Introduction to ESPResSo

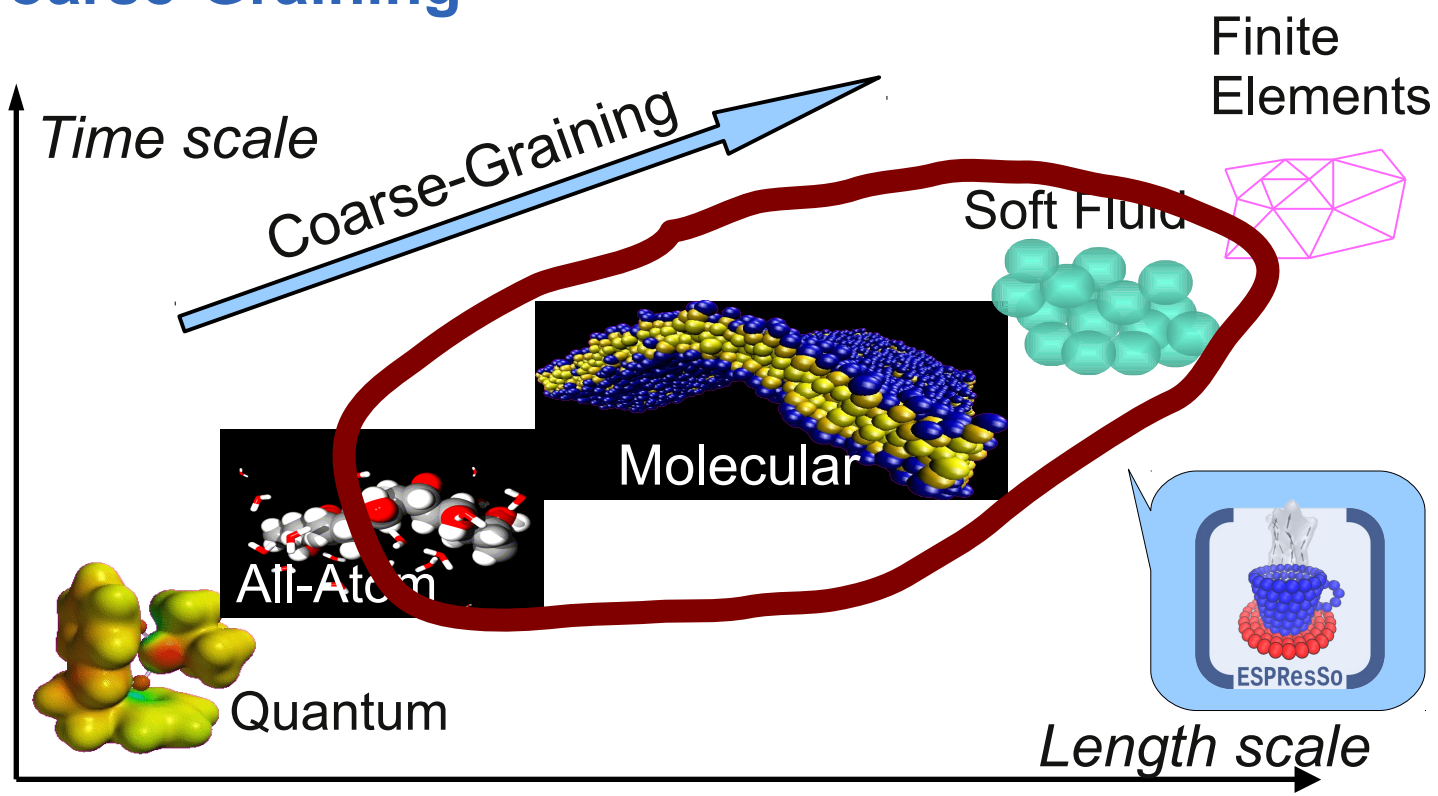
Olaf Lenz

Institut für Computerphysik, Universität Stuttgart
Stuttgart, Germany

University of Stuttgart
Germany

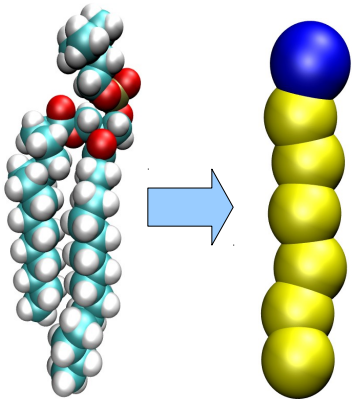
INSTITUTE FOR COMPUTATIONAL PHYSICS

Coarse-Graining



- All-Atom Models
  - Model all atoms and their interactions with semi-quantitative parameters
  - Only small systems and short times can be simulated
- Coarse-Grained Models
  - Only model “important” degrees of freedom
  - Allows for much larger time and length scales

Why yet another Simulation Package?



- Coarse-grained Bead-spring models:
  Combine several atoms into a single bead
- Often combined with other methods
  - Special interactions (DPD, Gay-Berne ellipsoids, ...)
  - Special integrators (MCPD, Hybrid MC/MD, ...)
  - Combined with lattice models (Lattice-Boltzmann, MEMD, ...)
  - Uncommon simulation protocols (Simulated annealing, Parallel tempering, …)
  - Special constraints (Walls, Pores, …)
- Standard MD simulation packages (GROMACS, NAMD, AMBER, …) are not flexible enough to deal with these models

⇨ **Package must be flexible!**

- In research, new methods are developed
- Building new methods into highly optimized code (GROMACS, NAMD, AMBER, …) is very hard

⇨ **Package must be extensible!**

Philosophy

- ES is intended as a research tool and a production platform
- ES provides the methods for coarse-grained simulations
- However, an understanding of the methods is required to be able to use ESPResSo
- ES can not check whether what you do makes sense!

Golden Rules

1. ESPResSo can *not* be used as a black box
2. ESPResSo does *not* do the physics for you

Methods

- **Integrators and ensembles**: Velocity-Verlet algorithm (NVE), Langevin thermostat (NVT), Barostat by Dünweg (NPT), Generalized Hybrid Monte-Carlo, Quarternion integrator for non-spherical particles or point-like dipoles, ...
- **Nonbonded interactions**: Lennard-Jones, Gay-Berne, Buckingham,, …
- **Bonded interactions**: harmonic, FENE, tabulated, bond-angle interaction, dihedral interaction, ...
- **Long-range interactions**: for electrostatics: P³M, MMM1D, MMM2D, Ewald, ELC and MEMD; for point-like dipoles: dipolar P³M, ScaFaCoS (FMM, ...), ...
- **Hydrodynamic interactions**: DPD, Lattice-Boltzmann fluid (on GPGPU) coupled to particle simulation
- **Constraints**: Particles can be fixed in any directions; walls, pores, spheres...
- **Analysis**: energy components, pressure tensor, forces, distribution functions, structure factors, polymer-specific analysis functions (radius of gyration, ...), output to VMD
- **...and it is continuously growing...**

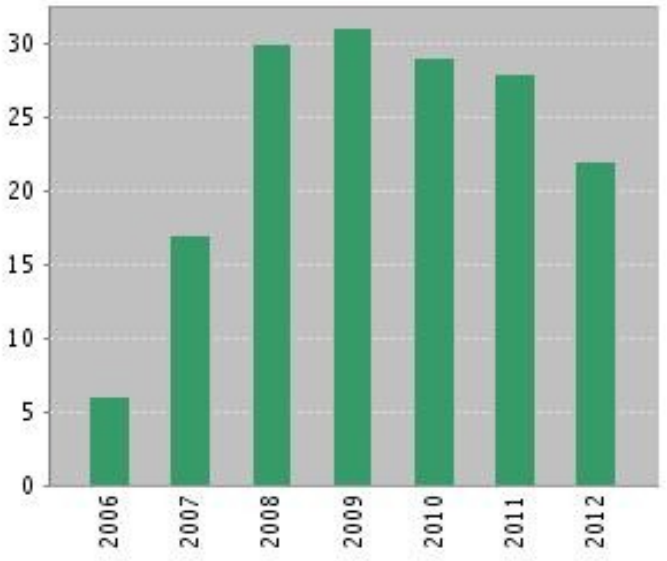
Availability

- Free, open-source
- Source code hosted at GNU Savannah (not a GNU core project, though)
- GNU General Public License (GPLv3)
  - Code may be freely downloaded, modified and redistributed
  - Provided that the GPL is kept
- Portable
  - Can be easily compiled and run on all POSIX platforms (think: Unix) known to us
  - Windows?
- Distribution packages exist for
  - Gentoo Linux (Christoph Junghans)
  - Fedora Linux (Thomas Spura; in progress)
  - … anybody interested in packaging for other distributions?

https://savannah.nongnu.org/projects/espressomd/

Distribution

- 163 Citations of the 2006 article (Web of Knowledge)
- Used by ~20 scientific working groups

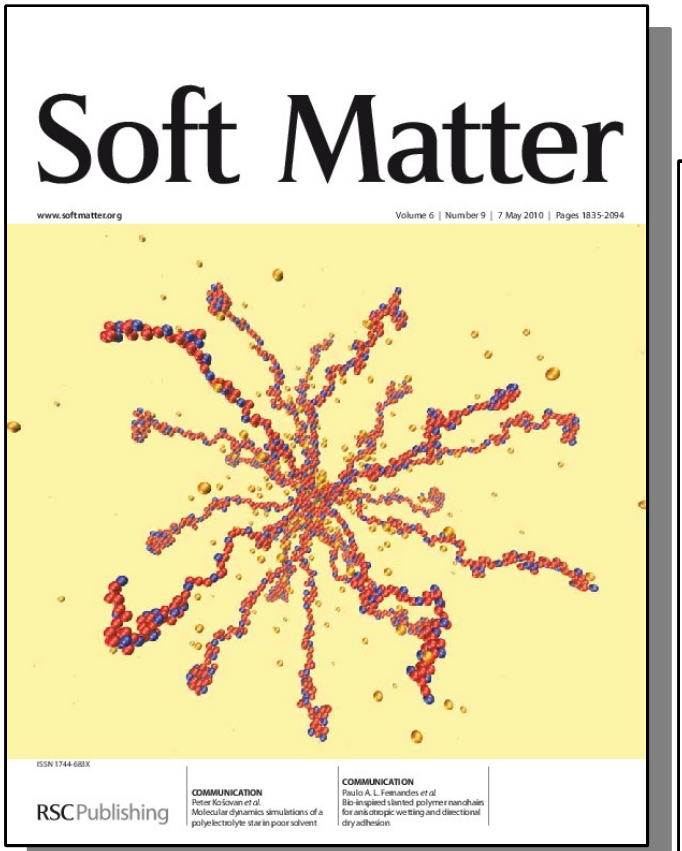


Citations per year

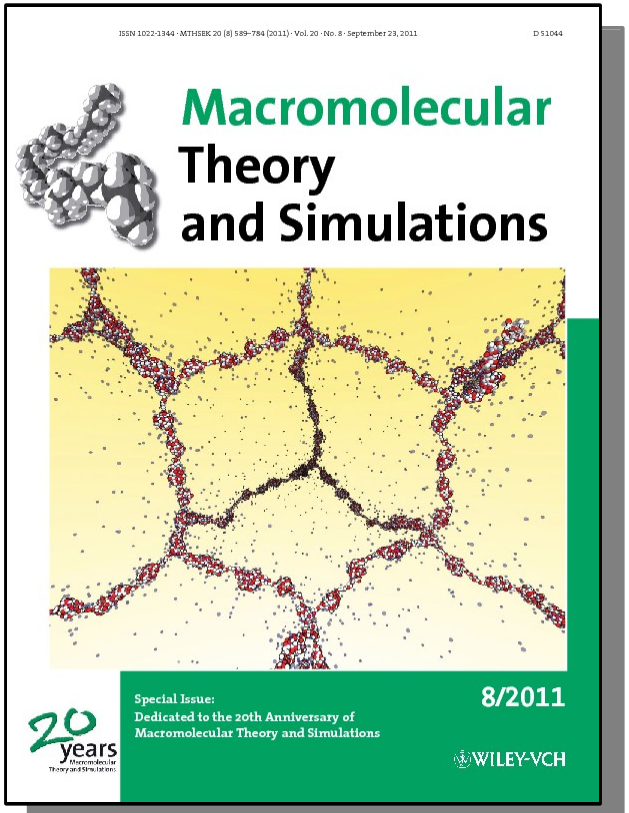
(from Web of Science, including self-citations)

Impact

2007



2010



2011

Code

- From Ohloh (https://www.ohloh.net/p/ESPResSo_MD)
- 4,946 commits from 59 Contributors
- ~ 123,650 lines of code
- Estimate: ~31 person years, ~1.7M$ cost

In a Nutshell, ESPResSo Soft Matter Simulation Software...

...has had 4,946 commits made by 59 contributors
representing 123,650 lines of code

...is mostly written in C
with a well-commented source code

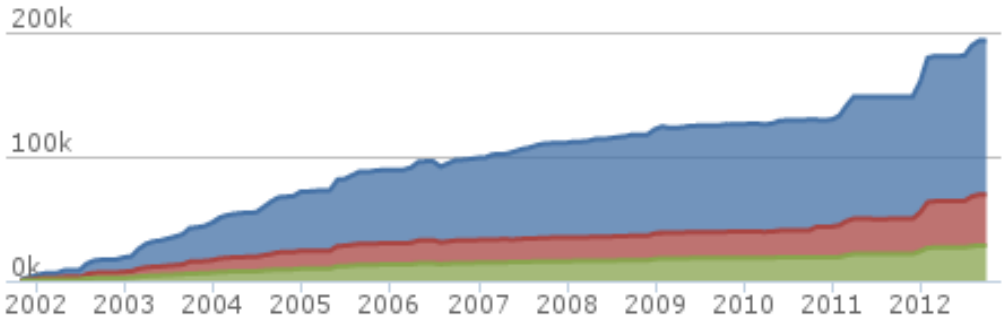
...has a well established, mature codebase
maintained by a large development team
with stable year-over-year commits

...took an estimated 31 years of effort (COCOMO model)
starting with its first commit in November, 2001
ending with its most recent commit 2 days ago

Languages

C	58%	Tcl	22%
TeX/LaTeX	13%	11 Other	7%

Lines of Code



Project Cost Calculator

Include: All Code

Average Salary (per year): $ 55000 .00

Codebase Size
123,650 lines

Estimated Effort
31 person-years

Estimated Cost
$ 1,689,603 *

*Using the Basic COCOMO Model

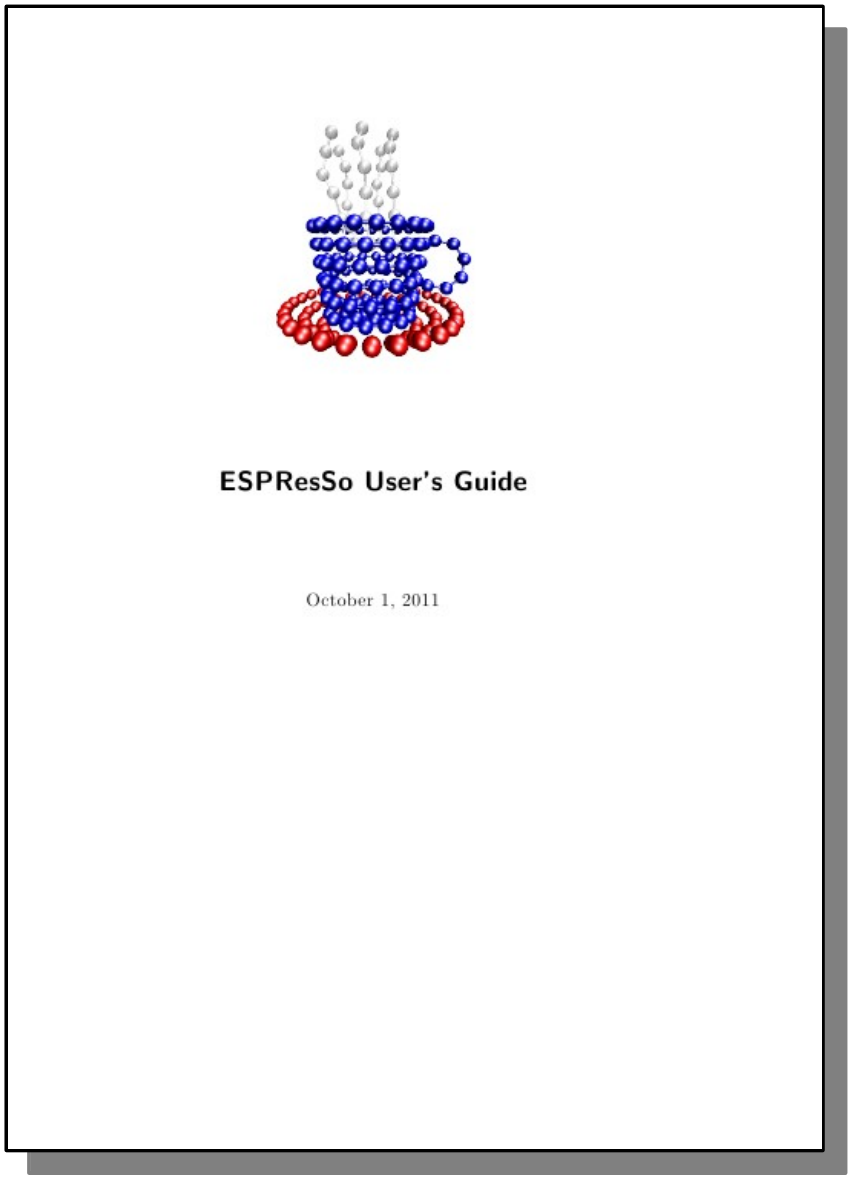
Web resources

- Home page
  http://espressomd.org
  - Hosted at ICP
  - Central resource for users
  - Downloads
  - Documentation → next slide
  - Community and Support
    - (Link to) Bug tracker
    - (Link to) Mailing lists
    - (Link to) Wiki
  - Developer's Zone (in the wiki)
    - Developer's docs
    - (Link to) Savannah project
    - (Link to) Source code repository

- Savannah Project Page
  https://savannah.nongnu.org/projects/espressomd/
  - Hosted at GNU Savannah servers
  - Download area
    - Releases and NEWS
  - Mailing list
    espresso-users@nongnu.org
    - Only mailings from members are accepted
  - Bug tracker
    - Report bugs in releases!
  - Mostly intended for Developers
    - Mailing list
      espresso-devel@nongnu.org
    - News
    - Source code repository
    - Task manager
    - Patch manager

User Documentation

- User's Guide
  - PDF document
  - In release package
    - Off-line
    - Matches the release
  - On web site (from build server)
    - Up-to-date
    - Contains ToDo-Boxes
  - Outline
    - Introduction
    - First steps: Quick start
    - Rest: Reference manual
- FAQ (on home page)
  - Not very complete
  - Please contribute!
- Mailing list archive
- Bug tracker



Getting Help

- RTFM!
  - FAQ
  - User's Guide
  - Mailing list archives
- Use Mailing List
  - Include version, OS, features
  - Also send replies to the list
    - If you send huge files, better provide a link
    - In a long, detailed discussion you can just send a summary at the end
  - Please remember: the developers are not paid for replying!
  - Please do not write to developers personally
    - All mailings are archived so others can benefit
    - Mailing list reaches everybody

Requirements

- C-Compiler (GNU CC is best tested)
- Bourne shell, GNU make
- Tcl
  - Including headers / devel package!
- Optional
  - FFTW
    - Including headers
    - Required for P3M
  - MPI
    - e.g. OpenMPI, MPICH
    - Including headers
    - Required for parallel execution
    - Useful to know how to use it
  - CUDA
    - For LB-GPU code
    - More in the future?

FFTW

Open MPI: Open Source HPC

MPICH2

NVIDIA CUDA

Compiling

- Necessary evil
  - Few binary packages exist
  - For optimal performance, recompilation is necessary
- Typically 3 steps
  - Configure code `configure`
    - Use `--help` to get options
    - Use `CPPFLAGS` and `LDFLAGS` when libraries are installed non-standard
    - Logfile `config.log` contains additional information
  - Compile code `make`
    - Use `-j` *np* to compile in parallel
  - Run testsuite `make check`
    - Use `processors="1 2"` to specify the numbers of tasks
- Installation is usually *not* required
- Separate *source* and *build* dir
  - The Source dir is where the source code resides
  - The Build dir is where all files created by the compilation are created
  - No file in the Source dir is modified
  - Call configure from the build dir

```
cd $builddir;
$srcdir/configure
```

Activating and Deactivating Features

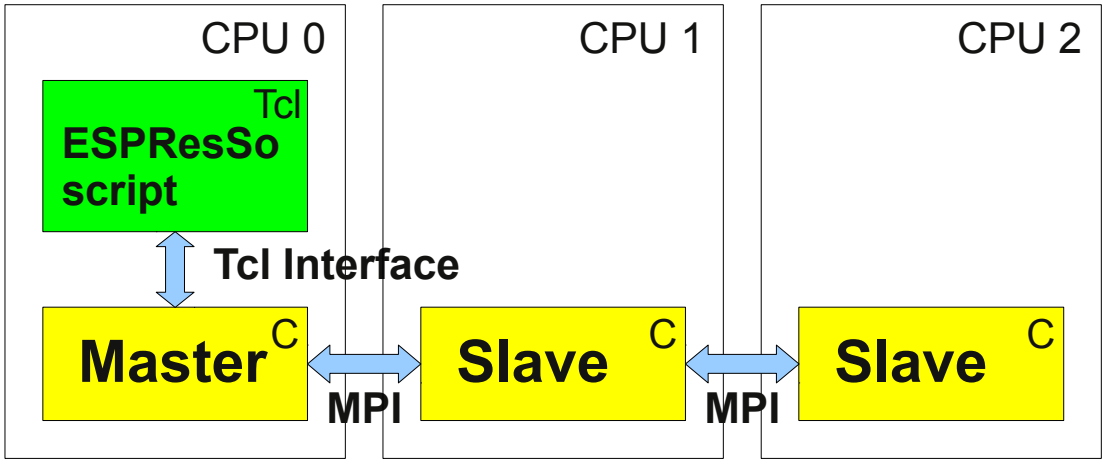
- ESPResSo supports various different features
- Not all features are compiled in
- To check, call `code_info`
- Create file `myconfig.h` in build or source dir to change the default set of features
- Use minimal set of features for optimal performance
- The term “features” is probably not well chosen
  - The code has a lot of features that do not have a compiler switch
  - Goal: remove all features

```
#define PARTIAL_PERIODIC
#define ELECTROSTATICS
#define DIPOLES
#define ROTATION
#define ROTATIONAL_INERTIA
#define MDLC
#define EXTERNAL_FORCES
#define CONSTRAINTS
#define MASS
#define EXCLUSIONS
#define COMFORCE
#define COMFIXED
#define MOLFORCES
#define MODES
#define BOND_VIRTUAL
```

Example myconfig.h

Architecture

- Simulation core
  - Written in ANSI C
  - MPI parallelized
  - Optimized



- Control layer
  - Simulation core is controlled via the scripting language Tcl
  - High-level Tcl commands to control the simulation and analyze the system
  - A simulation is defined by an “ESPResSo script”
  - Tcl script is *not* executed in parallel!

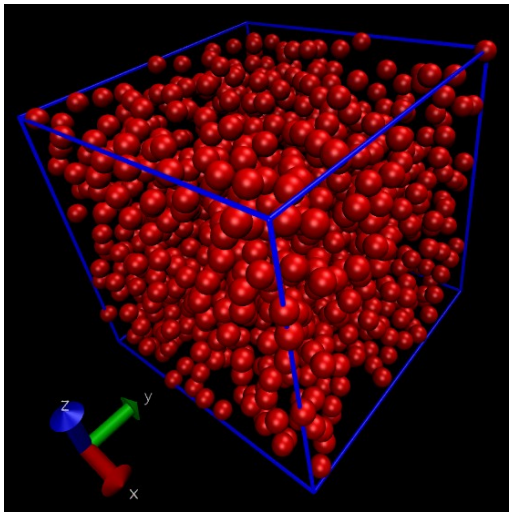
```
.
.
setmd box_l 10.0 10.0 10.0
integrate 1000

# compute kinetic energy
set e_kin \
  [analyze energy kinetic]
.
.
```

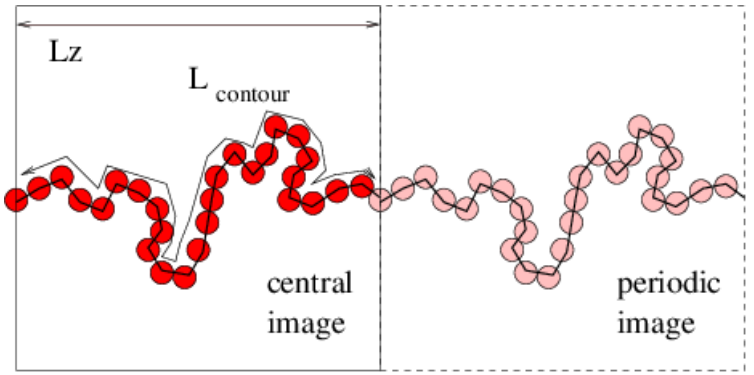
Example ESPResSo script

Writing an ESPResSo Script

- Example files: `lj.tcl` and `stretched_polymer.tcl`
  - `lj.tcl`: Lennard-Jones fluid
  - `stretched_polymer.tcl`: Stretched polymer
- Outline
  - Set up the system
  - Set up the particles
  - Set up the interactions
  - Running the simulation
    - Warmup integration
    - Main integration
  - Analysis
- Sections correspond roughly to chapters in UG
- Detailed command syntax can be found in UG



Snapshot of the LJ system



Schema of the stretched polymer

Setting Up the System: Global Variables

- Set global variables with `setmd varname value`
- e.g. Box size, Periodicity, Time step, Skin size, Cell size, …
- Many are set to sensible defaults
- Get global variable with `setmd varname`
- Many commands can be used without argument to get information

```
# define the system size
setmd box_l $box_size $box_size $box_size

# set up the integrator time step
setmd time_step 0.01

# the skin has no effect on the result, only on the speed
setmd skin 0.4

# uncomment the following to output the box size
#puts [setmd box_l]
```

Setting Up the System: Thermostat

- Command `thermostat`
- Misnomer: used to set ensemble (e.g. Barostat)
- Turn on Langevin thermostat

  `thermostat langevin temperature gamma`
- Turn off thermostat `thermostat off`
- Other “thermostats”
  - npt_isotropic (NPT)
  - Generalized Hybrid Monte-Carlo (GHMC; NPT and more)
  - DPD (see Friday)

```
# set up the thermostat
set langevin_gamma 1.0
thermostat langevin $temperature $langevin_gamma
```

Setting Up the System: More

- `Nemd:` "Non-equilibrium MD": special method for creating a shear flow
- `cellsystem`: Changing the cell system
  - Turn on Domain decomposition (default)
  - Turn off Verlet lists
  - Turn off Cell lists (nsquare)
  - Use "layered" system (only for MMM2D)
- `adress`: Turn on ADResS
- `cuda`: Set up CUDA device
- `collision_detection`: Turn on collision detection
  → Recent developments
- `reactions`: Turn on reactions → Recent developments

Setting Up Particles

- Create a single particle: `part` *`pid arguments`*
  - *pid* specifies a numeric id
  - Holes in *pid* order cost memory
- Possible arguments: position (required in first call), velocity, charge, mass, type
- Create bonds to other particles (`bond`)
- Fix particle in one or more directions (`fix`)
- Apply external force to particle (`ext_force`)
- Set individual temperature
- Make it virtual → Recent developments
- Delete a particle (`delete`)
- Get particle properties `part print` *`arguments`*

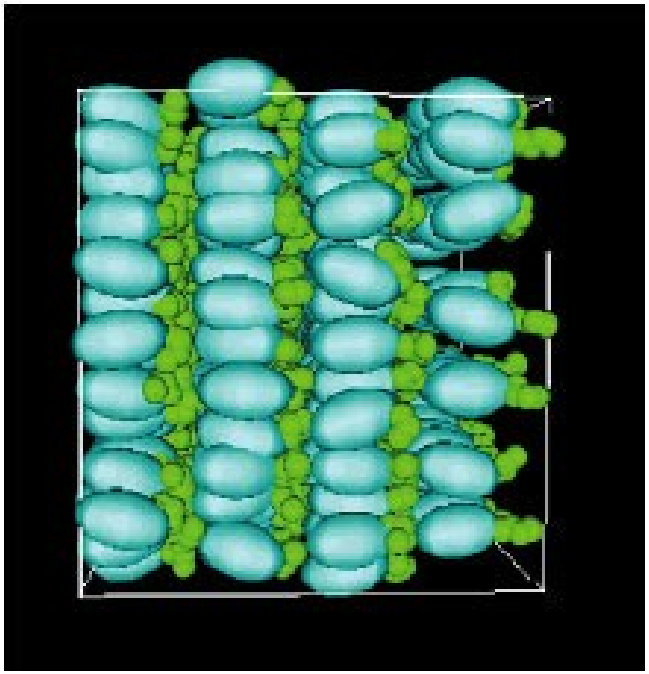
```
# generate $n_part particles at random positions
for {set i 0} { $i < $n_part } {incr i} {
    set x [expr $box_size*[t_random]]
    set y [expr $box_size*[t_random]]
    set z [expr $box_size*[t_random]]
    part $i pos $x $y $z type 0
}
```

Creating Groups of Particles

- Tcl commands to create many particles at once
  - Polymers `polymer`
  - Counterions `counterions`
  - Salt `salt`
  - Diamond polymer networks `diamond`
  - Icosaeder `icosaeder`
  - Copy existing particles `copy_particles`
- Create extended objects `constraint`
  - Walls, Spheres, Cylinders, Pores, Rods, Rhomboid, Planes
  - External magnetic field

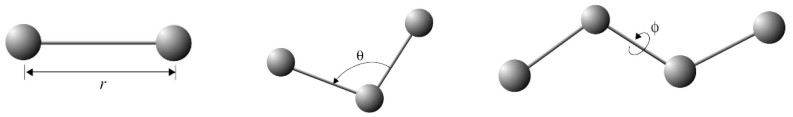
Oriented Particles

- Feature `ROTATION`
- Particles have an orientation
- Useful for
  - GB-ellipsoids
    (coarse-grained liquid crystals)
  - Directional Lennard-Jones
  - Point-like dipoles
    (→ dipolar P3M, Thursday)
- Quarternion integrator
- Roughly 30% slower!
- Currently, all particles have the rotational degrees of freedom
- Switchable rotation is under review



Setting Up Interactions

- *Non-bonded* Short-range Interactions
  - Work between particle *types*
  - Lennard-Jones, Morse, Buckingham, Smooth-step (DPD), …
  - Tabulated
- Bonded interactions
  - Work between two (or more) specific particles (can be set in `part`)
  - Have a *bondid*
  - Bond-length 2-body interaction: Harmonic, FENE, …
  - Bond-angle 3-body interaction: Harmonic, Cosine, …
  - Dihedral 4-body interaction
  - 2-body interactions can be made rigid
    - Not well-tested
- Long-range Interactions
  - Electrostatics
  - Magnetostatics (point-like dipoles)
  - Hydrodynamic interactions
  - → next days
- No force fields built in!




```
set lj_epsilon 1.0
set lj_sigma 1.0
set lj_cutoff 2.5
inter 0 0 lennard-jones \
  $lj_epsilon $lj_sigma \
  $lj_cutoff
puts "Interactions:\n[inter]"
```

Exclusions

- Often, neighboring particles in a chain should not interact via non-bonded interactions
- Two possibilities
  - Bonded subtracted LJ potential `lj_subst`
  - Exclusions (Feature `EXCLUSIONS`)
    - Explicitly exclude interactions between particles `part exclude`
    - Automatically exclude some interactions `part auto_exclusions`

Running the simulation

- Main integrator: Velocity Verlet
- Do a number of integration steps `integrate` *steps*
- Use `integrate 0` to update the forces or positions of virtual sites
- Warmup integration
  - Cap the maximal force `inter ljforcecap` *F_max*
  - Prevents overlapping particles and very high forces
  - Do steps until the large forces disappear
- Main integration
  - Switching between Tcl and C has an overhead
  - Do as many steps in a single `integrate` command
- Advanced commands for integration
  - Parallel tempering
  - Metadynamics

```
for { set i 0 } { $i < 100 } { incr i } {
    integrate 1000
    .
    .
```

Analysis

- Analysis can be done on Tcl level!
  - Use `part print` to get particle positions, velocities, etc.
  - Allows for anything you can think of
  - … but maybe slow
    - slow numerics in Tcl
    - not parallel
- Parallel Analysis in C `analyze`
  - Many different observables
    - Energies, pressures, stress tensor…
    - Minimal distances, RDF, structure factor, …
    - Polymer observables: end-to-end distance, radius of gyration, …
  - Analyzing several configurations
    - Store configuration `analyze append`
    - Retreive stored configurations `analyze configs`
    - Some analysis commands can handle stored configs (e.g. `analyze <rdf>`)
- More Advanced Analysis → Recent developments

File I/O and Visualization

- No general ESPResSo-format
  - What should be stored in the file?
    - Particle positions, box size
    - Interactions, Tcl variables?
- No real checkpointing
  - It is not clear what belongs to the state of the simulation
    - e.g. RNG state, number of tasks, …
- Blockfile format
  - Allows to write specified blocks of information
    `blockfile $f write particles`
  - Can read in these blocks
    `blockfile $f read auto`
- Jump into the main loop needs to be done manually

- Visualize best with VMD
  - Off-line
    - Recommended: Create VTF files
    - `writevsf` to output the structure into the VTF file
    - `writevcf` to output a configuration into the VTF file
    - Can also create PSF and PDB
  - On-line
    - VMD has a protocol for on-line visualization
    - In general, off-line is more useful

That's all.
Thanks for your attention!