Worksheet

Tutorial 1: Introduction of ESPResSo

Olaf Lenz

October 9, 2012

1 Good to know

- You can do the tutorial either on one of the ICP computers, or on your own laptop insofar it runs Linux or another POSIX operating system.
- The homepage of ESPResSo can be found at

  `http://espressomd.org`
- The ESPResSo User's Guide can be found on the home page (under the "Quick links"), or in the ESPResSo sources.
- Also, it might be useful to have a look at the FAQ on the ESPResSo homepage.
- For visualizing your simulation, the tool VMD is helpful. If you want to use your own computer, you can download VMD from its homepage [1]. On the ICP computers, you can start VMD from the command line:

```
vmd &
```

- All files used in the tutorial can be found on the workshop's homepage

  `http://espressomd.org/wordpress/ess2012/`

2 Building ESPResSo

Compiling ESPResSo is one of the necessary evils of using ESPResSo, as you can add and remove some of the many features of ESPResSo. This is necessary, as on the one hand some features cannot work together, and on the other hand the code simply runs faster when not all features are activated.

[1] `http://www.ks.uiuc.edu/Development/Download/download.cgi?PackageName=VMD`

2.1 Prerequisites

ESPResSo only has one mandatory prerequisite, namely the Tcl scripting language interpreter (>= Tcl 8.3)[2]. If you want to use ESPResSo in parallel, you will furthermore need an MPI implementation[3]. Electrostatics, *i.e.* the P^{3}M algorithm, that will be needed in the second tutorial, additionally requires the FFTW library [4].

- All Linux distributions that we know of and many other POSIX systems provide packages for Tcl, MPI and FFTW, thus they can be easily installed on the machine, provided that you are administrator of the machine.
- If you are not the administrator of the machine, you can usually download Tcl and compile the packages yourself and install them into your home directory. Only in the case of MPI on HPC clusters this might be problematic.
- Many distributions (for example Ubuntu, Debian, SUSE) split the packages into the binary package itself (*e.g.* `tcl`) plus an extra "development package" (*e.g.* `tcldevel`) that contains the include files of the package. To be able to compile ESPResSo, you will need to install *both* packages!
- In Linux distributions, you can usually choose between a number of MPI implementations, namely OpenMPI, MPICH or LAM/MPI. OpenMPI is the descendant of LAM/MPI, so you should prefer OpenMPI over LAM/MPI. It is usually not a good idea to install more than one MPI implementation at once, as this might easily confuse ESPResSo's build system (and the build system of other packages, too)!

2.2 Compiling the source code

- The source code of ESPResSo can be downloaded from ESPResSo's homepage [5].
- You can unpack the sources using

```
tar -xvf Espresso-3.1.1.tar.gz
```

- In general, compiling ESPResSo requires the following two commands

```
configure
make
```

- To activate or deactivate the various features of ESPResSo, you can create a file `myconfig.h` in your ESPResSo directory. A template for this file is the file

[2]`http://www.tcl.tk`
[3]*e.g.* `http://www.open-mpi.org`
[4]`http://www.fftw.org`
[5]`http://espressomd.org/wiki/Download`

myconfig-sample.h. Simply copy the file and rename it, and uncomment the features that you want to activate. The file will then automatically be used during compilation. Note, that if you change the file, it is enough to run make, it is not required to run configure again.

- If all prerequisites of ESPResSo are installed in the standard places on your system, this might already be enough to compile ESPResSo, maybe even in parallel. In other cases, you will have to give some options to configure or define some variables. To find out what is needed, have a look at the configure online help (configure --help), the User's Guide or the FAQ.
- When you compile ESPResSo on a multicore machine, you can speed up compilation by using make -j*n*, where *n* is the number of processes that will be started in parallel. A good choice for *n* is the number of cores plus one.
- After you have successfully compiled ESPResSo, you can use

  ```
  make check
  ```

  to run a testsuite that will run several small tests on ESPResSo.
- Once you got it compiled, you can run ESPResSo via

  ```
  Espresso script.tcl
  ```

Suggested exercises

- Try to compile ESPResSo, either on the ICP computers, on a laptop, or wherever you like to try. The tutors are around and can try to help you!

3 The sample scripts

- The scripts generate .dat and/or .rdf files. These files contain tabular data that can be plotted using for example gnuplot or xmgrace:

  ```
  xmgrace -nxy lj-0.3.rdf
  ```

- The scripts also generate .vtf-files that contain the trajectories of the corresponding simulations. You can visualize them using VMD.

  ```
  vmd lj-0.3.vtf
  ```

  Note, that by default ESPResSo writes out all positions in absolute coordinates, *i.e.* they are not folded back into the primary simulation box. When you want to visualize the system, you will either have to write out the positions in folded coordinates (look into the User's Guide how to do this!), or use the PBCTools that come as part of VMD to wrap the coordinates once they are loaded into VMD. To do that, use the following command in VMD's console:

```
    pbc wrap -all
```

For this tutorial, we have prepared two sample scripts.

3.1 Lennard-Jones fluid

The script `lj.tcl` is a simulation script for a Lennard-Jones fluid. The script demonstrates well how to use ESPResSo with a very simple system. It can be run in parallel, although it doesn't scale very well due to the small system size!

Suggested exercises

- Go through the script line by line and try to understand what is happening.
- Run the script for different values of the temperature and observe the different phases of the LJ system. Have a look at the different radial distribution functions (`lj-*.rdf`), and visualize the system at the different phases. You can compare the results directly to Hansen and Verlet's article from 1969[6]!
- Increase the system size and investigate ESPResSo scaling behavior for larger systems.
- You can also start two parallel tasks of ESPResSo using the command

  ```
      mpiexec -n 2 Espresso lj.tcl
  ```

  Investigate the parallel scaling of ESPResSo. Remember that ESPResSo scales better for larger systems!
- Extend the script so that the simulation stores the current simulation state into a file and that you can restart the simulation from there. The `blockfile` command might be handy for this.

3.2 Stretched polymer

The script `stretched_polymer.tcl` models a single polymer that can be stretched. The script demonstrates how to use bonds and exclusions to create molecules, and how to use Tcl to create complex simulation setups.

The polymer consists of beads ("monomers"), that are bound to each other via harmonic spring potentials and form a chain. To generate a polymer of seemingly infinite length with various degrees of stretching, the polymer was periodically repeated along the z-axis of the system, *i.e.* the last monomer of the chain was bound to the first monomer of the chain over the periodic boundary (see figure 1). In that case, the degree of stretching can be controlled via the system length L_z along this axis, and is controlled by the relation between the system length L_z and the contour length L_{contour} of the polymer.

[6]`http://link.aps.org/doi/10.1103/PhysRev.184.151`

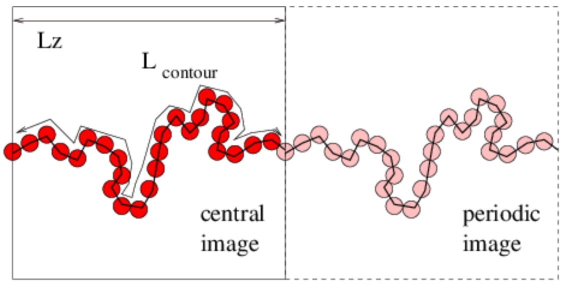

Figure 1: Sketch of the system with a stretched, periodically repeated polymer. The degree of stretching is defined by the relation of the contour length of the polymer L_{contour} to the system length in the axis of stretching L_z.

Suggested exercises

- The goal is to create 10 equilibrated, statistically independent polymer configurations for further analysis. To be able to do this, you will have to do the following things:
  - Determine when the system is equilibrated. To do that, have a look at the plots of the different observables.
  - Determine the autocorrelation time of the system, *e.g.* by using the function `uwerr` on the different observables. Now you know how many simulation steps need to be done between statistically independent configurations.
  - Once you have determined the autocorrelation time, adapt the parameters of the simulation such that it generates independent configurations.