1 Purpose of *run4amber*

run4amber performs MD simulations under various conditions, implicit or explicit solvent, using serial, parallel (MPI) or CUDA (GPU) infrastructure. The purpose is to automatically concatenate the different stages of an MD simulation in a single routine: minimisation, heat-up, equilibration, and production without intermdiate intervention of the user.

2 File types required or generated by *run4amber*

***.prm: input** Amber parameter-topology file of the system;

***.crd: input** Amber coordinate file of the system;

***.in: output** files with command sequences for the Amber modules *sander* or *pmemd* (and their variants); these are `minimize.in`, `mdheating.in`, `mdequilibration.in`, and `mdproduction.in`; the respective names are self-explaining;

restrt.*: output Amber restart files (continuously regenerated during the various phases of the run, i.e., `.min`, `.heat`, `.equil`, `.prod`; they are required to transit from one stage of simulation to the next; they are NETCDF formatted (i.e., binary);

do not remove while simulation is running!

mdinfo.*: output files with momentary info about the various stages `.min`, `.heat`, `.equil`, and `.prod`; useful only if you want to follow the progress of the simulation;

mdout.*: output files with very detailed information about the progress of the various stages `.min`, `.heat`, `.equil`, and `.prod`; useful especially to track possible problems, e.g., the `mdout.heat` file can be used follow if the required temperature (default = 300 K) is finally reached, the `mdout.equil` contains among others the density and is useful to verify if an explicit-water simulation reaches a density closer 1.0 g/cm3;

mdcrd.* : output trajectory files (binary NETCDF format) for the various stages `.min`, `.heat`, `.equil`, and `.prod`; especially `mdcrd.prod` is relevant, containing the complete production trajectory.

All output files have default names assigned automatically (no user influence on that). They will be overwritten without warning upon starting another simulation in the same folder! Therefore, it is good practice to create new folders for every new simulation when using *run4amber*.

For implicit-solvent simulations, no equilibration files are generated since this type of simulation uses only the minimisation, heating, and production phases, assuming that at the end of the heating process, the system is equilibrated enough to proceed to the production phase.

3 Running *run4amber*

For help, just type `run4amber` and RETURN. You then get this:

```
-------------------------------------
 run4amber version 1.1
 Romain M. Wolf (February 2019)
-------------------------------------
Usage: run4amber [options]

Options:
  -h, --help            show this help message and exit
  -p FILE, --prm=FILE   parameter-topology file    (default: com.leap.prm)
  -c FILE, --crd=FILE   Amber coordinate file      (default: com.leap.crd)
  -n INT, --np=INT      number of processors                (default: 4)
  --solv=STRING         explicit [exp] or implicit [imp]    (default: exp)
  --cuda                use GPU (CUDA) if possible           (default: no)
  --mask=STRING         Amber restraint mask           (default: no mask)
  --force=FLOAT         restr.force const. (kcal/mol/A2)      (default: 0)
  --ms=INT              minimization steps                 (default: 1000)
  --hs=INT              heat-up time [picoseconds]          (default: 100)
  --es=INT              equilibration time [picoseconds]    (default: 100)
  --ps=INT              production time [picoseconds]   (default: 1000 ps)
  --freq=INT            frame write frequency       (default: every 10 ps)
```

4 Command line options

`-p, --prm` must be followed by the Amber-compatible parameter-topology file of the system;

`-c, --crd` must be followed by the Amber-compatible coordinate file, corresponding to parameter-topology file;

`-n, --np` specifies the number of processors to be used; if no *sander.MPI* Amber routine is found, it is assumed that a single processor should be used and the number of processors automatically reverts to 1;

`--solv` specifies the solvent model: `exp` for explicit (TIP3P) water (default), `imp` for implicit (Generalized-Born with GB option `igb=5, gbsa=0`);

`--cuda` can be used to select GPU as processing unit (if installed and configured correctly);

`--mask` can be used to select atoms on which harmonic restraints are imposed; the selection must follow the Amber 'mask' notation; put the mask between `' '`, especially for complex mask expression;

`--force` specifies the force constant to be applied to restrained atoms (in $\frac{kcal}{mol \cdot \mathring{A}^2}$);

`--ms` specifies the maximum number of steps for the minimiser;

> **entering a negative value** will run the absolute number of steps and then stop without executing the rest of the simulation

`--hs` specifies the **picoseconds** to be used for heating up the system from 0 to 300 K;

`--es` specifies the **picoseconds** to be used for equilibration, i.e., adjusting the density of the system;

`--ps` defines the total length in **picoseconds** for the production phase;

`--freq` defines the frequency at which frames are written to the trajectory; example: with the default values, every 10 picoseconds over 1000 picoseconds would result in 100 frames saved to the trajectory output file;

> be careful with this value: too few saved frames (large frequency value) may make the trajectory worthless for later analysis, too many saved frames (small frequency) will lead to very large output files

5 Some technical details

5.1 Using GPU via `--cuda`

This only works if you have the right equipment configured correctly to run the Amber module *pmemd.cuda*. *run4amber* does not verify this in detail. Also, only runs on a single GPU are set up via *run4amber*.

5.2 Minimization

The minimisation part is not attempting to reach a very low residual gradient. It only tries to remove severe bumps that might lead to unstable MD phases later. If in doubt, you may increase the default number of minimiser steps (`--ms` option) from the default 1000 steps to something higher.

5.3 Heating

The heat-up phase brings the temperature linearly from 0 to 300 K over the specified time (100 picoseconds by default). It is run under constant volume (NVT). You can verify if the required temperature has been reached (within a few K) by looking at the final entries of the `mdout.heat` file. The 100 ps default is mostly OK, except maybe for very large systems with large amounts of water.

5.4 Equilibration

In the case of explicit water, the equilibration is required to adjust the density of the system to ca. 1 g/cm3 for a water box that extends by 12 Å or more on all sides of the periodic box beyond the solute. The equilibration phase is run under constant pressure (NPT). While the 'pressure' oscillates usually between unrealistic values (negative as well as positive), the density should slowly approach ca. 1 g/cm3. You can verify this by looking at the final lines of the output file `mdout.equil`. If the density is far from 1.0 g/cm3, stop the simulation and increase the equilibration time beyond the default of 100 ps.

On GPU runs (`--cuda`), the equilibration is switched to non-GPU, using then the maximum of processors specified by `-n, --np` (default 4). If the number of required processors is greater than 1, it runs *sander.MPI*, otherwise, a single-processor *sander* is executed, which of course slows down that part of the MD simulation. In the case of **implicit solvent** (GB), the NPT equilibration is skipped.

5.5 Production

If minimisation, heating, and equilibration (in case of explicit water) have created a 'stable' system, the production phase should run without issues. There is no 'optimum' length for an MD simulation. This is of course largely dependent on the purpose of the simulation. For later usage with techniques like MMGB/SA, anything between 1000 and 10000 ps (1 to 10 nanoseconds) is generally OK. Given the stochastic nature of MD, it is sometimes more useful to compute average properties of the molecular system (e.g., energy terms) based a few short to medium-long simulations rather than averaging over one single longer trajectory.

5.6 Running on SGE cluster

To run *run4amber* on the cluster, you can use an SGE file like this:

```
#$ -S /bin/bash
#$ -pe smp 8
#$ -cwd
#$ -l h_rt=8000000
#$ -j y
#$ -R y
#$ -V
run4amber ...followed by the command line option
```

and then submit it the usual way:

```
  qsub -N <jobname> <sgefilename>
```