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]. We will 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. All files necessary for this tutorial are included in folder `example`. This folder contains two sub folders, `database` and `PI-example`.
The folder `example/database` contains the files necessary to create a polymeric chain. In *Assemble!* chains are represented as sequences of monomeric building blocks. Building blocks are described by their coordinates (`*.pdb` files, see User Manual section 3.1.1)) and topology (i.e. atom connectivity, `*.txt` files, section 3.1.2). In this example, two building blocks are provided, cis- and trans- isopropene. These are described by `cis-PI-monomer*` and `trans-PI-monomer*` files, respectively.

In order to generate the files needed to launch a molecular dynamics simulation, an additional piece of information is necessary: a force field. Force fields describe atomic interactions (specific equilibrium distances, force constants, ...). In this example the TraPPE-UA force field [2] is provided (`trappe.ff.txt`, User Manual section 3.1.3)

The folder `example/PI-example` contains all the files *Assemble!* should produce in this tutorial, as well as a small molecular dynamics simulation output, and its related analysis.

1. **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. The interface is conceived to be used from top to bottom. At any moment the user can save the state of the interface with `Files > Save as...`. A previously saved interface state can be loaded with `Files > Load...`.

2. **Load monomers and create a database**
   The first step is to define a database of monomeric building blocks using the topmost section of the interface. This database will associate the coordinates files (*.pdb) and related topology files (*.txt files) to unique one-letter codes. A polymeric chain will be defined by a string of one-letter codes.

By pressing the button `Add` a popup menu called `Database Editor` will appear (Figure 1). We will now create an entry for trans- isopropene. In the `name` field, type `T` (our one-letter code of choice). In the `pdb` field select the file `trans-PI-monomer.pdb` and in topology select the file `trans-PI-monomer.txt`. Confirm the selections pressing `OK!`. Repeat this operation to create a database entry for cis- isopropene (name it C, load files `cis-PI-monomer.pdb` and `cis-PI-monomer.txt`).

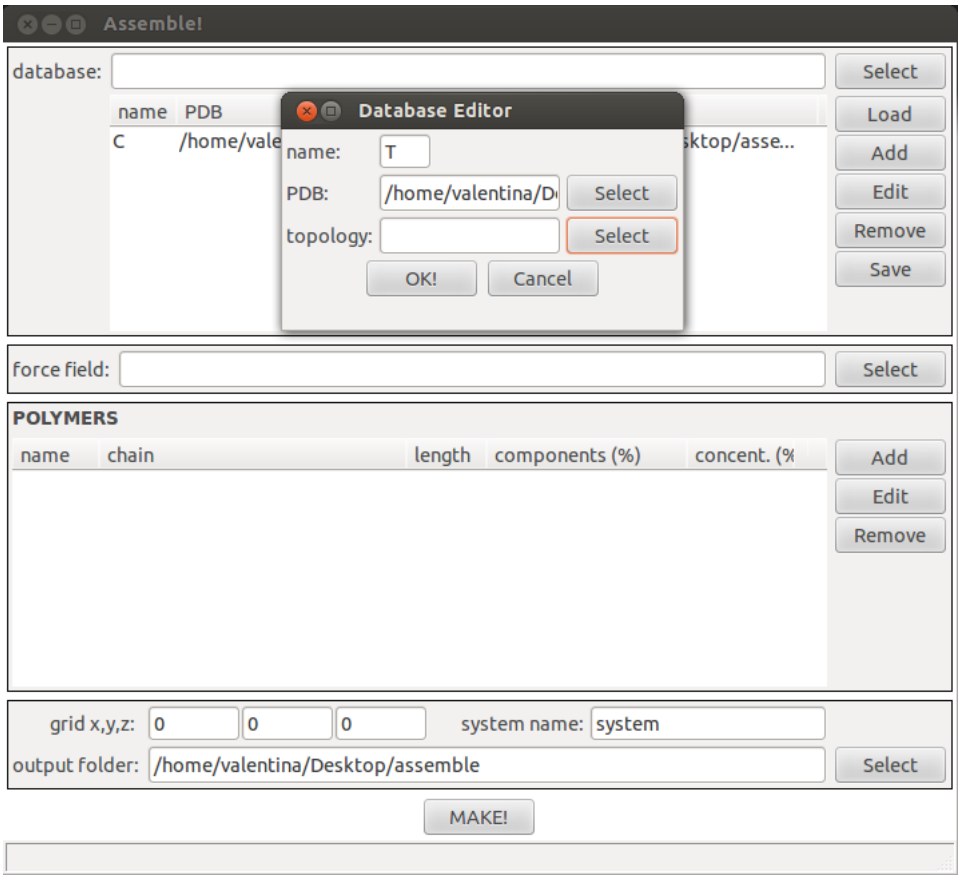


Figure 1: *Adding monomer information into Assemble! database*

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!*.

3. **Load force field file**
   In this phase, the force field to be used (in our case TraPPE-UA) must be indicated. In the force field section of the interface, select the

file `example/database/forcefield/trappe.ff.txt`. *Assemble!* will cross check force field file, coordinates and topology and will warn the user if there are mismatches.

4. **Describe polymers of interest**
   This is the creative part of *Assemble!*, where the user describes the polymers to be included in the system using the `POLYMERS` section of the interface. By pressing `Add` a popup window called `polymer editor` will appear, requiring information about the polymers's name (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 (in our case, `T` and `C`). For each defined chain, a value indicating the concentration within the final system should also be provided. After the `Polymer editor` is completely filled up, add the newly defined polymer to the polymers database pressing *OK!*. At this stage, *Assemble!* will check for all parameters consistency, warning the user in case of error.

   In this example we generate five chains, each having a 20% concentration in the final system. Note that the concentrations of all polymers should add up to 100%. In case a single polymer sequence is provided, *Assemble!* will automatically fill in concentration to 100%. Fill the fields of `Polymer Editor` with the following values (press `OK!` after every entry is complete):

   - *name* PI1, *chain* TTTCTCTC, *concentr.* 20
   - *name* PI2, *chain* CCTCTCCTT, *concentr.* 20
   - *name* PI3, *chain* TCCTTCC, *concentr.* 20
   - *name* PI4, *chain* CTCTCCTC, *concentr.* 20
   - *name* PI5, *chain* CCTCCT, *concentr.* 20

   The following information is not necessary for this Tutorial, but describes another *Assemble* capability: the definition of a chain by concentration of individual monomers. 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

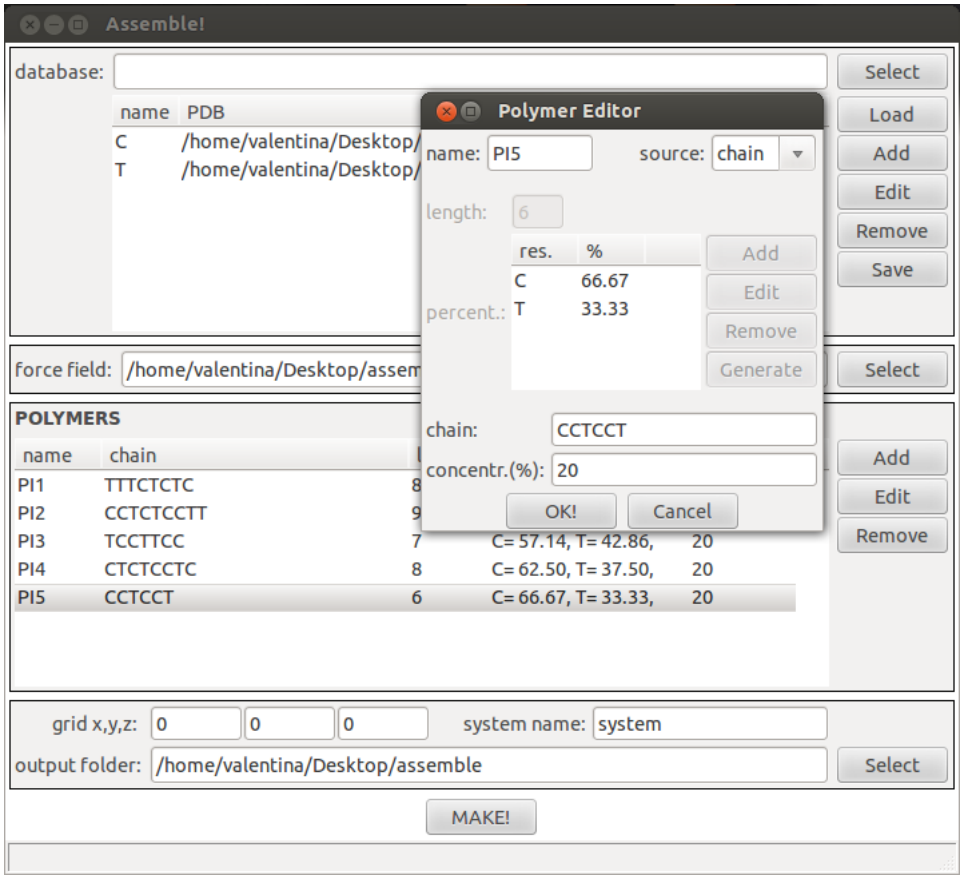


Figure 2: *Generation of polymer sequences*

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.

5. **Define polymeric system**
   The bottom section of the interface allows to generate a simulation box containing several randomly arranged copies of the previously defined polymers. Every polymer 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). 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. Insert these three values in the three fields `grid x,y,z`. 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.

6. **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.

   The log file informs us that we obtained a simulation box (see Figure 3 (a)) containing:

   - PI1 20.3703703704 %

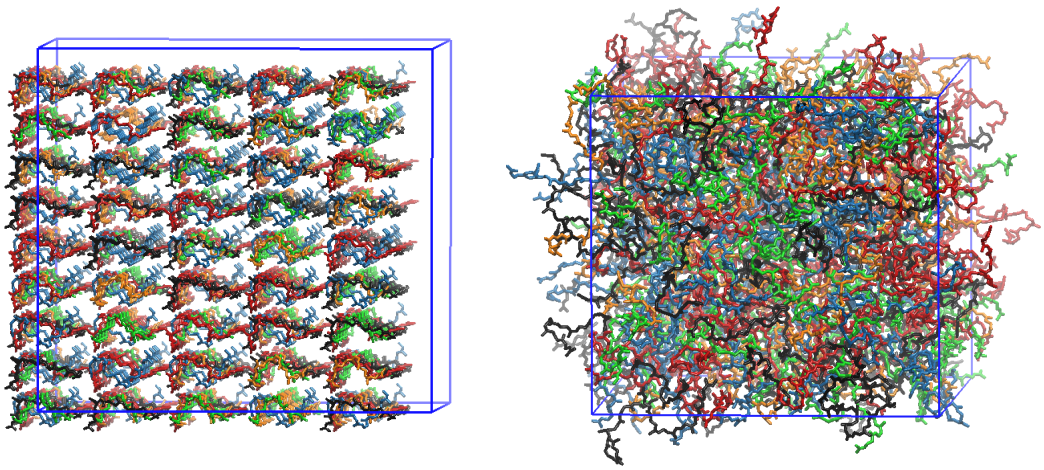
- PI2 23.7037037037 %
- PI3 18.1481481481 %
- PI4 20.5555555556 %
- PI5 17.2222222222 %

Cis/trans distribution:

- C 54.9711815562%
- T 45.0288184438%

Number average of polymerisation is 7.71111111111, and box size 11.71456 x 10.76650 x 11.28805 nm^3.

Please note that in your execution you might get slightly different values. This is caused by *Assemble!* randomly filling up the system grid with different molecules. If you think that the concentrations of different monomers are too far from being 20 percent each, you can press `MAKE!` again.



(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*

Generated data will include:

- for every polymer chain, coordinates (in `.gro` and `.pdb`), as well as an include topology file (`.itp`).
- for the full polymeric box, the system's coordinates (in `.gro` file) and its associated topology in a (`.top` file) The topology file links together all `.itp` files of individual polymers.
- an index file `.ndx`, that is useful for system's analysis (it contains information about which atoms are part of which polymer).
- a log file of all *Assemble!*'s work.

The user is encouraged to inspect the individual generated polymers and the resulting packed system. This terminates the tutorial about using *Assemble!* through its graphical interface. Note that an identical result can be obtained by launching *Assemble!* from a terminal, and providing an input file (not covered in this tutorial, see input file description in User Manual, section 3.1.5). *Assemble!* graphical interface is able to save and load input files with `File > Save as ...` and `File > Load ...`, respectively.

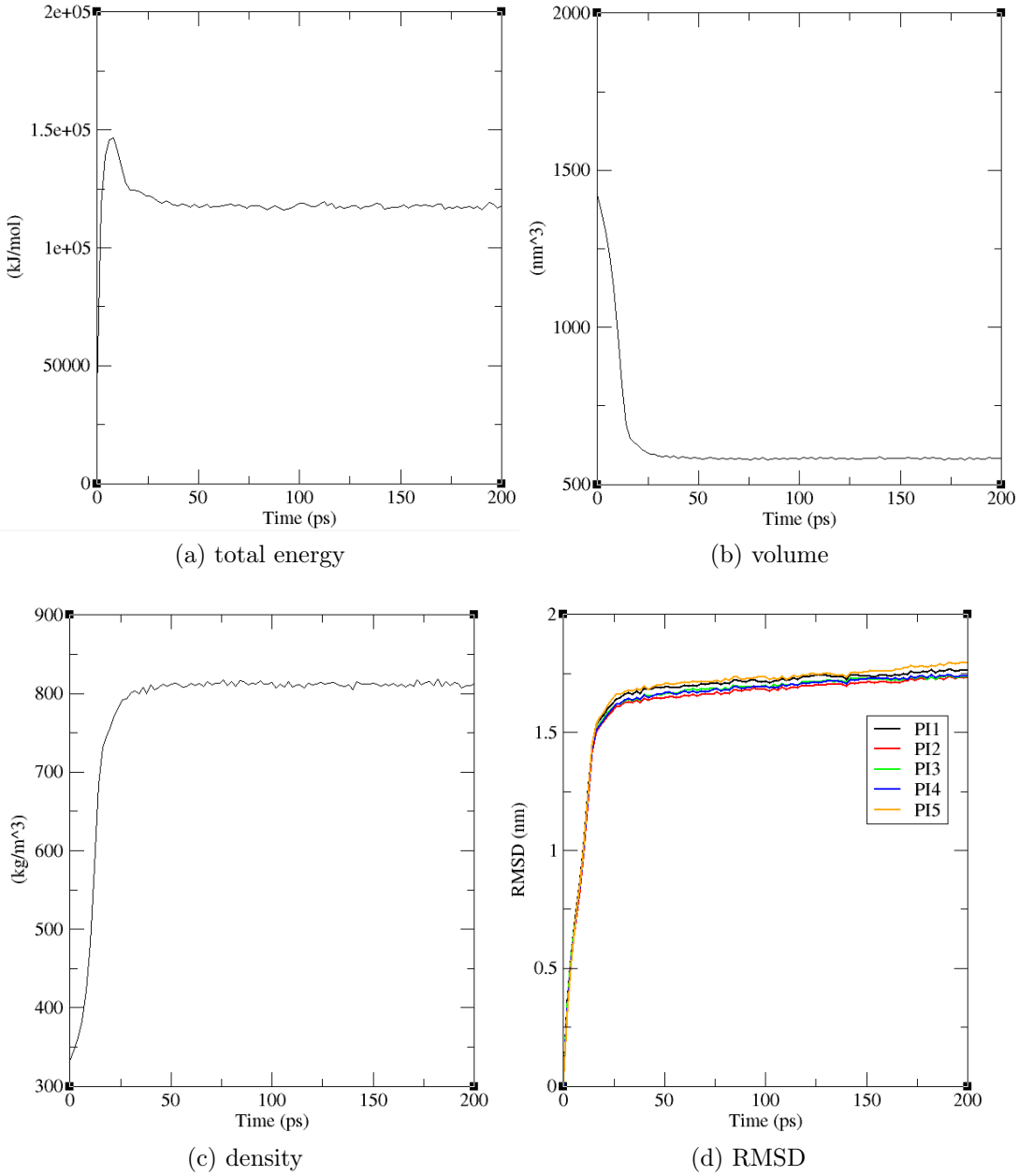
In the final part of this Tutorial, we demonstrate that *Assemble!*'s output can be directly submitted to Gromacs for molecular dynamics simulation. This requires Gromacs to be available on your computer, a molecular visualization software (e.g. VMD 3,...) and a graph plotting tool (e.g. xmgrace). Please note that all the expected simulation outputs are already provided in the folder `example/PI-example`.

1. **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 the input file `example/example-PI/em.mdp` for energy minimisation, and `example/example-PI/equil.mdp` for system equilibration over 200 ps in NPT ensemble at 350 K and 5 bar, using a 2 fs time step. Please note that the Gromacs input file provided here is consistent with Gromacs 4.5.
2. **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.

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 dependent on the chain length, but also on its monomer order sequence.



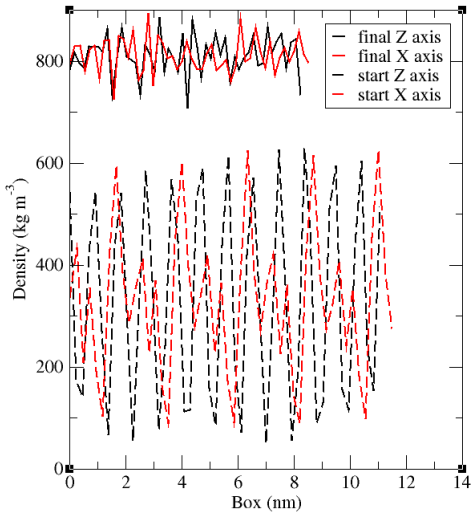
(a) total energy

(b) volume

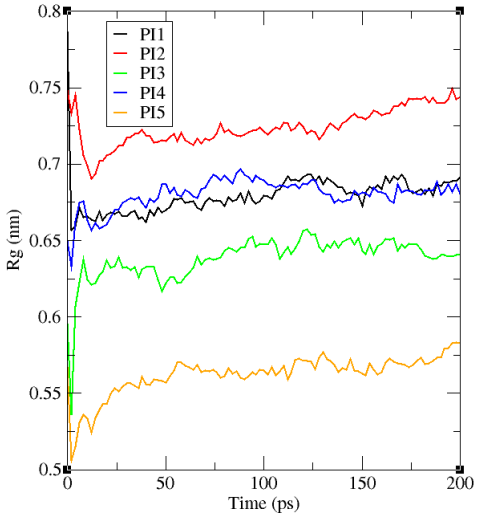
(c) density

(d) RMSD

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



(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.