

Goals and Intention

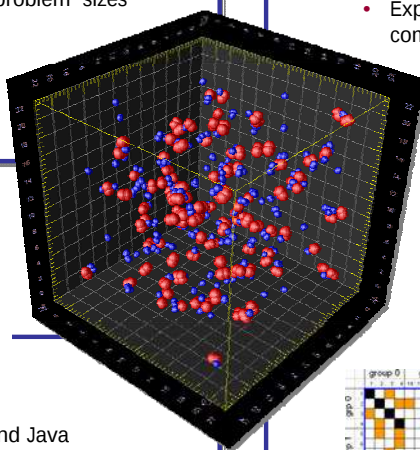
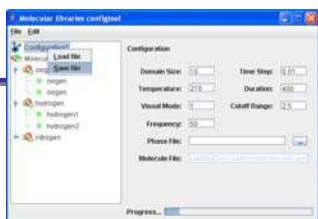
MolDyn is primarily conceived as a collaboration between students with different backgrounds that is aimed at delivering a scientific software product from scratch by applying Software Engineering practices.

Software Engineering challenges:

- Developing a fully working application for the simulation of phenomena at atomic level
- Respecting milestones, discussion of functional requirements with the client, distribution of roles, documentation of the project
- Project Planning

Technical challenges:

- Take account of rotational energy including thermostat
- Support for the simulation of mixtures
- Implementation based on $O(N)$ algorithms with respect to the number of molecules (assuming the molecules are equally distributed in space)
- Parallel implementation to support large problem sizes with a nearly linear scale-up
- Easy to use graphical interface



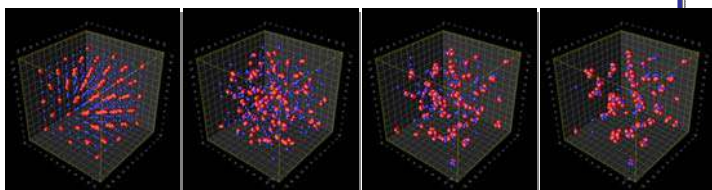
Implementation and results

Implementation:

- Platform independent source code in C++ and Java
- XML files:
 - Configuration file which contains initialisation parameters and phase space
 - Molecular library which stores already composed molecules
- Configuration tool for specifying molecule types and simulation parameters
- Checkpoints which enable the user to interrupt and resume a simulation run
- Parallelisation with MPI 2.0
- Visualisation in OpenDX

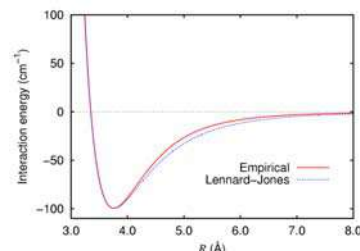
Results:

Visual output of realistic molecule movement and clustering

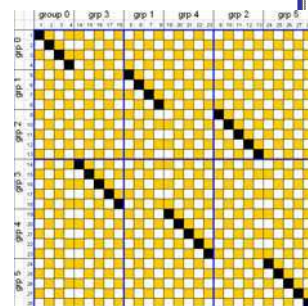
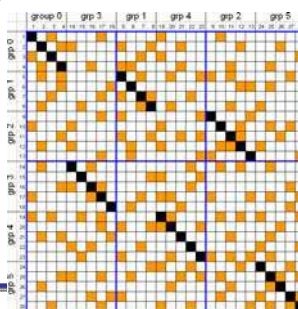


Beyond the Code

- Numerical solution of Newton's equations of motion based on the empirical model of the Lennard-Jones 6-12 potential



- Molecules are treated as rigid bodies
- Dimensionless formulation of the translational and rotational equations avoids numerical problems due to small time and length scales
- Time integration performed by explicit leap frog algorithm
- Use of the linked cell algorithm to implement the cut-off range to achieve $O(N)$ complexity
 - Exploitation of Newton's third law to further reduce computational costs
- Parallelisation using the force decomposition scheme as opposed to domain decomposition and the replicated data approach. This results in a scale-up of $O(N/\sqrt{P})$.
- Parallel linked cells



Upcoming and Future Work

For this release:

- Completion of the parallel output
- Enhancement of the GUI
- Parallel linked cells
- Documentation and short user manual

For future releases:

- Tuning of the program towards specific applications
- More sophisticated configuration set-up
- Cache optimisation