INF INF INF
region rmid block ${xmin} ${xmax} INF INF INF INF
```

The first command includes all the atoms of type 1 (i.e. all the atoms here) in a group named `carbon_atoms`. The variable x_{max} corresponds to the coordinate of the last atoms along x minus 0.5 Å, and x_{min} to the coordinate of the first atoms along x plus 0.5 Å. Then, three regions are defined, corresponding to the following: $x < x_{min}$, (`rbot`, for region bottom), $x_{min} > x > x_{max}$ (`rmid`, for region middle), and $x > x_{max}$ (`rtop`, for region top).

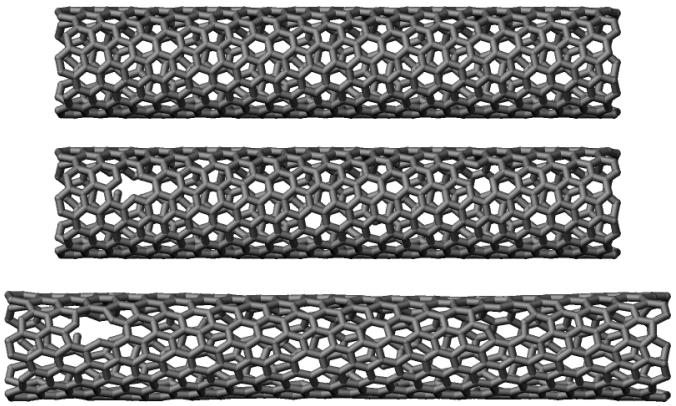

Figure 11. The unbreakable CNT simulated during Tutorial 2 before the removal of atoms (top), after the removal of 10 atoms from the `rmid` region (middle), and after deformation (bottom).

So far, variables have been referenced dynamically during the run using the `v_` prefix, which evaluates the variable as it evolves over time. Here, a dollar sign ($) is used to expand the variable immediately at the time the input script is read.

Finally, let us define 3 groups of atoms corresponding to the atoms in each of the 3 regions by adding to `unbreakable.lmp` just before the `run 0` command:

```
group cnt_top region rtop
group cnt_bot region rbot
group cnt_mid region rmid
set group cnt_top mol 1
set group cnt_bot mol 2
set group cnt_mid mol 3
```

With the three `set` commands, we assign unique, otherwise unused molecule IDs to atoms in those three groups. A molecule ID is an integer that groups atoms into a 'molecule' for bookkeeping purposes, and can be useful for tracking and post-processing. We will use these IDs later to assign different colors to these groups of atoms.

Run the simulation using LAMMPS. The number of atoms in each group is given in the «Output» window. It is an important check to make sure that the number of atoms in each group corresponds to what is expected, as shown here:

```
700 atoms in group carbon_atoms
10 atoms in group cnt_top
10 atoms in group cnt_bot
680 atoms in group cnt_mid
```

Finally, to start from a less ideal state and create a system with some defects, let us randomly delete a small fraction of the carbon atoms. To avoid deleting atoms that are too close to the edges, let us define a new region named `rdel` that starts at 2 Å from the CNT edges:

https://doi.org/10.33011/livecoms.6.1.3037
Living J. Comp. Mol. Sci. 2025, 6(1), 3037

```
variable xmax_del equal ${xmax}-2
variable xmin_del equal ${xmin}+2
region rdel block ${xmin_del} ${xmax_del} INF INF INF INF
group rdel region rdel
delete_atoms random fraction 0.02 no rdel NULL 2793 bond yes
```

The `delete_atoms` command randomly deletes 2 % of the atoms from the `rdel` group, here about 10 atoms (compare the top and the middle panels in Fig. 11).

The molecular dynamics

Let us give an initial temperature to the atoms of the group `cnt_mid` by adding the following commands to `unbreakable.lmp`:

```
reset_atoms id sort yes
velocity cnt_mid create 300 48455 mom yes rot yes
```

Re-setting the atom IDs is necessary before using the `velocity` command when atoms were deleted, which is done here with the `reset_atoms` command. The `velocity` command assigns random initial velocities to the atoms of the middle group `cnt_mid` from a uniform distribution, ensuring an initial temperature of T = 300 K for these atoms.

Let us specify the thermalization and the dynamics of the system. Add the following lines into `unbreakable.lmp`:

```
fix mynve1 cnt_top nve
fix mynve2 cnt_bot nve
fix mynvt cnt_mid nvt temp 300 300 100
```

The `fix nve` commands are applied to the atoms of `cnt_top` and `cnt_bot`, respectively, and will ensure that the positions of the atoms from these groups are recalculated at every step. The `fix nvt` does the same for the `cnt_mid` group, while also applying a Nosé-Hoover thermostat with desired temperature of 300 K [34, 35].

> The Nosé-Hoover thermostat only controls the temperature of the atoms belonging to the specified `cnt_mid` group. Atoms outside this group are not affected.

To immobilize the atoms at the edges, let us add the following commands to `unbreakable.lmp`:

```
fix mysf1 cnt_top setforce 0 0 0
fix mysf2 cnt_bot setforce 0 0 0
velocity cnt_top set 0 0 0
velocity cnt_bot set 0 0 0
```

The two `setforce` commands cancel the forces applied on the atoms of the two edges, respectively. The cancellation of the forces is done at every step, and along all 3 directions of space, x, y, and z, due to the use of `0 0 0`. Although the force on these atoms is set to zero, the `fix` stores the force vector

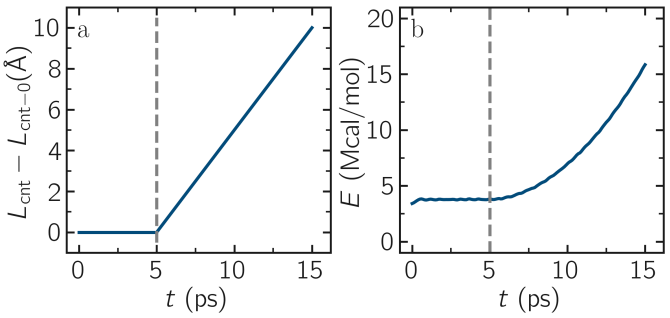

Figure 12. a) Evolution of the length L_{cnt} of the CNT with time, as simulated during Tutorial 2. The CNT starts deforming at t = 5 ps, and L_{cnt-0} is the CNT initial length. b) Evolution of the total energy E of the system with time t. Here, the potential is OPLS-AA, and the CNT is unbreakable.

acting on the group *before* cancellation, which can later be extracted for analysis (see below). The two `velocity` commands set the initial velocities along x, y, and z to 0 for the atoms of `cnt_top` and `cnt_bot`, respectively. As a consequence of these last four commands, the atoms of the edges will remain immobile during the simulation (or at least they would if no other command was applied to them).

> The `velocity set` command adjusts the velocities of a group of atoms immediately but has no effect *during* the simulation. When `velocity set` is used in combination with `setforce 0 0 0`, as is the case here, the initial velocity will persist during the entire simulation, thus producing a constant velocity motion or no motion at all.

Outputs

Next, to measure the strain and stress applied to the CNT, let us create a variable for the distance L_{cnt} between the two edges, as well as a variable F_{cnt} for the force applied on the edges:

```
variable Lcnt equal xcm(cnt_top,x)-xcm(cnt_bot,x)
variable Fcnt equal f_mysf1[1]-f_mysf2[1]
```

Here, the force is extracted from the fixes `mysf1` and `mysf2` using `f_`, similarly to the use of `v_` to call a variable, and `c_` to call a compute, as seen in Tutorial 1.

Let us also add a `dump image` command to visualize the system every 500 steps:

```
dump viz all image 500 myimage-*.ppm element type size &
  1000 400 zoom 6 shiny 0.3 fsaa yes bond atom 0.8 &
  view 0 90 box no 0.0 axes no 0.0 0.0
dump_modify viz pad 9 backcolor white adiam 1 0.85 bdiam 1 1.0
```

Let us run a small equilibration step to bring the system to the required temperature before applying any deformation. Replace the `run 0 post no` command in `unbreakable.lmp` with the following lines:

https://doi.org/10.33011/livecoms.6.1.3037
Living J. Comp. Mol. Sci. 2025, 6(1), 3037

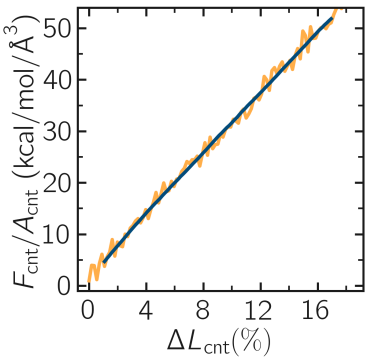

Figure 13. Stress applied on the CNT during deformation, F_{cnt}/A_{cnt}, where F_{cnt} is the force and A_{cnt} the CNT surface area, as a function of the strain, $\Delta L_{cnt} = (L_{cnt} - L_{cnt\text{-}0}/L_{cnt\text{-}0})$, where L_{cnt} is the CNT length and $L_{cnt\text{-}0}$ the CNT initial length, as simulated during Tutorial 2. Here, the potential is OPLS-AA, and the CNT is unbreakable. The orange line shows the raw data, and the blue line represents a time-averaged curve.

```
compute Tmid cnt_mid temp
thermo 100
thermo_style custom step temp etotal v_Lcnt v_Fcnt
thermo_modify temp Tmid line yaml

timestep 1.0
run 5000
```

With the `thermo_modify` command, we specify to LAMMPS that the temperature T_{mid} of the middle group, `cnt_mid`, must be outputted, instead of the temperature of the entire system. This choice is motivated by the presence of frozen parts with an effective temperature of 0 K, which makes the average temperature of the entire system less relevant. The `thermo_modify` command also imposes the use of the YAML format that can easily be read by Python (see below).

Let us impose a constant velocity deformation on the CNT by combining the `velocity set` command with previously defined `fix setforce`. Add the following lines in the `unbreakable.lmp` file, right after the last `run 5000` command:

```
velocity cnt_top set 0.0005 0 0
velocity cnt_bot set −0.0005 0 0

run 10000
```

The chosen velocity for the deformation is 100 m/s, or 0.001 Å/fs.

Run the simulation using LAMMPS. As can be seen from the variable L_{cnt}, the length of the CNT increases linearly over time for $t > 5$ ps (Fig. 12 a), as expected from the imposed constant velocity. What you observe in the «Slide Show» windows should resembles Fig. 11. The total energy of the system shows a non-linear increase with t once the deformation starts, which is expected from the typical dependency of bond energy with bond distance, $U_b = k_b (r - r_0)^2$ (Fig. 12 b).

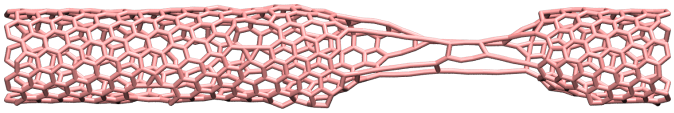

Figure 14. CNT with broken bonds. This image was generated using VMD [24, 25] with the «DynamicBonds» representation.

Importing YAML log file into Python

Let us import the simulation data into Python, and generate a stress-strain curve. Here, the stress is defined as F_{cnt}/A_{cnt}, where $A_{cnt} = \pi r_{cnt}^2$ is the surface area of the CNT, and $r_{cnt} = 5.2$ Å the CNT radius. The strain is defined as $(L_{cnt} - L_{cnt\text{-}0})/L_{cnt\text{-}0}$, where $L_{cnt\text{-}0}$ is the initial CNT length.

Right-click inside the «Output» window, and select «Export YAML data to file». Call the output `unbreakable.yaml`, and save it within the same folder as the input files, where a Python script named `unbreakable-yaml-reader.py` should also be located. When executed using Python, this .py file first imports the `unbreakable.yaml` file. Then, a certain pattern is identified and stored as a string character named 'docs'. The string is then converted into a list, and F_{cnt} and L_{cnt} are extracted. The stress and strain are then calculated, and the result is saved in a data file named `unbreakable.dat` using the NumPy 'savetxt' function. 'thermo[0]' can be used to access the information from the first minimization run, and 'thermo[1]' to access the information from the second MD run. The data extracted from the `unbreakable.yaml` file can then be used to plot the stress-strain curve, see Fig. 13.

3.2.2 Breakable bonds

When using a conventional molecular force field, as we have just done, the bonds between the atoms are non-breakable. Let us perform a similar simulation and deform a small CNT again, but this time with a reactive force field that allows bonds to break if the applied deformation is large enough.

Input file initialization

Open the input named `breakable.lmp` that should have been downloaded next to `unbreakable.lmp` during the tutorial setup. There are only a few differences with the previous input. First, the AIREBO force field requires the `metal` units setting instead of `real` for OPLS-AA. A second difference is the use of `atom_style atomic` instead of `molecular`, since no explicit bond information is required with AIREBO. The following commands are setting up the AIREBO force field:

```
pair_style airebo 3.0
pair_coeff * * CH.airebo C
```


https://doi.org/10.33011/livecoms.6.1.3037
Living J. Comp. Mol. Sci. 2025, 6(1), 3037

The AIREBO force field is a many-body potential, where interactions are not only between pairs of atoms, but also triples and quadruples representing angle and dihedral interactions. This means that there are different rules for the `pair_coeff` command: there must be only one command that covers all permutations of atom types by using two '*' wildcards. After the potential file follows a list of elements. These element names are used to look up the parameter sets in the potential file. There must be a list with as many elements as atom types following the filename. In our system, there is only one atom type (1), which is mapped to the element 'C' in the `pair_coeff` command. Which elements are supported is determined by the contents of the potential file.

Here, `CH.airebo` is the file containing the parameters for AIREBO, and must be placed next to `breakable.lmp`.

With `metal` units, time values are in units of picoseconds (10^{-12} s) instead of femtoseconds (10^{-15} s) in the case of `real` units. It is important to keep this in mind when setting parameters that are expressed units containing time, such as the timestep or the time constant of a thermostat, or velocities.

Since bonds, angles, and dihedrals do not need to be explicitly set when using AIREBO, some simplification must be made to the `.data` file. The new `.data` file is named `breakable.data`, and must be placed within the same folder as the input file. Just like `unbreakable.data`, the `breakable.data` contains the information required for placing the atoms in the box, but no bond/angle/dihedral information. Another difference between the `unbreakable.data` and `breakable.data` files is that, here, a larger distance of 120 Å was used for the box size along the *x*-axis, to allow for larger deformation of the CNT.

Start the simulation

Here, let us perform a similar deformation as the previous one. In `breakable.lmp`, replace the `run 0 post no` line with:

```
fix mysf1 cnt_bot setforce 0 0 0
fix mysf2 cnt_top setforce 0 0 0
velocity cnt_bot set 0 0 0
velocity cnt_top set 0 0 0

variable Lcnt equal xcm(cnt_top,x)-xcm(cnt_bot,x)
variable Fcnt equal f_mysf1[1]-f_mysf2[1]

dump viz all image 500 myimage.*.ppm type type size 1000 400 &
  zoom 4 shiny 0.3 adiam 1.5 box no 0.01 view 0 90 &
  shiny 0.1 fsaa yes
dump_modify viz pad 5 backcolor white acolor 1 gray
```

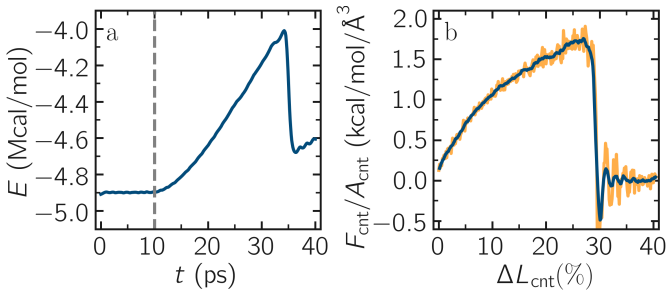

Figure 15. a) Evolution of the total energy E of the CNT with time t. b) Stress applied on the CNT during deformation, F_{cnt}/A_{cnt}, where F_{cnt} is the force and A_{cnt} the CNT surface area, as a function of the strain, $\Delta L_{cnt} = (L_{cnt} - L_{cnt\text{-}0}/L_{cnt\text{-}0})$, where L_{cnt} is the CNT length and $L_{cnt\text{-}0}$ the CNT initial length, as simulated during Tutorial 2. Here, the potential is AIREBO, and the CNT is breakable. The orange line shows the raw data, and the blue line represents a time-averaged curve.

```
compute Tmid cnt_mid temp
thermo 100
thermo_style custom step temp etotal v_Lcnt v_Fcnt
thermo_modify temp Tmid line yaml

timestep 0.0005
run 10000
```

Note the relatively small timestep of 0.0005 ps (= 0.5 fs) used. Reactive force fields like AIREBO usually require a smaller timestep than conventional ones. When running `breakable.lmp` with LAMMPS, you can see that the temperature deviates from the target temperature of 300 K at the start of the equilibration, but that after a few steps, it reaches the target value.

Bonds cannot be displayed by the `dump image` when using the `atom_style atomic`, as it contains no bonds. A tip for displaying bonds with the present system using LAMMPS is provided at the end of the tutorial. You can also use external tools like VMD or OVITO (see the tip for tutorial 3).

Launch the deformation

After equilibration, let us set the velocity of the edges equal to 75 m/s (or 0.75 Å/ps) and run for a longer duration than previously. Add the following lines into `breakable.lmp`:

```
velocity cnt_top set 0.75 0 0
velocity cnt_bot set -0.75 0 0

run 30000
```

Run the simulation. Some bonds are expected to break before the end of the simulation (Fig. 14).

Looking at the evolution of the energy, one can see that the total energy E is initially increasing with the deformation.

https://doi.org/10.33011/livecoms.6.1.3037
Living J. Comp. Mol. Sci. 2025, 6(1), 3037

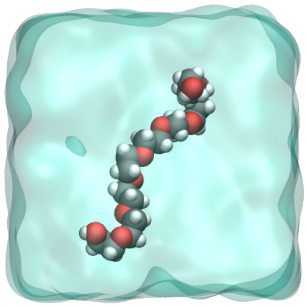

Figure 16. The polymer molecule (PEG - polyethylene glycol) solvated in water as simulated during Tutorial 3. Water molecules are represented as a transparent continuum field for clarity.

When bonds break, the energy relaxes abruptly, as can be seen near t = 32 ps in Fig. 15 a. Using a similar script as previously, i.e., `unbreakable-yaml-reader.py`, import the data into Python and generate the stress-strain curve (Fig. 15 b). The stress-strain curve reveals a linear (elastic) regime where $F_{\mathrm{cnt}} \propto \Delta L_{\mathrm{cnt}}$ for $\Delta L_{\mathrm{cnt}} < 5\,\%$, and a non-linear (plastic) regime for $5\,\% < \Delta L_{\mathrm{cnt}} < 25\,\%$.

Tip: bonds representation with AIREBO

In the input file `solution/breakable-with-tip.lmp`, which is an alternate solution for `breakable.lmp`, a trick is used to represent bonds while using AIREBO. A detailed explanation of the script is beyond the scope of the present tutorial. In short, the trick is to use AIREBO with the `molecular` atom style, and use the `fix bond/break` and `fix bond/create/angle` commands to update the status of the bonds during the simulation:

```
fix break all bond/break 1000 1 2.5
fix form all bond/create/angle 1000 1 1 2.0 1 aconstrain 90.0 180
```

This "hack" works because AIREBO does not pay any attention to bonded interactions and computes the bond topology dynamically inside the pair style. Thus adding bonds of bond style `zero` does not add any interactions but allows the visualization of them with `dump image`. It is required to change the `special_bonds` setting to disable any neighbor list exclusions as they are common for force fields with explicit bonds.

```
bond_style zero
bond_coeff 1 1.4
special_bonds lj/coul 1.0 1.0 1.0
```

3.3 Tutorial 3: Polymer in water

The goal of this tutorial is to use LAMMPS to solvate a small hydrophilic polymer molecule (PEG - polyethylene glycol) in a reservoir of water (Fig. 16). Once the water reservoir is properly equilibrated at the desired temperature and pressure, the polymer molecule is added and a constant stretching force is applied to both ends of the polymer. The evolution of the polymer length is measured as a function of time. The GROMOS 54A7 force field [36] is used for the PEG, the SPC/Fw model [37] is used for the water, and the long-range Coulomb interactions are solved using the PPPM solver [38]. This tutorial was inspired by a publication by Liese and coworkers, in which molecular dynamics simulations are compared with force spectroscopy experiments, see Ref. 39.

> When mixing different force fields, as is done here with GROMOS and SPC/Fw, users should exercise caution. The choices made in these tutorials prioritize progressive learning of LAMMPS functionality over strict physical accuracy. While GROMOS is commonly used with water models from the SPC family [40], the inter-compatibility of force fields is not generally guaranteed.

3.3.1 Preparing the water reservoir

In this tutorial, the water reservoir is first prepared in the absence of the polymer. A rectangular box of water is created and equilibrated at ambient temperature and pressure. The SPC/Fw water model is used [37], which is a flexible variant of the rigid SPC (simple point charge) model [41]. To set up this tutorial, select «`Start Tutorial 3`» from the «`Tutorials`» menu of LAMMPS–GUI and follow the instructions. The editor should display the following content corresponding to `water.lmp`:

```
units real
atom_style full
bond_style harmonic
angle_style harmonic
dihedral_style harmonic
pair_style lj/cut/coul/long 10
kspace_style ewald 1e-5
special_bonds lj 0.0 0.0 0.5 coul 0.0 0.0 1.0 angle yes
```

With the unit style `real`, masses are in g/mol, distances in Å, time in fs, and energies in kcal/mol. With the `atom_style full`, each atom is a dot with a mass and a charge that can be linked by bonds, angles, dihedrals, and/or impropers. The `bond_style`, `angle_style`, and `dihedral_style` commands define the potentials for the bonds, angles, and dihedrals used in the simulation, here `harmonic`. With the `pair_style` named `lj/cut/coul/long`, atoms interact through both a Lennard-Jones (LJ) potential and Coulomb interactions. The value of 10 Å is the cutoff, and the `kspace_style` command defines the long-range solver for the Coulomb interactions [42]. Finally, the `special_bonds` command, which was already seen in Tutorial 2, sets the LJ and Coulomb weighting factors for the interaction between neighboring atoms.

https://doi.org/10.33011/livecoms.6.1.3037
Living J. Comp. Mol. Sci. 2025, 6(1), 3037

With Coulomb interactions, additional rules apply to the `pair_coeff` command: (a) atom type values only matter for assignment of LJ potential parameters; (b) for Coulomb interactions, there are no parameters outside the cutoff, and when using a `coul/long` pair style, that cutoff can only be set globally for all atoms with the `pair_style` command; (c) for Coulomb interactions, only the per-atom charge and any `special_bonds` exclusions are relevant.

Let us create a 3D simulation box of dimensions $6 \times 3 \times 3$ nm^3, and make space for 8 atom types (2 for the water, 6 for the polymer), 7 bond types (1 for the water, 6 for the polymer), 8 angle types (1 for the water, 7 for the polymer), and 4 dihedral types (only for the polymer). Copy the following lines into `water.lmp`:

```
region box block −30 30 −15 15 −15 15
create_box 8 box &
bond/types 7 &
angle/types 8 &
dihedral/types 4 &
extra/bond/per/atom 3 &
extra/angle/per/atom 6 &
extra/dihedral/per/atom 10 &
extra/special/per/atom 14
```

The `extra/x/per/atom` commands reserve memory for adding bond topology data later. We use the file `parameters.inc` to set all the parameters (masses, interaction energies, bond equilibrium distances, etc). Thus add to `water.lmp` the line:

```
include parameters.inc
```

This tutorial uses type labels [12] to map each numeric atom type to a string (see the `parameters.inc` file): `labelmap atom 1 OE 2 C 3 HC 4 H 5 CPos 6 OAlc 7 OW 8 HW` Therefore, the oxygen and hydrogen atoms of water (respectively types 7 and 8) can be referred to as 'OW' and 'HW', respectively. Similar maps are used for the bond types, angle types, and dihedral types.

Let us create water molecules. To do so, let us import a molecule template called `water.mol` and then randomly create 700 molecules. Add the following lines into `water.lmp`:

```
molecule h2omol water.mol
create_atoms 0 random 700 87910 NULL mol h2omol 454756 &
  overlap 1.0 maxtry 50
```

The first parameter is 0, meaning that the atom types from the `water.mol` file will be used. The `overlap 1.0` option of the `create_atoms` command ensures that no atoms are placed exactly in the same position, as this would cause the simulation to crash. The `maxtry 50` asks LAMMPS to try at most 50 times to insert the molecules, which is useful in case some insertion attempts are rejected due to overlap. In some cases, depending on the system and the values of `overlap` and `maxtry`, LAMMPS may not create the desired number of molecules. Always check the number of created atoms in the `log` file (or in the `«Output»` window), where you should see:

```
Created 2100 atoms
```

When LAMMPS fails to create the desired number of molecules, a WARNING appears. The molecule template called `water.mol` must be downloaded and saved next to `water.lmp`. This template contains the necessary structural information of a water molecule, such as the number of atoms, or the IDs of the atoms that are connected by bonds and angles.

Then, let us organize the atoms of types OW and HW of the water molecules in a group named `H2O` and perform a small energy minimization. The energy minimization is mandatory here because of the small `overlap` value of 1 Å chosen in the `create_atoms` command. Add the following lines into `water.lmp`:

```
group H2O type OW HW
minimize 1.0e−4 1.0e−6 100 1000
reset_timestep 0
```

Resetting the step of the simulation to 0 using the `reset_timestep` command is optional. It is used here because the number of iterations performed by the `minimize` command is usually not a round number, since the minimization stops when one of four criteria is reached, which can disrupt the intended frequency of outputs such as `dump` commands that depend on the timestep count. We will use `fix npt` to control the temperature and pressure of the molecules with a Nosé-Hoover thermostat and baro-

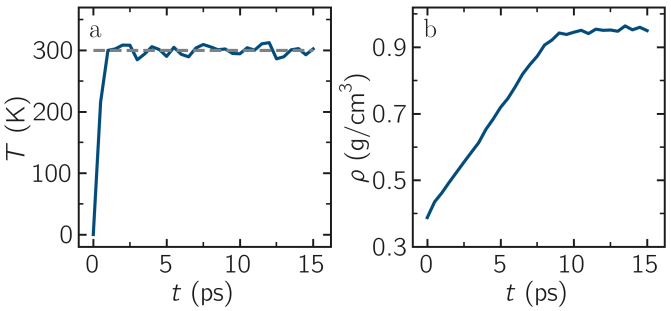

Figure 17. a) Temperature, T, of the water reservoir from Tutorial 3 as a function of the time, t. The horizontal dashed line is the target temperature of 300 K. b) Evolution of the system density, ρ, with t.

https://doi.org/10.33011/livecoms.6.1.3037
Living J. Comp. Mol. Sci. 2025, 6(1), 3037

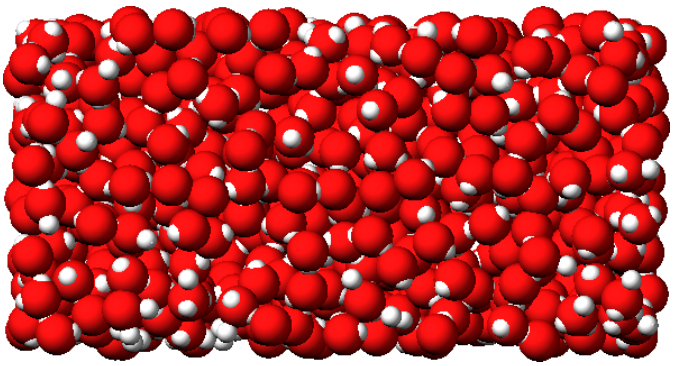

Figure 18. The water reservoir from Tutorial 3 after equilibration. Oxygen atoms are in red, and hydrogen atoms are in white.

stat, respectively [34, 35, 43]. Add the following line into `water.lmp`:

```
fix mynpt all npt temp 300 300 100 iso 1 1 1000
```

The `fix npt` allows us to impose both a temperature of 300 K (with a damping constant of 100 fs), and a pressure of 1 atmosphere (with a damping constant of 1000 fs). With the `iso` keyword, the three dimensions of the box will be re-scaled isotropically, maintaining the same proportion in all directions.

Let us output the system into images by adding the following commands to `water.lmp`:

```
dump viz all image 250 myimage-*.ppm type type &
  shiny 0.1 box no 0.01 view 0 90 zoom 3 size 1000 600
dump_modify viz backcolor white &
  acolor OW red acolor HW white &
  adiam OW 3 adiam HW 1.5
```

Let us also extract the volume and density, among others, every 500 steps:

```
thermo 500
thermo_style custom step temp etotal vol density
```

With the real units system, the volume is in $Å^3$, and the density is in g/cm^3.

Finally, let us set the timestep to 1.0 fs, and run the simulation for 15 ps by adding the following lines into `water.lmp`:

```
timestep 1.0
run 15000

write_restart water.restart
```

The final state is saved in a binary file named `water.restart`. Run the input using LAMMPS. The system reaches its equilibrium temperature after just a few picoseconds, and its equilibrium density after approximately 10 picoseconds (Fig. 17). A snapshot of the equilibrated system can also be seen in Fig. 18.

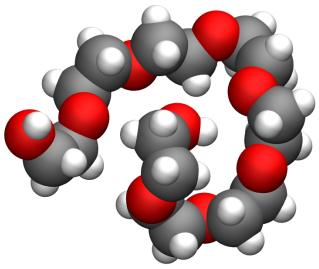

Figure 19. The PEG molecule from Tutorial 3. The carbon atoms are in gray, the oxygen atoms in red, and the hydrogen atoms in white.

The binary file created by the `write_restart` command contains the complete state of the simulation, including atomic positions, velocities, and box dimensions (similar to `write_data`), but also the groups, the compute, or the `atom_style`. Use the «`Inspect Restart`» option of the LAMMPS–GUI to vizualize the content saved in `water.restart`.

3.3.2 Solvating the PEG in water

Now that the water reservoir is equilibrated, we can safely add the PEG polymer to the water. The PEG molecule topology was downloaded from the ATB repository [40, 44]. It has a formula $C_{16}H_{34}O_9$, and the parameters are taken from the GROMOS 54A7 force field [36] (Fig. 19).

Open the file named `merge.lmp` that was downloaded alongside `water.lmp` during the tutorial setup. It only contain one line:

```
read_restart water.restart
```

Most of the commands that were initially present in `water.lmp`, such as the `units` of the `atom_style` commands do not need to be repeated, as they were saved within the `.restart` file. There is also no need to re-include the parameters from the `.inc` file. The `kspace_style` command, however, is not saved by the `write_restart` command and must be repeated. Since Ewald summation is not the most efficient choice for such dense system, let us use PPPM (for particle-particle particle-mesh) for the rest of the tutorial. Add the following command to `merge.lmp`:

```
kspace_style pppm 1e-5
```

Using the molecule template for the polymer called `peg.mol`, let us create a single molecule in the middle of the box by adding the following commands to `merge.lmp`:

```
molecule pegmol peg.mol
create_atoms 0 single 0 0 0 mol pegmol 454756
```


https://doi.org/10.33011/livecoms.6.1.3037
Living J. Comp. Mol. Sci. 2025, 6(1), 3037

Let us create a group for the atoms of the PEG (the previously created group H2O was saved by the restart and can be omitted):

```
group PEG type C CPos H HC OAlc OE
```

Water molecules that are overlapping with the PEG must be deleted to avoid future crashing. Add the following line into `merge.lmp`:

```
delete_atoms overlap 2.0 H2O PEG mol yes
```

Here the value of 2.0 Å for the overlap cutoff was fixed arbitrarily and can be chosen through trial and error. If the cutoff is too small, the simulation will crash because atoms that are too close to each other undergo forces that can be extremely large. If the cutoff is too large, too many water molecules will unnecessarily be deleted.

Let us use the `fix npt` to control the temperature, as well as the pressure by allowing the box size to be rescaled along the *x*-axis:

```
fix mynpt all npt temp 300 300 100 x 1 1 1000
```

Let us also use the `recenter` command to always keep the PEG at the position (0, 0, 0):

```
fix myrct PEG recenter 0 0 0 shift all
```

Note that the `recenter` command has no impact on the dynamics, it simply repositions the frame of reference so that any drift of the system is ignored, which can be convenient for visualizing and analyzing the system. However, be aware that using `fix recenter` can sometimes mask underlying issues in the simulation, such as a net momentum or the so-called "flying ice cube syndrome" [7].

Let us create images of the systems:

```
dump viz all image 250 myimage-*.ppm type type size 1100 600 &
  box no 0.1 shiny 0.1 view 0 90 zoom 3.3 fsaa yes bond atom 0.8
dump_modify viz backcolor white acolor OW red adiam OW 0.2 &
  acolor OE darkred adiam OE 2.6 acolor HC white adiam HC 1.4 &
  acolor H white adiam H 1.4 acolor CPos gray adiam CPos 2.8 &
  acolor HW white adiam HW 0.2 acolor C gray adiam C 2.8 &
  acolor OAlc darkred adiam OAlc 2.6
thermo 500
```

Finally, to perform a short equilibration and save the final state to a `.restart` file, add the following lines to the input:

```
timestep 1.0
run 10000

write_restart merge.restart
```

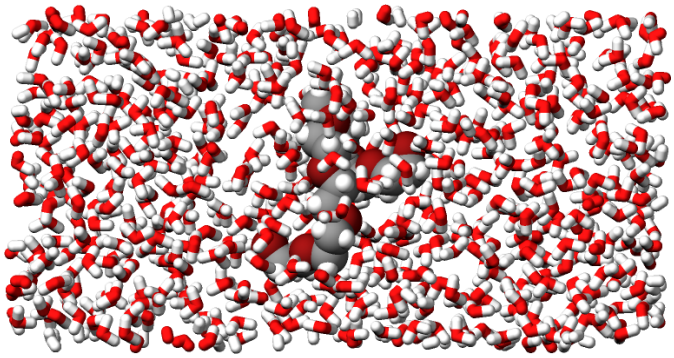

Figure 20. The PEG molecule solvated in water during Tutorial 3.

Run the simulation using LAMMPS. From the outputs, you can make sure that the temperature remains close to the target value of 300 K throughout the entire simulation, and that the volume and total energy are almost constant, indicating that the system was in a reasonable configuration from the start. See a snapshot of the system in Fig. 20.

3.3.3 Stretching the PEG molecule

Here, a constant force is applied to both ends of the PEG molecule until it stretches. Open the file named `pull.lmp`, which only contains two lines:

```
kspace_style pppm 1e-5
read_restart merge.restart
```

Next, we'll create new atom groups, each containing a single oxygen atom. The atoms of type OAlc correspond to the hydroxyl (alcohol) group oxygen atoms located at the ends of the PEG molecule, which we will use to apply the force. Add the following lines to `pull.lmp`:

```
group ends type OAlc
variable xcm equal xcm(ends,x)
variable oxies atom type==label2type(atom,OAlc)
variable end1 atom v_oxies*(x>v_xcm)
variable end2 atom v_oxies*(x<v_xcm)
group topull1 variable end1
group topull2 variable end2
```

These lines identify the oxygen atoms (type OAlc) at the ends of the PEG molecule and calculates their center of mass along the *x*-axis. It then divides these atoms into two groups, `end1` (i.e., the OAlc atom to the right of the center) and `end2` (i.e., the OAlc atom to the right of the center), for applying force during the stretching process.

Add the following `dump` command to create images of the system:

```
dump viz all image 250 myimage-*.ppm type &
  type shiny 0.1 box no 0.01 &
  view 0 90 zoom 3.3 fsaa yes bond atom 0.8 size 1100 600
dump_modify viz backcolor white &
```


https://doi.org/10.33011/livecoms.6.1.3037
Living J. Comp. Mol. Sci. 2025, 6(1), 3037

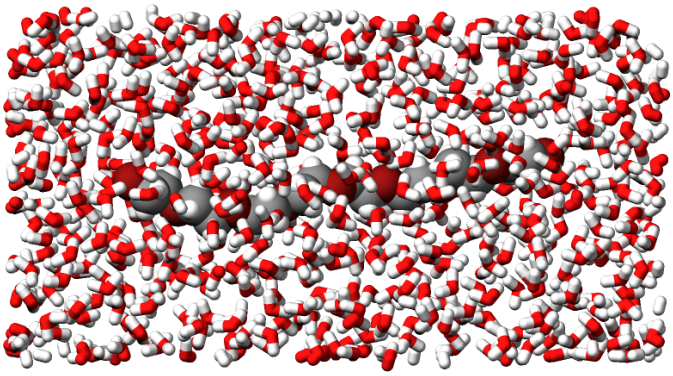

Figure 21. PEG molecule stretched along the *x* direction in water as simulated during Tutorial 3.

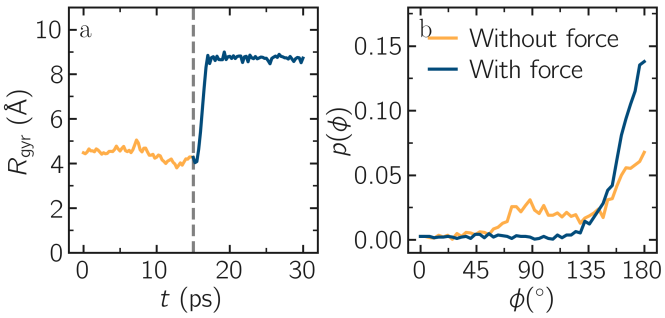

Figure 22. a) Evolution of the radius of gyration R_{gyr} of the PEG molecule from Tutorial 3, with the force applied starting at t = 15 ps. b) Histograms of the dihedral angles of type 1 in the absence (orange) and in the presence (blue) of the applied force.

```
acolor OW red acolor HW white &
acolor OE darkred acolor OAlc darkred &
acolor C gray acolor CPos gray &
acolor H white acolor HC white &
adiam OW 0.2 adiam HW 0.2 &
adiam C 2.8 adiam CPos 2.8 adiam OAlc 2.6 &
adiam H 1.4 adiam HC 1.4 adiam OE 2.6
```

Let us use a single Nosé-Hoover thermostat applied to all the atoms, and let us keep the PEG in the center of the box, by adding the following lines to `pull.lmp`:

```
timestep 1.0
fix mynvt all nvt temp 300 300 100
fix myrct PEG recenter 0 0 0 shift all
```

To investigate the stretching of the PEG molecule, let us compute its radius of gyration [45] and the angles of its dihedral constraints using the following commands:

```
compute rgyr PEG gyration
compute dphi PEG dihedral/local phi
```

The radius of gyration can be directly printed with the `thermo_style` command:

```
thermo_style custom step temp etotal c_rgyr
thermo 250
dump mydmp all local 100 pull.dat index c_dphi
```

By contrast with the radius of gyration (compute `rgyr`), the dihedral angle ϕ (compute `dphi`) is returned as a vector by the `compute dihedral/local` command and must be written to a file using the `dump local` command.

Finally, let us simulate 15 picoseconds without any external force:

```
run 15000
```

This initial run will serve as a benchmark to quantify the changes caused by the applied force in later steps. Next, let us apply a force to the two selected oxygen atoms using two `addforce` commands, and then run the simulation for an extra 15 ps:

```
fix myaf1 topull1 addforce 10 0 0
fix myaf2 topull2 addforce −10 0 0
run 15000
```

Each applied force has a magnitude of 10 kcal/mol/Å, corresponding to 0.67 nN. This value was chosen to be sufficiently large to overcome both the thermal agitation and the entropic contributions from the molecules.

Run the `pull.lmp` file using LAMMPS. From the generated images of the system, you should observe that the PEG molecule eventually aligns in the direction of the applied force (as seen in Fig. 21). The evolutions of the radius of gyration over time indicates that the PEG quickly adjusts to the external force (Fig. 22 a). Additionally, from the values of the dihedral angles printed in the `pull.dat` file, you can create a histogram of dihedral angles for a specific type. For example, the angle ϕ for dihedrals of type 1 (C-C-OE-C) is shown in Fig. 22 b.

Tip: using external visualization tools

Trajectories can be visualized using external tools such as VMD or OVITO [25, 27]. To do so, the IDs and positions of the atoms must be regularly written to a file during the simulation. This can be accomplished by adding a `dump` command to the input file. For instance, create a duplicate of `pull.lmp` and name it `pull-with-tip.lmp`. Then, replace the existing `dump` and `dump_modify` commands with:

```
dump mydmp all atom 1000 pull.lammpstrj
```

Running the `pull-with-tip.lmp` file using LAMMPS will generate a trajectory file named `pull.lammpstrj`, which can be opened in OVITO or VMD.

https://doi.org/10.33011/livecoms.6.1.3037
Living J. Comp. Mol. Sci. 2025, 6(1), 3037

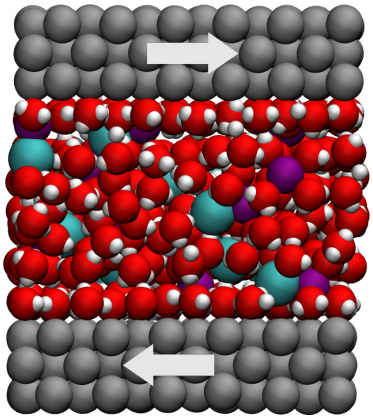

Figure 23. The electrolyte confined in a nanometer slit pore as simulated during Tutorial 4. Na^+ ions are represented as purple spheres, Cl^- ions as cyan spheres, water molecules are colored in red and white, and the walls are colored in gray. The arrows indicate the imposed lateral motion of the walls.

Since the default trajectory dump file does not contain information about topology and elements, it is usually preferred to first write out a data file and import it directly (in the case of OVITO) or convert it to a PSF file (for VMD). This allows the topology to be loaded before *adding* the trajectory file to it. When using LAMMPS–GUI, this process can be automated through the «View in OVITO» or «View in VMD» options in the «Run» menu. Afterwards only the trajectory dump needs to be added. Alternatively, the `dump custom` command can be combined with `dump` command to include element names in the dump file and simplify visualization.

Microstates collected during a simulation in the form of a trajectory can be analyzed within LAMMPS using the `rerun` command. This is particularly useful, for example, for computing properties not set up in the original simulation without having to run it again. A possible use of the `rerun` command is estimating the self-diffusion coefficient by using the `compute msd` command [1].

3.4 Tutorial 4: Nanosheared electrolyte

The objective of this tutorial is to simulate an electrolyte nanoconfined and sheared between two walls (Fig. 23). The density and velocity profiles of the fluid in the direction normal to the walls are extracted to highlight the effect of confining a fluid on its local properties. This tutorial demonstrates key concepts of combining a fluid and a solid in the same simulation. A major difference from the previous tutorial, Polymer in water, is that here a rigid four-point water model named TIP4P/2005 is used [13].

Four-point water models such as TIP4P/2005 are widely used as they offer a good compromise between accuracy and computational cost [46].

3.4.1 System preparation

The fluid and walls must first be generated, followed by equilibration at the desired temperature and pressure.

System generation

To set up this tutorial, select «Start Tutorial 4» from the «Tutorials» menu of LAMMPS–GUI and follow the instructions. The editor should display the following content corresponding to `create.lmp`:

```
boundary p p f
units real
atom_style full
bond_style harmonic
angle_style harmonic
pair_style lj/cut/tip4p/long O H O−H H−O−H 0.1546 12.0
kspace_style pppm/tip4p 1.0e−5
kspace_modify slab 3.0
```

These lines are used to define the most basic parameters, including the atom style, the forms of the non-bonded, bond, and angle potentials, as well as other specifics of the non-bonded interactions. Here, `lj/cut/tip4p/long` imposes a Lennard-Jones potential with a cut-off at 12 Å and a long-range Coulomb potential. The parameters `O`, `H`, `O-H`, and `H-O-H` correspond respectively to the oxygens, hydrogens, O-H bonds, and H-O-H angle constraints of the water molecules; their definitions, provided by the `labelmap` commands, will be clarified below.

So far, the commands are relatively similar to those in the previous tutorial, Polymer in water, with two major differences: the use of `lj/cut/tip4p/long` instead of `lj/cut/coul/long`, and `pppm/tip4p` instead of `pppm`. When using `lj/cut/tip4p/long` and `pppm/tip4p`, the interactions resemble the conventional Lennard-Jones and Coulomb interactions, except that they are specifically designed for the four-point water model. As a result, LAMMPS automatically adds the fourth point to the water molecules, assigning type O atoms as oxygen and type H atoms as hydrogen. The fourth massless atom (M) of the TIP4P water molecule does not have to be defined explicitly, and the value of 0.1546 Å corresponds to the O-M distance of the TIP4P-2005 water model [13]. All other atoms in the simulation are treated as usual, with long-range Coulomb interactions. Another novelty, here, is the use of `kspace_modify slab 3.0` that is combined with the non-periodic boundaries along the *z* coordinate: `boundary p p f`. With the `slab` option, the system is treated as periodical along *z*, but with an empty

https://doi.org/10.33011/livecoms.6.1.3037
Living J. Comp. Mol. Sci. 2025, 6(1), 3037

volume inserted between the periodic images of the slab, and the interactions along *z* effectively turned off.

Let us create the box and the label maps by adding the following lines to `create.lmp`:

```
lattice fcc 4.04
region box block −3 3 −3 3 −5 5
create_box 5 box bond/types 1 angle/types 1 &
   extra/bond/per/atom 2 extra/angle/per/atom 1 &
   extra/special/per/atom 2
labelmap atom 1 O 2 H 3 Na+ 4 Cl− 5 WALL
labelmap bond 1 O−H
labelmap angle 1 H−O−H
```

The `lattice` command defines the unit cell. Here, the face-centered cubic (fcc) lattice with a scale factor of 4.04 has been chosen for the future positioning of the atoms of the walls. The `region` command defines a geometric region of space. By choosing xlo = −3 and xlo = 3, and because we have previously chosen a lattice with a scale factor of 4.04, the region box extends from −12.12 Å to 12.12 Å along the *x* direction. The `create_box` command creates a simulation box with 5 types of atoms: the oxygen and hydrogen of the water molecules, the two ions (Na^+, Cl^-), and the atoms from the walls. The simulation contains 1 type of bond and 1 type of angle (both required by the water molecules). The parameters for these bond and angle constraints will be given later. The `extra/ (...)` keywords are for memory allocation. Finally, the `labelmap` commands assign alphanumeric type labels to each numeric atom type, bond type, and angle type, concepts already introduced in previous tutorials.

Now, we can add atoms to the system. First, let us create two sub-regions corresponding respectively to the two solid walls, and create a larger region from the union of the two regions. Then, let us create atoms of type WALL within the two regions. Add the following lines to `create.lmp`:

```
region rbotwall block −3 3 −3 3 −4 −3
region rtopwall block −3 3 −3 3 3 4
region rwall union 2 rbotwall rtopwall
create_atoms WALL region rwall
```

Atoms will be placed in the positions of the previously defined lattice, thus forming fcc solids.

To add the water molecules, the molecule template called `water.mol` must be located next to `create.lmp`. The template contains all the necessary information concerning the water molecule, such as atom positions, bonds, and angles. Add the following lines to `create.lmp`:

```
region rliquid block INF INF INF INF −2 2
molecule h2omol water.mol
create_atoms 0 region rliquid mol h2omol 482793
```

Within the last three lines, a region named `rliquid` is created based on the last defined lattice, `fcc 4.04`. `rliquid` will be used for introducing the water molecules in the simulation domain. The `molecule` command opens up the molecule template called `water.mol`, and names the associated molecule `h2omol`. The new molecules are placed on the `fcc 4.04` lattice by the `create_atoms` command. The first parameter is 0, meaning that the atom types from the `water.mol` file will be used. The number `482793` is a seed that is required by LAMMPS, it can be any positive integer.

Finally, let us create 30 ions (15 Na^+ and 15 Cl^-) in between the water molecules, by adding the following commands to `create.lmp`:

```
create_atoms Na+ random 15 5802 rliquid overlap 0.3 maxtry 500
create_atoms Cl− random 15 9012 rliquid overlap 0.3 maxtry 500
set type Na+ charge 1
set type Cl− charge −1
```

Each `create_atoms` command will add 15 ions at random positions within the `rliquid` region, ensuring that there is no `overlap` with existing molecules. Feel free to increase or decrease the salt concentration by changing the number of desired ions. To keep the system charge neutral, always insert the same number of Na^+ and Cl^-, unless there are other charges in the system. The charges of the newly added ions are specified by the two `set` commands.

Before starting the simulation, we need to define the parameters of the simulation: the mass of the 5 atom types (O, H, Na^+, Cl^-, and wall), the pairwise interaction parameters (in this case, for the Lennard-Jones potential), and the bond and angle parameters. Copy the following lines into `create.lmp`:

```
include parameters.inc
include groups.inc
```

Both `parameters.inc` and `groups.inc` files must be located next to `create.lmp`.

The `parameters.inc` file contains the masses, as follows:

```
mass O 15.9994
mass H 1.008
mass Na+ 22.990
mass Cl− 35.453
mass WALL 26.9815
```

Each `mass` command assigns a mass in g/mol to an atom type. The `parameters.inc` file also contains the pair coefficients:

```
pair_coeff O O 0.185199 3.1589
pair_coeff H H 0.0 1.0
pair_coeff Na+ Na+ 0.04690 2.4299
```


https://doi.org/10.33011/livecoms.6.1.3037
Living J. Comp. Mol. Sci. 2025, 6(1), 3037

```
pair_coeff Cl− Cl− 0.1500 4.04470
pair_coeff WALL WALL 11.697 2.574
pair_coeff O WALL 0.4 2.86645
```

Each `pair_coeff` assigns the depth of the LJ potential (in kcal/mol), and the distance (in Ångströms) at which the particle-particle potential energy is 0. As noted in previous tutorials, with the important exception of `pair_coeff O WALL`, pairwise interactions were only assigned between atoms of identical types. By default, LAMMPS calculates the pair coefficients for the interactions between atoms of different types (i and j) by using geometric average: $\epsilon_{ij} = \sqrt{\epsilon_{ii}\epsilon_{jj}}$, $\sigma_{ij} = \sqrt{\sigma_{ii}\sigma_{jj}}$. However, if the default value of 1.472 kcal/mol was used for $\epsilon_{\text{O-WALL}}$, the solid walls would be extremely hydrophilic, causing the water molecules to form dense layers. As a comparison, the water-water energy $\epsilon_{\text{O-O}}$ is only 0.185199 kcal/mol. Therefore, to make the walls less hydrophilic, the value of $\epsilon_{\text{O-WALL}}$ was reduced.

Finally, the `parameters.inc` file contains the following two lines:

```
bond_coeff O−H 0 0.9572
angle_coeff H−O−H 0 104.52
```

The `bond_coeff` command, used here for the O-H bond of the water molecule, sets both the spring constant of the harmonic potential and the equilibrium bond distance of 0.9572 Å. The force constant can be 0 for a rigid water molecule because the SHAKE algorithm, which will be used in the input at a later step, will constrain the intramolecular structure of the water molecules (see below) [47, 48]. Similarly, the `angle_coeff` command for the H-O-H angle of the water molecule sets the force constant of the angular harmonic potential to 0 and the equilibrium angle to 104.52°.

Alongside `parameters.inc`, the `groups.inc` file contains several `group` commands to define groups of atoms based on their types:

```
group H2O type O H
group Na type Na+
group Cl type Cl−
group ions union Na Cl
group fluid union H2O ions
```

The `groups.inc` file also defines the `walltop` and `wallbot` groups, which contain the WALL atoms located in the $z > 0$ and $z < 0$ regions, respectively:

```
group wall type WALL
region rtop block INF INF INF INF 0 INF
region rbot block INF INF INF INF INF 0
group top region rtop
group bot region rbot
group walltop intersect wall top
group wallbot intersect wall bot
```

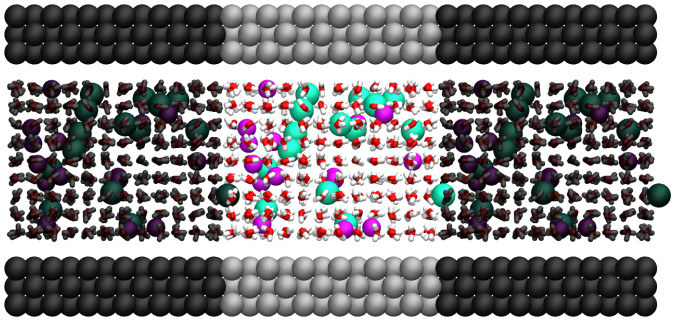

Figure 24. Side view of the system. Periodic images are represented in darker colors. Water molecules are in red and white, Na^+ ions in purple, Cl^- ions in lime, and wall atoms in gray. Note the absence of atomic defect at the cell boundaries.

Currently, the fluid density between the two walls is slightly too high. To avoid excessive pressure, let us add the following lines into `create.lmp` to delete about 15 % of the water molecules:

```
delete_atoms random fraction 0.15 yes H2O NULL 482793 mol yes
```

To create an image of the system, add the following `dump image` into `create.lmp` (see also Fig. 24):

```
dump mydmp all image 200 myimage−*.ppm type type &
  shiny 0.1 box no 0.01 view 90 0 zoom 1.8
dump_modify mydmp backcolor white &
  acolor O red adiam O 2 &
  acolor H white adiam H 1 &
  acolor Na+ blue adiam Na+ 2.5 &
  acolor Cl− cyan adiam Cl− 3 &
  acolor WALL gray adiam WALL 3
```

Finally, add the following lines into `create.lmp`:

```
run 0

write_data create.data nocoeff
```

The `run 0` command initializes the simulation, which is required for cleanly saving the state, but it does not advance positions or velocities. The `write_data` command generates a file called `system.data` containing the information required to restart the simulation from the final configuration produced by this input file. With the `nocoeff` option, the parameters from the force field are not included in the `.data` file. Run the `create.lmp` file using LAMMPS, and a file named `create.data` will be created alongside `create.lmp`.

Energy minimization

Let us move the atoms and place them in more energetically favorable positions before starting the actual molecular

https://doi.org/10.33011/livecoms.6.1.3037
Living J. Comp. Mol. Sci. 2025, 6(1), 3037

dynamics simulation. Open the `equilibrate.lmp` file that was downloaded alongside `create.lmp` during the tutorial setup. Same as before, it contains the following lines:

```
boundary p p f
units real
atom_style full
bond_style harmonic
angle_style harmonic
pair_style lj/cut/tip4p/long O H O−H H−O−H 0.1546 12.0
kspace_style pppm/tip4p 1.0e−5
kspace_modify slab 3.0

read_data create.data

include parameters.inc
include groups.inc
```

The only difference from the previous input is that, instead of creating a new box and new atoms, we open the previously created `create.data` file.

Now, let us use the SHAKE algorithm to maintain the shape of the water molecules [47, 48] by adding the following line to the script.

```
fix myshk H2O shake 1.0e−5 200 0 b O−H a H−O−H kbond 2000
```

Here the SHAKE algorithm applies to the `O-H` bond and the `H-O-H` angle of the water molecules. The `kbond` keyword specifies the force constant that will be used to apply a restraint force when used during minimization. This last keyword is important here, because the spring constants of the rigid water molecules were set to 0 (see the `parameters.inc` file).

LAMMPS provides several ways to keep molecules rigid during a simulation. The `fix shake` command is appropriate and efficient for constraining individual bonds or bonds and angles within small molecules like water while using a per-atom time integration fix command like `fix nve` or `fix nvt`. However, it fails for linear molecules like CO_2 or constraining larger or more complex objects. In such cases, the `fix rigid` family of commands can be used to perform time integration for translation and rotation of groups of atoms as rigid bodies.

Let us also create images of the system and control the printing of thermodynamic outputs by adding the following lines to `equilibrate.lmp`:

```
  dump mydmp all image 1 myimage−*.ppm type type &
  shiny 0.1 box no 0.01 view 90 0 zoom 1.8
dump_modify mydmp backcolor white &
  acolor O red adiam O 2 &
  acolor H white adiam H 1 &
  acolor Na+ blue adiam Na+ 2.5 &
  acolor Cl− cyan adiam Cl− 3 &
  acolor WALL gray adiam WALL 3

thermo 1
thermo_style custom step temp etotal press
```

Let us perform an energy minimization by adding the following lines to `equilibrate.lmp`:

```
minimize 1.0e−6 1.0e−6 1000 1000
reset_timestep 0
```

When running the `equilibrate.lmp` file with LAMMPS, you should observe that the total energy of the system is initially very high but rapidly decreases. From the generated images of the system, you will notice that the atoms and molecules are moving to adopt more favorable positions.

System equilibration

Let us equilibrate further the entire system by letting both fluid and wall relax at ambient temperature. Here, the commands are written within the same `equilibrate.lmp` file, right after the `reset_timestep` command.

Let us do a molecular dynamics simulation using the Nosé-Hoover thermostat. Add the following lines to `equilibrate.lmp`:

```
fix mynvt all nvt temp 300 300 100
fix myshk H2O shake 1.0e−5 200 0 b O−H a H−O−H
fix myrct all recenter NULL NULL 0
timestep 1.0
```

As mentioned previously, the `fix recenter` does not influence the dynamics, but will keep the system in the center of the box, which makes the visualization easier. Then, add the following lines into `equilibrate.lmp` for the trajectory visualization:

```
undump mydmp
dump mydmp all image 250 myimage−*.ppm type type &
  shiny 0.1 box no 0.01 view 90 0 zoom 1.8
dump_modify mydmp backcolor white &
  acolor O red adiam O 2 &
  acolor H white adiam H 1 &
  acolor Na+ blue adiam Na+ 2.5 &
  acolor Cl− cyan adiam Cl− 3 &
  acolor WALL gray adiam WALL 3
```

The `undump` command is used to cancel the previous `dump` command. Then, a new `dump` command with a larger dumping period is used.

https://doi.org/10.33011/livecoms.6.1.3037
Living J. Comp. Mol. Sci. 2025, 6(1), 3037

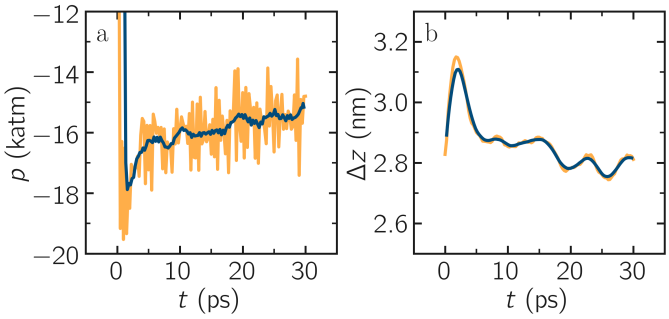

Figure 25. a) Pressure, p, of the nanosheared electrolyte system simulated in Tutorial 4 as a function of the time, t. b) Distance between the walls, Δz, as a function of t. The orange line shows the raw data, and the blue line represents a time-averaged curve.

Just like the `undump` command can cancel an active `dump`, other objects defined in a LAMMPS input script can be cancelled when no longer needed. For example, you can use `unfix` to remove a previously defined `fix`, and `uncompute` to delete a `compute`.

To monitor the system equilibration, let us print the distance between the two walls. Add the following lines to `equilibrate.lmp`:

```
variable walltopz equal xcm(walltop,z)
variable wallbotz equal xcm(wallbot,z)
variable deltaz equal v_walltopz-v_wallbotz

thermo 250
thermo_style custom step temp etotal press v_deltaz
```

The first two variables extract the z coordinate of the centers of mass of the two walls. The `deltaz` variable is then used to calculate the difference between the two variables `walltopz` and `wallbotz`, i.e. the distance in the z direction between the two centers of mass of the walls.

Finally, let us run the simulation for 30 ps by adding a `run` command to `equilibrate.lmp`:

```
run 30000

write_data equilibrate.data nocoeff
```

Run the `equilibrate.lmp` file using LAMMPS. Both the pressure and the distance between the two walls show oscillations at the start of the simulation but eventually stabilize at their equilibrium values toward the end of the simulation (Fig. 25).

Note that it is generally recommended to run a longer equilibration. In this case, the slowest process in the system is likely ionic diffusion. Therefore, the equilibration period should, in principle, exceed the time required for the ions to diffuse across the size of the pore, i.e. $H_{\text{pore}}^2/D_{\text{ions}}$. Using $H_{\text{pore}} \approx 1.2$ nm as the final pore size and $D_{\text{ions}} \approx 1.5 \cdot 10^{-9}$ m^2/s as the typical diffusion coefficient for sodium chloride in water at room temperature [49], one finds that the equilibration should be on the order of one nanosecond.

3.4.2 Imposed shearing

From the equilibrated configuration, let us impose a lateral motion on the two walls and shear the electrolyte. Open the last input file named `shearing.lmp`. It starts as follows:

```
boundary p p f
units real
atom_style full
bond_style harmonic
angle_style harmonic
pair_style lj/cut/tip4p/long O H O-H H-O-H 0.1546 12.0
kspace_style pppm/tip4p 1.0e-5
kspace_modify slab 3.0

read_data equilibrate.data

include parameters.inc
include groups.inc
```

To address the dynamics of the system, add the following lines to `shearing.lmp`:

```
compute Tfluid fluid temp/partial 0 1 1
fix mynvt1 fluid nvt temp 300 300 100
fix_modify mynvt1 temp Tfluid

compute Twall wall temp/partial 0 1 1
fix mynvt2 wall nvt temp 300 300 100
fix_modify mynvt2 temp Twall

fix myshk H2O shake 1.0e-5 200 0 b O-H a H-O-H
fix myrct all recenter NULL NULL 0
timestep 1.0
```

One key difference with the previous input is that, here, two thermostats are used, one for the fluid (`mynvt1`) and one for the solid (`mynvt2`). The combination of `fix_modify` with `compute temp` ensures that the correct temperature values are used by the thermostats. Using `compute` commands for the temperature with `temp/partial 0 1 1` is intended to exclude the x coordinate from the thermalization, which is important since a large velocity will be imposed along the x direction.

Then, let us impose the velocity of the two walls by adding the following commands to `shearing.lmp`:

https://doi.org/10.33011/livecoms.6.1.3037
Living J. Comp. Mol. Sci. 2025, 6(1), 3037

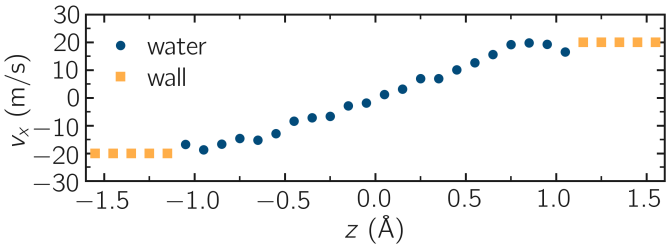

Figure 26. Velocity profiles for water (blue) and walls (orange) along the *z*-axis as simulated in Tutorial 4.

```
fix mysf1 walltop setforce 0 NULL NULL
fix mysf2 wallbot setforce 0 NULL NULL
velocity wallbot set -2e-4 NULL NULL
velocity walltop set 2e-4 NULL NULL
```

The `setforce` commands cancel the forces on `walltop` and `wallbot` in the *x* direction. As a result, the atoms in these two groups will not experience any forces along *x* from their interaction with rest of the system. Consequently, in the absence of external forces, these atoms will conserve the initial velocities imposed by the two `velocity` commands. As seen previously, although the forces on these atoms are set to zero, the `fix setforce` still stores the forces acting on the group before cancellation, which can later be extracted for analysis (see below).

Finally, let us generate images of the systems and print the values of the forces exerted by the fluid on the walls, as given by `f_mysf1[1]` and `f_mysf2[1]`. Add these lines to `shearing.lmp`:

```
dump mydmp all image 250 myimage-*.ppm type type &
  shiny 0.1 box no 0.01 view 90 0 zoom 1.8
dump_modify mydmp backcolor white &
  acolor O red adiam O 2 &
  acolor H white adiam H 1 &
  acolor Na+ blue adiam Na+ 2.5 &
  acolor Cl- cyan adiam Cl- 3 &
  acolor WALL gray adiam WALL 3

thermo 250
thermo_modify temp Tfluid
thermo_style custom step temp etotal f_mysf1[1] f_mysf2[1]
```

Let us also extract the density and velocity profiles using the `chunk/atom` and `ave/chunk` commands. When deployed as below, these commands discretize the simulation domain into spatial bins and compute and output average properties of the atoms belonging to each bin, here the velocity along *x* (`vx`) within the bins. Add the following lines to `shearing.lmp`:

```
compute cc1 H2O chunk/atom bin/1d z 0.0 0.25
compute cc2 wall chunk/atom bin/1d z 0.0 0.25
compute cc3 ions chunk/atom bin/1d z 0.0 0.25
fix myac1 H2O ave/chunk 10 15000 200000 &
  cc1 density/mass vx file shearing-water.dat
fix myac2 wall ave/chunk 10 15000 200000 &
  cc2 density/mass vx file shearing-wall.dat
fix myac3 ions ave/chunk 10 15000 200000 &
  cc3 density/mass vx file shearing-ions.dat

run 200000
```

Here, a bin size of 0.25 Å is used for the density profiles generated by the `ave/chunk` commands, and three `.dat` files are created for the water, the walls, and the ions, respectively. With values of `10 15000 200000`, the velocity `vx` will be evaluated every 10 steps during the final 150,000 steps of the simulations. The result will be averaged and printed only once at the 200,000th step.

Run the simulation using LAMMPS. The averaged velocity profile for the fluid is plotted in Fig. 26. As expected for such a Couette flow geometry, the fluid velocity increases linearly along *z*, and is equal to the walls velocities at the fluid-solid interfaces (no-slip boundary conditions).

From the force applied by the fluid on the solid, one can extract the stress within the fluid, which enables the measurement of its viscosity η according to

$$\eta = \tau/\dot{\gamma} \tag{2}$$

where τ is the stress applied by the fluid on the shearing wall, and $\dot{\gamma}$ the shear rate [50]. Here, the shear rate is approximately $\dot{\gamma} = 20 \cdot 10^9\ \mathrm{s}^{-1}$ (Fig. 26), the average force on each wall is given by `f_mysf1[1]` and `f_mysf2[1]` and is approximately 2.7 kcal/mol/Å. Using a surface area for the walls of $A = 6 \cdot 10^{-18}\ \mathrm{m}^2$, one obtains an estimate for the shear viscosity for the confined fluid of $\eta = 3.1\ \mathrm{mPa \cdot s}$ using Eq. (2).

The viscosity calculated at such a high shear rate may differ from the expected *bulk* value. In general, it is recommended to use a lower value for the shear rate. Note that for lower shear rates, the signal-to-noise ratio is smaller, and longer simulations are needed. Another point to consider is that the viscosity of a fluid next to a solid surface is typically larger than in bulk due to interaction with the walls [51]. Therefore, one expects the present simulation to yield a viscosity that is slightly higher than what would be measured in the absence of walls.

3.5 Tutorial 5: Reactive silicon dioxide

The objective of this tutorial is to demonstrate how the reactive force field ReaxFF can be used to calculate the partial charges of a system undergoing deformation, as well as the formation and breaking of chemical bonds [14, 52]. The system simulated in this tutorial is a block of silicon dioxide SiO_2

https://doi.org/10.33011/livecoms.6.1.3037
Living J. Comp. Mol. Sci. 2025, 6(1), 3037

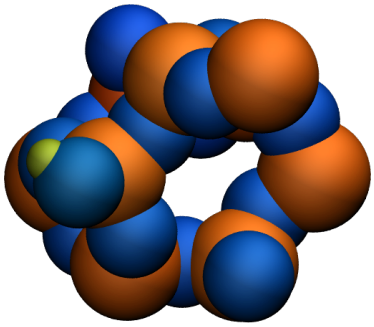

Figure 27. A portion of the silicon dioxide structure as simulated during Tutorial 5. Atoms are colored by their charges: the hydrogen atoms appear as small greenish spheres, silicon atoms as large orange spheres, and oxygen atoms as blue spheres of intermediate size.

(Fig. 27) which is deformed until it ruptures. Particular attention is given to the evolution of atomic charges during deformation, with a focus on tracking chemical reactions resulting from the deformation over time.

3.5.1 Prepare and relax

The first step is to relax the structure with ReaxFF, which which will be achieved using molecular dynamics. To ensure the system equilibrates properly, we will monitor certain parameters over time, such as the system volume. To set up this tutorial, select «Start Tutorial 5» from the «Tutorials» menu of LAMMPS–GUI and follow the instructions. The editor should display the following content corresponding to `relax.lmp`:

```
units real
atom_style full

read_data silica.data
```

So far, the input is very similar to what was seen in the previous tutorials. Some basic parameters are defined (`units` and `atom_style`), and a `.data` file is imported by the `read_data` command.

The initial topology given by `silica.data` corresponds to a small amorphous silica structure. This structure was generated in a prior simulation using the Vashishta force field [53]. If you open the `silica.data` file, you will find in the `Atoms` section that all silicon atoms have a charge of q = 1.1 e, and all oxygen atoms have a charge of q = –0.55 e.

Assigning the same charge to all atoms of the same type is common with many force fields, including the force fields used in the previous tutorials. This changes once ReaxFF is used: the charge of each atom will adjust to its local environment through charge equilibration.

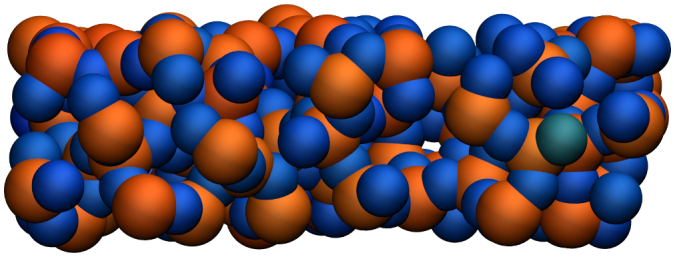

Figure 28. A slice of the amorphous silica simulated during Tutorial 5, where atoms are colored by their charges. Dangling oxygen groups appear in greenish, bulk Si atoms with a charge of about 1.8 e appear in red/orange, and bulk O atoms with a charge of about –0.9 e appear in blue.

Next, copy the following three crucial lines into the `relax.lmp` file:

```
pair_style reaxff NULL safezone 3.0 mincap 150
pair_coeff * * ffield.reax.CHOFe Si O
fix myqeq all qeq/reaxff 1 0.0 10.0 1.0e−6 reaxff maxiter 400
```

In this case, the `pair_style reaxff` is used without a control file (see note below). The `safezone` and `mincap` keywords are added to prevent allocation issues, which sometimes can trigger segmentation faults and `bondchk` errors. The `pair_coeff` command uses the `ffield.reax.CHOFe` file, which should have been downloaded during the tutorial set up. Finally, the `fix qeq/reaxff` is used to perform charge equilibration [54], which occurs at every step. The values 0.0 and 10.0 represent the low and the high cutoffs, respectively, and 1.0e – 6 is the tolerance, i.e., the precision to which the atomic charges are equilibrated during the charge equilibration process. The `maxiter` sets an upper limit to the number of attempts to equilibrate the charge.

The `pair_style reaxff` command optionally accepts a control file, which defines control variables such as global parameters of the ReaxFF potential, as well as performance and output settings. If no control file is provided, as in this tutorial, LAMMPS uses its default values, which correspond to those in Adri van Duin's original stand-alone ReaxFF code [14].

Next, add the following commands to the `relax.lmp` file to track the evolution of the charges during the simulation:

```
group grpSi type Si
group grpO type O
variable qSi equal charge(grpSi)/count(grpSi)
variable qO equal charge(grpO)/count(grpO)
variable vq atom q
```

The definition of the equal style variables qSi and qO make use of functions pre-defined within LAMMPS that allow

https://doi.org/10.33011/livecoms.6.1.3037
Living J. Comp. Mol. Sci. 2025, 6(1), 3037

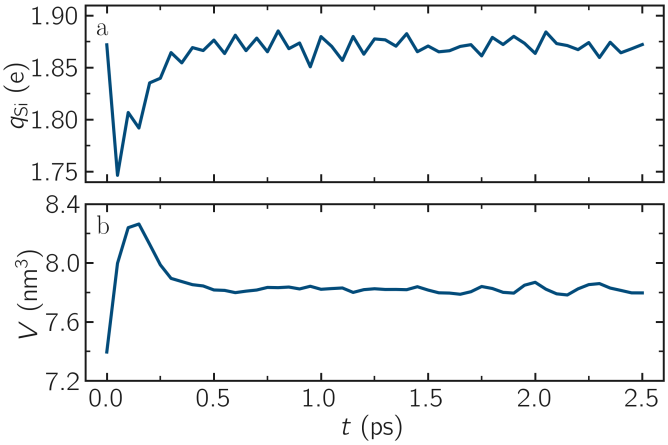

Figure 29. a) Average charge per atom of the silicon, q_{Si}, atoms as a function of time, t, during equilibration of the SiO_2 system from Tutorial 5. b) Volume of the system, V, as a function of t.

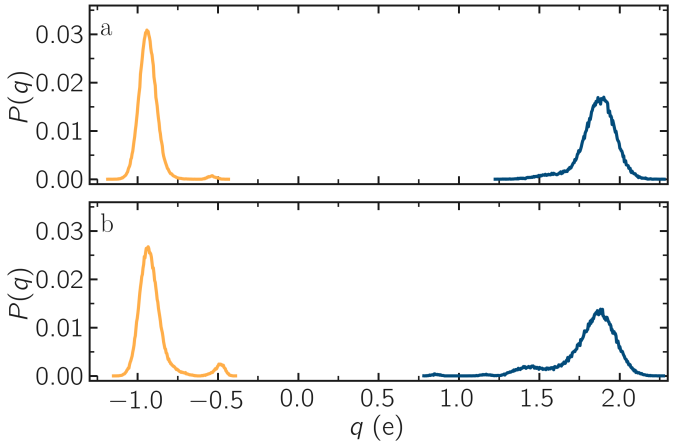

Figure 30. a) Probability distributions of charge of silicon (positive, blue) and oxygen (negative, orange) atoms during the equilibration of the SiO_2 system from Tutorial 5. b) Same probability distributions as in panel (a) after the deformation.

calculating the total charge of atoms belonging to a group (charge()) and the total number of atoms in the group (count()). To print the averaged charges `qSi` and `qO` using the `thermo_style` command, and create images of the system. Add the following lines to `relax.lmp`:

```
thermo 100
thermo_style custom step temp etotal press vol v_qSi v_qO
dump viz all image 100 myimage-*.ppm q &
  type shiny 0.1 box no 0.01 view 180 90 zoom 2.3 size 1200 500
dump_modify viz adiam Si 2.6 adiam O 2.3 backcolor white &
  amap -1 2 ca 0.0 3 min royalblue 0 green max orangered
```

Here, the atoms are colored by their charges `q`, ranging from royal blue (when q = –1 e) to orange-red (when q = 2 e).

We can generate histograms of the charges for each atom type using `fix ave/histo` commands:

```
fix myhis1 grpSi ave/histo 10 500 5000 -1.5 2.5 1000 v_vq &
  file relax-Si.histo mode vector
fix myhis2 grpO ave/histo 10 500 5000 -1.5 2.5 1000 v_vq &
  file relax-O.histo mode vector
```

The `fix ave/histo` command samples values over a group of atoms and builds a histogram over a specified range divided into bins. In this tutorial, it is used to monitor the charge distributions of silicon and oxygen atoms. The parameters `10 500 5000` specify how often the histogram is updated and averaged, `-1.5 2.5` set the value range, `1000` is the number of bins, and `v_vq` is the variable being histogrammed.

We can also use `fix reaxff/species` to evaluate what species are present within the simulation. It will be useful later when the system is deformed, and bonds are broken:

```
fix myspec all reaxff/species 5 1 5 relax.species element Si O
```

Here, the information will be printed every 5 steps in a file called `relax.species`. Let us perform a very short run using the anisotropic NPT command and relax the density of the system:

```
velocity all create 300.0 32028
fix mynpt all npt temp 300.0 300.0 100 aniso 1.0 1.0 1000
timestep 0.5

run 5000

write_data relax.data nofix
```

The `write_data` command is used with the `nofix` keyword to print a data file without extra sections from the `reaxff/species` command. Run the `relax.lmp` file using LAMMPS. As seen from `relax.species`, only one species is detected, called `O384Si192`, representing the entire system.

> With the `aniso` keyword, the three dimensions of the simulation box can change independently. This is particularly relevant for solids and other systems where anisotropic stresses may develop.

As the simulation progresses, the charge of every atom fluctuates because it is adjusting to the local environment of the atom (Fig. 29 a). It is also observed that the averaged charges for silicon and oxygen atoms fluctuate significantly at the beginning of the simulation, corresponding to a rapid change in the system volume, which causes interatomic distances to shift quickly (Fig. 29 b). The atoms with the most extreme charges are located at structural defects, such as dangling oxygen groups (Fig. 28). Finally, the generated `.histo` files can be used to plot the probability distributions, $P(q)$ (see Fig. 30 a).

https://doi.org/10.33011/livecoms.6.1.3037
Living J. Comp. Mol. Sci. 2025, 6(1), 3037

3.5.2 Deform the structure

Let us apply a deformation to the structure to force some Si–O bonds to break (and eventually re-assemble). Open the `deform.lmp` file, which must contain the following lines:

```
units real
atom_style full

read_data relax.data

pair_style reaxff NULL safezone 3.0 mincap 150
pair_coeff * * ffield.reax.CHOFe Si O
fix myqeq all qeq/reaxff 1 0.0 10.0 1.0e−6 reaxff maxiter 400

group grpSi type Si
group grpO type O
variable qSi equal charge(grpSi)/count(grpSi)
variable qO equal charge(grpO)/count(grpO)
variable vq atom q

thermo 200
thermo_style custom step temp etotal press vol v_qSi v_qO
dump viz all image 100 myimage−*.ppm q &
  type shiny 0.1 box no 0.01 view 180 90 zoom 2.3 size 1200 500
dump_modify viz adiam Si 2.6 adiam O 2.3 backcolor white &
  amap −1 2 ca 0.0 3 min royalblue 0 green max orangered

fix myhis1 grpSi ave/histo 10 500 5000 −1.5 2.5 1000 v_vq &
  file deform−Si.histo mode vector
fix myhis2 grpO ave/histo 10 500 5000 −1.5 2.5 1000 v_vq &
  file deform−O.histo mode vector
fix myspec all reaxff/species 5 1 5 deform.species element Si O
```

The only difference with the previous `relax.lmp` file is the path to the `relax.data` file.

Next, let us use `fix nvt` instead of `fix npt` to apply a Nosé-Hoover thermostat without a barostat:

```
fix mynvt all nvt temp 300.0 300.0 100
timestep 0.5
```

Here, no barostat is used because the change in the box volume will be imposed by the `fix deform`, see below.

Let us run for 5000 steps without deformation, then apply the `fix deform` to progressively elongate the box along the *x*-axis during 25000 steps. Add the following line to `deform.lmp`:

```
run 5000

fix mydef all deform 1 x erate 5e−5

run 25000

write_data deform.data nofix
```

The `fix deform` command applies a continuous deformation by elongating the simulation box along the x-axis at a constant engineering shear strain rate, specified by `erate`, of 5×10^{-5} fs^{-1}.

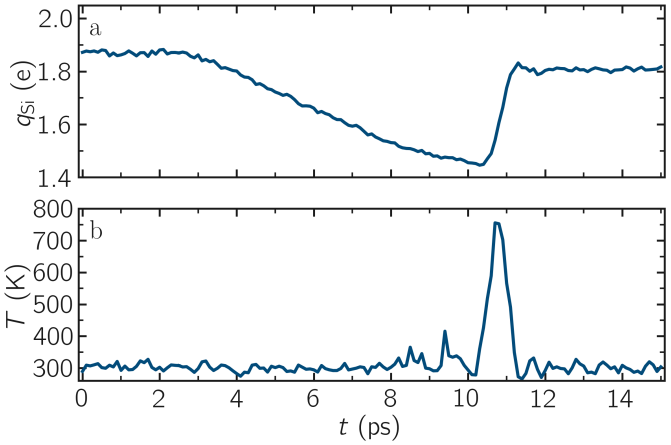

Figure 31. a) Average charge per atom of the silicon, q_{Si}, atoms as a function of time, t, during deformation of the SiO_2 system from Tutorial 5. The break down of the silica structure occurs near t = 11 ps. b) Temperature, T, of the system as a function of t.

Run the `deform.lmp` file using LAMMPS. During the deformation, the charge values progressively evolve until the structure eventually breaks down. After the structure breaks down, the charges equilibrate near new average values that differ from the initial averages (Fig. 31 a). The difference between the initial and the final charges can be explained by the presence of defects, as well as new solid/vacuum interfaces, and the fact that surface atoms typically have different charges compared to bulk atoms (Fig. 32). You can also see a sharp increase in temperature during the rupture of the material (Fig. 31 b).

You can examine the charge distribution after deformation, as well as during deformation (Fig. 30 b). As expected, the final charge distribution slightly differs from the previously calculated one. If no new species were formed during the simulation, the `deform.species` file should look like this:

```
# Timestep No_Moles No_Specs O384Si192
         5 1 1 1
(...)
# Timestep No_Moles No_Specs O384Si192
     30000 1 1 1
```

Sometimes, O_2 molecules are formed during the deformation. If this occurs, a new column `O2` appears in the `deform.species` file.

3.5.3 Decorate the surface

Under ambient conditions, some of the surface SiO_2 atoms become chemically passivated by forming covalent bonds with hydrogen (H) atoms [55]. We will add hydrogen atoms randomly to the cracked silica and observe how the system evolves. To do so, we first need to modify the previously generated data file `deform.data` and make

https://doi.org/10.33011/livecoms.6.1.3037
Living J. Comp. Mol. Sci. 2025, 6(1), 3037

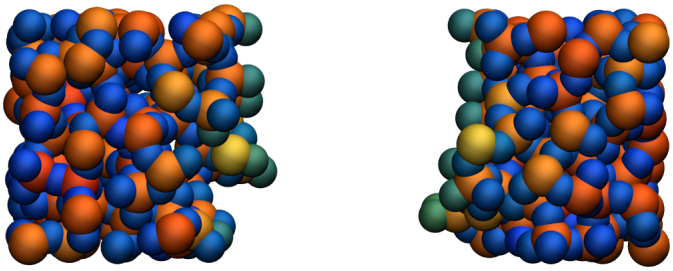

Figure 32. Amorphous silicon oxide after deformation during Tutorial 5. The atoms are colored by their charges. Dangling oxygen groups appear in greenish, bulk Si atoms with a charge of about 1.8 e appear in red/orange, and bulk O atoms with a charge of about –0.9 e appear in blue.

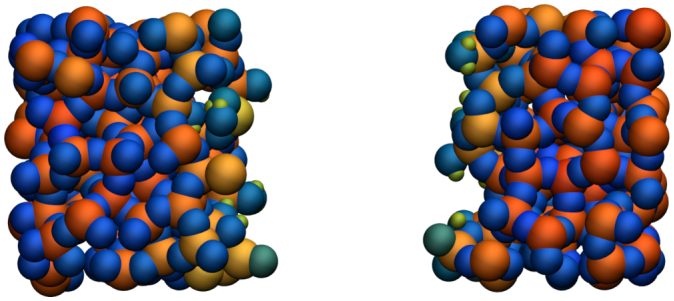

Figure 33. Cracked silicon oxide after the addition of hydrogen atoms during Tutorial 5. The atoms are colored by their charges, with the newly added hydrogen atoms appearing as small greenish spheres.

space for a third atom type. Copy `deform.data`, name the copy `deform-mod.data`, and modify the first lines of `deform-mod.data` as follows:

```
576 atoms
3 atom types

(...)

Atom Type Labels

1 Si
2 O
3 H

Masses

Si 28.0855
O 15.999
H 1.008

(...)
```

Open the `decorate.lmp` file, which must contain the following lines:

```
units real
atom_style full

read_data deform-mod.data
displace_atoms all move -12 0 0 # optional

pair_style reaxff NULL safezone 3.0 mincap 150
pair_coeff * * ffield.reax.CHOFe Si O H
fix myqeq all qeq/reaxff 1 0.0 10.0 1.0e-6 reaxff maxiter 400
```

The `displace_atoms` command is used to move the center of the crack near the center of the box. This step is optional but makes for a nicer visualization. A different value for the shift may be needed in your case, depending on the location of the crack. A difference with the previous input is that three atom types are specified in the `pair_coeff` command, i.e. `Si O H`.

Then, let us adapt some familiar commands to measure the charges of all three types of atoms, and output the charge values into log files:

```
group grpSi type Si
group grpO type O
group grpH type H
variable qSi equal charge(grpSi)/count(grpSi)
variable qO equal charge(grpO)/count(grpO)
variable qH equal charge(grpH)/(count(grpH)+1e-10)

thermo 5
thermo_style custom step temp etotal press v_qSi v_qO v_qH

dump viz all image 100 myimage-*.ppm q &
  type shiny 0.1 box no 0.01 view 180 90 zoom 2.3 size 1200 500
dump_modify viz adiam Si 2.6 adiam O 2.3 adiam H 1.0 &
  backcolor white amap -1 2 ca 0.0 3 min royalblue &
  0 green max orangered

fix myspec all reaxff/species 5 1 5 decorate.species &
  element Si O H
```

The commands above are, once again, similar to the ones of the previous script. Here, the +1e–10 was added to the denominator of the `variable qH` to avoid dividing by 0 at the beginning of the simulation, as no hydrogen atoms exists in the simulation domain yet. Finally, let us create a loop with 10 steps, and create two hydrogen atoms at random locations at every step:

```
fix mynvt all nvt temp 300.0 300.0 100
timestep 0.5

label loop
variable a loop 10

variable seed equal 35672+${a}
create_atoms 3 random 2 ${seed} NULL overlap 2.6 maxtry 50

run 2000

next a
jump SELF loop
```


https://doi.org/10.33011/livecoms.6.1.3037
Living J. Comp. Mol. Sci. 2025, 6(1), 3037

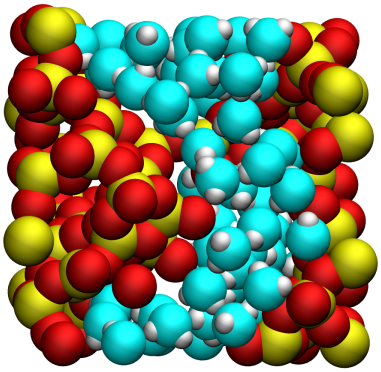

Figure 34. Water molecules (H_2O) adsorbed in cracked silica (SiO_2) material as simulated during Tutorial 6. The oxygen atoms of the water molecules are represented in cyan, the silicon atoms in yellow, and the oxygen atoms of the solid in red.

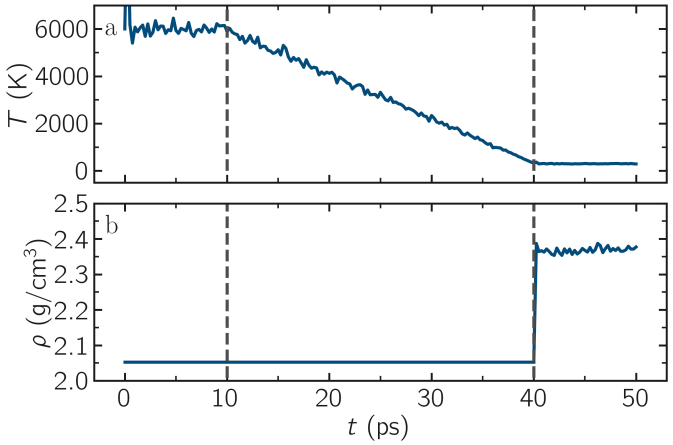

Figure 35. a) Temperature, T, as a function of time, t, during the annealing of the silica system from Tutorial 6. b) System density, ρ, during the annealing process. The vertical dashed lines mark the transition between the different phases of the simulation.

Run the simulation with LAMMPS. When the simulation is over, it can be seen from the `decorate.species` file that all the created hydrogen atoms reacted with the SiO_2 structure to form surface groups (such as hydroxyl (-OH) groups).

```
(...)
# Timestep No_Moles No_Specs H20O384Si192
  20000 1 1 1
```

At the end of the simulation, hydroxyl (-OH) groups can be seen at the interfaces (Fig. 33).

3.6 Tutorial 6: Water adsorption in silica

The objective of this tutorial is to combine molecular dynamics and grand canonical Monte Carlo simulations to compute the adsorption of water molecules in cracked silica material (Fig. 34). This tutorial illustrates the use of the grand canonical ensemble in molecular simulation, an open ensemble where the number of atoms or molecules in the simulation box can vary. By using this combination, we simulate water in a nanoporous SiO_2 structure at a specified chemical potential.

3.6.1 Generation of the silica block

To begin this tutorial, select «`Start Tutorial 6`» from the «`Tutorials`» menu of LAMMPS–GUI and follow the instructions. The editor should display the following content corresponding to `generate.lmp`:

```
units metal
boundary p p p
atom_style full
pair_style vashishta
neighbor 1.0 bin
neigh_modify delay 1
```

The main difference from some of the previous tutorials is the use of the `Vashishta` pair style. The Vashishta potential implicitly models atomic bonds through energy terms dependent on interatomic distances and angles [53].

Let us create a box for two atom types, `Si` of mass 28.0855 g/mol and `O` of mass 15.9994 g/mol. Add the following lines to `generate.lmp`:

```
region box block −18.0 18.0 −9.0 9.0 −9.0 9.0
create_box 2 box
labelmap atom 1 Si 2 O
mass Si 28.0855
mass O 15.9994
create_atoms Si random 240 5802 box overlap 2.0 maxtry 500
create_atoms O random 480 1072 box overlap 2.0 maxtry 500
```

In line with what is done in previous tutorials, the `create_atoms` commands are used to place 240 Si atoms and 480 O atoms, respectively, in the region previously defined. This corresponds to an initial density of approximately 2 g/cm^3, which is close to the expected final density of amorphous silica at 300 K.

Now, specify the potential parameters by indicating that the first atom type is `Si` and the second is `O`:

```
pair_coeff * * SiO.1990.vashishta Si O
```

Ensure that the `SiO.1990.vashishta` file is located in the same directory as `generate.lmp`.

Next, add a `dump image` command to `generate.lmp` to follow the evolution of the system with time:

```
dump viz all image 250 myimage−*.ppm type type &
  shiny 0.1 box no 0.01 view 180 90 zoom 3.4 size 1700 700
dump_modify viz backcolor white &
  acolor Si yellow adiam Si 2.5 &
  acolor O red adiam O 2
```


https://doi.org/10.33011/livecoms.6.1.3037
Living J. Comp. Mol. Sci. 2025, 6(1), 3037

Let us also print the box volume and system density, alongside the temperature and total energy:

```
thermo 250
thermo_style custom step temp etotal vol density
```

Finally, let us implement the annealing procedure which consists of three consecutive runs. This procedure was inspired by Ref. [56]. First, to melt the system, a 10 ps run at T = 6000 K is performed:

```
velocity all create 6000 8289 rot yes dist gaussian
fix mynvt all nvt temp 6000 6000 0.1
timestep 0.001
run 10000
```

Next, a second run, during which the system is cooled down from T = 6000 K to T = 300 K, is implemented as follows:

```
fix mynvt all nvt temp 6000 300 0.1
run 30000
```

In this case, the initial and final target temperatures set for the Nosé-Hoover thermostat is different, causing it to evolve linearly within the number of timesteps evoked in the `run` command. In the third run, the system is equilibrated at the final desired conditions, T = 300 K and p = 1 atm, using an anisotropic pressure coupling:

```
unfix mynvt

fix mynpt all npt temp 300 300 0.1 aniso 1 1 1
run 10000

write_data generate.data
```

Here, an anisotropic barostat is used. As previously mentioned, anisotropic barostats adjust the dimensions independently, which is generally suitable for a solid phase.

Run the simulation using LAMMPS. From the «`Charts`» window, the temperature evolution can be observed, showing that it closely follows the desired annealing procedure (Fig. 35 a). The evolution of the box dimensions over time confirms that the box is deforming during the last stage of the simulation (Fig. 35 b). After the simulation completes, the final microstate attained during the dynamics and the system topology will be written to a LAMMPS data file called `generate.data` which will be located next to `generate.lmp` (Fig. 36).

3.6.2 Cracking the silica

Open the `cracking.lmp` file, which must contain the following familiar lines:

```
units metal
boundary p p p
atom_style full
```

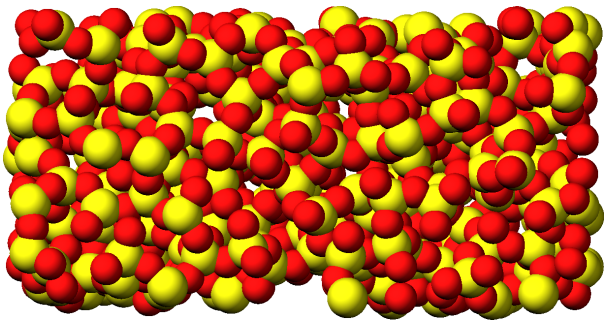

Figure 36. Amorphous silica (SiO_2) simulated during Tutorial 6. Silicon atoms are represented in yellow, and oxygen atoms in red.

```
pair_style vashishta
neighbor 1.0 bin
neigh_modify delay 1

read_data generate.data

pair_coeff * * SiO.1990.vashishta Si O

dump viz all image 250 myimage-*.ppm type type &
  shiny 0.1 box no 0.01 view 180 90 zoom 3.4 size 1700 700
dump_modify viz backcolor white &
  acolor Si yellow adiam Si 2.5 &
  acolor O red adiam O 2

thermo 250
thermo_style custom step temp etotal vol density
```

Let us progressively increase the size of the box in the x direction, forcing the silica to deform and eventually crack. To achive this, the `fix deform` command is used, with a rate of 0.005 ps^{-1}. Add the following lines to the `cracking.lmp` file:

```
timestep 0.001
fix nvt1 all nvt temp 300 300 0.1
fix mydef all deform 1 x erate 0.005
run 50000

write_data cracking.data
```

As discussed, the `fix nvt` command integrates the Nosé-Hoover equations of motion and is employed to control the temperature of the system. As observed from the generated images, the atoms progressively adjust to the changing box dimensions. At some point, bonds begin to break, leading to the appearance of dislocations (Fig. 37).

Although the Nosé-Hoover equations were originally formulated to sample the NVT ensemble, using the `fix nvt` command does not guarantee that a simulation actually samples the NVT ensemble.

https://doi.org/10.33011/livecoms.6.1.3037
Living J. Comp. Mol. Sci. 2025, 6(1), 3037

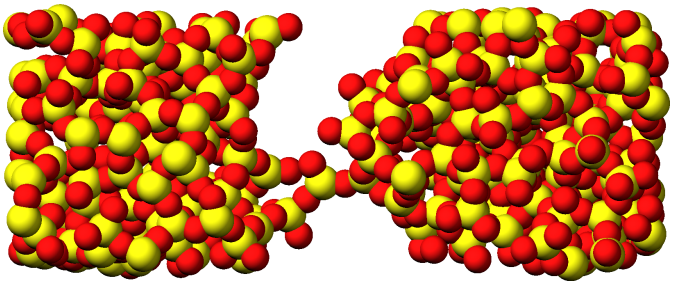

Figure 37. Block of silica from Tutorial 6 after deformation. Silicon atoms are represented in yellow, and oxygen atoms in red. The crack was induced by the imposed deformation of the box along the *x*-axis (i.e., the horizontal axis).

3.6.3 Adding water

To add the water molecules to the silica, we will employ the Monte Carlo method in the grand canonical ensemble (GCMC). In short, the system is placed into contact with a virtual reservoir containing pure water at a given thermodynamic state, and multiple attempts to insert water molecules at random positions are made. In the grand canonical ensemble, each attempt is either accepted or rejected based on internal energy and chemical potential, μ considerations. For further details, please refer to classical textbooks like Ref. 1.

Adapting the pair style

For this next step, we need to specify the force field used to model the interactions in the system. The TIP4P/2005 model is employed for the water [13], while no interaction within silica is defined, as it will be seen farther below. This is because the atoms of the silica will remain frozen during this part of the simulation. Only the cross-interactions between water and silica need to be defined. Open the `gcmc.lmp` file, which should contain the following lines:

```
units metal
boundary p p p
atom_style full
neighbor 1.0 bin
neigh_modify delay 1
pair_style lj/cut/tip4p/long OW HW OW-HW HW-OW-HW &
    0.1546 10
kspace_style pppm/tip4p 1.0e-5
bond_style harmonic
angle_style harmonic
```

The PPPM solver [38] is specified with the `kspace` command, and is used to compute the long-range Coulomb interactions associated with `tip4p/long`. Finally, the form of the bond and angle potentials of the water molecules are defined; however, as previously discussed, these specifications are not critical since TIP4P/2005 is a rigid water model.

In practice, it is possible to use both `vashishta` and `lj/cut/tip4p/long` pair styles at the same time by employing the `pair_style hybrid` command. However, hybridizing force fields should be done with caution, as there is no guarantee that the resulting force field will produce meaningful results.

The water molecule template called `H2O.mol` must be downloaded and located next to `gcmc.lmp`.

Before going further, we need to make a few changes to our data file. Currently, the `cracking.data` file includes only two atom types, but we require four. Copy the previously generated `cracking.data`, and name the duplicate `cracking-mod.data`. Make the following changes to the beginning of `cracking-mod.data` to ensure it matches the following format (with 4 atom types, 1 bond type, 1 angle type, the proper type labels, and four masses):

```
720 atoms
4 atom types
1 bond types
1 angle types

2 extra bond per atom
1 extra angle per atom
2 extra special per atom

-22.470320800269317 22.470320800269317 xlo xhi
-8.579178758211475 8.579178758211475 ylo yhi
-8.491043517346204 8.491043517346204 zlo zhi

Atom Type Labels

1 Si
2 O
3 OW
4 HW

Bond Type Labels

1 OW-HW

Angle Type Labels

1 HW-OW-HW

Masses

1 28.0855
2 15.9994
3 15.9994
4 1.008

Atoms # full

(...)
```


https://doi.org/10.33011/livecoms.6.1.3037
Living J. Comp. Mol. Sci. 2025, 6(1), 3037

Doing so, we anticipate that there will be 4 atom types in the simulations, with the oxygens and hydrogens of H_2O having types `OW` and `HW`, respectively. There will also be 1 bond type (`OW-HW`) and 1 angle type (`OW-HW-HW`). The `extra bond`, `extra angle`, and `extra special` lines are here for memory allocation.

We can now proceed to complete the `gcmc.lmp` file by adding the system definition:

```
read_data cracking-mod.data
molecule h2omol H2O.mol
create_atoms 0 random 3 3245 NULL mol h2omol 4585 &
  overlap 2.0 maxtry 50

group SiO type Si O
group H2O type OW HW
```

After reading the data file and defining the `h2omol` molecule from the `H2O.txt` file, the `create_atoms` command is used to include three water molecules in the system. Then, add the following `pair_coeff` (and `bond_coeff` and `angle_coeff`) commands to `gcmc.lmp` in order to set the potential parameters:

```
pair_coeff * * 0 0
pair_coeff Si OW 0.0057 4.42
pair_coeff O OW 0.0043 3.12
pair_coeff OW OW 0.008 3.1589
pair_coeff HW HW 0.0 0.0
bond_coeff OW-HW 0 0.9572
angle_coeff HW-OW-HW 0 104.52
```

Pair coefficients for the `lj/cut/tip4p/long` pair style are defined between O(H_2O) and between H(H_2O) atoms, as well as between O(SiO_2)-O(H_2O) and Si(SiO_2)-O(H_2O). Thus, the fluid-fluid and the fluid-solid interactions will be adressed with by the `lj/cut/tip4p/long` potential. The `bond_coeff` and `angle_coeff` commands set the `OW-HW` bond length to 0.9572 Å, and the `HW-OW-HW` angle to 104.52°, respectively [13].

> The pair coefficients for interactions between Si(SiO_2) and O(SiO_2) are set by the first command, `pair_coeff * * 0 0`, which effectively means that they do not interact. This is acceptable here because the silica atoms remain frozen during this part of the tutorial.

Add the following lines to `gcmc.lmp` as well:

```
variable oxygen atom type==label2type(atom,OW)
group oxygen dynamic all var oxygen
variable nO equal count(oxygen)

fix shak H2O shake 1.0e-5 200 0 b OW-HW &
  a HW-OW-HW mol h2omol
```

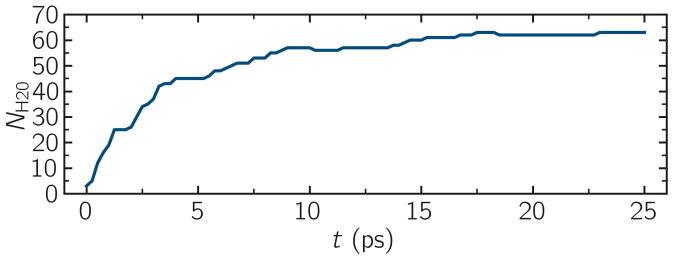

Figure 38. Number of water molecules, N_{H2O}, as a function of time, t, as extracted from Tutorial 6.

The number of oxygen atoms from water molecules (i.e. the number of molecules) is calculated by the `nO` variable. As already discussed in other tutorials, the SHAKE algorithm is used to maintain the shape of the water molecules over time [47, 48].

> Here, a variable of style 'atom' is used. Such variable defines a per-atom property, i.e., it evaluates the specified expression separately for each atom. This is often used to select atoms based on their properties or types.

Finally, let us create images of the system using `dump image`:

```
dump viz all image 250 myimage-*.ppm type type &
  shiny 0.1 box no 0.01 view 180 90 zoom 3.4 size 1700 700
dump_modify viz backcolor white &
  acolor Si yellow adiam Si 2.5 &
  acolor O red adiam O 2 &
  acolor OW cyan adiam OW 2 &
  acolor HW white adiam HW 1
```

GCMC simulation

To prepare for the GCMC simulation, let us add the following lines into `gcmc.lmp`:

```
compute ctH2O H2O temp
compute_modify thermo_temp dynamic/dof yes
compute_modify ctH2O dynamic/dof yes
fix mynvt H2O nvt temp 300 300 0.1
fix_modify mynvt temp ctH2O
timestep 0.001
```

Here, the `fix nvt` applies only to the water molecules, so the atoms in the silica remain fixed. The `compute_modify` command with the `dynamic/dof yes` option is used for water to account for the fact that the number of molecules is not constant.

Finally, let us use the `fix gcmc` and perform the grand canonical Monte Carlo steps. Add the following lines into `gcmc.lmp`:

```
variable tfac equal 5.0/3.0
fix fgcmc H2O gcmc 100 100 0 0 65899 300 -0.5 0.1 &
```


https://doi.org/10.33011/livecoms.6.1.3037
Living J. Comp. Mol. Sci. 2025, 6(1), 3037

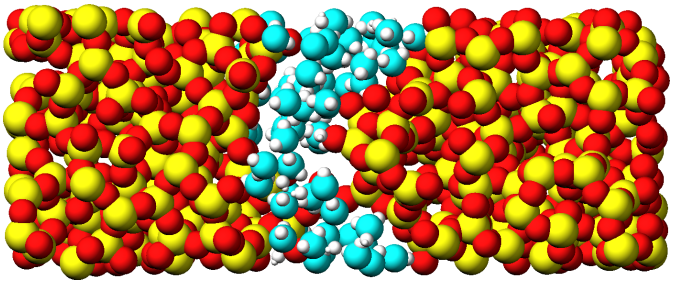

Figure 39. Snapshot of the silica system after the adsorption of water molecules during Tutorial 6. The oxygen atoms of the water molecules are represented in cyan, the silicon atoms in yellow, and the oxygen atoms of the solid in red.

```
  mol h2omol tfac_insert ${tfac} shake shak &
  full_energy pressure 100
```

The `fix gcmc` command performs grand canonical Monte Carlo moves to insert, delete, or swap molecules. Here, 100 attempts are made every 100 steps. The `mol h2omol` keyword specifies the molecule type being inserted/deleted, while `shake shak` enforces rigid molecular constraints during these moves. With the `pressure 100` keyword, a fictitious reservoir with a pressure of 100 atmospheres is used. The `tfac_insert` option ensures the correct estimate for the temperature of the inserted water molecules by taking into account the internal degrees of freedom.

At a pressure of p = 100 bar, the chemical potential of water vapor at T = 300 K can be calculated using as $\mu = \mu_0 + RT \ln(\frac{p}{p_0})$, where μ_0 is the standard chemical potential (typically taken at a pressure p_0 = 1 bar), R = 8.314 J/mol·K is the gas constant, T = 300 K is the temperature.

Finally, let us print some information and run for 25 ps:

```
thermo 250
thermo_style custom step temp etotal v_nO &
  f_fgcmc[3] f_fgcmc[4] f_fgcmc[5] f_fgcmc[6]

run 25000
```

The `f_` keywords extract the Monte Carlo move statistics which is computed (and can be extracted) by the `fix gcmc` command.

When using the pressure argument, LAMMPS ignores the value of the chemical potential (here μ = –0.5 eV, which corresponds roughly to ambient conditions, i.e. to a relative humidity RH ≈ 50 % [57].) The large pressure value of 100 bars was chosen to ensure that some successful insertions of molecules would occur during the short duration of this simulation.

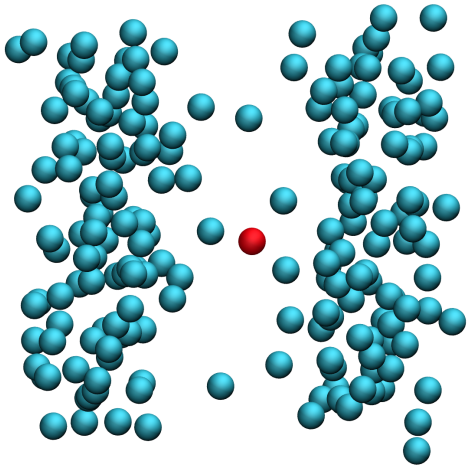

Figure 40. System simulated during Tutorial 7. The pink atom explores the energetically unfavorable central area of the simulation box thanks to the additional potential imposed during umbrella sampling.

Running this simulation using LAMMPS, one can see that after a few GCMC steps, the number of molecules starts increasing. Once the crack is filled with water molecules, the total number of molecules reaches a plateau (Figs. 38-39). The final number of molecules depends on the imposed pressure, temperature, and the interaction between water and silica (i.e. its hydrophilicity). Note that GCMC simulations of such dense phases are usually slow to converge due to the very low probability of successfully inserting a molecule. Here, the short simulation duration was made possible by the use of a high pressure.

3.7 Tutorial 7: Free energy calculation

The objective of this tutorial is to measure the free energy profile of particles through a barrier potential using two methods: free sampling and umbrella sampling [1, 2, 58] (Fig. 40). To simplify the process and minimize computation time, the barrier potential will be imposed on the atoms using an additional force, mimicking the presence of a repulsive area in the middle of the simulation box without needing to simulate additional atoms. The procedure is valid for more complex systems and can be adapted to many other situations, such as measuring adsorption barriers near an interface or calculating translocation barriers through a membrane [59--63].

3.7.1 Method 1: Free sampling

The most direct way to estimate a free energy profile is to sample the Boltzmann distribution using a classical (i.e. unbiased) molecular dynamics simulation, and compute relative Gibbs free energies from the relative probabilities of states

https://doi.org/10.33011/livecoms.6.1.3037
Living J. Comp. Mol. Sci. 2025, 6(1), 3037

using

$$\Delta G = -RT \ln(\rho/\rho_0), \tag{3}$$

where ΔG is the free energy difference, R is the gas constant, T is the temperature, ρ is the density, and ρ_0 is a reference density. As an illustration, let us apply this method to a simple system that consists of a particles in a box in the presence of a position-dependent repulsive force that makes the center of the box a less favorable area to explore.

Basic LAMMPS parameters

To begin this tutorial, select «Start Tutorial 7» from the «Tutorials» menu of LAMMPS–GUI and follow the instructions. The editor should display the following content corresponding to `free-sampling.lmp`:

```
variable sigma equal 3.405
variable epsilon equal 0.238
variable U0 equal 1.5*${epsilon}
variable dlt equal 1.0
variable x0 equal 10.0

units real
atom_style atomic
pair_style lj/cut $(2^(1/6)*v_sigma)
pair_modify shift yes
boundary p p p
```

Here, we begin by defining variables for the Lennard-Jones parameters (σ and ϵ) and for the repulsive potential parameters U, which are U_0, δ, and x_0 [see Eqs. (4-5) below]. The cut-off value of $2^{1/6}\sigma = 3.822$ was chosen to create a Weeks-Chandler-Andersen (WCA) potential, which is a truncated and purely repulsive LJ potential [64]. The potential is also shifted to be equal to 0 at the cut-off using the `pair_modify` command. The unit system is `real`, in which energy is in kcal/mol, distance in Ångströms, or time in femtosecond, has been chosen for practical reasons: the WHAM algorithm used in the second part of the tutorial automatically assumes the energy to be in kcal/mol.

> The syntax `$(...)`, where a dollar sign is followed by parentheses, allows you to evaluate a numeric formula immediately, without having to assign it to a named variable first.

System creation and settings

Let us define the simulation box and randomly add atoms by addying the following lines to `free-sampling.lmp`:

```
region myreg block -50 50 -15 15 -50 50
create_box 1 myreg
create_atoms 1 random 200 34134 myreg overlap 3 maxtry 50

mass * 39.95
pair_coeff * * ${epsilon} ${sigma}
```

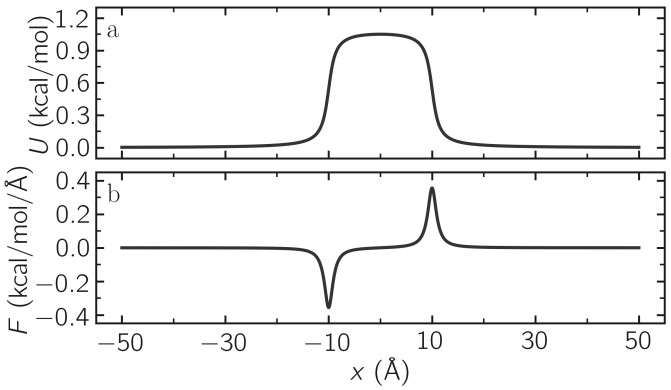

Figure 41. Potential U given in Eq. (4) (a) and force F given in Eq. (5) (b) as functions of the coordinate x. Here, $U_0 = 0.36$ kcal/mol, $\delta = 1.0$ Å, and $x_0 = 10$ Å.

> In the `pair_coeff` command, the first two asterisks `* *` indicate that the parameters apply to all atom types in the simulation.

The variables U_0, δ, and x_0, defined in the previous subsection, are used here to create the repulsive potential, restricting the atoms from exploring the center of the box:

$$U = U_0 \left[\arctan\left(\frac{x+x_0}{\delta}\right) - \arctan\left(\frac{x-x_0}{\delta}\right)\right]. \tag{4}$$

Taking the derivative of the potential with respect to x, we obtain the expression for the force that will be imposed on the atoms:

$$F = \frac{U_0}{\delta}\left[\frac{1}{(x-x_0)^2/\delta^2+1} - \frac{1}{(x+x_0)^2/\delta^2+1}\right]. \tag{5}$$

Fig. 41 shows the potential U and force F along the x-axis. With $U_0 = 1.5\epsilon = 0.36$ kcal/mol, U_0 is of the same order of magnitude as the thermal energy $k_BT = 0.24$ kcal/mol, where $k_B = 0.002$ kcal/mol/K is the Boltzmann constant and $T = 119.8$ K is the temperature used in this simulation. Under these conditions, particles are expected to frequently overcome the energy barrier due to thermal agitation.

We impose the force $F(x)$ to the atoms in the simulation using the `fix addforce` command. Add the following lines to `free-sampling.lmp`:

```
variable U atom ${U0}*atan((x+${x0})/${dlt})&
    -${U0}*atan((x-${x0})/${dlt})
variable F atom ${U0}/((x-${x0})^2/${dlt}^2+1)/${dlt}&
    -${U0}/((x+${x0})^2/${dlt}^2+1)/${dlt}
fix myadf all addforce v_F 0.0 0.0 energy v_U
```

Next, we use the Newtonian equations of motion with a Langevin thermostat by combining the `fix nve` with a `fix langevin` command:

```
fix mynve all nve
fix mylgv all langevin 119.8 119.8 500 30917
```


https://doi.org/10.33011/livecoms.6.1.3037
Living J. Comp. Mol. Sci. 2025, 6(1), 3037

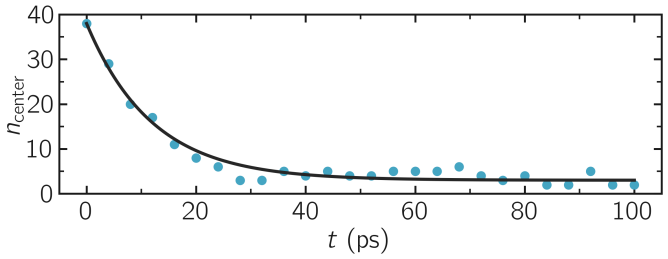

Figure 42. Evolution of the number of atoms n_{center} in the central region `mymes` as a function of time t during equilibration. The dark line is $n_{center} = 22\exp(-t/160) + 5$ and serves as a guide for the eyes. Here, U_0 = 0.36 kcal/mol, δ = 1.0 Å, and x_0 = 10 Å.

When combining these two commands, the MD simulation operates in the NVT ensemble, maintaining a constant number of atoms N, constant volume V, and a temperature T that fluctuates around a target value.

> LAMMPS documentation suggests using damping constants for thermostats that are approximately 100 times the timestep value. In this case, a value of 500 is used, resulting in a relatively weak coupling to the thermostat.

To ensure that the equilibration time is sufficient, we will track the evolution of the number of atoms in the central - energetically unfavorable - region, defined below under the name `mymes`, using the `n_center` variable:

```
region mymes block −${x0} ${x0} INF INF INF INF
variable n_center equal count(all,mymes)
thermo_style custom step temp etotal v_n_center
thermo 10000

dump viz all image 5000 myimage−*.ppm type type &
     shiny 0.1 box yes 0.01 view 180 90 zoom 6 &
     size 1600 500 fsaa yes
dump_modify viz backcolor white acolor 1 cyan &
     adiam 1 3 boxcolor black
```

A `dump image` command was also added for system visualization. The other commands should also be familiar from previous tutorials.
Finally, let us perform an equilibration of 50000 steps, using a timestep of 2 fs, corresponding to a total duration of 100 ps:

```
timestep 2.0
run 50000
```

Run the simulation with LAMMPS. The number of atoms in the central region, n_{center}, reaches its equilibrium value after approximately 40 ps (Fig. 42). A snapshot of the equilibrated system is shown in Fig. 43.

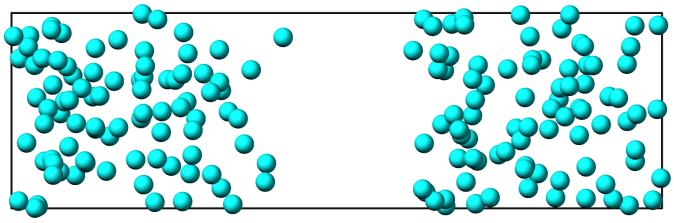

Figure 43. Snapshot of the system simulated during the free sampling step of Tutorial 7. The atoms density is the lowest in the central part of the box, `mymes`. Here, U_0 = 0.36 kcal/mol, δ = 1.0 Å, and x_0 = 10 Å.

Run and data acquisition

Once the system is equilibrated, we will record the density profile of the atoms along the x-axis using the `ave/chunk` command. Add the following line to `free-sampling.lmp`:

```
reset_timestep 0

thermo 200000

compute cc1 all chunk/atom bin/1d x 0.0 2.0
fix myac all ave/chunk 100 20000 2000000 &
     cc1 density/number file free−sampling.dat

run 2000000
```

Here, the `chunk/atom` command discretizes the simulation domain into spatial bins of size 2 Å along the x direction, and the `fix ave/chunk` command outputs the number density of atoms within each bin to the file `free-sampling.dat`. The step count is reset to 0 using `reset_timestep` to synchronize it with the output times of `fix ave/chunk`. Run the simulation using LAMMPS.

Data analysis

Once the simulation is complete, the density profile from `free-sampling.dat` shows that the density in the center of the box is about two orders of magnitude lower than inside the reservoir (Fig. 44 a). Next, we plot $-RT\ln(\rho/\rho_{bulk})$, where ρ/ρ_{bulk} is the the density ratio, and compare it with the imposed potential U from Eq. (4) (Fig. 44 b). The reference density, ρ_{bulk} = 0.0009 Å^{-3}, was estimated by measuring the density of the reservoir from the density profiles. The agreement between the MD results and the imposed energy profile is excellent, despite some noise in the central part, where fewer data points are available due to the repulsive potential.

The limits of free sampling

Increasing the value of U_0 reduces the average number of atoms in the central region, making it difficult to achieve a high-resolution free energy profile within reasonable simulation times. For example, running the same simulation with

https://doi.org/10.33011/livecoms.6.1.3037
Living J. Comp. Mol. Sci. 2025, 6(1), 3037

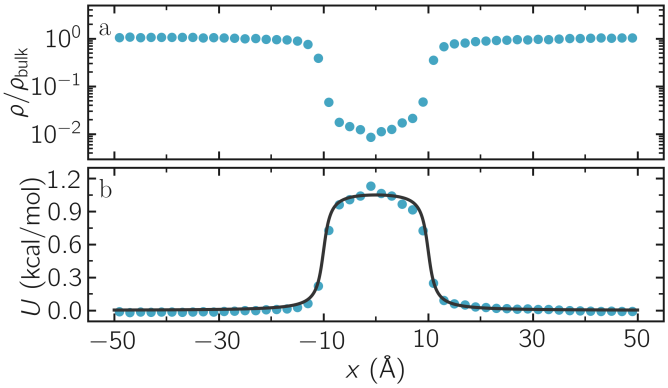

Figure 44. a) Fluid density, ρ, along the x direction. b) Potential, U, as a function of x measured using free sampling (blue disks) compared to the imposed potential given in Eq. (4) (dark line). Here, U_0 = 0.36 kcal/mol, δ = 1.0 Å, x_0 = 10 Å, and the measured reference density in the reservoir is ρ_{bulk} = 0.0009 Å^{-3}.

$U_0 = 10\epsilon$, corresponding to $U_0 \approx 10k_BT$, results in no atoms exploring the central part of the simulation box during the simulation. In such a case, employing an enhanced sampling method is recommended, as done in the next section.

3.7.2 Method 2: Umbrella sampling

Umbrella sampling is a biased molecular dynamics method in which additional forces are added to a chosen atom to force it to explore the more unfavorable areas of the system [1, 2, 58]. Here, to encourage one of the atoms to explore the central region of the box, we apply a potential V and force it to move along the x-axis. The chosen path is called the axis of reaction. Several simulations (called windows) will be conducted with varying positions for the center of the applied biasing. The results will be analyzed using the weighted histogram analysis method (WHAM) [65, 66], which allows for the removal of the biasing effect and ultimately deduces the unbiased free energy profile.

LAMMPS input script

Open the file named `umbrella-sampling.lmp`, which should contain the following lines:

```
variable sigma equal 3.405
variable epsilon equal 0.238
variable U0 equal 10*${epsilon}
variable dlt equal 1.0
variable x0 equal 10
variable k equal 0.5

units real
atom_style atomic
pair_style lj/cut $(2^(1/6)*v_sigma)
pair_modify shift yes
boundary p p p
```

The first difference from the previous case is the larger value for the repulsive potential parameter U_0, which makes the central area of the system very unlikely to be visited by free particles. The second difference is the introduction of the variable k, which will be used for the biasing potential.

Let us create a simulation box with two atom types, including a single particle of type 2, by adding the following lines to `umbrella-sampling.lmp`:

```
region myreg block −50 50 −15 15 −50 50
create_box 2 myreg
create_atoms 2 single 0 0 0
create_atoms 1 random 199 34134 myreg overlap 3 maxtry 50
```

Next, we assign the same mass and LJ parameters to both atom types 1 and 2, and place the atoms of type 2 into a group named `topull`:

```
mass * 39.948
pair_coeff * * ${epsilon} ${sigma}
group topull type 2
```

Then, the same potential U and force F are applied to all the atoms, together with the same `fix nve` and `fix langevin` commands:

```
variable U atom ${U0}*atan((x+${x0})/${dlt})&
    −${U0}*atan((x−${x0})/${dlt})
variable F atom ${U0}/((x−${x0})^2/${dlt}^2+1)/${dlt}&
    −${U0}/((x+${x0})^2/${dlt}^2+1)/${dlt}
fix myadf all addforce v_F 0.0 0.0 energy v_U

fix mynve all nve
fix mylgv all langevin 119.8 119.8 500 30917
```

Next, we perform a brief equilibration to prepare for the umbrella sampling run:

```
thermo 5000

dump viz all image 5000 myimage−*.ppm type type &
    shiny 0.1 box yes 0.01 view 180 90 zoom 6 &
    size 1600 500 fsaa yes
dump_modify viz backcolor white acolor 1 cyan &
acolor 2 red adiam 1 3 adiam 2 3 boxcolor black

timestep 2.0
run 50000
```

So far, our code resembles that of Method 1, except for the additional particle of type 2. Particles of types 1 and 2 are identical. However, the particle of type 2 will also be exposed to the biasing potential V, which forces it to explore the central part of the box (Fig. 45), thus justifying the definition of two atom types.

Now, we create a loop with 15 steps and progressively move the center of the bias potential by increments of 0.4 nm. Add the following lines to `umbrella-sampling.lmp`:

```
variable a loop 15
label loop
```


https://doi.org/10.33011/livecoms.6.1.3037
Living J. Comp. Mol. Sci. 2025, 6(1), 3037

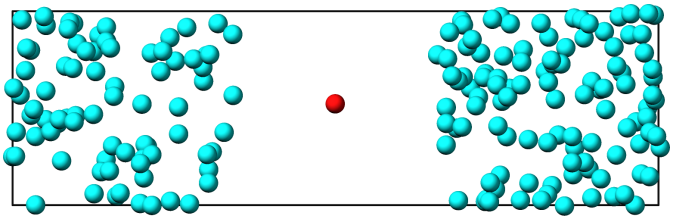

Figure 45. Snapshot of the system simulated during the umbrella sampling step of Tutorial 7, showing type-1 atoms in cyan and the type-2 atom in red. Only the type-2 atom explores the central part of the box, `mymes`, due to the additional biasing potential V. Parameters are U_0 = 2.38 kcal/mol, δ = 1.0 Å, and x_0 = 10 Å.

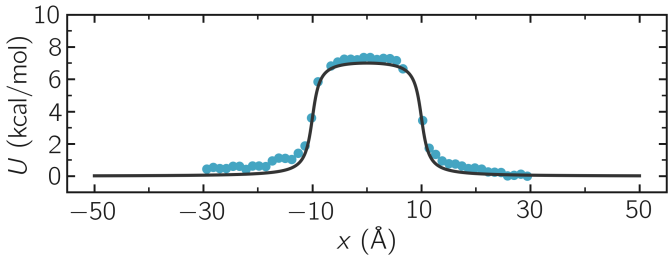

Figure 46. The potential, U, as a function of x, measured using umbrella sampling during Tutorial 7 (blue disks), is compared to the imposed potential given in Eq. (4) (dark line). Parameters are U_0 = 2.38 kcal/mol, δ = 1.0 Å, and x_0 = 10 Å.

```
variable xdes equal 4*${a}-32
variable xave equal xcm(topull,x)
fix mytth topull spring tether ${k} ${xdes} 0 0 0

run 20000

fix myat1 all ave/time 10 10 100 &
    v_xave v_xdes file umbrella-sampling.${a}.dat

run 200000
unfix myat1
next a
jump SELF loop
```

The definition of a variable of loop style serves the same purpose as in (Tutorial 5), and we highlight here the particular utility of using its value to distinguish the files written by the `fix ave_time` command for the different bias potentials. The `spring` command imposes the additional harmonic potential V with the previously defined spring constant k to the atoms in the group `topull`. The center of the harmonic potential, x_{des}, successively takes values from –28 Å to 28 Å. For each value of x_{des}, an equilibration step of 40 ps is performed, followed by a step of 400 ps during which the position of the particle of type 2 along the x-axis, x_{ave}, is saved in data files named `umbrella-sampling.i.dat`, where i ranges from 1 to 15. Run the `umbrella-sampling.lmp` file using LAMMPS.

The value of k should be chosen with care: if k is too small the particle won't follow the biasing potential, and if k is too large there will be no overlapping between the different windows, leading to poor reconstruction of the free energy profile.

WHAM algorithm

To generate the free energy profile from the particle positions saved in the `umbrella-sampling.i.dat` files, we use the WHAM [65, 66] algorithm as implemented by Alan Grossfield [67]. You can download it from Alan Grossfield's website. Make sure you download the WHAM code version 2.1.0 or later which introduces the `units` command-line option used below. The executable called `wham` generated by following the instructions from the website must be placed next to `umbrella-sampling.lmp`. To apply the WHAM algorithm to our simulation, we need a metadata file containing:

- the paths to all the data files,
- the values of x_{des},
- the values of k.

Download the `umbrella-sampling.meta` file and save it next to `umbrella-sampling.lmp`. Then, run the WHAM algorithm by typing the following command in the terminal:

```
./wham units real -30 30 50 1e-8 119.8 0 \
    umbrella-sampling.meta umbrella-sampling.dat
```

where -30 and 30 are the boundaries, 50 is the number of bins, 1e-8 is the tolerance, and 119.8 is the temperature in Kelvin. A file called `umbrella-sampling.dat` is created, containing the free energy profile in kcal/mol. The resulting PMF can be compared with the imposed potential U, showing excellent agreement (Fig. 46). Remarkably, this excellent agreement is achieved despite the very short calculation time and the high value for the energy barrier. Achieving similar results through free sampling would require performing extremely long and computationally expensive simulations.

3.8 Tutorial 8: Reactive Molecular Dynamics

The goal of this tutorial is to create a model of a carbon nanotube (CNT) embedded in a polymer melt made of polystyrene (PS) (Fig. 47). The REACTER protocol is used to simulate the polymerization of styrene monomers, and the polymerization reaction is followed in time [15, 68, 69]. In contrast with AIREBO (Tutorial 2) and ReaxFF (Tutorial 5), the REACTER protocol relies on the use of a *classical* force field that does not inherently model bond formation or breaking, but instead couples with an external algorithm to simulate polymerization reactions.

https://doi.org/10.33011/livecoms.6.1.3037
Living J. Comp. Mol. Sci. 2025, 6(1), 3037

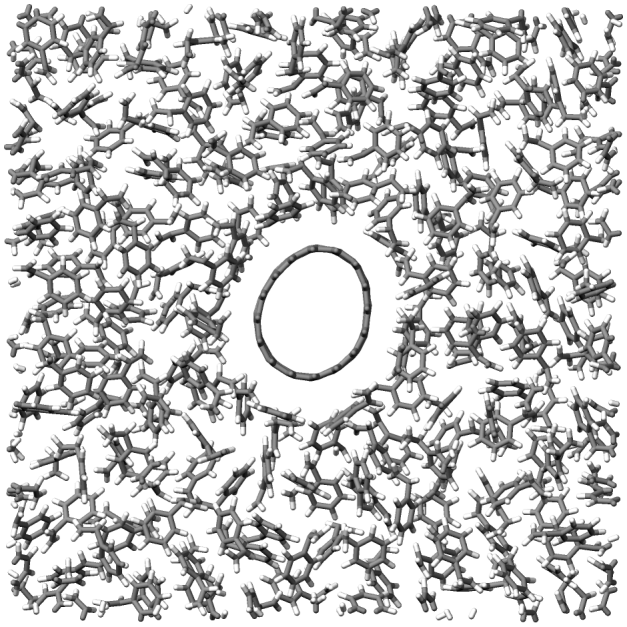

Figure 47. Initial configuration for Tutorial 8. The system consists of 200 styrene molecules packed around a single-walled CNT, with a mass density for the whole system of 0.9 g/cm^3.

3.8.1 Creating the system

To begin this tutorial, select «Start Tutorial 8» from the «Tutorials» menu of LAMMPS–GUI and follow the instructions. The editor should display the following content corresponding to `mixing.lmp`:

```
units real
boundary p p p
atom_style full

kspace_style pppm 1.0e−5
pair_style lj/class2/coul/long 8.5
angle_style class2
bond_style class2
dihedral_style class2
improper_style class2

pair_modify tail yes mix sixthpower
special_bonds lj/coul 0 0 1
```

The `class2` `pair_styles` compute a 6/9 Lennard-Jones potential [70]. The *class2* bond, angle, dihedral, and improper styles are used as well, see the documentation for a description of the respective potential form they, each, prescribe. The `tail yes` option adds long-range van der Waals tail corrections to the energy and pressure. The `mix sixthpower` imposes the following mixing rule for the calculation of the cross coefficients:

$$\sigma_{ij} = 2^{-1/6}(\sigma_i^6 + \sigma_j^6)^{1/6}, \text{ and}$$
$$\epsilon_{ij} = \frac{2\sqrt{\epsilon_i\epsilon_j}\sigma_i^3\sigma_j^3}{\sigma_i^6 + \sigma_j^6}.$$

Let us read the `CNT.data` file, which contains a periodic single-walled CNT. Add the following line to `mixing.lmp`:

```
read_data CNT.data extra/special/per/atom 20
```

The CNT is approximately 1.1 nm in diameter and 1.6 nm in length, oriented along the *x*-axis. The simulation box is initially 12.0 nm in the two other dimensions before densification, making it straightforward to fill the box with styrene. To add 200 styrene molecules to the simulation box, we will use the `styrene.mol` molecule template file. Include the following commands to `mixing.lmp`:

```
molecule styrene styrene.mol
create_atoms 0 random 200 8305 NULL overlap 2.75 &
    maxtry 500 mol styrene 7687
```

Finally, let us use the `minimize` command to reduce the potential energy of the system:

```
minimize 1.0e−4 1.0e−6 100 1000
reset_timestep 0
```

These commands were covered in earlier tutorials and should already be familiar.

Then, let us densify the system to a target value of 0.9 g/cm^3 by imposing the shrinking of the simulation box at a constant rate. The dimension parallel to the CNT axis is maintained fixed because the CNT is periodic in that direction. Add the following commands to `mixing.lmp`:

```
velocity all create 530 9845 dist gaussian rot yes
fix mynvt all nvt temp 530 530 100

fix mydef all deform 1 y erate −0.0001 z erate −0.0001
variable rho equal density
fix myhal all halt 10 v_rho > 0.9 error continue

thermo 200
thermo_style custom step temp pe etotal press density

run 9000
```

The `fix halt` command is used to stop the box shrinkage once the target density is reached, and the other commands should be familiar from previous tutorials.

For the next stage of the simulation, we will use `dump image` to output images every 200 steps:

```
dump viz all image 200 myimage−*.ppm &
  type type shiny 0.1 box no 0.01 size 1000 1000 &
```


https://doi.org/10.33011/livecoms.6.1.3037
Living J. Comp. Mol. Sci. 2025, 6(1), 3037

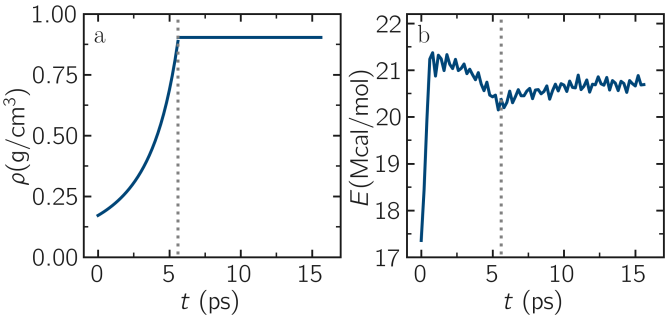

Figure 48. a) Evolution of the density, ρ, as a function of the time, t, during equilibration of the system from Tutorial 8. b) Evolution of the total energy, E, of the system. The vertical dashed lines mark the transition between the different phases of the simulation.

```
  view 90 0 zoom 1.8 fsaa yes bond atom 0.5
dump_modify viz backcolor white &
  acolor cp gray acolor c=1 gray &
  acolor c= gray acolor c1 deeppink &
  acolor c2 deeppink acolor c3 deeppink &
  adiam cp 0.3 adiam c=1 0.3 &
  adiam c= 0.3 adiam c1 0.3 &
  adiam c2 0.3 adiam c3 0.3 &
  acolor hc white adiam hc 0.15
```

For the following 10 ps, let us equilibrate the densified system in the constant-volume ensemble, and write the final state of the system in a file named `mixing.data`:

```
unfix mydef
unfix myhal
reset_timestep 0

group CNT molecule 1
fix myrec CNT recenter NULL 0 0 units box shift all

run 10000

write_data mixing.data
```

For visualization purposes, the atoms of the CNT `group` are moved to the center of the box using `fix recenter`. As the time progresses, the system density, ρ, gradually converges toward the target value of 0.9 g/cm^3 (Fig. 48 a). Meanwhile, the total energy of the system initially evolves rapidly, reflecting the densification process, and then eventually stabilizes (Fig. 48 b). The final state is shown in Fig. 47.

3.8.2 Reaction templates

The REACTER protocol enables the modeling of chemical reactions using classical force fields. The user must provide a molecule template for the reactants, a molecule template for the products, and a `reaction map` file that provides an atom mapping between the two templates. The reaction map file also includes additional information, such as which atoms act as initiators for the reaction and which serve as edge atoms to connect the rest of a long polymer chain in the simulation.

There are three reactions to define: (1) the polymerization of two styrene monomers, (2) the addition of a styrene monomer to the end of a growing polymer chain, and (3) the linking of two polymer chains. Download the three files associated with each reaction. The first reaction uses the prefix 'M-M' for the pre-reaction template, post-reaction template, and reaction map file:

- `M-M_pre.mol`,
- `M-M_post.mol`,
- `M-M.rxnmap`.

The second reaction uses the prefix 'M-P',

- `M-P_pre.mol`,
- `M-P_post.mol`,
- `M-P.rxnmap`.

The third reaction uses the prefix 'P-P',

- `P-P_pre.mol`,
- `P-P_post.mol`,
- `P-P.rxnmap`.

Here, the file names for each reaction use the abbreviation 'M' for monomer and 'P' for polymer.

The data stored in molecule templates include atom coordinates, partial charges, molecule IDs, atom types, and interaction types for bonds, angles, dihedrals and impropers. The map files contain information about the reaction. The first mandatory section of the map files begins with the keyword "InitiatorIDs" and lists the two atom IDs of the initiator atom pair in the pre-reacted molecule template. The second mandatory section begins with the keyword "Equivalences" and lists a one-to-one correspondence between atom IDs of the pre- and post-reacted templates. Some atoms in the pre-reacted template that are not reacting may have missing topology with respect to the simulation. For example, the pre-reacted template may contain an atom that, in the simulation, is currently connected to the rest of a long polymer chain. These are referred to as edge atoms, and are also specified in the map file in the "EdgeIDs" section.

3.8.3 Simulating the reaction

The first step, before simulating the reaction, is to import the previously generated configuration. Open the file named `polymerize.lmp`, which should contain the following lines:

```
units real
boundary p p p
atom_style full
```


https://doi.org/10.33011/livecoms.6.1.3037
Living J. Comp. Mol. Sci. 2025, 6(1), 3037

```
kspace_style pppm 1.0e-5
pair_style lj/class2/coul/long 8.5
angle_style class2
bond_style class2
dihedral_style class2
improper_style class2

pair_modify tail yes mix sixthpower
special_bonds lj/coul 0 0 1

read_data mixing.data &
  extra/bond/per/atom 5 &
  extra/angle/per/atom 15 &
  extra/dihedral/per/atom 15 &
  extra/improper/per/atom 25 &
  extra/special/per/atom 25
```

Here, the `read_data` command is used to import the previously generated `mixing.data` file. All other commands have been introduced in earlier parts of the tutorial.

Then, let us import all six molecules templates using the `molecule` command:

```
molecule mol1 M-M_pre.mol
molecule mol2 M-M_post.mol
molecule mol3 M-P_pre.mol
molecule mol4 M-P_post.mol
molecule mol5 P-P_pre.mol
molecule mol6 P-P_post.mol
```

In order to follow the evolution of the reaction with time, let us generate images of the system using `dump image`:

```
dump viz all image 200 myimage-*.ppm &
  type type shiny 0.1 box no 0.01 size 1000 1000 &
  view 90 0 zoom 1.8 fsaa yes bond atom 0.5
dump_modify viz backcolor white &
  acolor cp gray acolor c=1 gray &
  acolor c= gray acolor c1 deeppink &
  acolor c2 gray acolor c3 deeppink &
  adiam cp 0.3 adiam c=1 0.3 &
  adiam c= 0.3 adiam c1 0.3 &
  adiam c2 0.3 adiam c3 0.3 &
  acolor hc white adiam hc 0.15
```

Let us use `fix bond/react` by adding the following line to `polymerize.lmp`:

```
fix rxn all bond/react &
  stabilization yes statted_grp 0.03 &
  react R1 all 1 0 3.0 mol1 mol2 M-M.rxnmap &
  react R2 all 1 0 3.0 mol3 mol4 M-P.rxnmap &
  react R3 all 1 0 5.0 mol5 mol6 P-P.rxnmap
```

With the `stabilization` keyword, the `fix bond/react` command will stabilize the atoms involved in the reaction using the `fix nve/limit` command with a maximum displacement of 0.03 Å. The `fix nve/limit` command functions similar to `fix nve`, but restricts how far atoms can move in a single time step, even with very large forces. By default, each reaction is stabilized for 60 time steps. Each `react` keyword corresponds to a reaction, e.g., a transformation of `mol1` into `mol2`. Implementation details about each reaction, such as the reaction distance cutoffs and the frequency with which to search for reaction sites, are also specified in this command.

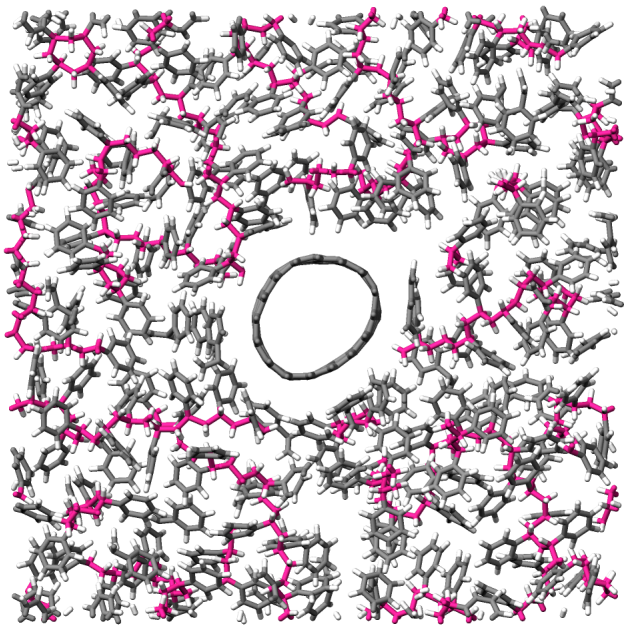

Figure 49. Final configuration for Tutorial 8. The atoms from the formed polymer named `c1`, `c2`, and `c3` are colored in pink.

The command `fix bond/react` creates several groups of atoms that are dynamically updated to track which atoms are being stabilized and which atoms are undergoing dynamics with the system-wide time integrator (here, `fix nvt`). When reaction stabilization is employed, there should not be a time integrator acting on the group `all`. Instead, the group of atoms not currently undergoing stabilization is named by appending '_REACT' to the user-provided prefix.

Add the following commands to `polymerize.lmp` to carry out the dynamics using a Nosé-Hoover thermostat while ensuring that the CNT remains centered in the simulation box:

```
fix mynvt statted_grp_REACT nvt temp 530 530 100
group CNT molecule 1 2 3
fix myrec CNT recenter NULL 0 0 shift all

thermo 1000
thermo_style custom step temp press density f_rxn[*]
```


https://doi.org/10.33011/livecoms.6.1.3037
Living J. Comp. Mol. Sci. 2025, 6(1), 3037

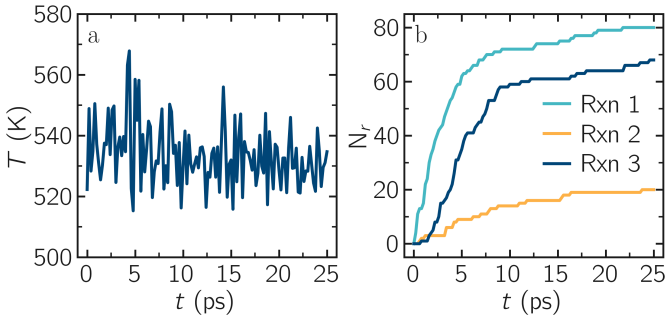

Figure 50. a) Evolution of the system temperature, *T*, as a function of the time, *t*, during the polymerization step of Tutorial 8. b) Evolution of the three reaction counts, corresponding respectively to the polymerization of two styrene monomers (Rxn 1), the addition of a styrene monomer to the end of a growing polymer chain (Rxn 2), and to the linking of two polymer chains (Rxn 3).

```
run 25000
```

Here, the `thermo_style custom` command is used to print the cumulative reaction counts which are calculated by `fix rxn` and thus can be extracted from it. Run the simulation using LAMMPS. As the simulation progresses, polymer chains are observed forming (Fig. 49). During this reaction process, the temperature of the system remains well-controlled (Fig. 50 a), while the number of reactions, N_r, increases with time (Fig. 50 b).

Author Contributions

S.G. conceived and wrote the original online tutorials and underlying Sphinx documentation for lammpstutorials.github.io. C.A. tested the tutorials extensively, providing feedback that improved their clarity, accuracy, and usability. J.G. is the principal author of `fix bond/react` and `type labels` support in LAMMPS. He revised the tutorials to incorporate type labels and wrote Tutorial 8. A.K. developed the LAMMPS–GUI software and assisted in revising the tutorials for use with it. All authors participated in the revision and finalization of the manuscript.

Potentially Conflicting Interests

There are no conflicting interests to declare.

Funding Information

S.G. acknowledges funding from the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement N° 101065060. A.K. acknowledges financial support by Sandia National Laboratories under POs 2149742 and 2407526. This work used Expanse at the San Diego Supercomputer Center through allocation MAT240081 from the Advanced Cyberinfrastructure Coordination Ecosystem: Services & Support (ACCESS) program.

Author Information

ORCID:

Simon Gravelle: 0000-0003-2149-6706
Cecilia M. S. Alvares: 0009-0007-6237-6512
Jacob R. Gissinger: 0000-0003-0031-044X
Axel Kohlmeyer: 0000-0001-6204-6475

References

[1] **Frenkel D**, Smit B. Understanding Molecular Simulation: From Algorithms to Applications. Elsevier; 2023. https://doi.org/10.1016/B978-0-12-267351-1.X5000-7.

[2] **Allen MP**, Tildesley DJ. Computer Simulation of Liquids. Oxford university press; 2017. https://doi.org/10.1093/oso/9780198803195.001.0001.

[3] **van Gunsteren WF**, Dolenc J, Mark AE. Molecular Simulation as an Aid to Experimentalists. Current Opinion in Structural Biology. 2008; 18(2):149--153. https://doi.org/10.1016/j.sbi.2007.12.007.

[4] LAMMPS Homepage;. Accessed: 2024-07-15. https://www.lammps.org.

[5] **Thompson AP**, Aktulga HM, Berger R, Bolintineanu DS, Brown WM, Crozier PS, in't Veld PJ, Kohlmeyer A, Moore SG, Nguyen TD, et al. LAMMPS-A Flexible Simulation Tool for Particle-Based Materials Modeling at the Atomic, Meso, and Continuum Scales. Computer Physics Communications. 2022; 271:108171. https://doi.org/10.1016/j.cpc.2021.108171.

[6] LAMMPS Online Documentation for Latest Stable Version;. Accessed: 2024-07-15. https://docs.lammps.org/stable.

[7] **Wong-Ekkabut J**, Karttunen M. The Good, the Bad and the User in Soft Matter Simulations. Biochimica et Biophysica Acta (BBA)-Biomembranes. 2016; 1858(10):2529--2538. https://doi.org/10.1016/j.bbamem.2016.02.004.

[8] **van Gunsteren WF**, Daura X, Hansen N, Mark AE, Oostenbrink C, Riniker S, Smith LJ. Validation of Molecular Simulation: An Overview of Issues. Angewandte Chemie International Edition. 2018; 57(4):884--902. https://doi.org/10.1002/anie.201702945.

[9] **Prasad S**, Mobley D, Braun E, Mayes H, Monroe J, Zuckerman D, et al. Best Practices for Foundations in Molecular Simulations [Article v1. 0]. Living Journal of Computational Molecular Science. 2018; 1:1--28. https://doi.org/10.33011/livecoms.1.1.5957.

[10] **Jorgensen WL**, Maxwell DS, Tirado-Rives J. Development and Testing of the OPLS All-Atom Force Field on Conformational Energetics and Properties of Organic Liquids. Journal of the American Chemical Society. 1996; 118(45):11225--11236. https://doi.org/10.1021/ja9621760.

[11] **Stuart SJ**, Tutein AB, Harrison JA. A Reactive Potential for Hydrocarbons With Intermolecular Interactions. The Journal of Chemical Physics. 2000; 112(14):6472--6486. https://doi.org/10.1063/1.481208.

[12] **Gissinger JR**, Nikiforov I, Afshar Y, Waters B, Choi Mk, Karls DS, Stukowski A, Im W, Heinz H, Kohlmeyer A, et al. Type Label Framework for Bonded Force Fields in LAMMPS. The Journal of Physical Chemistry B. 2024; 128(13):3282--3297. https://doi.org/10.1021/acs.jpcb.3c08419.

[13] **Abascal JL**, Vega C. A General Purpose Model for the Condensed Phases of Water: TIP4P/2005. The Journal of Chemical Physics. 2005; 123(23). https://doi.org/10.1063/1.2121687.

[14] **van Duin AC**, Dasgupta S, Lorant F, Goddard WA. ReaxFF: A Reactive Force Field for Hydrocarbons. The Journal of Physical Chemistry A. 2001; 105(41):9396--9409. https://doi.org/10.1021/jp004368u.

[15] **Gissinger JR**, Jensen BD, Wise KE. REACTER: A Heuristic Method for Reactive Molecular Dynamics. Macromolecules. 2020; 53(22):9953--9961. https://doi.org/10.1021/acs.macromol.0c02012.

[16] LAMMPS-GUI Online Documentation for Latest Stable Version;. Accessed: 2024-07-15. https://docs.lammps.org/stable/Howto_lammps_gui.html.

[17] **Barrat JL**, Hansen JP. Basic Concepts for Simple and Complex Liquids. Cambridge University Press; 2003. https://doi.org/10.1017/CBO9780511606533.

[18] **Hansen JP**, McDonald IR. Theory of Simple Liquids: With Applications to Soft Matter. Academic press; 2013. https://doi.org/10.1016/C2010-0-66723-X.

[19] **SklogWiki contributors**, Main Page; n.d. Accessed: 2024-12-22. http://www.sklogwiki.org/SklogWiki/index.php/Main_Page.

[20] **Plimpton SJ**, Kohlmeyer A, Thompson AP, Moore SG, Berger R, LAMMPS: Large-Scale Atomic/Molecular Massively Parallel Simulator. Zenodo; 2024. https://doi.org/10.5281/zenodo.3726416.

[21] LAMMPS Releases Page on GitHub;. Accessed: 2024-12-26. https://github.com/lammps/lammps/releases.

[22] **van Rossum G**, Drake Jr FL. Python Reference Manual. Centrum voor Wiskunde en Informatica Amsterdam; 1995.

[23] **Hunter JD**. Matplotlib: A 2D Graphics Environment. Computing in Science & Engineering. 2007; 9(3):90--95. https://doi.org/10.1109/MCSE.2007.55.

[24] VMD Homepage;. Accessed: 2024-07-15. https://www.ks.uiuc.edu/Research/vmd.

[25] **Humphrey W**, Dalke A, Schulten K. VMD: Visual Molecular Dynamics. Journal of Molecular Graphics. 1996; 14(1):33--38. https://doi.org/10.1016/0263-7855(96)00018-5.

[26] OVITO Homepage;. Accessed: 2024-07-15. https://ovito.org.

[27] **Stukowski A**. Visualization and Analysis of Atomistic Simulation Data With OVITO--The Open Visualization Tool. Modelling and Simulation in Materials Science and Engineering. 2009; 18(1):015012. https://doi.org/10.1088/0965-0393/18/1/015012.

[28] **Wang X**, Ramírez-Hinestrosa S, Dobnikar J, Frenkel D. The Lennard-Jones Potential: When (Not) to Use It. Physical Chemistry Chemical Physics. 2020; 22(19):10624--10633. https://doi.org/10.1039/C9CP05445F.

[29] **Fischer J**, Wendland M. On the History of Key Empirical Intermolecular Potentials. Fluid Phase Equilibria. 2023; 573:113876. https://doi.org/10.1016/j.fluid.2023.113876.

[30] **Hestenes MR**, Stiefel E, et al. Methods of Conjugate Gradients for Solving Linear Systems, vol. 49. NBS Washington, DC; 1952. https://doi.org/10.6028/jres.049.044.

[31] **Schneider T**, Stoll E. Molecular-Dynamics Study of a Three-Dimensional One-Component Model for Distortive Phase Transitions. Physical Review B. 1978; 17(3):1302. https://doi.org/10.1103/PhysRevB.17.1302.

[32] **Grossfield A**, Zuckerman DM. Quantifying uncertainty and sampling quality in biomolecular simulations. Annual Reports in Computational Chemistry. 2009; 5:23--48.

[33] **Kohlmeyer A**, Vermaas J, TopoTools: Release 1.9. Zenodo; 2021. https://doi.org/10.5281/zenodo.598373.

[34] **Nosé S**. A Unified Formulation of the Constant Temperature Molecular Dynamics Methods. The Journal of Chemical Physics. 1984; 81(1):511--519. https://doi.org/10.1063/1.447334.

[35] **Hoover WG**. Canonical Dynamics: Equilibrium Phase-Space Distributions. Physical Review A. 1985; 31(3):1695. https://doi.org/10.1103/PhysRevA.31.1695.

[36] **Schmid N**, Eichenberger AP, Choutko A, Riniker S, Winger M, Mark AE, van Gunsteren WF. Definition and Testing of the GROMOS Force-Field Versions 54A7 and 54B7. European Biophysics Journal. 2011; 40:843--856. https://doi.org/10.1007/s00249-011-0700-9.

[37] **Wu Y**, Tepper HL, Voth GA. Flexible Simple Point-Charge Water Model With Improved Liquid-State Properties. The Journal of Chemical Physics. 2006; 124(2). https://doi.org/10.1063/1.2136877.

[38] **Luty BA**, van Gunsteren WF. Calculating Electrostatic Interactions Using the Particle-Particle Particle-Mesh Method With Nonperiodic Long-Range Interactions. The Journal of Physical Chemistry. 1996; 100(7):2581--2587. https://doi.org/10.1021/jp9518623.

[39] **Liese S**, Gensler M, Krysiak S, Schwarzl R, Achazi A, Paulus B, Hugel T, Rabe JP, Netz RR. Hydration Effects Turn a Highly Stretched Polymer From an Entropic Into an Energetic Spring. ACS Nano. 2017; 11(1):702--712. https://doi.org/10.1021/acsnano.6b07071.

[40] **Oostenbrink C**, Villa A, Mark AE, van Gunsteren WF. A Biomolecular Force Field Based on the Free Enthalpy of Hydration and Solvation: The GROMOS Force-Field Parameter Sets 53A5 and 53A6. Journal of Computational Chemistry. 2004; 25(13):1656--1676. https://doi.org/10.1002/jcc.20090.

[41] **Berendsen HJC**, Postma JPM, van Gunsteren WF, Hermans J. In: Pullman B, editor. Interaction Models for Water in Relation to Protein Hydration Dordrecht: Springer Netherlands; 1981. p. 331--342. https://doi.org/10.1007/978-94-015-7658-1_21.

[42] **Ewald PP**. Die Berechnung Optischer und Elektrostatischer Gitterpotentiale. Annalen der Physik. 1921; 369(3):253--287. https://doi.org/10.1002/andp.19213690304.

[43] **Martyna GJ**, Tobias DJ, Klein ML. Constant Pressure Molecular Dynamics Algorithms. The Journal of Chemical Physics. 1994; 101(5):4177--4189. https://doi.org/10.1063/1.467468.

[44] **Malde AK**, Zuo L, Breeze M, Stroet M, Poger D, Nair PC, Oostenbrink C, Mark AE. An Automated Force Field Topology Builder (ATB) and Repository: Version 1.0. Journal of Chemical Theory and Computation. 2011; 7(12):4026--4037. https://doi.org/10.1021/ct200196m.

[45] **Fixman M**. Radius of Gyration of Polymer Chains. The Journal of Chemical Physics. 1962; 36(2):306--310. https://doi.org/10.1063/1.1732501.

[46] **Kadaoluwa Pathirannahalage SP**, Meftahi N, Elbourne A, Weiss AC, McConville CF, Padua A, Winkler DA, Costa Gomes M, Greaves TL, Le TC, et al. Systematic Comparison of the Structural and Dynamic Properties of Commonly Used Water Models for Molecular Dynamics Simulations. Journal of Chemical Information and Modeling. 2021; 61(9):4521--4536. https://doi.org/10.1021/acs.jcim.1c00794.

[47] **Ryckaert JP**, Ciccotti G, Berendsen HJ. Numerical Integration of the Cartesian Equations of Motion of a System With Constraints: Molecular Dynamics of n-Alkanes. Journal of Computational Physics. 1977; 23(3):327--341. https://doi.org/10.1016/0021-9991(77)90098-5.

[48] **Andersen HC**. Rattle: A "Velocity" Version of the SHAKE Algorithm for Molecular Dynamics Calculations. Journal of Computational Physics. 1983; 52(1):24--34. https://doi.org/10.1016/0021-9991(83)90014-1.

[49] **Mills R**. A Remeasurement of the Self-Diffusion Coefficients of Sodium Ion in Aqueous Sodium Chloride Solutions. Journal of the American Chemical Society. 1955; 77(23):6116--6119. https://doi.org/10.1021/ja01628a008.

[50] **Gravelle S**, Kamal C, Botto L. Violations of Jeffery's Theory in the Dynamics of Nanographene in Shear Flow. Physical Review Fluids. 2021; 6(3):034303. https://doi.org/10.1103/PhysRevFluids.6.034303.

[51] **Wolde-Kidan A**, Netz RR. Interplay of Interfacial Viscosity, Specific-Ion, and Impurity Adsorption Determines Zeta Potentials of Phospholipid Membranes. Langmuir. 2021; 37(28):8463--8473. https://doi.org/10.1021/acs.langmuir.1c00868.

[52] **Zou C**, van Duin A. Investigation of Complex Iron Surface Catalytic Chemistry Using the ReaxFF Reactive Force Field Method. Jom. 2012; 64:1426--1437. https://doi.org/10.1007/s11837-012-0463-5.

[53] **Vashishta P**, Kalia RK, Rino JP, Ebbsjö I. Interaction Potential for SiO2: A Molecular-Dynamics Study of Structural Correlations. Physical Review B. 1990; 41(17):12197. https://doi.org/10.1103/PhysRevB.41.12197.

[54] **Rappe AK**, Goddard III WA. Charge Equilibration for Molecular Dynamics Simulations. The Journal of Physical Chemistry. 1991; 95(8):3358--3363. https://doi.org/10.1021/j100161a070.

[55] **Sulpizi M**, Gaigeot MP, Sprik M. The Silica-Water Interface: How the Silanols Determine the Surface Acidity and Modulate the Water Properties. Journal of Chemical Theory and Computation. 2012; 8(3):1037--1047. https://doi.org/10.1021/ct2007154.

[56] **Della Valle RG**, Andersen HC. Molecular Dynamics Simulation of Silica Liquid and Glass. The Journal of Chemical Physics. 1992; 97(4):2682--2689. https://doi.org/10.1063/1.463056.

[57] **Gravelle S**, Dumais J. A Multi-Scale Model for Fluid Transport Through a Bio-Inspired Passive Valve. The Journal of Chemical Physics. 2020; 152(1). https://doi.org/10.1063/1.5126481.

[58] **Kästner J**. Umbrella Sampling. Wiley Interdisciplinary Reviews: Computational Molecular Science. 2011; 1(6):932--942. https://doi.org/10.1002/wcms.66.

[59] **Wilson MA**, Pohorille A. Adsorption and Solvation of Ethanol at the Water Liquid--Vapor Interface: A Molecular Dynamics Study. The Journal of Physical Chemistry B. 1997; 101(16):3130--3135. https://doi.org/10.1021/jp962629n.

[60] **Makarov DE**. Computer Simulations and Theory of Protein Translocation. Accounts of Chemical Research. 2009; 42(2):281--289. https://doi.org/10.1021/ar800128x.

[61] **Gravelle S**, Botto L. Adsorption of Single and Multiple Graphene-Oxide Nanoparticles at a Water--Vapor Interface. Langmuir. 2021; 37(45):13322--13330. https://doi.org/10.1021/acs.langmuir.1c01902.

[62] **Loche P**, Bonthuis DJ, Netz RR. Molecular Dynamics Simulations of the Evaporation of Hydrated Ions From Aqueous Solution. Communications Chemistry. 2022; 5(1):55. https://doi.org/10.1038/s42004-022-00669-5.

[63] **Hayatifar A**, Gravelle S, Moreno BD, Schoepfer VA, Lindsay MB. Probing Atomic-Scale Processes at the Ferrihydrite-Water Interface With Reactive Molecular Dynamics. Geochemical Transactions. 2024; 25(1):10. https://doi.org/10.1186/s12932-024-00094-8.

[64] **Weeks JD**, Chandler D, Andersen HC. Role of Repulsive Forces in Determining the Equilibrium Structure of Simple Liquids. The Journal of Chemical Physics. 1971; 54(12):5237--5247. https://doi.org/10.1063/1.1674820.

[65] **Kumar S**, Rosenberg JM, Bouzida D, Swendsen RH, Kollman PA. The Weighted Histogram Analysis Method for Free-Energy Calculations on Biomolecules. I. The Method. Journal of Computational Chemistry. 1992; 13(8):1011--1021. https://doi.org/10.1002/jcc.540130812.

[66] **Kumar S**, Rosenberg JM, Bouzida D, Swendsen RH, Kollman PA. Multidimensional Free-Energy Calculations Using the Weighted Histogram Analysis Method. Journal of Computational Chemistry. 1995; 16(11):1339--1350. https://doi.org/10.1002/jcc.540161104.

[67] **Grossfield A**, An Implementation of WHAM: The Weighted Histogram Analysis Method Version 2.1.0; 2025. http://membrane.urmc.rochester.edu/content/wham/, accessed: 2025-03-16.

https://doi.org/10.33011/livecoms.6.1.3037
Living J. Comp. Mol. Sci. 2025, 6(1), 3037

[68] **Gissinger JR**, Jensen BD, Wise KE. Modeling Chemical Reactions in Classical Molecular Dynamics Simulations. Polymer. 2017; 128:211--217. https://doi.org/10.1016/j.polymer.2017.09.038.

[69] **Gissinger JR**, Jensen BD, Wise KE. Molecular Modeling of Reactive Systems With REACTER. Computer Physics Communications. 2024; 304:109287. https://doi.org/10.1016/j.cpc.2024.109287.

[70] **Sun H**. COMPASS: An Ab Initio Force-Field Optimized for Condensed-Phase Applications Overview With Details on Alkane and Benzene Compounds. The Journal of Physical Chemistry B. 1998; 102(38):7338--7364. https://doi.org/10.1021/jp980939v.

[71] Flatpak Homepage;. Accessed: 2024-08-09. https://flatpak.org.

[72] Run LAMMPS Online Documentation for Latest Stable Version;. Accessed: 2024-12-14. https://docs.lammps.org/stable/Run_head.html.

A Using LAMMPS–GUI

For simplicity, these tutorials reference keyboard shortcuts based on the assignments for Linux and Windows. macOS users should use the "Command" key (⌘) in place of the "Ctrl" key when using keyboard shortcuts.

A.1 Installation

Pre-compiled versions of LAMMPS–GUI are available for Linux, macOS, and Windows on the LAMMPS GitHub Release page [21]. The Linux version is provided in two formats: as compressed tar archive (.tar.gz) and as a Flatpak bundle [71]. The macOS version is distributed as a .dmg installer image, while the Windows version comes as an executable installer package.

A.1.1 Installing the Linux .tar.gz Package

Download the archive (e.g., LAMMPS-Linux-x86_64-GUI-22Jul2025.tar.gz) and unpack it. This will create a folder named LAMMPS–GUI containing the included commands, which can be launched directly using "./lammps-gui" or "./lmp", for example. Adding this folder to the PATH environment variable will make these commands accessible from everywhere, without the need for the "./" prefix.

A.1.2 Installing the Linux Flatpak Bundle

You have to have Flatpak support installed on Linux machine to be able to use the Flatpak bundle. Download the bundle file (e.g., LAMMPS-Linux-x86_64-GUI-22Jul2025.flatpak) and then install it using the following command:

```
flatpak install --user \
    LAMMPS-Linux-x86_64-GUI-22Jul2025.flatpak
```

This will integrate LAMMPS–GUI into your desktop environment (e.g., GNOME, KDE, XFCE) where it should appear in the "Applications" menu under "Science". Additionally, the ".lmp" file extension will be registered to launch LAMMPS–GUI when opening a file with this extension in the desktop's file manager.

You can also launch LAMMPS–GUI from the command-line using the following command:

```
flatpak run org.lammps.lammps-gui
```

Similarly, for launching the LAMMPS command-line executable, use:

```
flatpak run --command=lmp org.lammps.lammps-gui -in in.lmp
```

A.1.3 Installing the macOS Application Bundle

After downloading the macOS app bundle image file (e.g., LAMMPS-macOS-multiarch-GUI-22Jul2025.dmg), double-click on it. In the dialog that opens drag the LAMMPS–GUI app bundle into the Applications folder. To enable command-line access, follow the instructions in the README.txt file. These macOS app-bundles contain native executables for both, Intel and Apple CPUs.

After installation, you can launch LAMMPS–GUI from the Applications folder. Additionally, you can drag an input file onto the app or open files with the ".lmp" extension. Note that the LAMMPS–GUI app bundle is currently not cryptographically signed, so macOS may initially prevent it from launching. If this happens, you need to adjust the settings in the "Security & Privacy" system preferences dialog to allow access.

A.1.4 Installing the Windows package

Download the LAMMPS–GUI installer for Windows (e.g., LAMMPS-Win10-64bit-GUI-22Jul2025.exe). Windows may warn you that the file is from an unknown developer and was downloaded from the internet. This happens because neither the installer nor the LAMMPS–GUI application (or any other included applications) have been cryptographically signed. You will need to choose to keep the file, and when launching the installer, confirm that you want to run it despite the warning.

After installation, a new entry should appear in the Start menu. Additionally, the ".lmp" file extension should be registered with Windows File Explorer to open LAMMPS–GUI when opening a file with the ".lmp" extension. The "lammps-gui" and "lmp" commands should also be available in the command-line.

https://doi.org/10.33011/livecoms.6.1.3037
Living J. Comp. Mol. Sci. 2025, 6(1), 3037

A.2 Opening, Editing, and Saving Files

LAMMPS–GUI can be launched from the command-line, as explained above, where you can either launch it without arguments or provide one file name as an argument. All other arguments will be ignored. For example:

```
lammps-gui input.lmp
```

Files can also be opened from the "File" menu. You can select a file through a dialog and then open it. Additionally, a history of the last five opened files is maintained, with entries to open them directly. Finally, the `Ctrl-O` keyboard shortcut can also be used to open a file. When integrated into a desktop environment, it is also possible to open files with a ".lmp" extension or use drag-and-drop.

For the most part, the editor window behaves like other graphical editors. You can enter, delete, or copy and paste text. When entering text, a pop-up window will appear with possible completions after typing the first two characters of the first word in a line. You can navigate the highlighted options using the up and down arrow keys, and select a completion by pressing the Enter key or using the mouse. You can also continue typing, and the selection in the pop-up will be refined. For some commands, there will be completion pop-ups for their keywords or when a filename is expected, in which case, the pop-up will list files in the current folder.

As soon as LAMMPS–GUI recognizes a command, it applies syntax highlighting according to built-in categories. This can help detect typos, since those may cause LAMMPS–GUI not to recognize the syntax and thus not apply or partially apply the syntax highlighting. When you press the `Tab` key, the line will be reformatted. Consistent formatting can improve the readability of input files, especially long and complex ones.

If the file in the editor has unsaved changes, the word "*modified*" will appear in the window title. The current input buffer can be saved by selecting "Save" or "Save As..." from the "File" menu. You can also click the "Save" icon on the left side of the status bar, or use the `Ctrl-S` keyboard shortcut.

When LAMMPS–GUI opens a file, it will *switch* the working directory to the folder that contains the input file. The same happens when saving to a different folder than the current working directory. The current working directory can be seen in the status bar at the bottom right. This is important to note because LAMMPS input files often require additional files for reading and may write output files (such as images, trajectory dumps, or averaged data files), which are typically expected to be in the same folder as the input file.

A.3 Running LAMMPS

From within the LAMMPS–GUI main window, LAMMPS can be started either from the «Run» menu by selecting the «Run LAMMPS from Editor Buffer» entry, using the keyboard shortcut Ctrl-Enter (Command-Enter on macOS), or by clicking the green «Run» button in the status bar. While LAMMPS is running, a message on the left side indicates that LAMMPS is active, along with the number of active threads. On the right side, a progress bar is displayed, showing the estimated progress of the current `run` or `minimize` command.

A.4 Creating Snapshot Images

Open the «Image Viewer» using either the «Create Image» option from the «Run» menu, the «Ctrl-I» keyboard shortcut, or click on the (right) palette button in the status bar. The image can be saved using the «Save As...» option from the «File» menu.

A.5 The Output Window

By default, when starting a run, the «Output» window opens to display the screen output of the running LAMMPS calculation. The text in the Output window is read-only and cannot be modified, but keyboard shortcuts for selecting and copying all or part of the text can be used to transfer it to another program: The keyboard shortcut «Ctrl-S» (or «Command-S» on macOS) can be used to save the Output buffer to a file. Additionally, the «Select All» and «Copy» functions, along with a «Save Log to File» option, are available through the context menu, which can be accessed by right-clicking within the text area of the «Output» window.

A.6 The Charts Window

By default, when starting a run, a «Charts» window opens to display a plot of the thermodynamic output from the LAMMPS calculation. From the «File» menu in the top-left corner, you can save an image of the currently displayed plot or export the data in various formats: plain text columns (for use with plotting tools like Gnuplot or XmGrace), CSV data (suitable for processing in Microsoft Excel, LibreOffice Calc, or Python with Pandas), or YAML (which can be imported into Python using PyYAML or Pandas). You can use the mouse to zoom in on the graph by holding the left button and dragging to select an area. To zoom out, right-click anywhere on the graph. You can reset the view by clicking the «lens» button located next to the data drop-down menu.

A.7 Preferences

The Preferences dialog allows customization of the behavior and appearance of LAMMPS–GUI. Among other options:

https://doi.org/10.33011/livecoms.6.1.3037
Living J. Comp. Mol. Sci. 2025, 6(1), 3037

- In the «General Settings» tab, the «Data update interval» setting allows you to define the time interval, in milliseconds, between data updates during a LAMMPS run. By default, the data for the «Charts» and «Output» windows is updated every 10 milliseconds. Set this to 100 milliseconds or more if LAMMPS–GUI consumes too many resources during a run. The «Charts update interval» controls the time interval between redrawing the plots in the «Charts» window, in milliseconds.
- The «Accelerators» tab enables you to select an accelerator package for LAMMPS to use. Only settings supported by the LAMMPS library and local hardware are available. The «Number of threads» field allows you to set the maximum number of threads for accelerator packages that utilize threading.
- The «Editor Settings» tab allows you to adjust the settings of the editor window. Select the «Auto-save on Run and Quit» option to automatically save changes made to the `.lmp` file upon closing LAMMPS–GUI.

See Ref. 16 for a full list of options.

B Running LAMMPS on the Command-Line without the GUI

LAMMPS can also be executed from the command-line on Linux, macOS, and Windows without using the GUI. This is the more common way to run LAMMPS. Both, the LAMMPS–GUI program and the LAMMPS command-line executable utilize the same LAMMPS library and thus no changes to the input file are required.

First, open a terminal or command-line prompt window and navigate to the directory containing the `input.lmp` file. Then execute:

```
lmp -in input.lmp
```

where `lmp` is the command-line LAMMPS command.

For parallel execution with 4 processors (via OpenMP threads where supported by the OPENMP package), use:

```
lmp -in input.lmp -pk omp 4 -sf omp
```

Running in parallel via MPI requires a specially compiled LAMMPS package and is not supported by the GUI. On supercomputers or HPC clusters, pre-compiled LAMMPS executables are typically provided by the facility's user support team. For more information, please refer to the facility's documentation or contact its user support staff.

See Ref. 72 for a complete description on how to run LAMMPS.