Assemble!
- Tutorial -

Matteo Degiacomi, University of Oxford; Valentina Erastova,
Durham University

In this document we present a working example of Assemble! GUI usage, aimed at the creation of a small polyisoprene system. A short equilibration of the obtained system is subsequently performed using Gromacs 4.5.4 [1] on a linux Ubuntu 12.04 workstation equipped with 3.30 GHz quad-core Intel i3. All files mentioned here are included folder *example*.

For this example we set up a simulation box containing a range of cis-trans-polyisoprene molecules with average of 8 monomer units, of which 40% are cis and 60% trans.

1. **Prepare monomers coordinates and topologies**
   To create a polymer chain, information about all the subunits must be available. For every subunit, coordinates in .pdb format (see User Manual, section 3.1.1) and topology (section 3.1.2) should be provided. Additionally, a Gromacs-like force field field describing the molecular interactions required by monomer topologies (section 3.1.3) must also be provided. In this tutorial, coordinates and topologies for cis- and trans- isoprene monomers, as well as the TraPPE-UA force field [2], are already given in folder *example/database*.

2. **Run Assemble!**
   Launch assemble either by typing `python Assemble_GUI.py` in the terminal, or double clicking on its icon. A graphical interface window will be displayed.

3. **Load monomers and create a database**
   Since this example assumes it is first time Assemble! is used, the user does not have neither a pre-made database file nor a saved session. We will therefore create them from scratch. Firstly, Assemble! requires information on monomer units. By clicking *Add*, a popup window (Figure 1) will appear, allowing the user to load PDB coordinates and topology files of a molecule, and associate them to a one-letter code. This information will constitute a database Assemble! will later exploit to create polymers.

   We can now add in our database a cis-isoprene monomer named C, and trans-isoprene monomer named T. In case of a mistake, it is possible to *Edit* or *Remove* a monomer from the database. When monomer information is added, it is also possible to *Save* the database file. The saved database file can be loaded into later sessions of Assemble!.

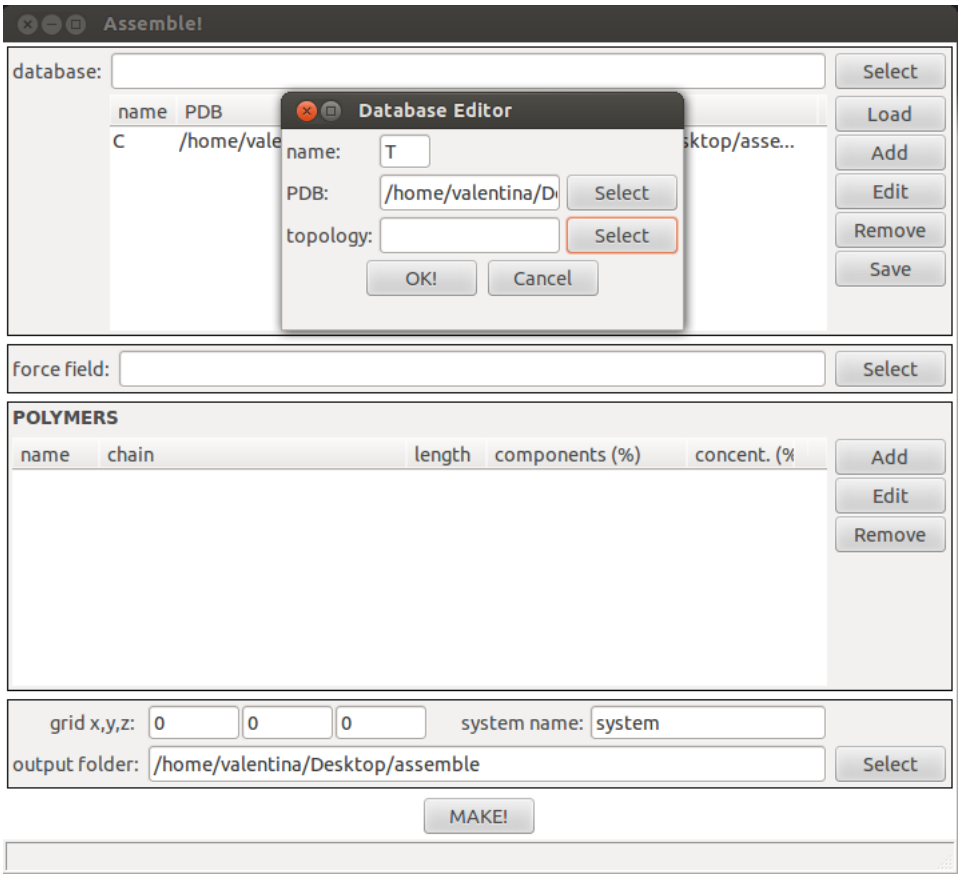


Figure 1: *Adding monomer information into Assemble! database*

4. **Load force field file**
   In this phase, the force field to be used (in our case TraPPE-UA) must be indicated. Assemble! will cross check force field file, coordinates and topology and will warn the user if there are mismatches.

5. **Describe polymers of interest**
   This is the creative part of Assemble!, where the user describes the polymers to be included in the system. By pressing *Add* a popup window will appear, requiring information about the system's name (should not be too long and must avoid white spaces and special characters), and the method for the polymeric sequence definition - *chain* or *percentage* (see figure 2). If the option *chain* is chosen the user will provide a chain sequence according to the one-letter monomer unit names defined in database. If the option *percentage* is chosen the Assemble! will create a random sequence of a desired length and containing given percentages of the monomers. The *Add* button must be clicked to provide information about the individual monomer percentages. A popup window will appear allowing to chose from a drop-down menu a one-letter name of the monomer (as defined in the previously loaded database), and a required percentage (or ratio) of the monomer in the chain. The chain sequences generated by pressing *Generate* button will be displayed in the *chain:* textbox. If the user is not satisfied with the sequence, the *Generate* button can be clicked again, leading to the creation of a different random chain. At this stage it is also possible to manually alter the generated chain sequence by changing *s*ource to *chain*. Please note that often the sequence cannot feature exactly the defined monomer percentages. The obtained monomer percentages in the generated polymer chain will be displayed in the main window. For each defined chain, a value indicating the concentration within the final system should also be provided. Note that polymer concentrations should add up to 100%. In case a single polymer sequence is provided, Assemble! will automatically fill in concentration to 100%.

   In this example we generate the following five chains, each having a 20% concentration in the final system:

   - chain PI1 sequence TTTCTCTC
   - chain PI2 sequence CCTCTCCTT
   - chain PI3 sequence TCCTTCC
   - chain PI4 sequence CTCTCCTC
   - chain PI5 sequence CCTCCT

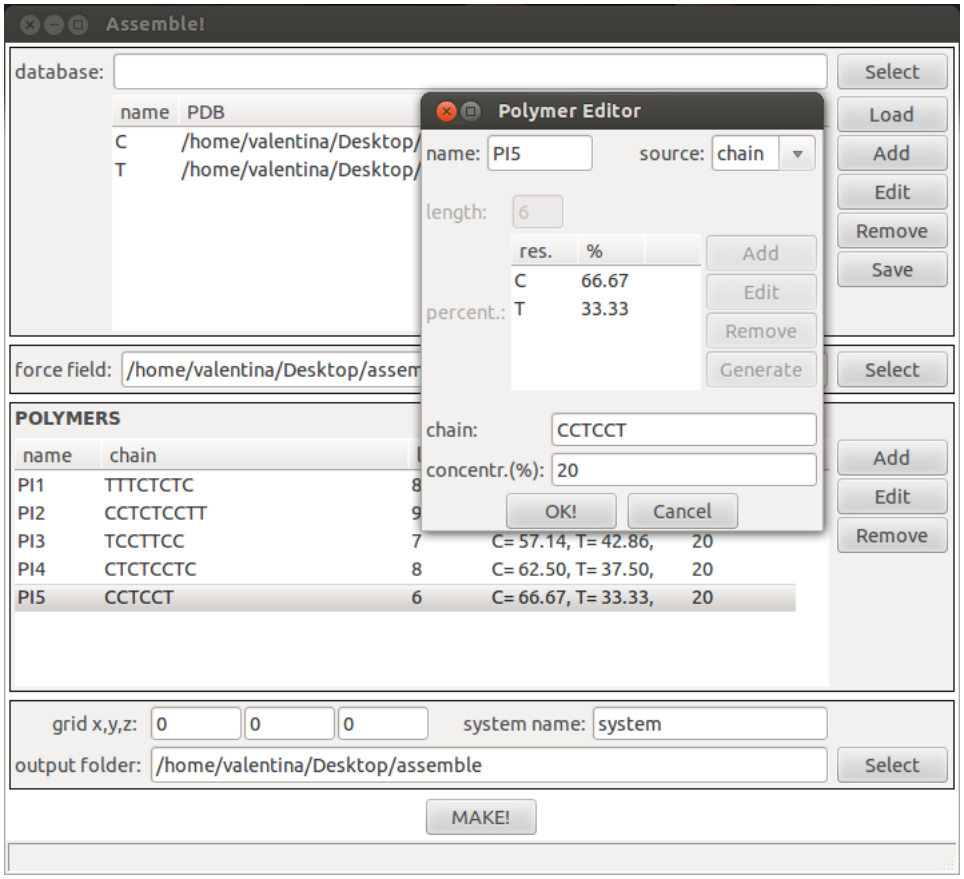


Figure 2: *Generation of polymer sequences*

6. **Define polymeric system**
   Every polymer described above will be aligned along its inertia tensor in order to minimise its bounding box (first principal axis along x, second along y and third along z). Several polymer copies can be randomly arranged into a simulation box. In Assemble!, a simulation box is defined by providing its grid dimensions, i.e. the amount of polymers to be arranged in the x, y and z direction. For Gromacs simulations it is always best to aim for a cubic box, so defining the grid is a trial and error process, luckily an easy one. Here we set our system to be a 5 x 9 x 12 grid.

7. **Output folder**
   Chose the name of the final system (default is *system*) and select a folder where all your precious work has to be saved. All data will be saved in a subfolder having the system's name. Data includes coordinates in both .gro and .pdb format for each polymer chain, an .itp for each polymer chain, a box of the polymers (.gro), its associated topology (.top), an index file (.ndx) for Gromacs runs and a log file of the Assemble! work. In this tutorial, the system we generate is named *example*. All produced files are provided in the *example* folder.

8. **MAKE!**
   Assemble! will create all defined polymers, pack them into a box of desired dimensions. When the system generation will be complete, a popup window will notify the user. During this step a lot of information will appear on the terminal. This information is also written into a log file in the chosen folder. Assemble! reports on the polymers generation and provides information about the generated system, namely: final polymer concentrations in the box, monomer distribution in the box, number average degree of polymerisation and the size of the final box. If the user is not satisfied with the obtained system, this is a good time to edit the inputs and generate a new system pressing *MAKE!* again.

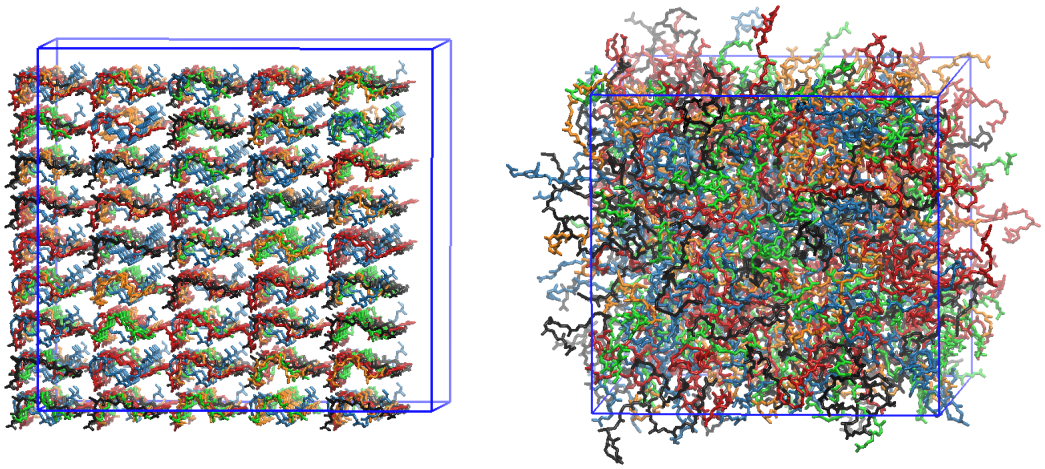
   In this example we obtain a box (see figure 3 (a)) containing:

   - PI1 20.3703703704 %
   - PI2 23.7037037037 %
   - PI3 18.1481481481 %
   - PI4 20.5555555556 %
   - PI5 17.2222222222 %

Cis/trans distribution:

- C 54.9711815562%
- T 45.0288184438%

Number average of polymerisation is 7.71111111111, and box size 11.71456 x 10.76650 x11.28805 nm^3.

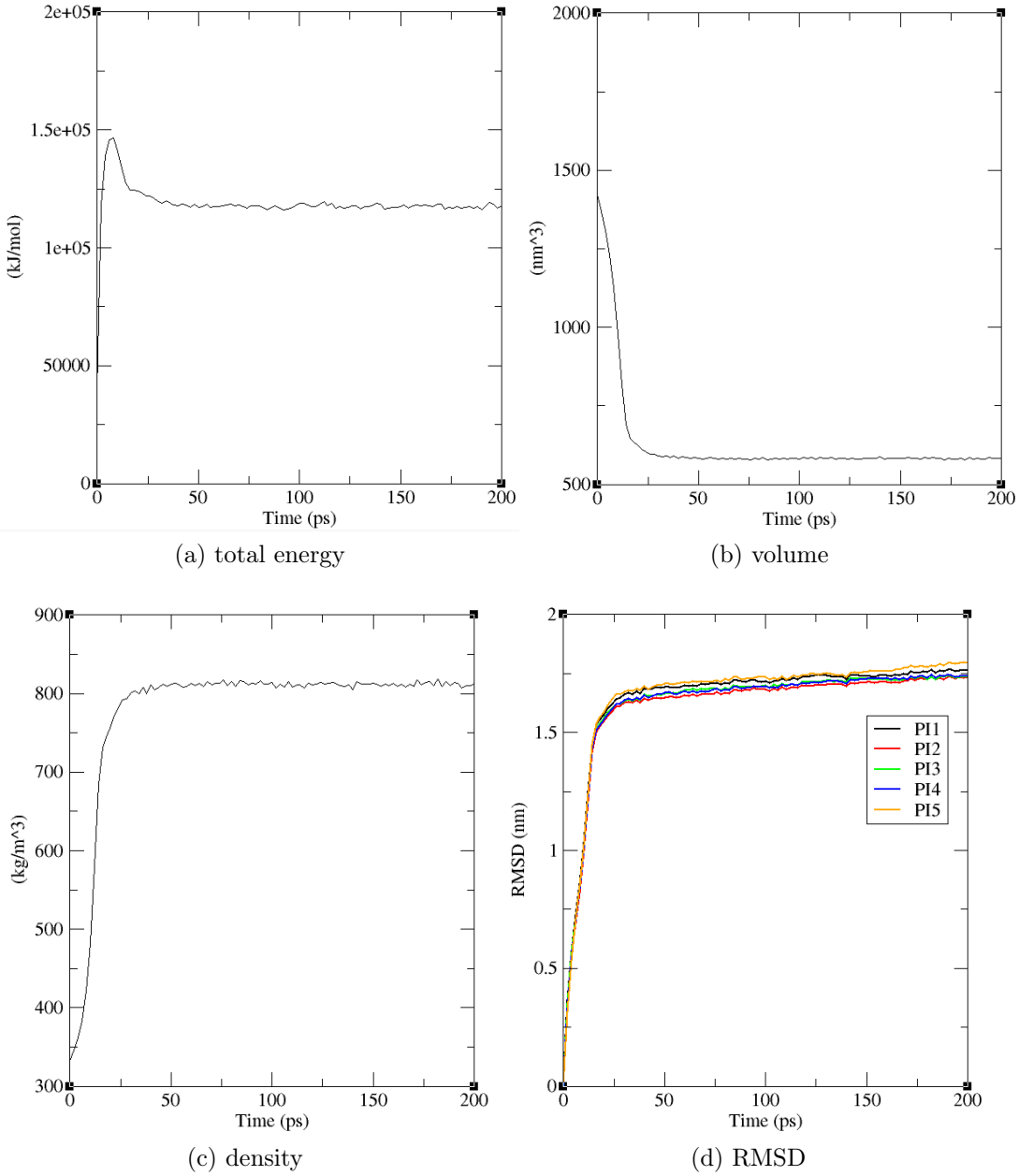


(a) 0 ps (b) 200 ps

Figure 3: *Polyisoprene system generated with Assemble! displayed with VMD at the start (a) and end (b) of 200 ps simulation. The polymers are coloured by type: PI1 - black, PI2 - red, PI3 - green, PI4 - blue and PI5 - orange*

9. **Simulate with Gromacs**
   Assemble! provides all necessary files for a Gromacs simulation. The user has only to provide the molecular dynamics parameter file (.mdp file), defining a simulation protocol. Information about .mdp files can be found online at manual.gromacs.org. For this example, we provide em.mdp for energy minimisation and equil.mdp for system equilibration over 200 ps in NPT ensemble at 350K and 5 bar, using a 2 fs time step.

10. **Simulation analysis**
    The final frame of simulation is given in figure 3 (b). The total energy, volume, density and root mean square deviation of atom distances evolution through the simulation are shown in figure 4. From the figure it can be seen that the system converges within the first 30 ps of simulation.



(a) total energy

(b) volume

(c) density

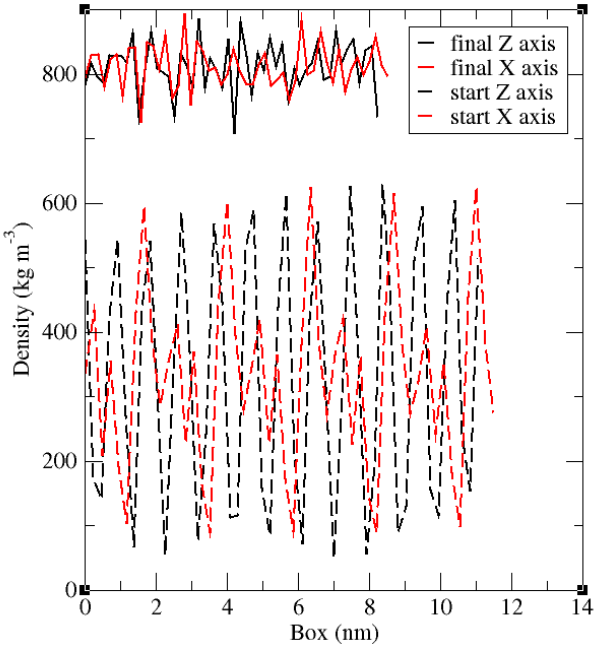
(d) RMSD

Figure 4: *Total energy (a), volume (b), density (c) and root mean square deviation of atom distances (d) trough 200 ps simulation*

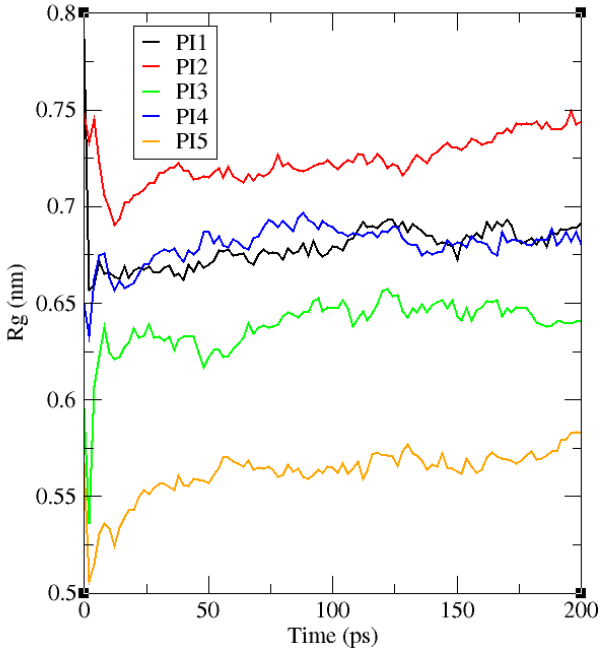
Figure 5 (a) shows the partial density of the system at the start (dashed line) and end (solid line) of the simulation. In the starting configuration the polymers are aligned along the inertia tensor, as shown in figure 3 (a). For this reason, a strong structuring in the partial density along both axes is visible. After 200 ns simulation the partial density is uniform, averaging at 800 $kg\ m^{-3}$ on both axes, matching the density of the system given on figure 4 (c). This indicates well intermixed melt.

A frequently used quantity to describe a polymeric system is the radius of gyration of its constituent molecules. Figure 5 (b) shows the evolution of radius of gyration through the simulation. A constant radius of gyration is achieved after 50 ps of the simulation for all of the components. Interestingly, the radius of gyration is not only dependant on the chain length, but also on its monomer order sequence.

With this example we have described the set up process for a polymeric system with Assemble!, leading to a straightforward input into the Gromacs molecular dynamics engine. Assemble! prepares a mixed and closely packed polymeric system, that not only reduces equilibration time, but also also minimises undesired formation of globular conformations likely to form in vacuum.



(a) partial density



(b) radius of gyration

Figure 5: *(a) Partial density of polyisoprene system projected on x (red) and z (black) axis of start (dashed) and final (solid) configurations; (b) radius of gyration per polymer type over 200 ps simulation*

References

[1] Sander Pronk, Szilárd Páll, Roland Schulz, Per Larsson, Pär Bjelkmar, Rossen Apostolov, Michael R Shirts, Jeremy C Smith, Peter M Kasson, David van der Spoel, et al. Gromacs 4.5: a high-throughput and highly parallel open source molecular simulation toolkit. *Bioinformatics*, page btt055, 2013.

[2] Collin D Wick, Marcus G Martin, and J Ilja Siepmann. Transferable potentials for phase equilibria. 4. united-atom description of linear and branched alkenes and alkylbenzenes. *The Journal of Physical Chemistry B*, 104(33):8008–8016, 2000.