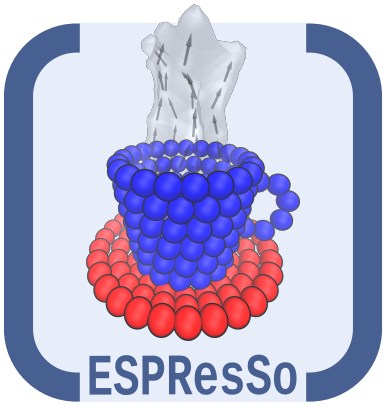

Setting up ESPResSo *

October 5, 2017

1 Introduction

ESPResSo is distributed as source code because this allows the user to disable features that are mutually exclusive or would otherwise decrease the performance unnecessarily. Users typically build a custom ESPResSo binary for a specific simulation, containing only the minimal set of features required for this application. This tutorial will guide you through this whole process from obtaining the source code of the development version, to configuring and building binaries, to compiling the documentation.

2 Getting the Source Code

2.1 Download the Source Code for the Development Version

For the development version, first clone the ESPResSo reposity from github by running

```
git clone https://github.com/espressomd/espresso.git
```

*For ESPResSo 3.4-dev-5437-g54a1137-dirty

in a terminal. This creates a local copy of the current ESPResSo repository in the folder `espresso`. Enter the newly create directory using

```
cd espresso
```

It is recommended to create a `build` directory with

```
mkdir build && cd build
```

that will contain all the data that is generated in the following. Next, run

```
cmake ..
```

to configure the build system.

2.2 Compiling ESPResSo

Now compile ESPResSo from within the build subfolder with the command

```
make -j 8
```

With the option `-j` you can specify how many threads are used for the make process which will speed up the compilation. It is good to know that you can maintain several build folders with different feature sets using the same ESPResSo source folder. They don't have to reside within the source directory. If you have installed the curses frontend to `cmake` you can see and modify the build options graphically by running

```
ccmake ..
```

before the `make` command. ESPResSo requires a number of external libraries depending on the feature set you plan to use. Most, if not all of these libraries will be available through your Linux distribution's repositories. If you manually install dependencies in a non-standard location, you have to specify this location with the `ccmake` command. If, for example, you plan to use GPU features and your cuda toolkit is not located under `/usr/local/cuda`, you will have to specify its location explicitly. Remember, after you changed configurations with `ccmake`, the new configuration set have to be built by using the `make -j 8` command again. This produces an executable named `pypresso` in the build directory which can be run by

```
./pypresso
```

2.3 Documentation

For the development code and upcoming releases see the Online Documentation for ESPResSo with python at `http://espressomd.org/html/doc/index.html`.

As an expert option, you can locally create this html documentation by running

```
make sphinx
```

in your build directory. The resulting web pages can then be browsed by opening `build/doc/spinx/html/index.html`.

2.4 Configuring features

After successfully running the `cmake` script, your build folder will contain a file `myconfig-sample.hpp`. Create a copy of this file called `myconfig.hpp` by executing

```
cp myconfig-sample.hpp myconfig.hpp
```

Then uncomment the respective lines in `myconfig.hpp` (by removing the leading slashes) to enable features your simulation requires. The User's Guide contains information about required features for every ESPResSo command. When you activate or deactivate features, you have to `rebuild` the binaries by running

```
make -j 8
```

in the build directory.