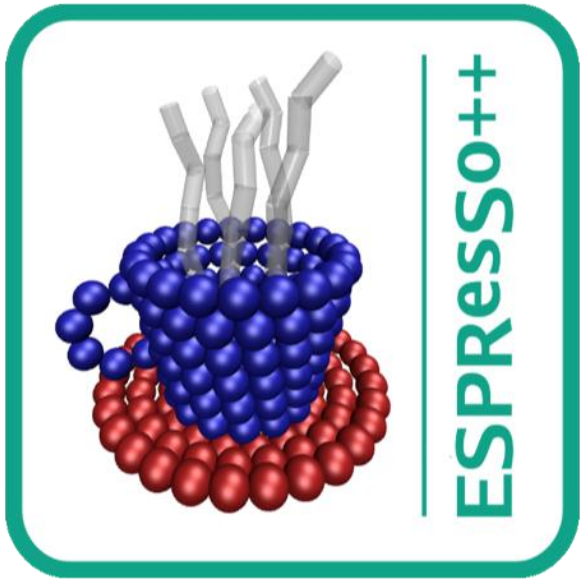
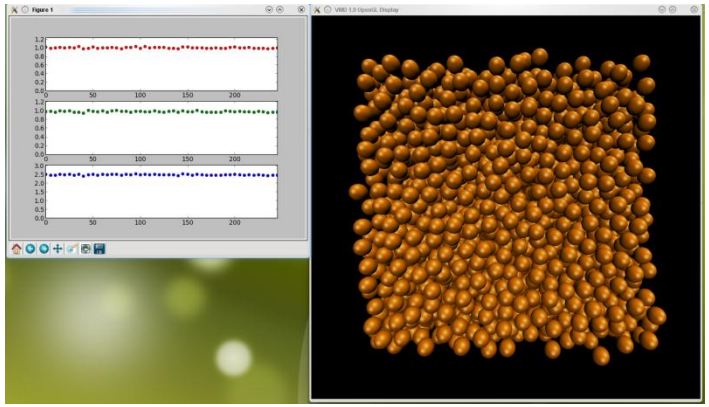


CECAM Tutorial

Stuttgart, Oct. 2012

ESPResSo++

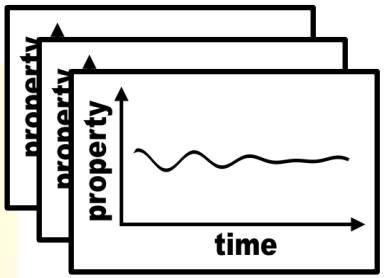
- **ESPResSo++** is an open-source simulation package designed to perform *molecular dynamics* and *monte carlo* simulations of condensed matter systems using traditional and state-of-the-art techniques.
- The software is **extensible** due to its modular object-oriented C++ core.
- It is **highly flexible** due to its Python scripting interface.
- The package can be used as a research platform as well as a production tool.



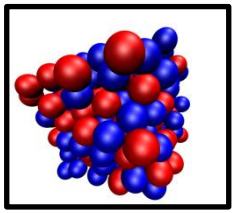
ESPResSo++

Key Features and Advantages

- Flexibility due to Python/C++ integration
- Extensibility due to object oriented design
- Easy script-guided creation of complex topologies
- Quick incorporation of new interactions and algorithms
- Use of standard short- and long range interactions: e.G. Lennard-Jones, Morse, FENE, OPLS, Dihedrals, Ewald Coulomb, etc.
- Efficient implementation of advanced algorithms: e.G. AdResS, Parallel Tempering, dynamic bonds, etc.
- On-the-fly analysis and visualization
- Applicable to large systems and large numbers of CPUs



Analysis



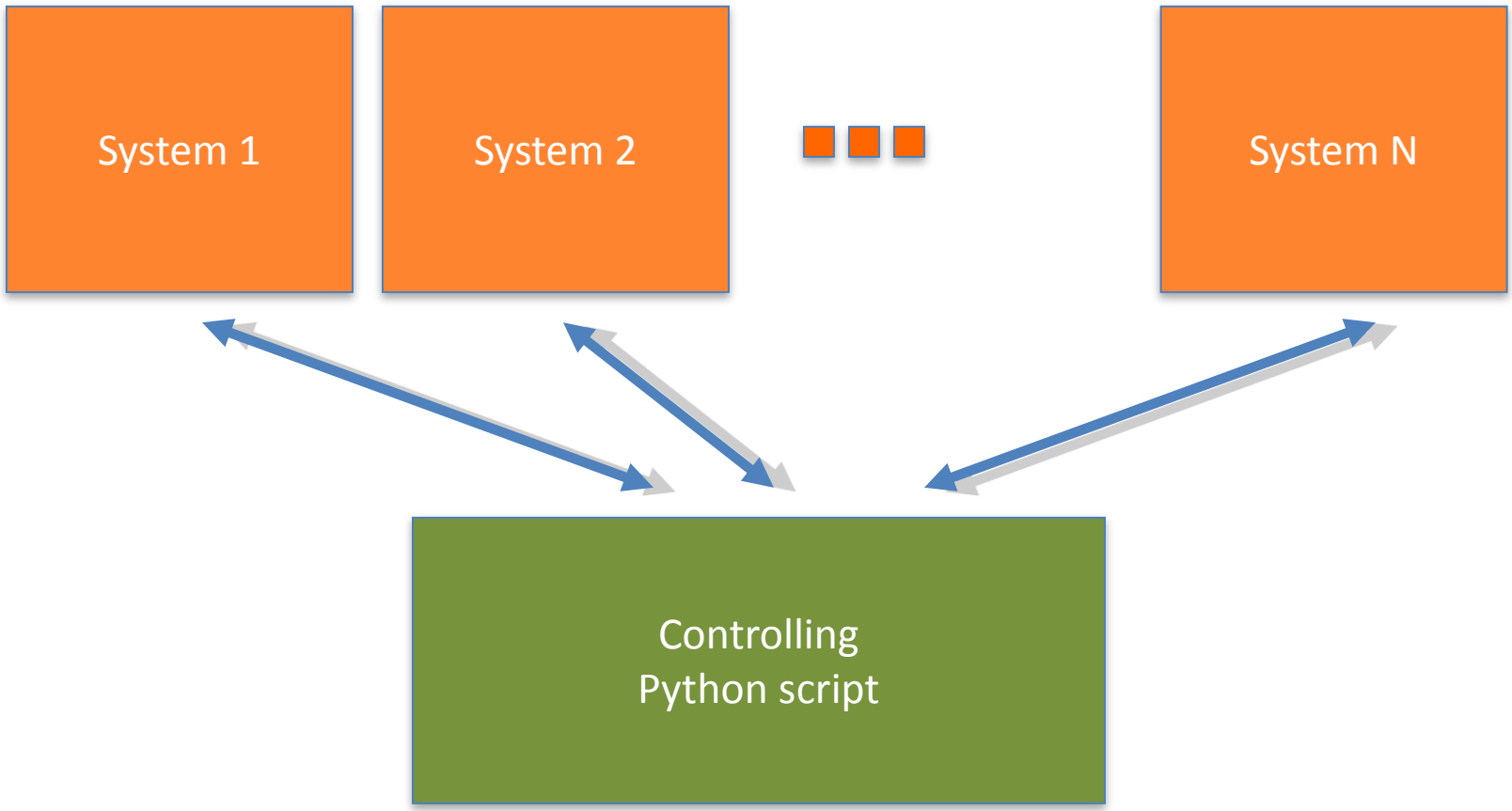
Visualization

NumPy SciPy PyMol VMD

Python scripting language

Advanced C++ Kernel
- Parallelization
- Time-critical algorithms
- Data storage
- Extensible design

Multiple parallel systems, controlled by one Python script:



e.G. for Parallel Tempering, Replica Exchange, …

ESPResSo++ Team:

Current developers:

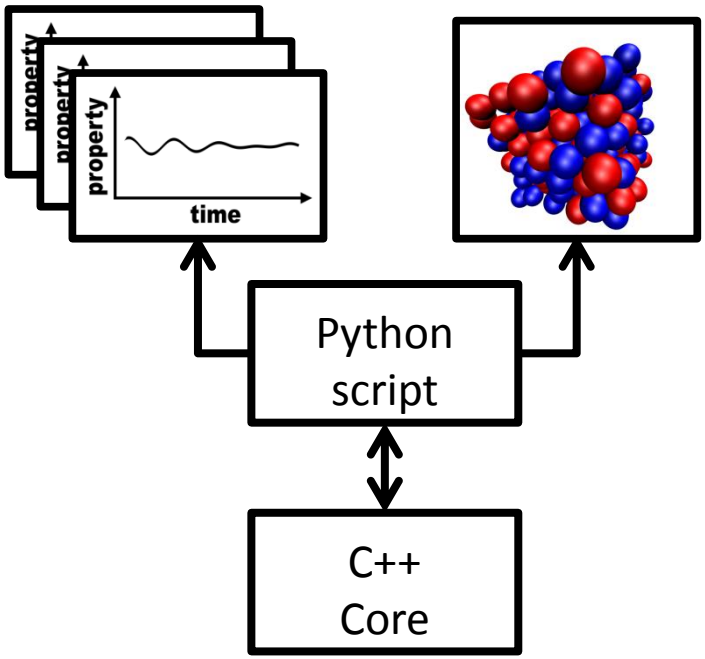
- Torsten Stuehn (MPIP)
- Vitalii Starchenko (MPIP)
- Stas Bevc (NIC, Slovenia)
- Konstantin Koschke (MPIP)
- Livia Moreira (MPIP)
- Raffaello Potestio (MPIP)
- Karsten Kreis (MPIP)

Former developers:

- Thomas Brandes (SCAI)
- Dirk Reith (SCAI)
- Axel Arnold (ICP)
- Olaf Lenz (ICP)
- Jonathan Halverson (BNL, USA)
- Victor Ruehle (Cambridge, UK)
- Christoph Junghans (LANL, USA)

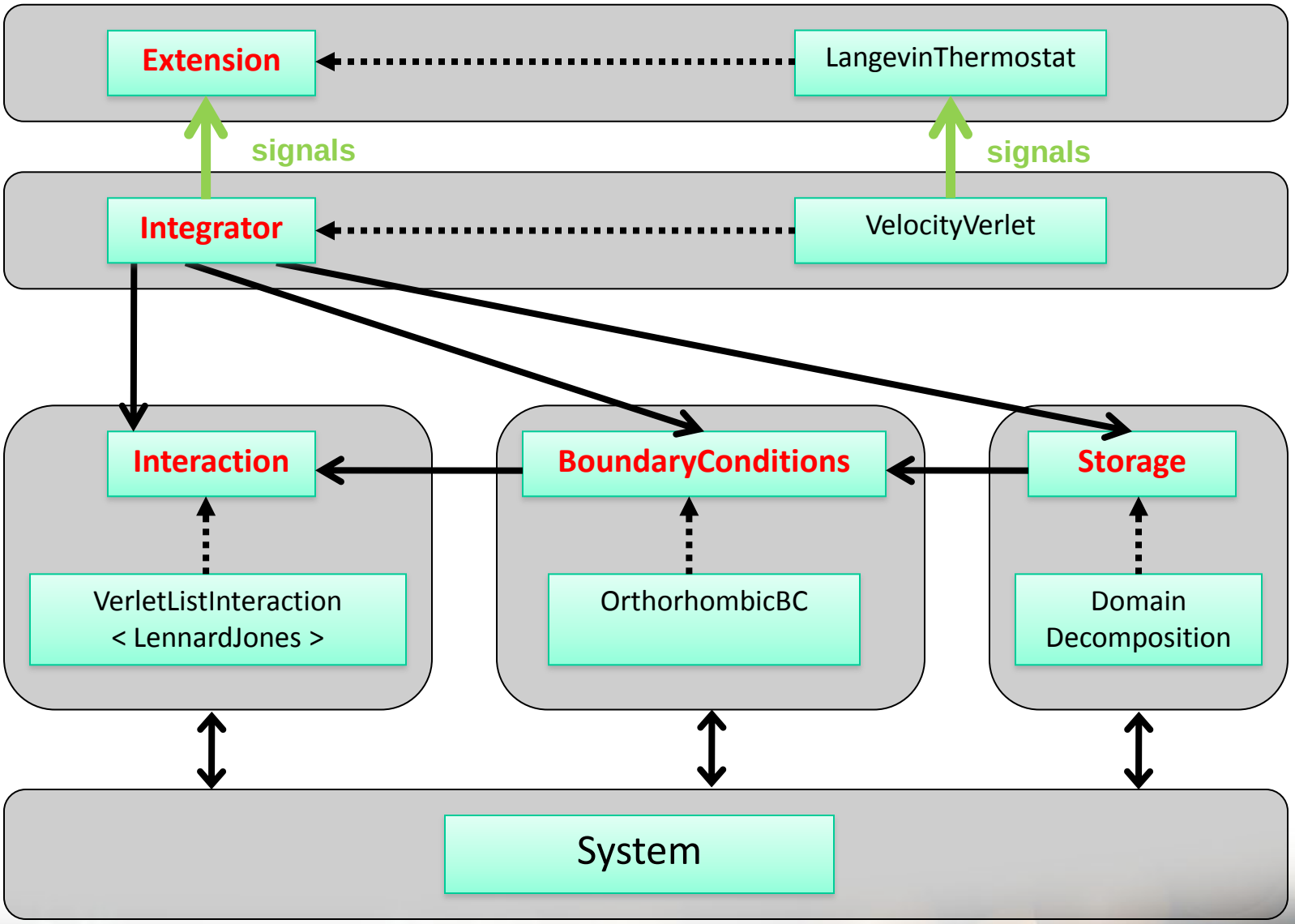
Outline of the tutorial

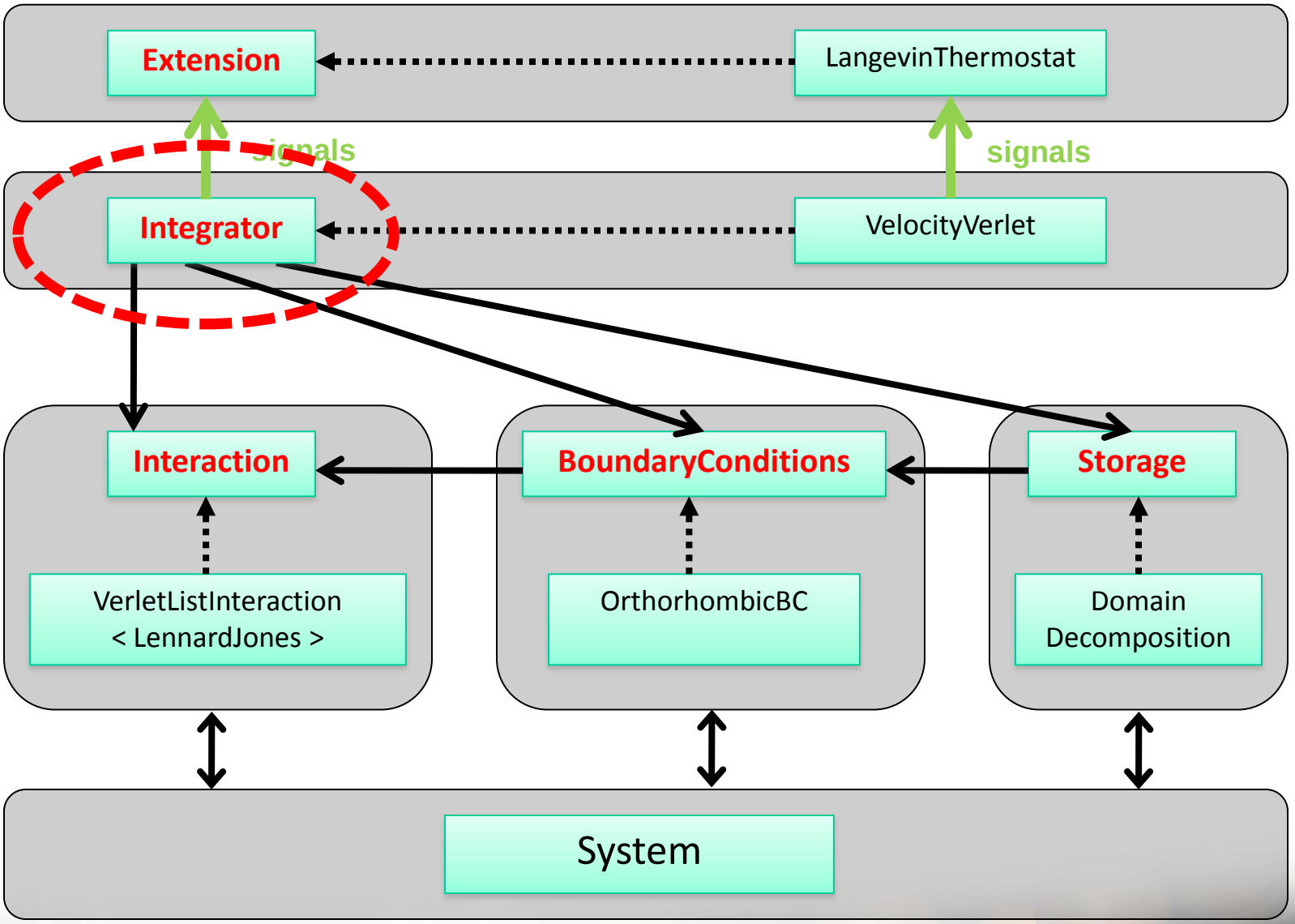
- Inside ESPReSo++ an overview of the C++ class structure
- Integrator, Interactions, Storage
- Setting up a simulation with „System“, „Integrator“ and „Interactions“
- How to equilibrate a polymer melt

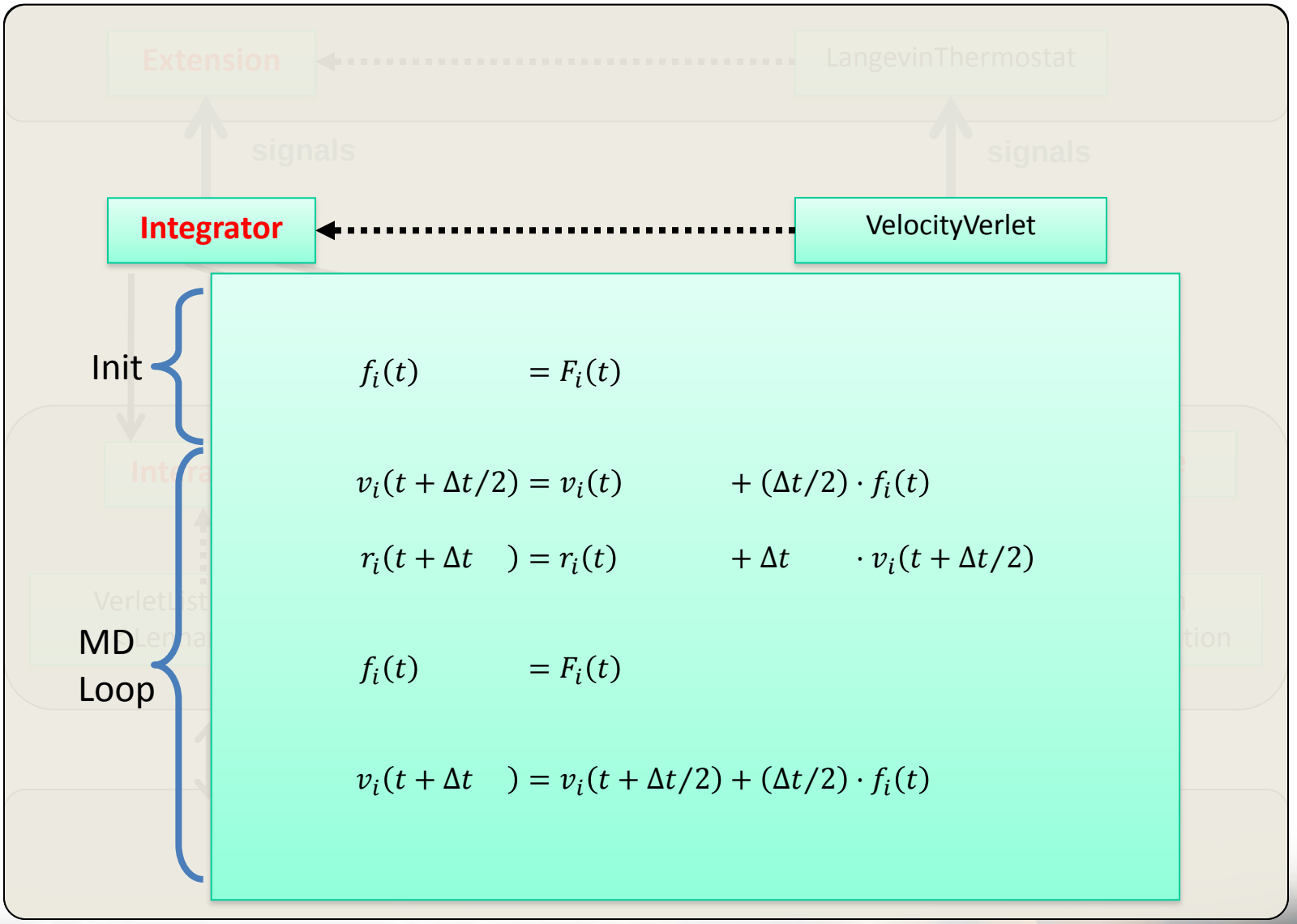


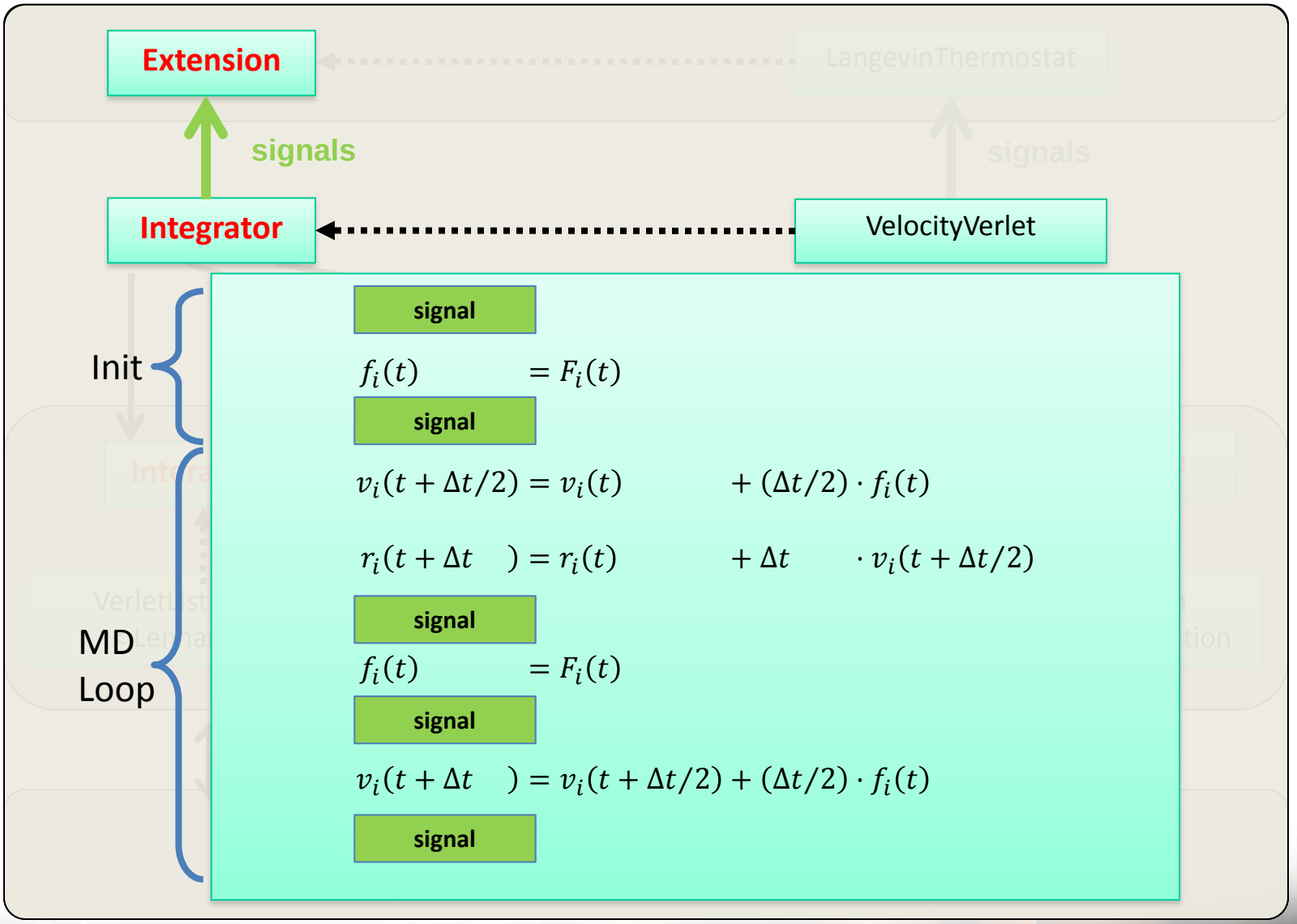
- Installation of ESPResSo++
- Basic System Setup
- Simple Lennard Jones System
- Advanced Lennard Jones System with graphical output
- Polymer Melt

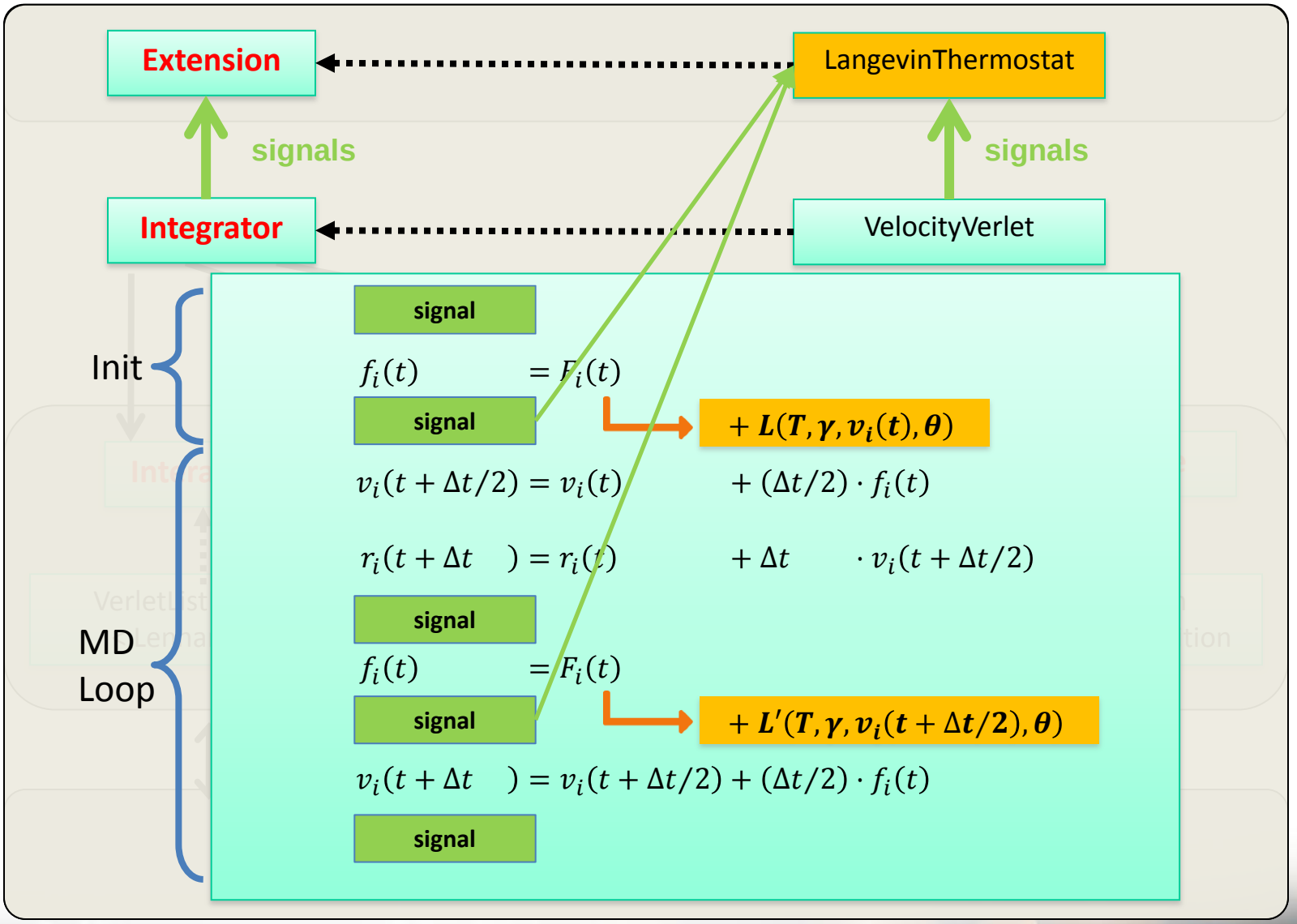
Inside ESPResSo++

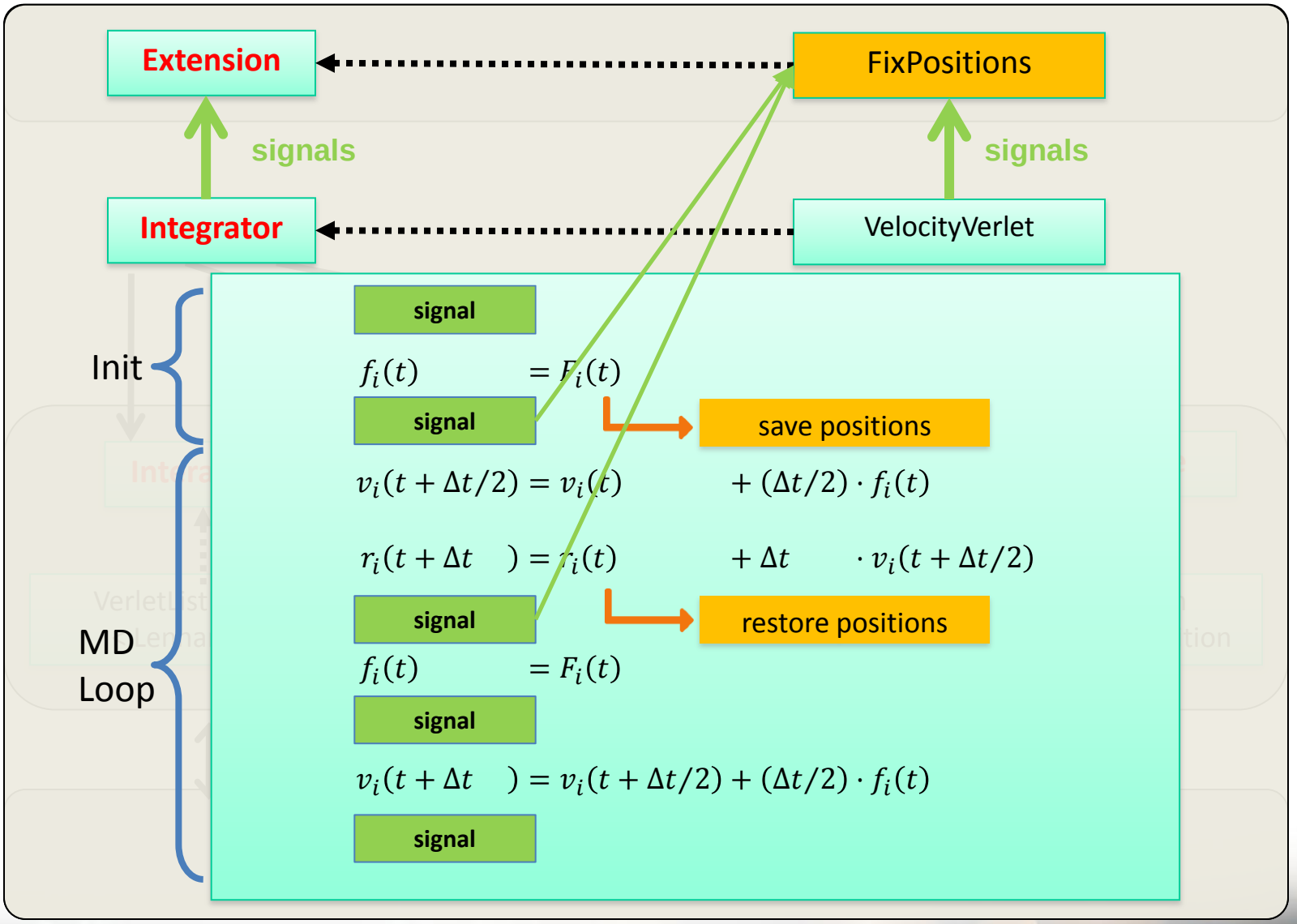


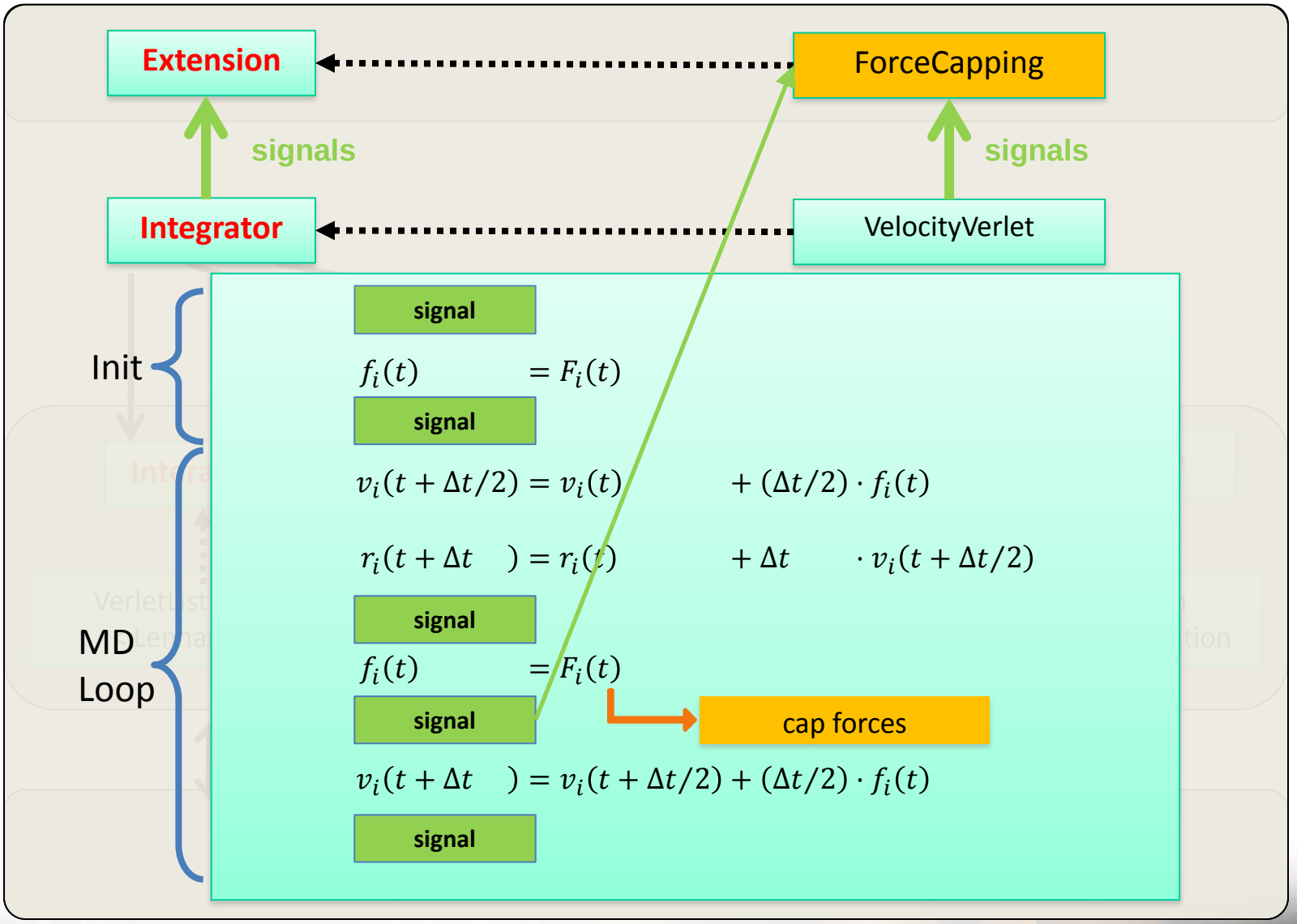

Extension
LangevinThermostat
signals
signals
Integrator
VelocityVerlet
Interaction
BoundaryConditions
Storage
VerletListInteraction
< LennardJones >
OrthorhombicBC
Domain
Decomposition
System

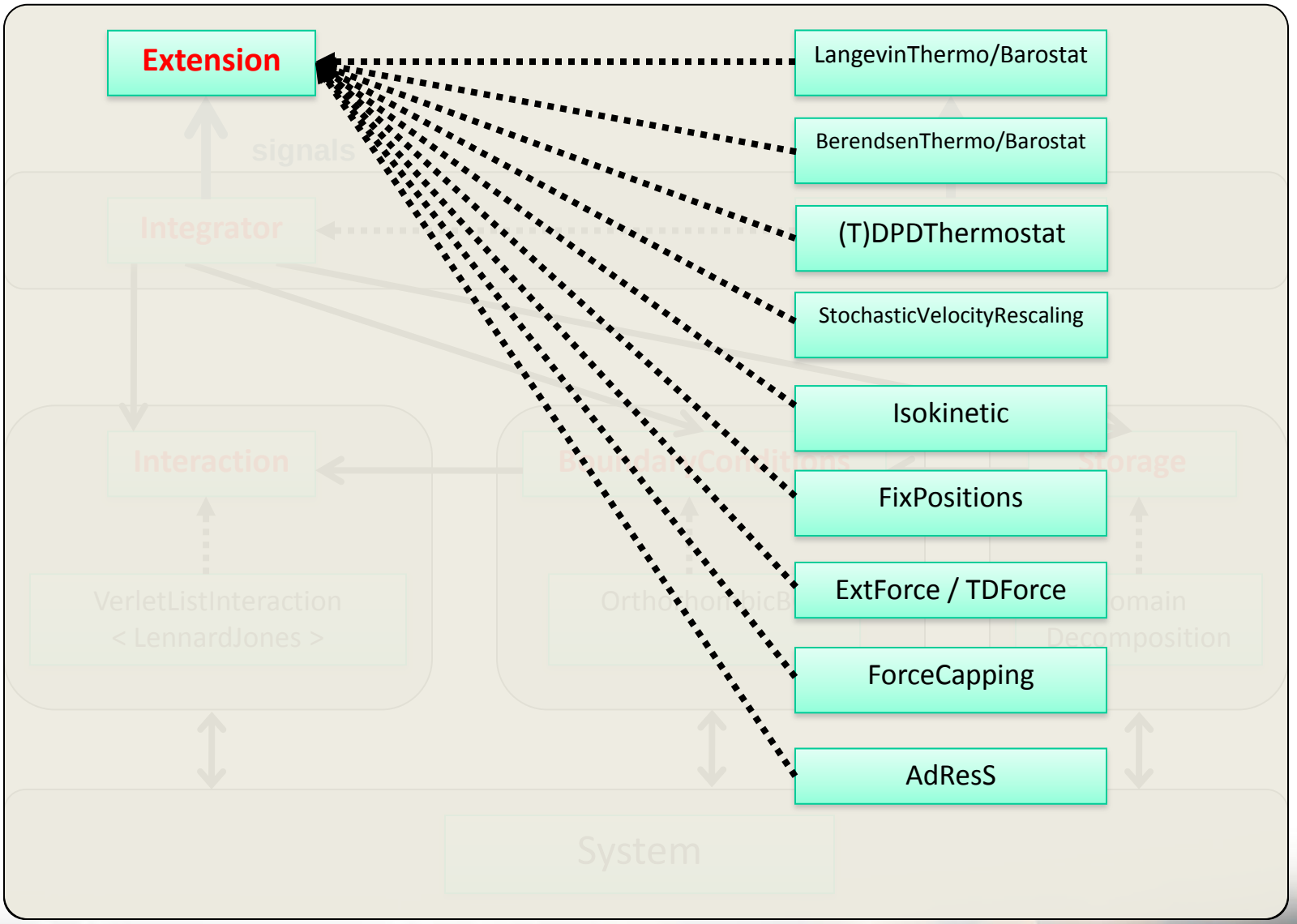

Extension
LangevinThermostat
signals
signals
Integrator
VelocityVerlet
Init
MD
Loop
$f_i(t) = F_i(t)$
$v_i(t + \Delta t/2) = v_i(t) + (\Delta t/2) \cdot f_i(t)$
$r_i(t + \Delta t) = r_i(t) + \Delta t \cdot v_i(t + \Delta t/2)$
$f_i(t) = F_i(t)$
$v_i(t + \Delta t) = v_i(t + \Delta t/2) + (\Delta t/2) \cdot f_i(t)$

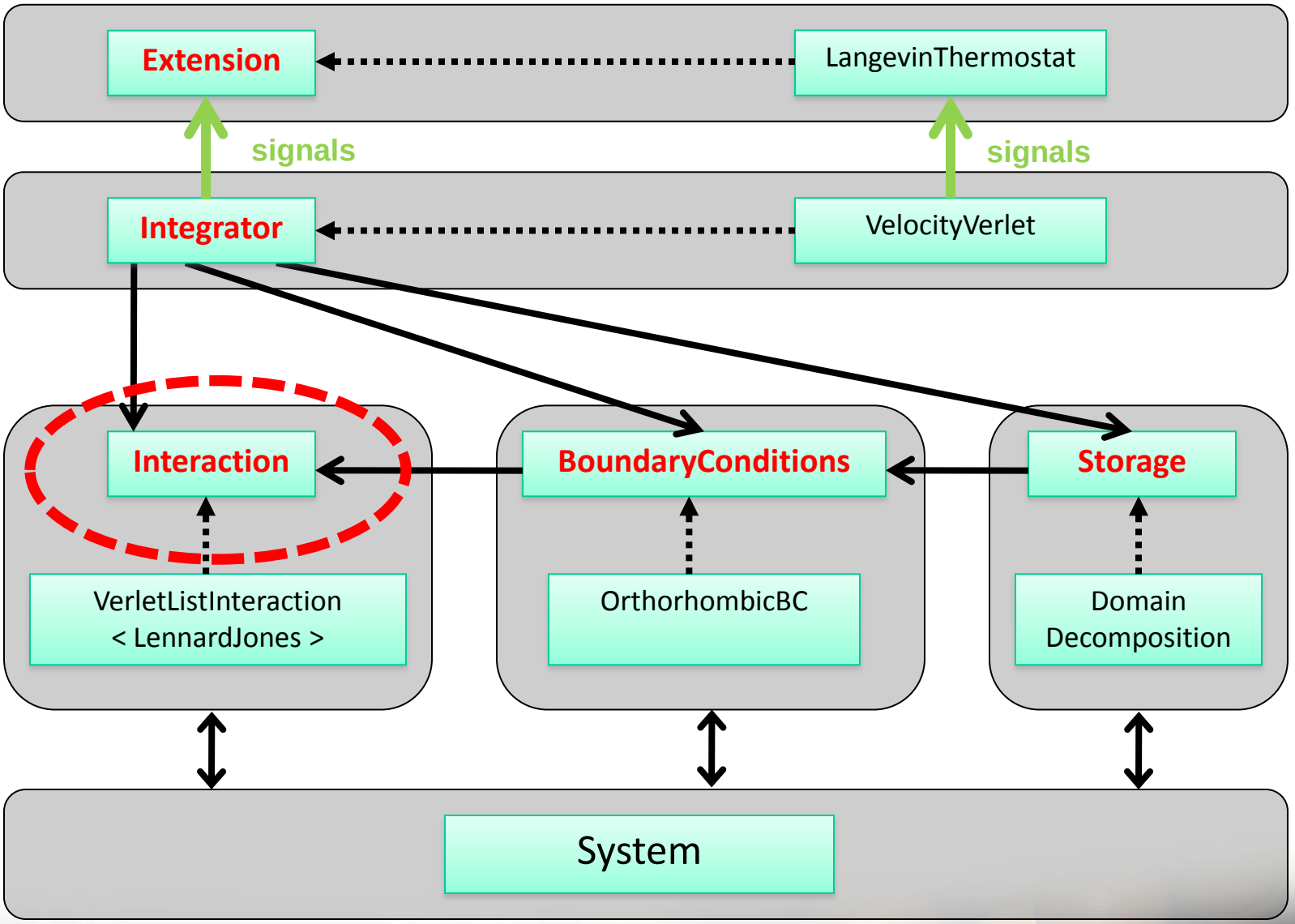

Extension
signals
Integrator
VelocityVerlet
Init
MD
Loop
signal
$f_i(t) = F_i(t)$
signal
$v_i(t + \Delta t/2) = v_i(t) + (\Delta t/2) \cdot f_i(t)$
$r_i(t + \Delta t) = r_i(t) + \Delta t \cdot v_i(t + \Delta t/2)$
signal
$f_i(t) = F_i(t)$
signal
$v_i(t + \Delta t) = v_i(t + \Delta t/2) + (\Delta t/2) \cdot f_i(t)$
signal

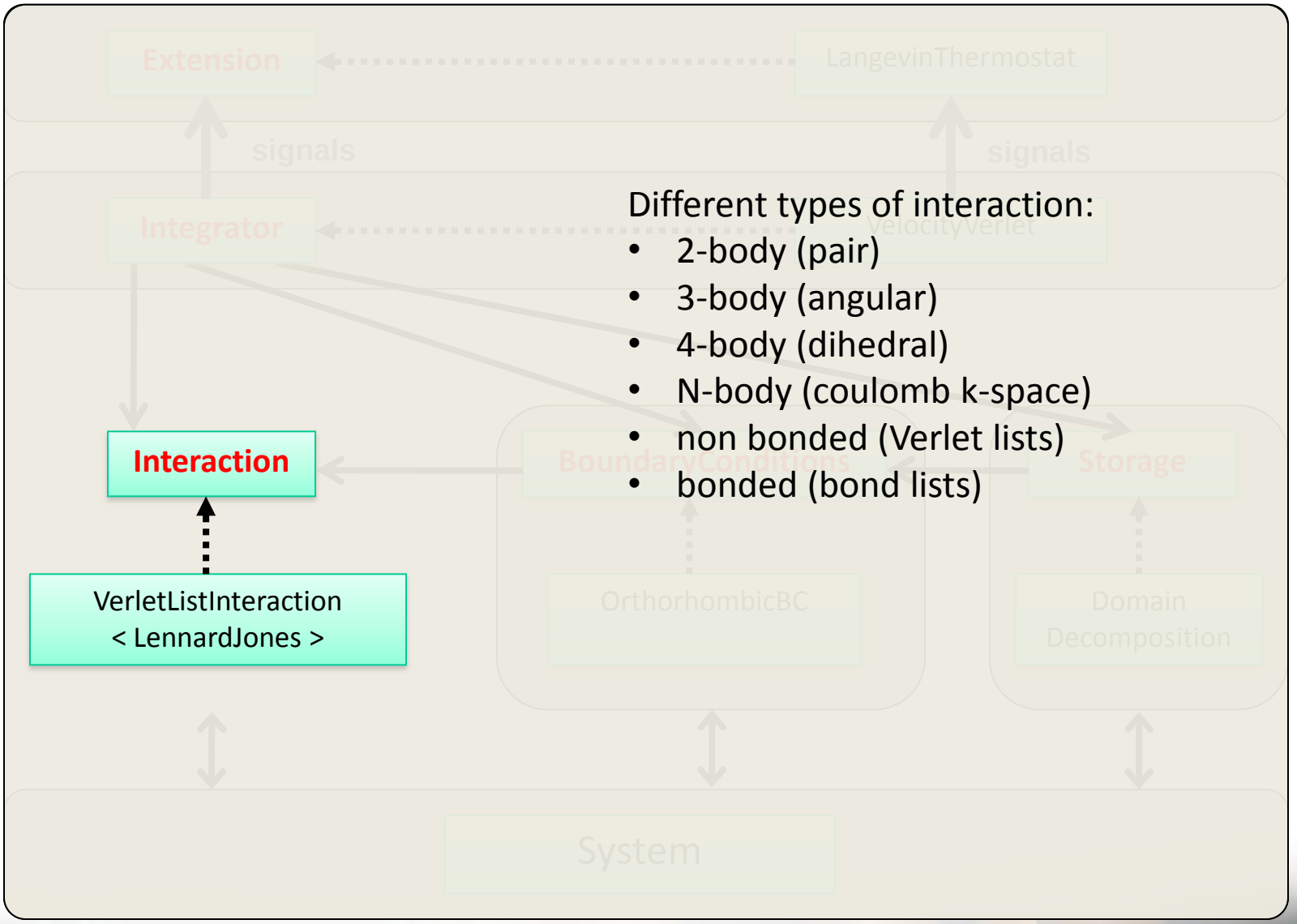

Extension
LangevinThermostat
signals
signals
Integrator
VelocityVerlet
Init
MD
Loop
signal
$f_i(t) = F_i(t)$
signal
$+ \boldsymbol{L}(\boldsymbol{T}, \boldsymbol{\gamma}, \boldsymbol{v_i}(\boldsymbol{t}), \boldsymbol{\theta})$
$v_i(t + \Delta t/2) = v_i(t) + (\Delta t/2) \cdot f_i(t)$
$r_i(t + \Delta t) = r_i(t) + \Delta t \cdot v_i(t + \Delta t/2)$
signal
$f_i(t) = F_i(t)$
signal
$+ \boldsymbol{L'}(\boldsymbol{T}, \boldsymbol{\gamma}, \boldsymbol{v_i}(\boldsymbol{t} + \Delta \boldsymbol{t}/\boldsymbol{2}), \boldsymbol{\theta})$
$v_i(t + \Delta t) = v_i(t + \Delta t/2) + (\Delta t/2) \cdot f_i(t)$
signal


Extension
FixPositions
signals
signals
Integrator
VelocityVerlet
Init
MD
Loop
signal
$f_i(t) = F_i(t)$
signal
save positions
$v_i(t + \Delta t/2) = v_i(t) + (\Delta t/2) \cdot f_i(t)$
$r_i(t + \Delta t) = r_i(t) + \Delta t \cdot v_i(t + \Delta t/2)$
signal
restore positions
$f_i(t) = F_i(t)$
signal
$v_i(t + \Delta t) = v_i(t + \Delta t/2) + (\Delta t/2) \cdot f_i(t)$
signal


Extension
ForceCapping
signals
signals
Integrator
VelocityVerlet
Init
MD
Loop
signal
$f_i(t) = F_i(t)$
signal
$v_i(t + \Delta t/2) = v_i(t) + (\Delta t/2) \cdot f_i(t)$
$r_i(t + \Delta t) = r_i(t) + \Delta t \cdot v_i(t + \Delta t/2)$
signal
$f_i(t) = F_i(t)$
signal
cap forces
$v_i(t + \Delta t) = v_i(t + \Delta t/2) + (\Delta t/2) \cdot f_i(t)$
signal

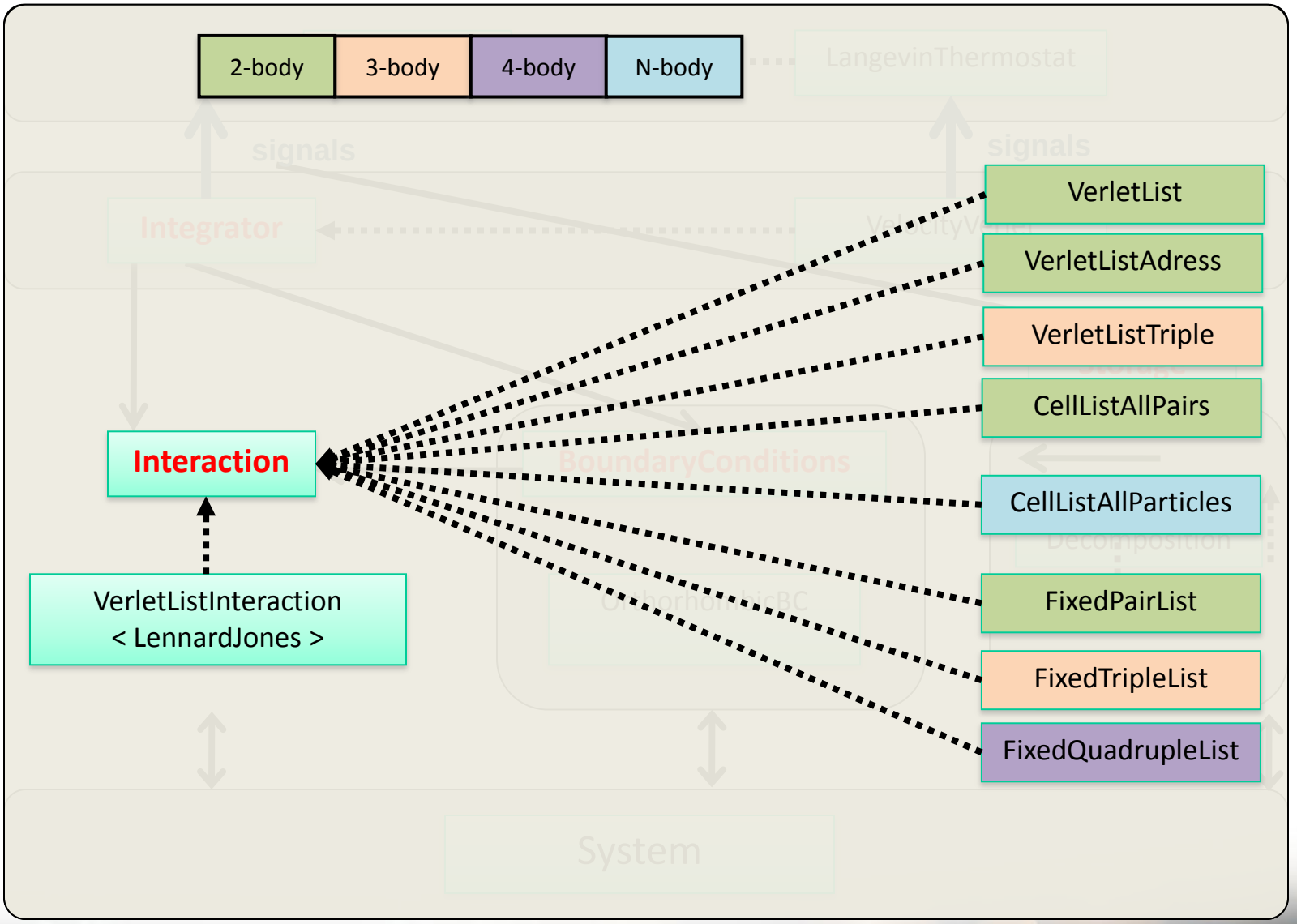

Extension
LangevinThermo/Barostat
BerendsenThermo/Barostat
(T)DPDThermostat
StochasticVelocityRescaling
Isokinetic
FixPositions
ExtForce / TDForce
ForceCapping
AdResS
signals
Integrator
Interaction
VerletListInteraction
< LennardJones >
Storage
Decomposition
System

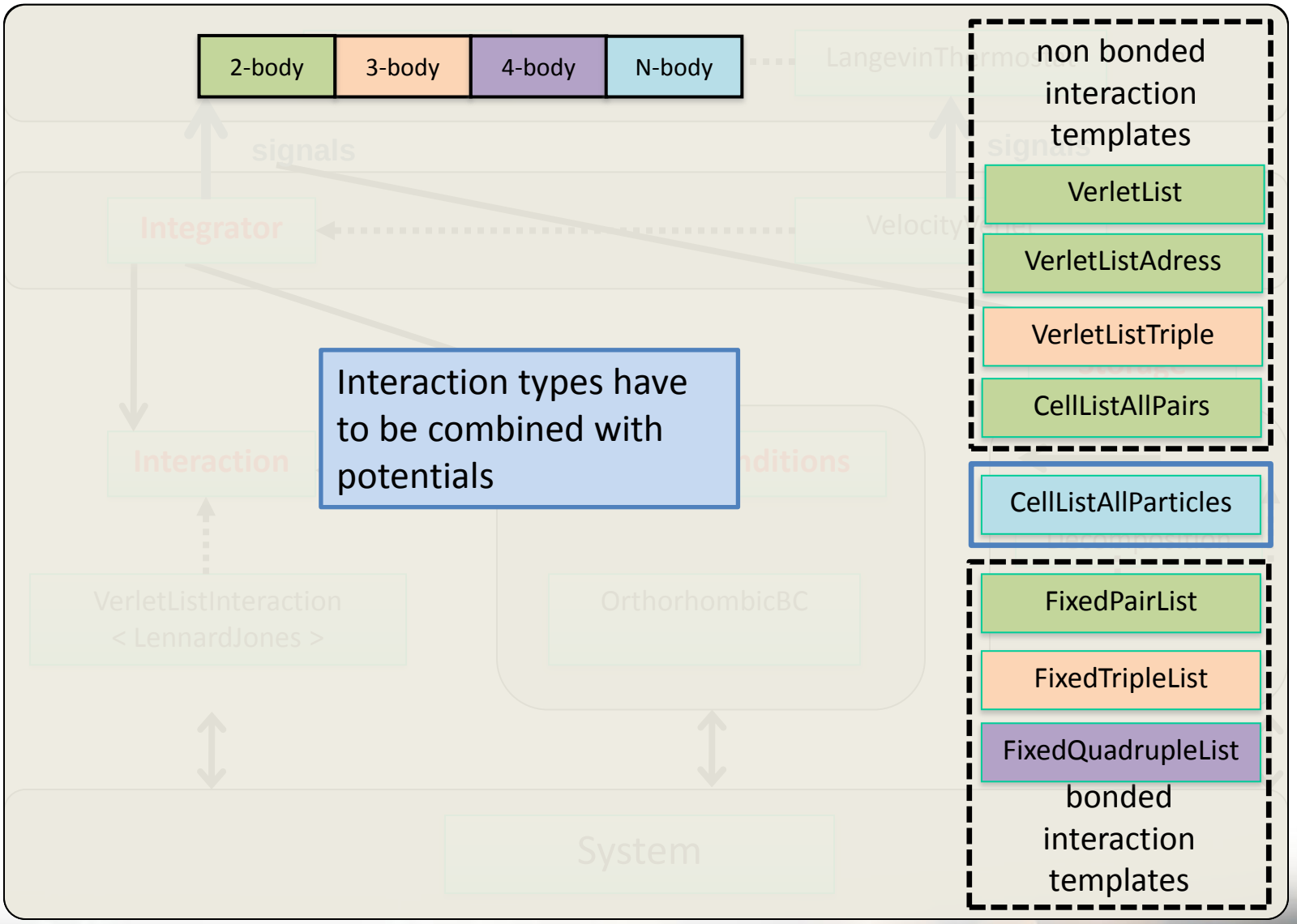

Extension
LangevinThermostat
signals
signals
Integrator
VelocityVerlet
Interaction
BoundaryConditions
Storage
VerletListInteraction
< LennardJones >
OrthorhombicBC
Domain
Decomposition
System

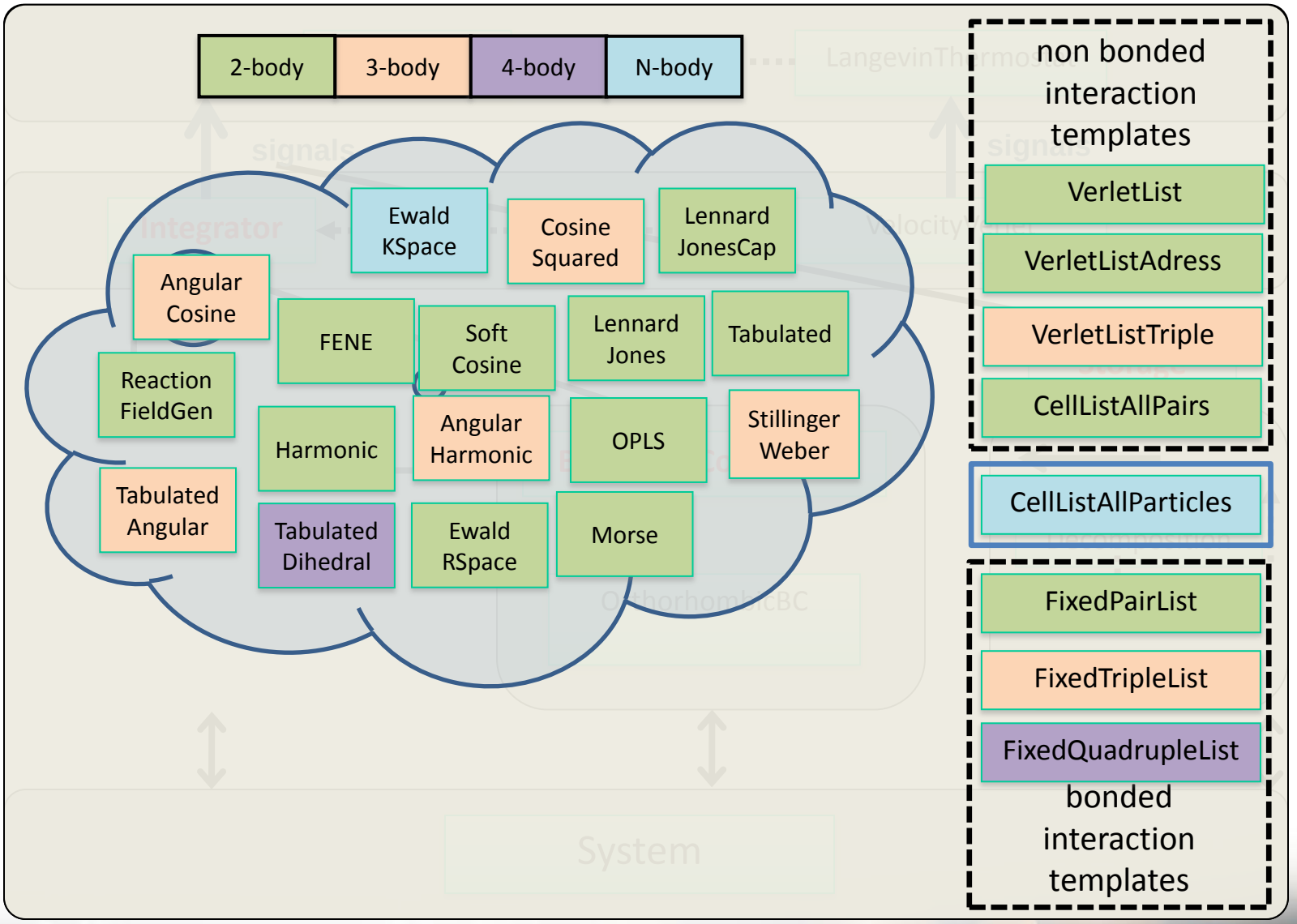


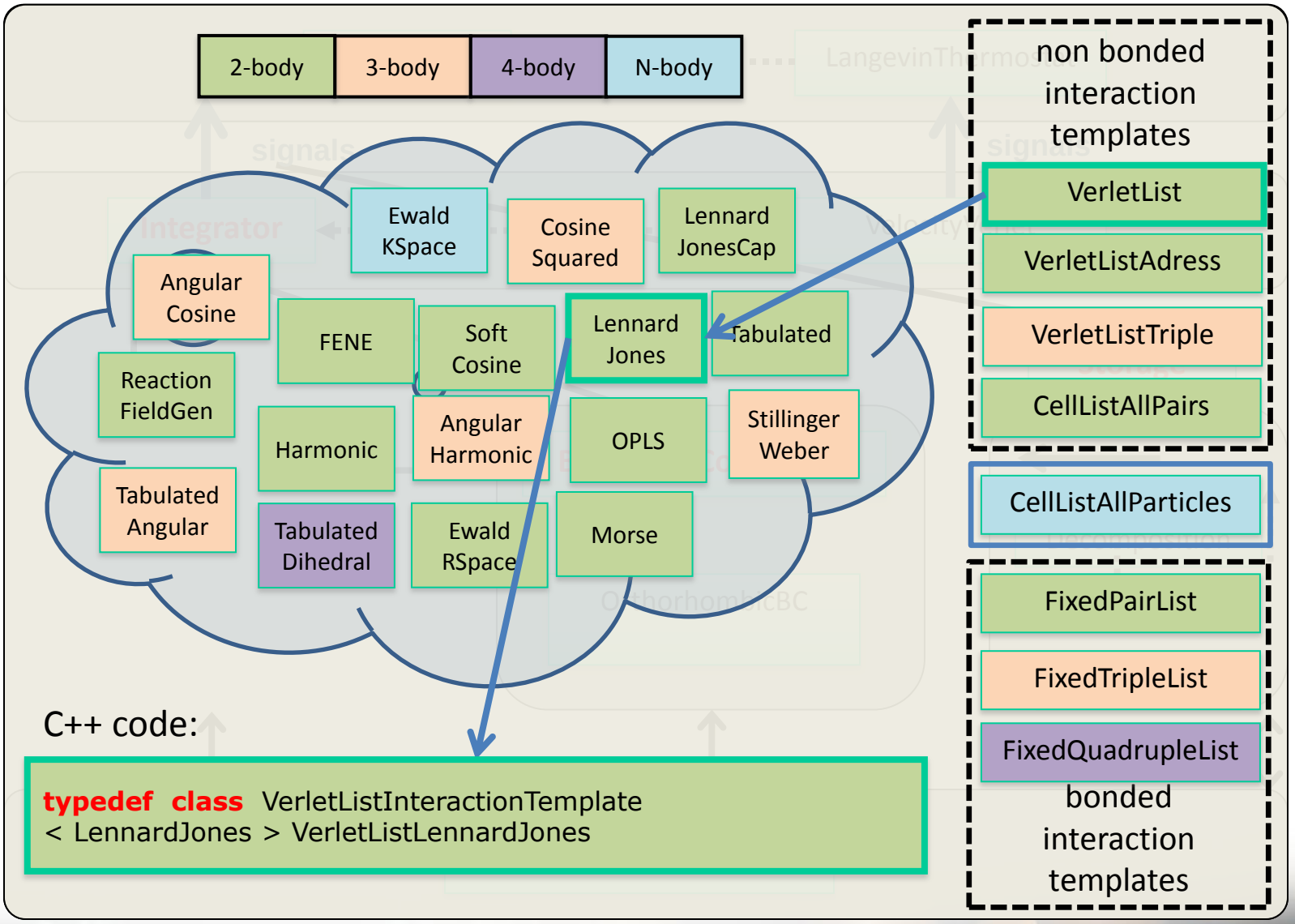
Different types of interaction:

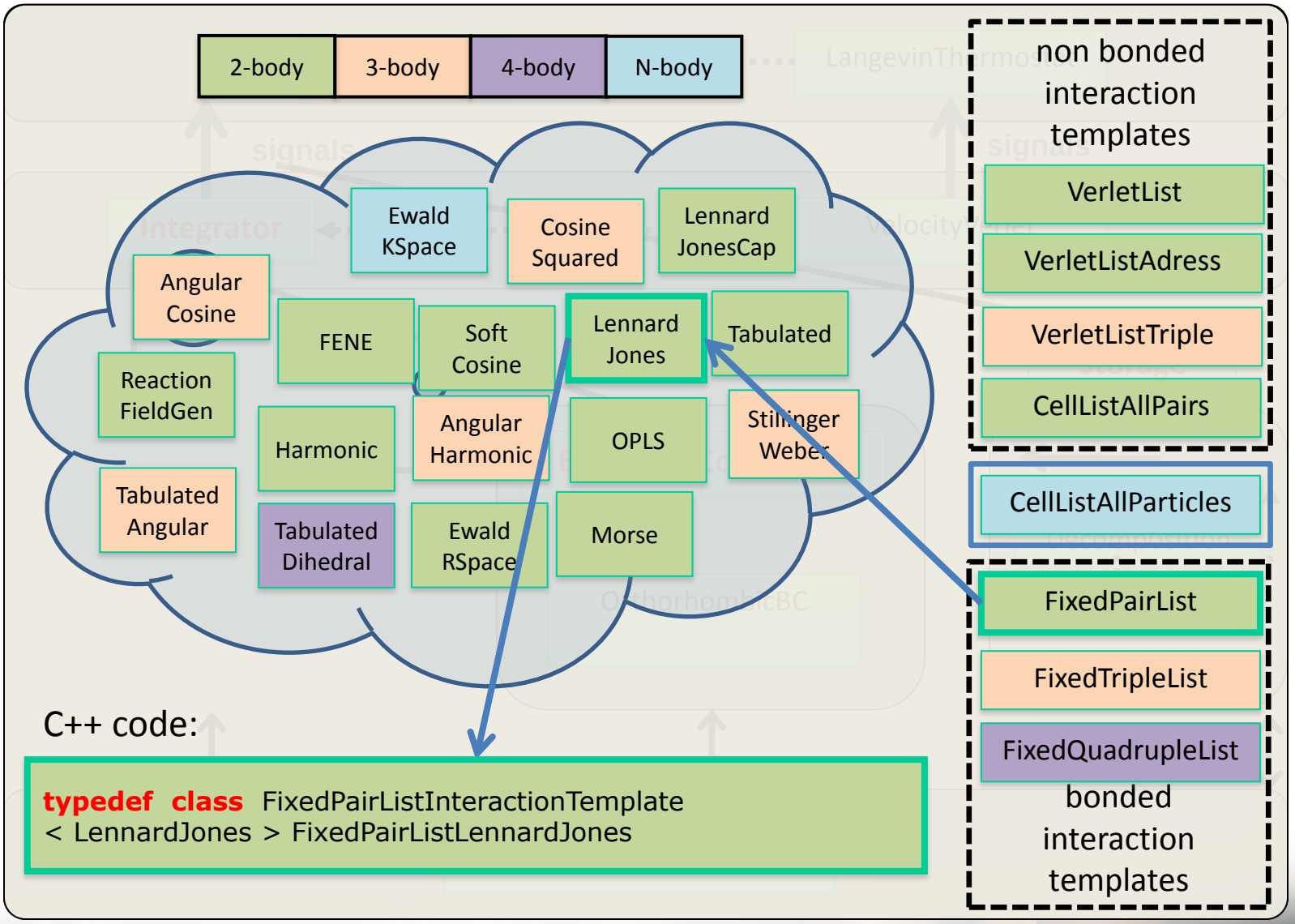
- 2-body (pair)
- 3-body (angular)
- 4-body (dihedral)
- N-body (coulomb k-space)
- non bonded (Verlet lists)
- bonded (bond lists)

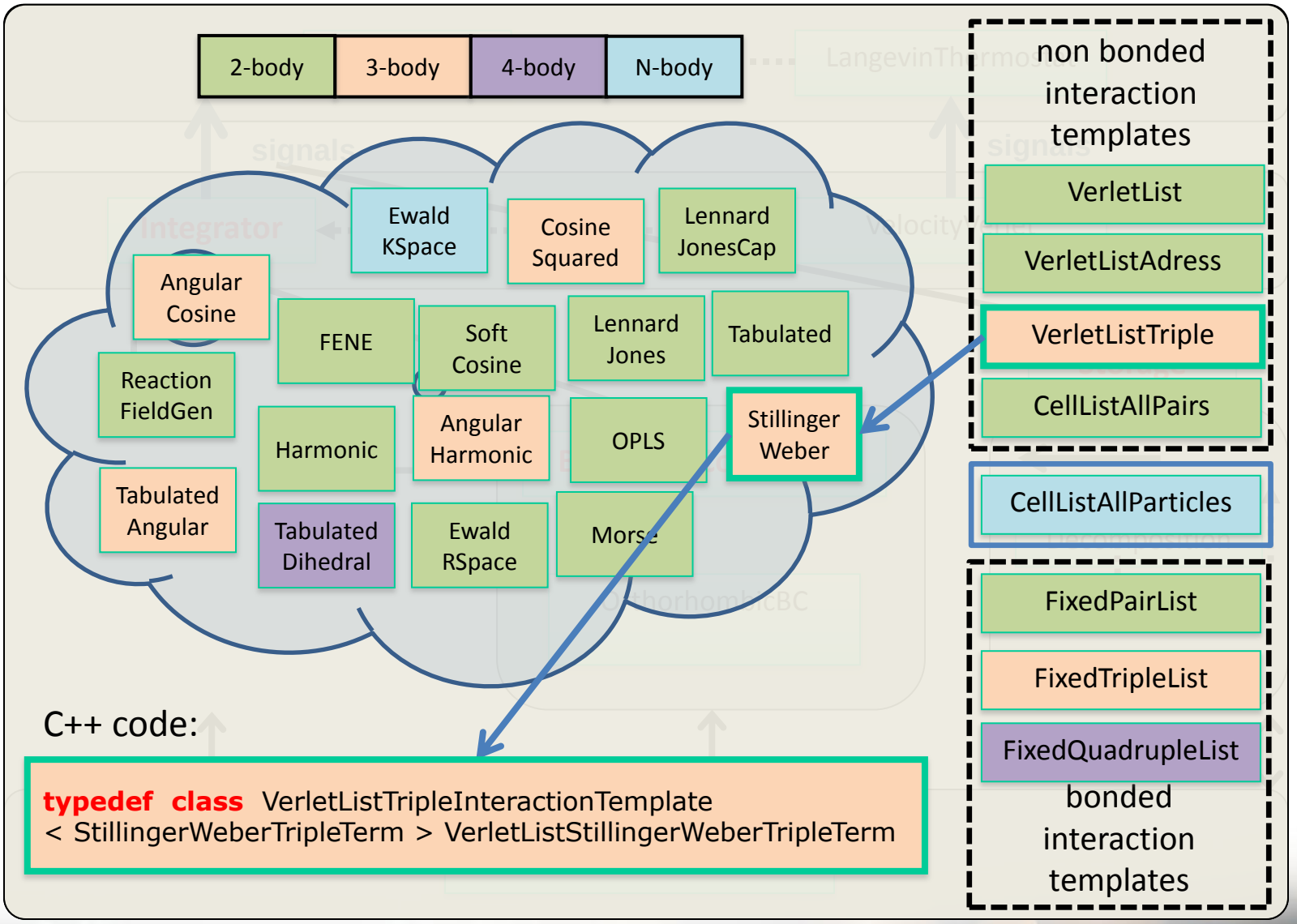

2-body
3-body
4-body
N-body
VerletList
VerletListAdress
VerletListTriple
CellListAllPairs
CellListAllParticles
FixedPairList
FixedTripleList
FixedQuadrupleList
Interaction
VerletListInteraction
< LennardJones >
signals
signals
Integrator
LangevinThermostat
BoundaryConditions
System

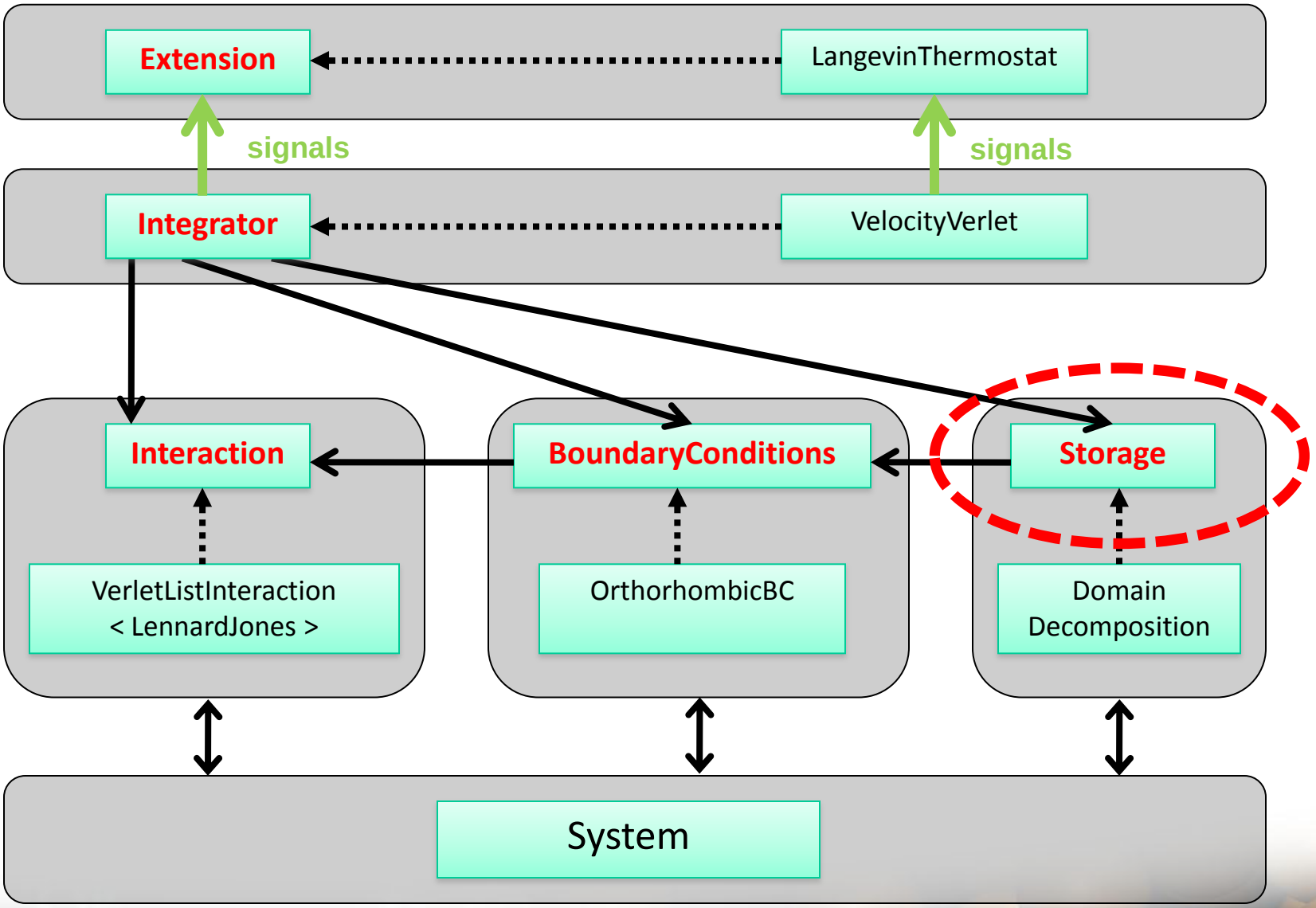

2-body
3-body
4-body
N-body
non bonded interaction templates
VerletList
VerletListAdress
VerletListTriple
CellListAllPairs
CellListAllParticles
FixedPairList
FixedTripleList
FixedQuadrupleList
bonded interaction templates
Interaction types have to be combined with potentials

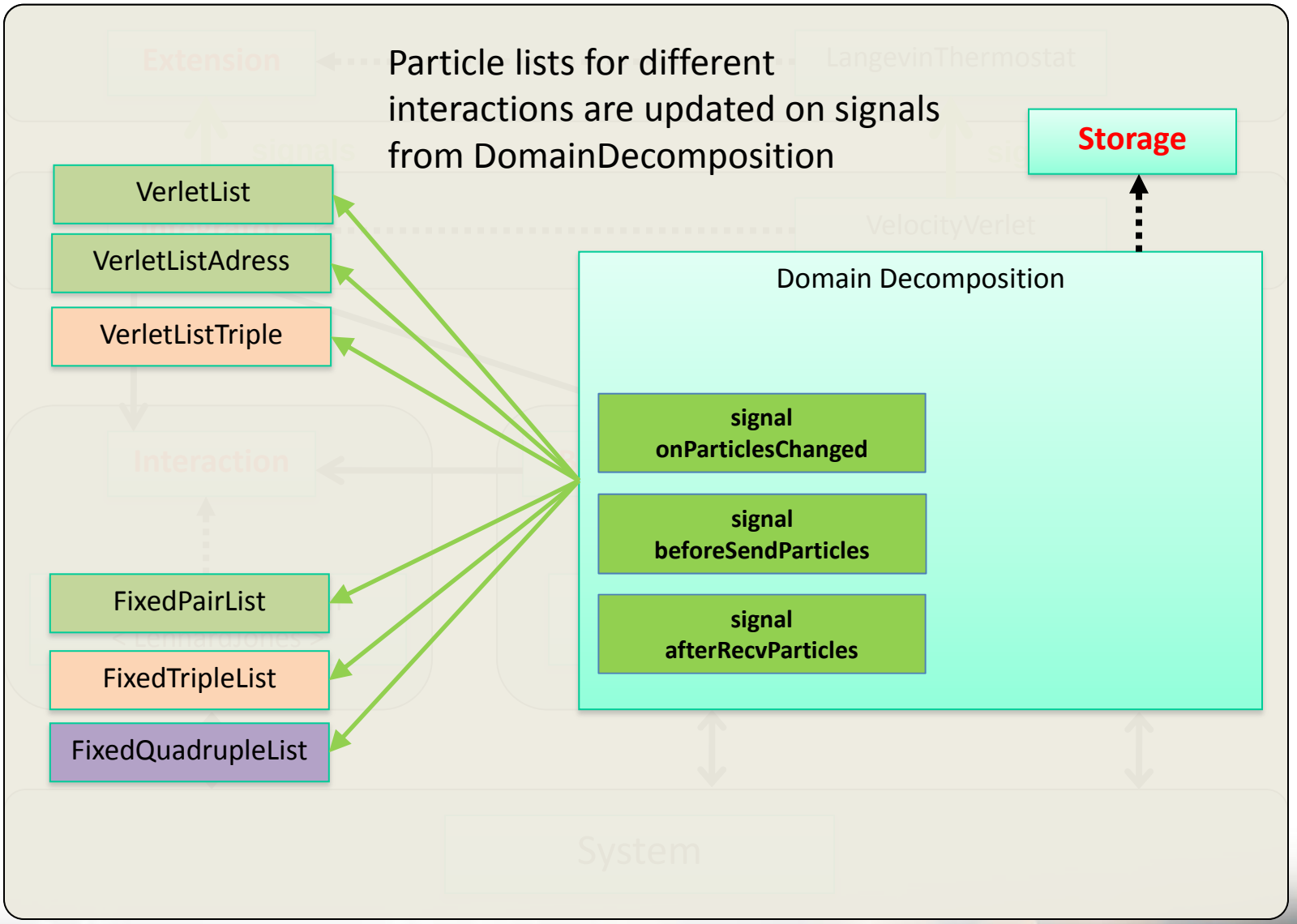

2-body
3-body
4-body
N-body
Ewald KSpace
Cosine Squared
Lennard JonesCap
Angular Cosine
FENE
Soft Cosine
Lennard Jones
Tabulated
Reaction FieldGen
Harmonic
Angular Harmonic
OPLS
Stillinger Weber
Tabulated Angular
Tabulated Dihedral
Ewald RSpace
Morse
non bonded interaction templates
VerletList
VerletListAdress
VerletListTriple
CellListAllPairs
CellListAllParticles
FixedPairList
FixedTripleList
FixedQuadrupleList
bonded interaction templates

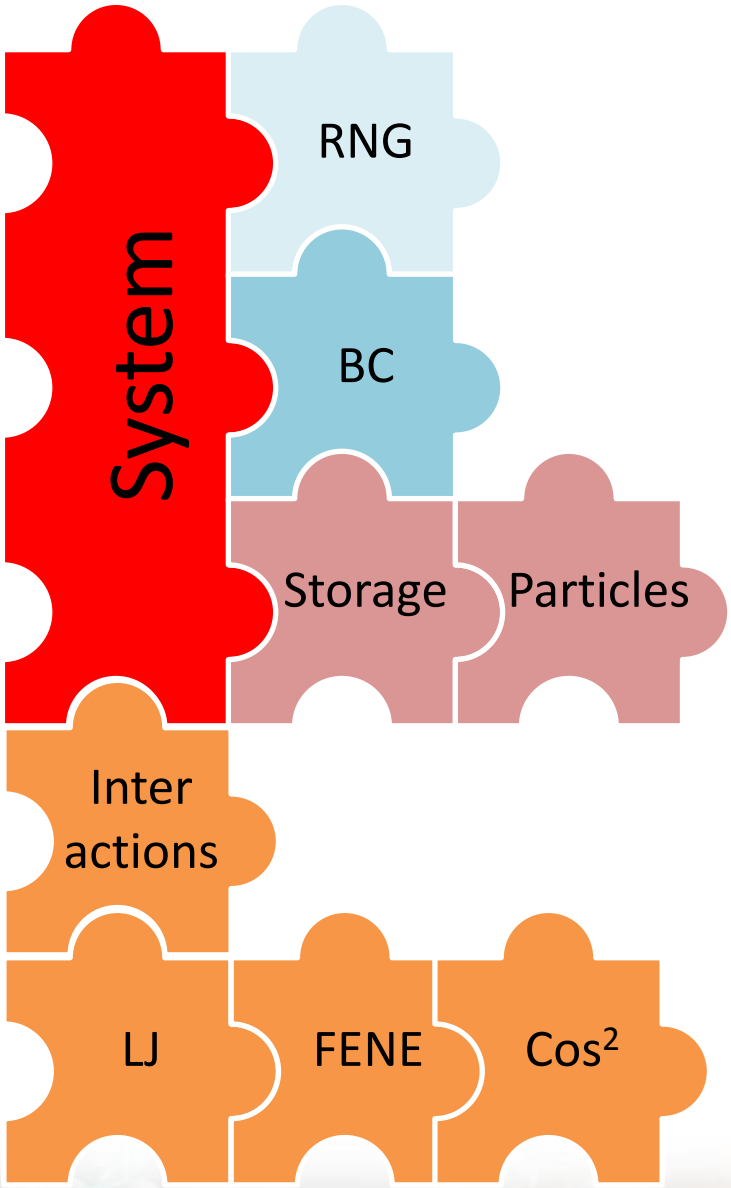

2-body
3-body
4-body
N-body
Ewald KSpace
Cosine Squared
Lennard JonesCap
Angular Cosine
FENE
Soft Cosine
Lennard Jones
Tabulated
Reaction FieldGen
Harmonic
Angular Harmonic
OPLS
Stillinger Weber
Tabulated Angular
Tabulated Dihedral
Ewald RSpace
Morse
C++ code:
typedef class VerletListInteractionTemplate
< LennardJones > VerletListLennardJones
non bonded interaction templates
VerletList
VerletListAdress
VerletListTriple
CellListAllPairs
CellListAllParticles
FixedPairList
FixedTripleList
FixedQuadrupleList
bonded interaction templates


2-body
3-body
4-body
N-body
Ewald KSpace
Cosine Squared
Lennard JonesCap
Angular Cosine
FENE
Soft Cosine
Lennard Jones
Tabulated
Reaction FieldGen
Harmonic
Angular Harmonic
OPLS
Stillinger Weber
Tabulated Angular
Tabulated Dihedral
Ewald RSpace
Morse
non bonded interaction templates
VerletList
VerletListAdress
VerletListTriple
CellListAllPairs
CellListAllParticles
FixedPairList
FixedTripleList
FixedQuadrupleList
bonded interaction templates
C++ code:
typedef class FixedPairListInteractionTemplate
< LennardJones > FixedPairListLennardJones

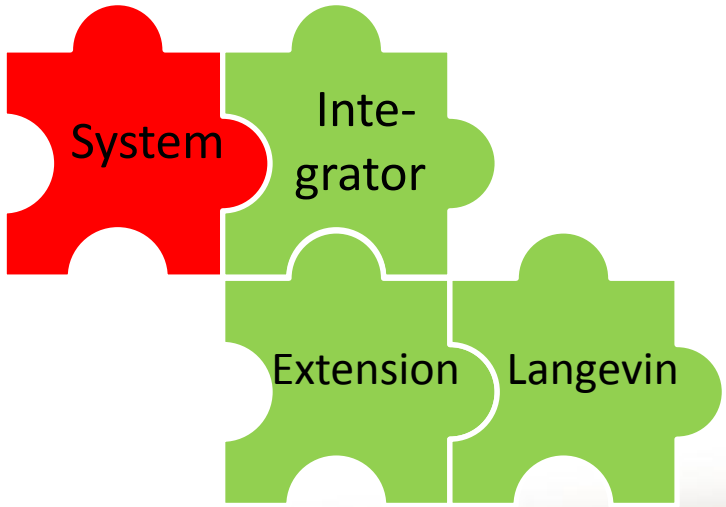

2-body
3-body
4-body
N-body
Ewald KSpace
Cosine Squared
Lennard JonesCap
Angular Cosine
FENE
Soft Cosine
Lennard Jones
Tabulated
Reaction FieldGen
Harmonic
Angular Harmonic
OPLS
Stillinger Weber
Tabulated Angular
Tabulated Dihedral
Ewald RSpace
Morse
non bonded interaction templates
VerletList
VerletListAdress
VerletListTriple
CellListAllPairs
CellListAllParticles
FixedPairList
FixedTripleList
FixedQuadrupleList
bonded interaction templates
C++ code:
typedef class VerletListTripleInteractionTemplate
< StillingerWeberTripleTerm > VerletListStillingerWeberTripleTerm

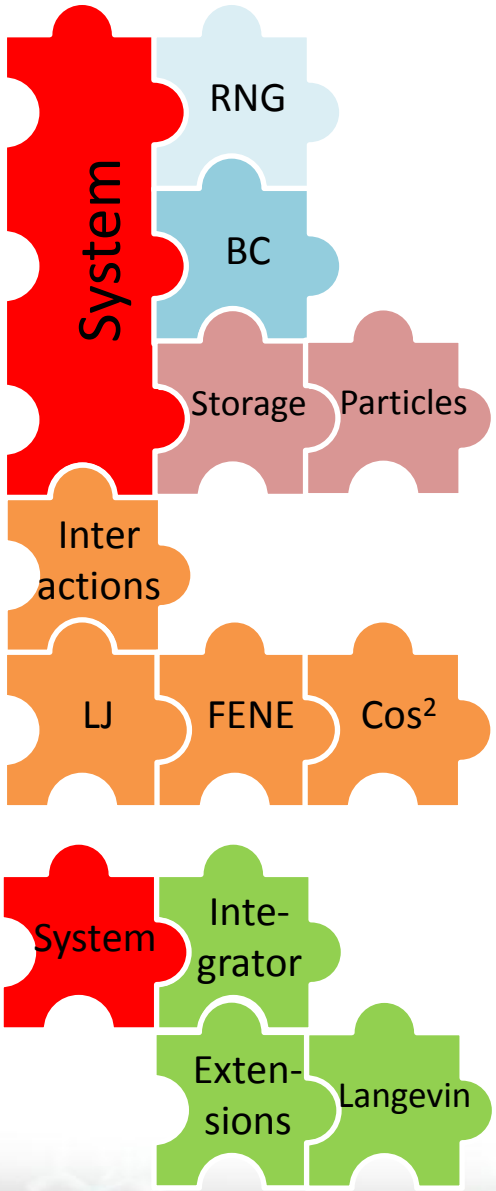

Extension
LangevinThermostat
signals
signals
Integrator
VelocityVerlet
Interaction
BoundaryConditions
Storage
VerletListInteraction
< LennardJones >
OrthorhombicBC
Domain
Decomposition
System


Particle lists for different interactions are updated on signals from DomainDecomposition
Storage
VerletList
VerletListAdress
VerletListTriple
Domain Decomposition
signal onParticlesChanged
signal beforeSendParticles
signal afterRecvParticles
FixedPairList
FixedTripleList
FixedQuadrupleList



Necessary modules to simulate a standard polymer melt with Langevin dynamics





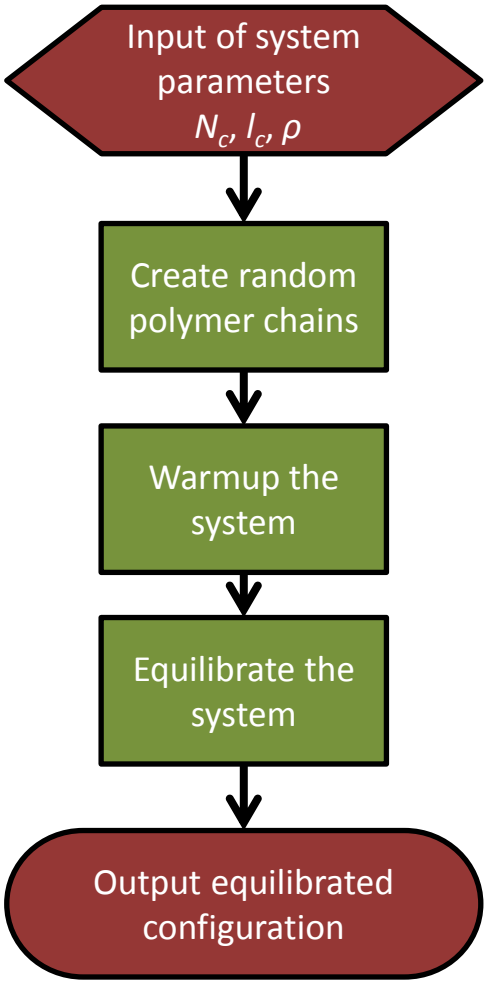
Corresponding Python script :

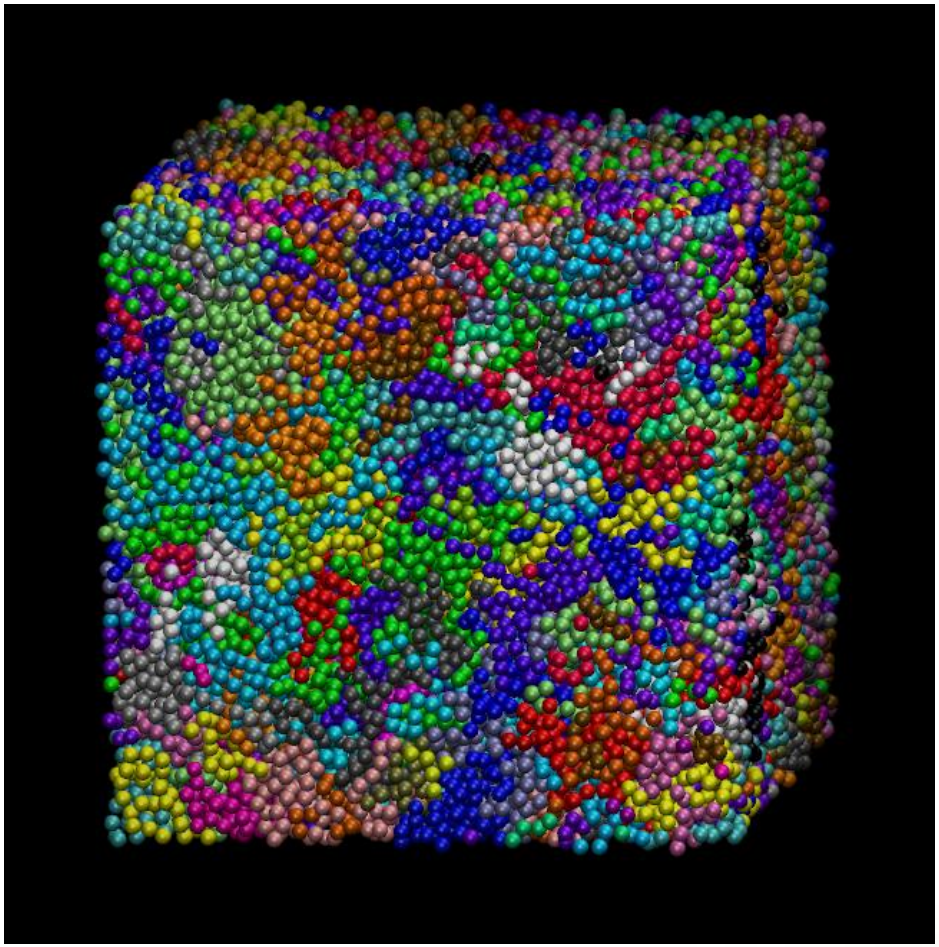
```
s1          = System()
s1.rng      = esutil.RNG()
s1.bc       = bc.OrthorhombicBC(<args>)
s1.storage =
   storage.DomainDecomposition(<args>)
      .
      . <create particlelist>
      .
s1.storage.addParticles(*property_list,
                        particle_list)
      .
      . <define interactions with parameters>
      .
s1.addInteraction(LJ)
s1.addInteraction(FENE)
s1.addInteraction(CosSq)

i1 = integrator.VelocityVerlet(s1)
      .
      . <create thermostat with parameters>
      .
i1.addExtension(langevin_thermostat)

i1.run(10000)
```

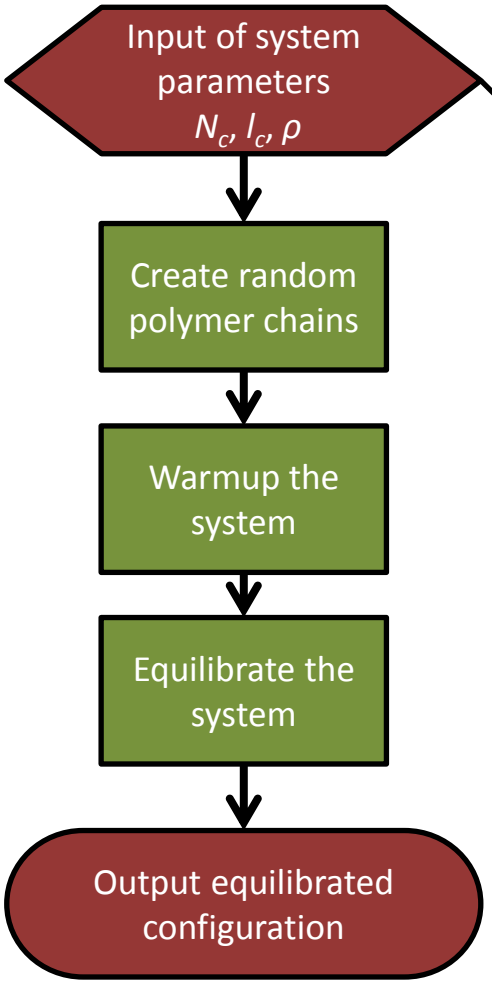
Creating an equilibrated configuration of a dense polymer melt





Snapshot of an equilibrated dense ring polymer melt, number of chains N_c=200, chain length l_c=200 particles, density ρ=0.85 [N/V^3]

Creating an equilibrated configuration of a dense polymer melt



Specify parameters of the simulation:

- Number of chains N_c
- Number of monomers per chain (chainlength) l_c
- Number density of the system ρ

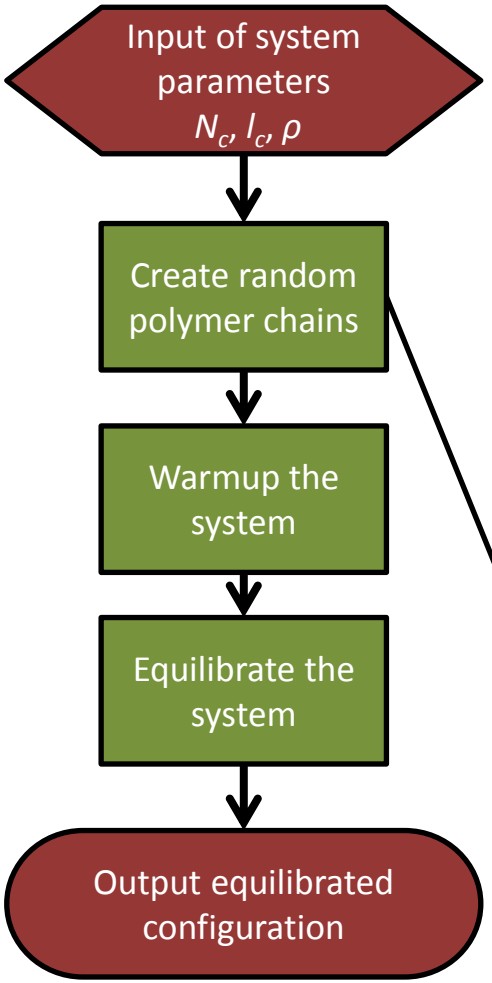
Average end-to-end distance of polymers:

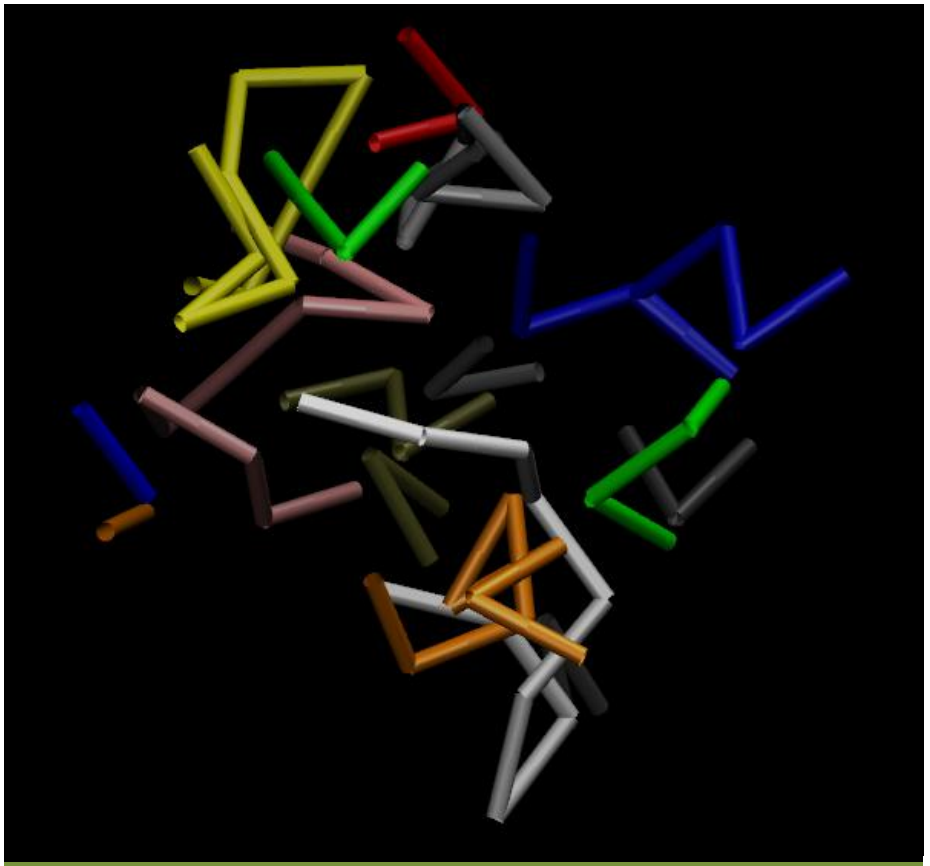
$$\langle R_{ee} \rangle \approx N^{1/2}$$

Equilibrated dense polymer melts have a random walk (RW) statistic.

Starting configurations should be random walks.

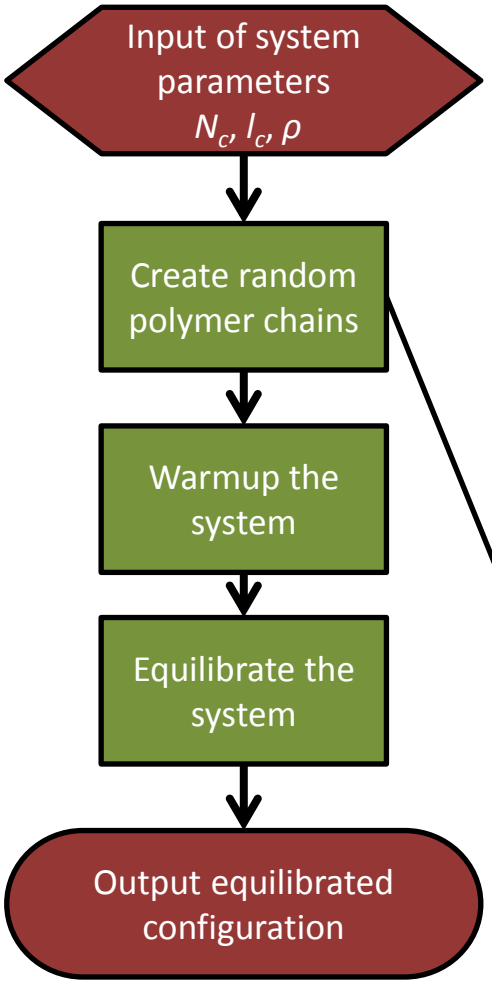
Creating an equilibrated configuration of a dense polymer melt

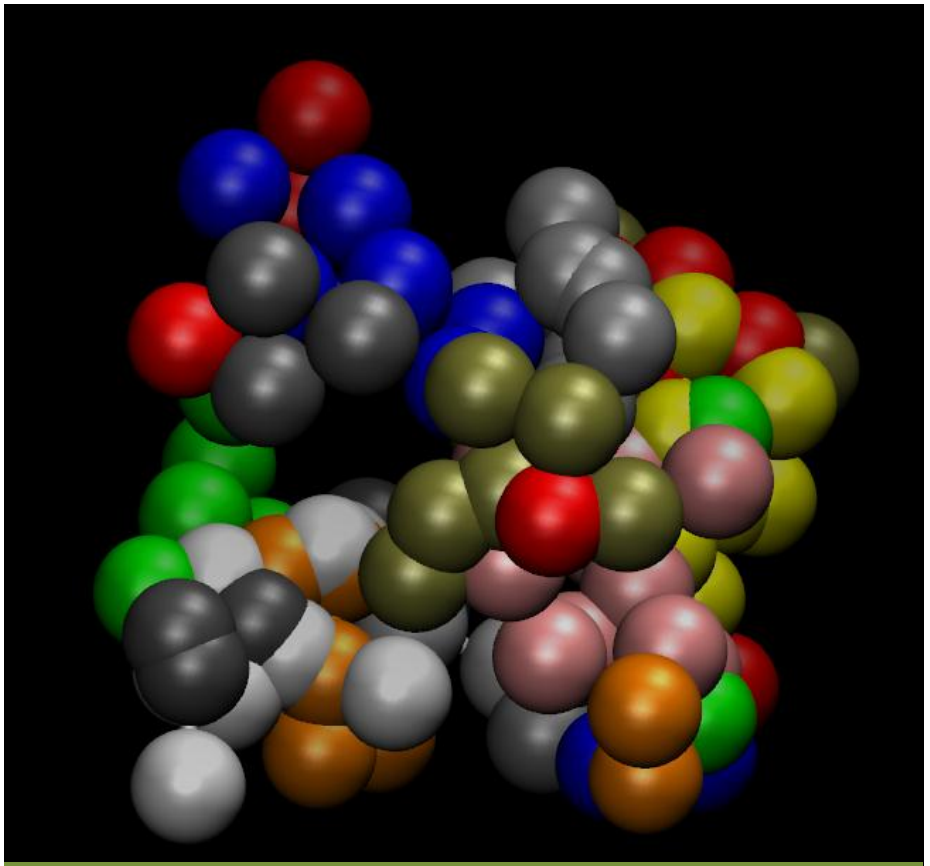




For each chain we have to define a random start position in the simulation box and then create a random walk with a fixed step length and a number of steps l_c .

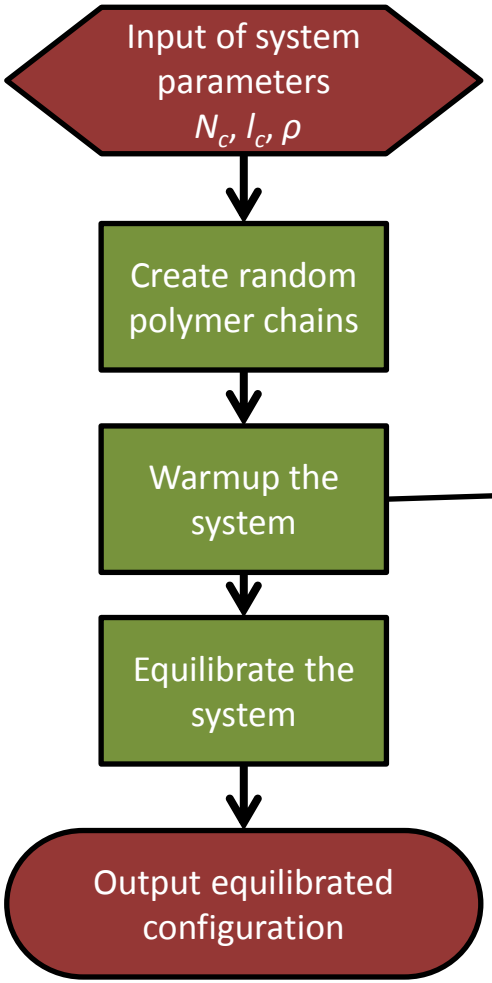
Creating an equilibrated configuration of a dense polymer melt





Chains and therefore particles in a dense random walk system are very likely to have a strong overlap. This results in extremely strong repulsive forces between these overlapping particles.

Creating an equilibrated configuration of a dense polymer melt



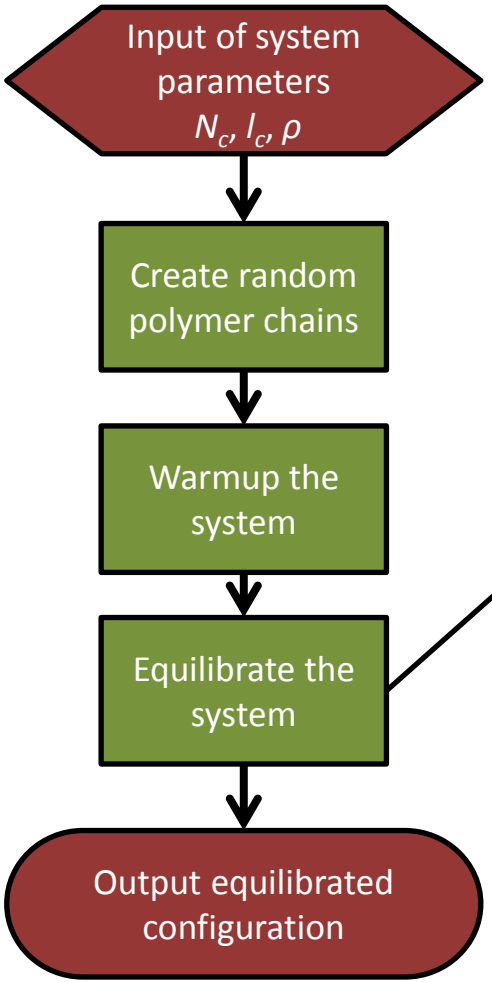
Solving the equations of motion by simply applying a Velocity-Verlet scheme to RW polymers will lead to a so called „explosion“ of the system due to numerical errors caused by extremely large numbers for the forces between overlapping particles.

Measures to be taken to avoid „explosion“:

- Decrease basic time step dt of the integrator
- Apply *force capping* to some or all of the interactions
- Gradually decrease the force capping radius
- Gradually increase epsilon or sigma of the WCA interaction

Depending on the density of the system and the polymer model (e.g. totally flexible or bending stiffness included) not all of the above mentioned steps may be necessary.

Creating an equilibrated configuration of a dense polymer melt



After the warmup evolve the system by running the Velocity-Verlet integrator with full potentials at the specified temperature for several thousands or millions of steps, depending on the chainlength and the interactions.

During the equilibration the following observables of the system should be measured in regular time intervals:

- Temperature of the system
- Total energy of the system

These will already give some hints about how long equilibration will take. Additionally the

- *mean squared displacement (msd)* of the center of mass of the polymer chains

could be measured. As soon as this property has reached a value of the order of the size of the *radius of gyration* equilibration should be finished.

Thank you for your attention !

Project website:

www.espresso-pp.de