Faunus - a flexible framework for Monte Carlo simulation

Stenqvist, Björn; Thuresson, Axel; Kurut Sabanoglu, Anil; Vacha, Robert; Lund, Mikael

Published in:
Molecular Simulation

DOI:
10.1080/08927022.2013.828207

2013

Molecular Simulation
Vol. 00, No. 00, Month 2013, 1–10

RESEARCH ARTICLE

Faunus – A Flexible Framework for Monte Carlo Simulation

Björn Stenqvist[a], Axel Thuresson[a], Anıl Kurut[a], Robert Vácha[b], and Mikael Lund[a*]

[a] *Department of Theoretical Chemistry, Lund University, POB 124, SE-22100 Lund, Sweden*; [b] *National Centre for Biomolecular Research, Faculty of Science and CEITEC - Central European Institute of Technology, Masaryk University, Kamenice 5, 625 00 Brno-Bohunice, Czech Republic*

(*August 2, 2013*)

Faunus is a set of building blocks or "statistical mechanical Lego" for constructing molecular simulation programs to study complex solutions including proteins, polymers, salts, phospholipid membranes, surfaces, and/or rigid macro-molecules. Current focus is on Metropolis Monte Carlo (MC) algorithms with support for anisotropic particles (multipolar, polarisable, sphero-cylindrical) and a flexible Hamiltonian. The design is inherently modular and it is trivial to extend functionality to cover new interaction potentials, geometries or moves. In this paper we present basic features, C++ design principles, and review selected applications. The latter includes splined pair potentials, two-dimensional parallel tempering of protein mixtures, and MC swap moves for modelling ion specific effects without ions.

Keywords: Metropolis Monte Carlo, Coarse Graining, C++ scientific programming.

1 Introduction

Faunus is an application programming interface (API) for building molecular simulation programs. It is written in C++, uses an object oriented design and aims to be flexible in terms of force fields, atom types, Monte Carlo (MC) moves, and simulation geometries.

It is an alternative to precompiled simulation packages and the end user will typically need some programming skills, albeit at an abstract level. A number of example programs for common simulation tasks are provided and minor modifications are often sufficient to get started.

We believe an API approach, also taken up by others [1–5], is beneficial since the user (i) is in *control* of the entire program flow, (ii) needs to *understand* the simulation steps, and (iii) can *modify* to accommodate new scenarios. In a teaching context these factors are especially important as students gain fundamental insight into simulation techniques. In our experience, letting students develop their own simulation programs leads to better understanding of the underlying theory and physical mechanisms. It can for example be a rewarding experience to discover that thermodynamic entropy enters a MC program with a single line of code! [1]

Let us briefly compare with the opposite but completely possible scenario, where a simulation package is used without appreciation of, say, statistical thermodynamics. While this may be a productive study, there is a distinct risk that either input, output or both is misinterpreted. By building the simulation by hand – although not necessarily from scratch – we argue that this risk can be lowered since (a) the user designed the program and is aware of all steps, and (b)

*Corresponding author. Email: mikael.lund@teokem.lu.se

[1] Namely the Metropolis acceptance criterion. [6]

ISSN: 0892-7022 print/ISSN 1029-0435 online

DOI: 10.1080/0892702YYxxxxxxxx
http://www.informaworld.com

code, input, and output are specific and minimalistic (see Listing 1). The trick is to find an API with a balanced abstraction level, i.e. in between low level programming and the "black-box" scenario described above. Faunus aims at precisely that.

Listing 1 Minimal C++ program for simulating charged Lennard-Jones particles in a periodic, cubic simulation box using the minimum image convention.

```
#include <faunus/faunus.h>
using namespace Faunus;                                    // Faunus namespace
typedef Space<Geometry::Cuboid> Tspace;                    // simulation geometry ("Space")
typedef Potential::CoulombLJ Tpair;                        // pair potential
int main() {
  atom.includefile("minimal.json");                        // load atom properties
  InputMap in("minimal.input");                            // load user input
  Energy::Nonbonded<Tspace,Tpair> pot(in);                 // non-bonded interactions, only
  Tspace spc(in);                                          // particles and geometry stored here
  Group salt(spc, in);                                     // group (range) containing all particles
  salt.addParticles(spc,in);                               // insert particles according to input file
  Move::AtomicTranslation<Tspace> mv(in,pot,spc);          // particle move class
  mv.setGroup(salt);                                       // move acts on salt group
  mv.move(1e5);                                            // move randomly 100000 times
  std::cout << spc.info() + pot.info() + mv.info();        // print final information
}
```

2 Main Features

2.1 *Flexible Hamiltonian*

Faunus is designed to handle custom force fields including arbitrary pair potentials, external potentials and many-body interactions. This is managed in a modular Hamiltonian class where terms are added as needed. Consider for example a particle system in the isothermal-isobaric ensemble (NpT) with bonded and non-bonded interactions. The corresponding Hamiltonian class is constructed by the following statement,

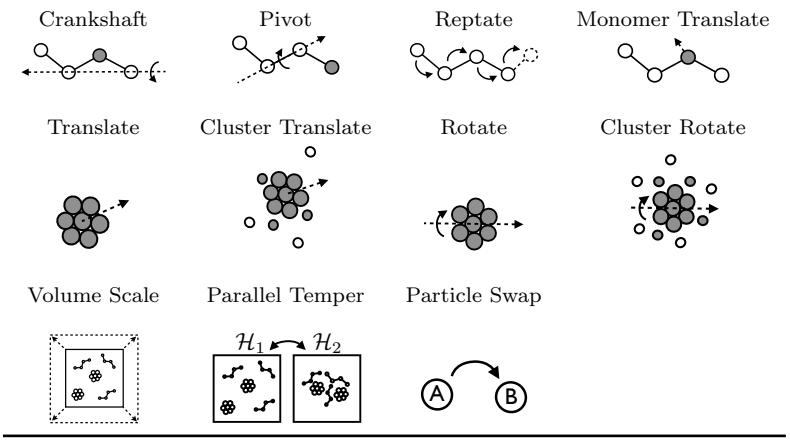
```
auto pot = Bonded() + Nonbonded() + ExternalPressure();
```

where the `auto` keyword (C++11) conveniently determines the resulting often nested and templated type. The first two terms should be self-explanatory while the last adds pV terms. A MC move that changes the simulation volume thus needs no information about pressure, p, as this is handled not by a move-specific acceptance criterion, but by the system Hamiltonian and will automatically be part of the calculated energy difference. This is particularly useful in connection with parallel tempering – more on this later in the text – where the full state energy must be deduced to exchange configurations.

2.2 *Monte Carlo Moves*

In contrast to Molecular Dynamics, particle propagation in MC is manually imposed using moves obeying the detailed balance criterium [6, 7]. Table 1 shows a selected list of available moves, following the common sequence: (i) trial move, (ii) energy change, (iii) Metropolis criterion, (iv) sample statistics. Each move is implemented as a polymorphic class with the items above as replaceable functions. Thus, new moves can be constructed either from scratch, or – *via* inheritance – by extending an existing move. The trial energy is calculated the system Hamiltonian as explained above. Since a MC move commonly involves only a subset of particles, the Hamiltonian provides fine grained control of the energy evaluation to increase run-time performance.

Table 1. Illustration of selected Monte Carlo moves available in Faunus.



2.3 *Pair Potentials*

Pair-wise, additive particle potentials all have a common interface and can be arbitrarily combined. For example, the following creates a new pair potential function by adding Coulomb to Lennard-Jones:

```
auto CoulombLJ = Coulomb() + LennardJones();
double uij = CoulombLJ(particle[i], particle[j], rij);
```

In turn this new potential can be added to another and so forth. In contrast to similar functionality in scripting languages, the above is implemented using generic programming techniques, i.e. templates [3, 8], that essentially eliminates the combination overhead at *compile* time. In the example above the plus operator is overloaded to generate a combined function object using a class template taking the parent pair potentials as template arguments.

Besides arbitrary pair potential combinations, Faunus has a general framework for automatic spline tabulation of one-dimensional functions. This can be used for a constant time-evaluation of complex potentials involving eg. root finding or trigonometric functions. So far the method of Andrea *et al.* [9] along with Hermite cubic and linear splines [10] are implemented. One line of code suffices to spline an existing pair potential,

```
PotentialTabulate<CoulombLJ> CoulombLJ_tab();
```

The potential and hence the table may depend on the particle type (charge, sizes etc.). This is automatically handled by the tabulation template that will create and store table data for each encountered pair of particles.

Depending on the algorithm used, a precision tolerance must be set on the energy and the force which is also tabulated. Fig. 1 shows a Weeks-Chandler-Andersen/Cosine squared attraction pair potential and the corresponding absolute tabulation error. To fulfill the specified energy and force tolerances of 0.001 k_BT and 0.001 k_BT/Å, the Andrea and Hermite cubic spline methods [9, 10] require 13 and 19 knots, respectively. Given a certain tolerance, the number of knots usually decrease with increasing degree of the polynomial. Finding the correct knot for an arbitrary distance is therefore faster, while evaluation of the polynomial is slower. For optimal performance the choice of tabulation method hence depends on the problem.

In many scenarios it is convenient to specify different potentials between different particle indexes. Consider for example a three-bead, coarse grained lipid [11] with the following internal bonds,

```
PotentialMap pot;
pot.add(0, 1, Potential::FENE() );
```

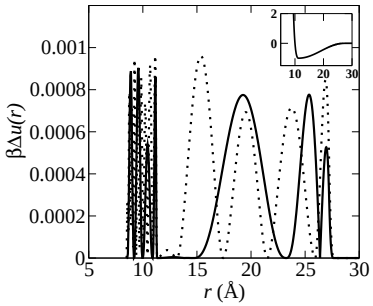


Figure 1. Pair potential tabulation error using Andrea *et al.* (full line) and Hermite cubic splines (dotted line) with $\beta u_{\mathrm{tol}} = 0.001$. The original Weeks-Chandler-Andersen pair potential combined with a cosine squared attraction is shown in the inset.

```
pot.add(1, 2, Potential::FENE() );
pot.add(0, 2, Potential::Harmonic() );
```

where the numbers refer to particle index. During simulation, the pair potential call is determined at run-time which may affect performance due to lack of function inlining. In contrast, a splined potential share a common function call – known at compile time – and only data differs between each particle pair. Inline optimisation is therefore at least theoretically possible and may lead to faster execution depending on table lookup overhead and pair potential complexity. Table 2 shows benchmarks of MC simulations of a bilayer formed using the above lipid model [11] as well as of an isotropic liquid where atoms interact via a Yokawa/Buckingham potential. The latter involves two exponentials, one square root, and one division.

Table 2. Relative speedup using splined pair potentials in MC simulations. The tabulated energy tolerance is 0.001 k_BT, resulting in a typical knot ratio of 1:2:40 between the three spline methods. Benchmark performed on a x86-64 CPU using double arithmetic precision.

System	Base	Andrea	Hermite	Linear
3-bead lipid bilayer[a]	1.0	1.2	1.3	1.0
Isotropic liquid[b]	1.0	1.5	1.4	1.0

[a]100 lipid molecules, isotension ensemble.

[b]2500 particles, canonical ensemble.

2.4 *Anisotropic Particles*

In addition to isotropic particles commonly used in molecular simulations, Faunus supports anisotropic particles where the interaction potential is orientational dependent. Particle here refers to the smallest, indivisible building block. All such particles are implemented using a matrix template library [12] and the multipole expansion below is for example coded directly using the elegant tensor form. In addition to expressive vector manipulation, the matrix library provides quaternions and transformation matrices used for geometric operations.

2.4.1 Multipoles

In addition to a centro-symmetric monopole, particles in Faunus can have dipolar and quadrupolar moments. A well-known example is the Stockmayer fluid, consisting of point dipolar Lennard-Jones particles [13]. Using atomic units and Einstein summation notation, the interaction energy between two charge distributions, $A = [\boldsymbol{r}_i, q_i]$ and $B = [\boldsymbol{r}_j, q_j]$, is given by a multipole expansion,

$$U = \sum_{i=1}^{N_A} \sum_{j=1}^{N_B} \frac{q_i q_j}{|\boldsymbol{R} + \boldsymbol{r}_j - \boldsymbol{r}_i|} \tag{1}$$

$$U = T_{\boldsymbol{R}} q^A q^B + \boldsymbol{T}^{\alpha}_{\boldsymbol{R}}(q^A \boldsymbol{\mu}^B_{\alpha} - \boldsymbol{\mu}^A_{\alpha} q^B) + \boldsymbol{T}^{\alpha\beta}_{\boldsymbol{R}} \left(\frac{1}{3} q^A \boldsymbol{\theta}^B_{\alpha\beta} - \boldsymbol{\mu}^A_{\alpha} \boldsymbol{\mu}^B_{\beta} + \frac{1}{3} \boldsymbol{\theta}^A_{\alpha\beta} q^B \right) + \dots \tag{2}$$

$$\boldsymbol{T}^{(n)}_{\boldsymbol{R}} = \boldsymbol{\nabla}^n \left(\frac{1}{|\boldsymbol{R}|} \right) \tag{3}$$

where the Taylor expansion is valid when the internal distances, $|\boldsymbol{r}_i|$ and $|\boldsymbol{r}_j|$, are much smaller than the separation between the expansion centers, R.

The first term in Eq. 2 is the ion-ion interaction, the second is the ion-dipole, and the third term accounts for dipole-dipole and ion-quadrupole interactions. In the case of charged particles, both ion-dipole and ion-quadrupole should be included as they are of equal order, i.e. they decay with the same distance dependence. While isotropic particle force fields by far dominate published molecular simulation studies, a number of multipolar force fields are being developed [14–16].

2.4.2 Polarisation

Particles may *become* anisotropic by acquiring an induced dipole moment when exposed to an electric or magnetic field. This is implemented with an iterative scheme where the induced moments of all polarisable particles are updated self-consistently. The field on the i'th particle with polarisability tensor α_i is

$$\mathbf{E}_i = \sum_{j \neq i}^{N} \left(-\boldsymbol{T}^{(1)}_{\boldsymbol{R}_{ij}} q_j + \boldsymbol{T}^{(2)}_{\boldsymbol{R}_{ij}} \boldsymbol{\mu}_j \right) \tag{4}$$

and the induced moment becomes $\boldsymbol{\mu}_{i,ind} = \boldsymbol{\alpha}_i \mathbf{E}_i$. The particle polarisability is implemented in such a way that both isotropic and anisotropic values can be used, i.e. $\boldsymbol{\alpha}_i$ can be either a scalar or a matrix.

2.4.3 Spherocylinders

The main body of this particle type is a *spherocylinder*, i.e. a cylinder with hemispherical caps at both ends, thus providing a smooth rod-like particle with a variable aspect ratio [17, 18] – see Figure 2A. An attractive angular wedge can be added to the side, creating a patchy spherocylinder (PSC). This patch can either run along the whole axis including the ends, or it can be limited to the cylindrical part (Fig. 2A and B). Used in implicit solvent, the attractive interaction should include all effective interactions between the studied particles including hydrophobic interactions, hydrogen bonds, salt bridges, charges, etc. The repulsive interaction is described using an isotropic Weeks-Chandler-Andersen potential which is a Lennard-Jones potential shifted and truncated at the minimum. The attractive interaction is determined from overlapping segments of the two PSC patches and has a cosine squared distance dependency. It is possible to further tune a switch range (distance from the potential minimum to the cutoff). The step change of the interaction at the side of the patch is smoothed by linear transition region (typically 5°).

This particle model was developed to study amphiphilic prolate objects such as helical peptides and protein fibrils. For more details see ref. [19].

2.5 *Simulation Boundaries*

Since we always simulate a finite system size, particles are confined in a simulation container with particular boundary conditions and shape. A number of such containers – here called *geometries* – are supported as shown in Table 3. The geometry handles (i) distance calculations, (ii) boundary conditions, and (iii) volume scaling. This task distribution makes it easy to implement any geometry without modifying code, nor obstruct run-time efficiency.

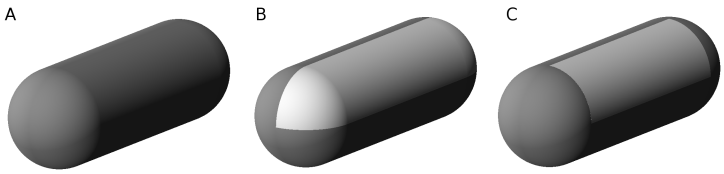
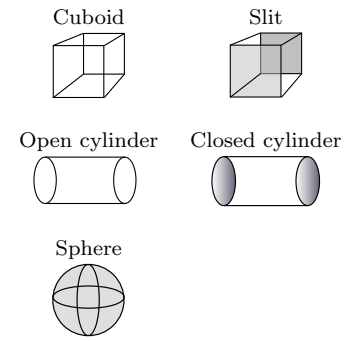


Figure 2. (A) Spherocylinder with isotropic repulsion, (B and C) spherocylinder with an attractive stripe that runs along the main axis. In (C) only the cylindrical part is attractive (light gray), while in (B) the patch is on the cylindrical part as well as on the hemispherical caps.

Table 3. Simulation geometries. Periodic boundaries can be applied in 3D (cuboid), 2D (slit), 1D (open cylinder) or not at all (sphere, closed cylinder).



3 Selected Applications

3.1 *Rare events – Parallel Tempering*

Parallel tempering, or "replica exchange", is a powerful technique to sample regions of coordinate space separated by large energy barriers [7, 20–23]. Although the majority of reported tempering studies are done via *temperature*, any parameter in the Hamiltonian can be used. In Faunus, tempering is implemented in a general way allowing the full Hamiltonian to exchange between replicas. Tempering in the grand canonical ensemble is, however, not yet supported.

We have previously used parallel tempering in the isothermal-isobaric ensemble (NpT) to construct phase diagrams of protein mixtures, where the gas-liquid transition is investigated as a function of salt concentration [24]. Since the studied proteins are oppositely charged, strongly bound aggregates form and the system tends to sample only high density configurations. To improve sampling of the dilute phase we used a two-dimensional array of NpT simulations at different pressures (p) and salt concentrations (proportional to κ). Parallel tempering was then used to exchange configurations according to the following acceptance criterion [25],

$$\begin{aligned} acc[(i,j)\rightarrow(j,i)] &= \min\Big\{1, \exp\Big(-\beta[U(\kappa_i,\mathbf{r}_j) \\ &\quad +U(\kappa_j,\mathbf{r}_i)-U(\kappa_i,\mathbf{r}_i)-U(\kappa_j,\mathbf{r}_j) \\ &\quad +p_iV_j+p_jV_i-p_iV_i-p_jV_j]\Big)\Big\} \\ &= \min\Big\{1, \exp\Big(-\beta[\Delta U+\Delta pV]\Big)\Big\} \end{aligned} \tag{5}$$

$1/\beta = k_BT$ is the thermal energy, κ is inverse Debye screening length, V is volume, U is the Hamiltonian. As already discussed, the system Hamiltonian in Faunus returns the full energy, including pV and other possible term(s) whereby no move-specific acceptance criteria for tempering is needed. This design makes it trivial to temper in any energy field parameter.

The exchange events between three selected replicas and the resulting Gibbs free energies are shown in Fig. 3. The choice of both pressure and salt concentration as tempering parameters is in this case particularly advantageous as a scan of these parameters is regardlessly needed to construct the phase diagram. Thus, in addition to much improved sampling, tempering offers a convenient way of parallelising serial MC code [22]. In contrast to MD, where all pair interactions are evaluated in each time step, a typical MC move updates only a subset of particles and the relative communication overhead between parallel runs may obstruct runtime performance. Parallel tempering offers an elegant solution around this issue.

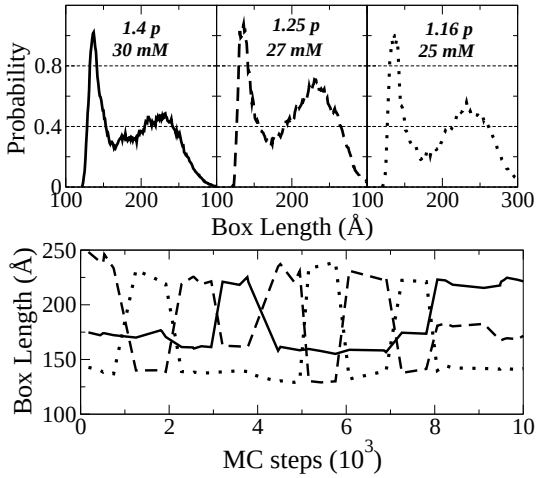


Figure 3. *Top*: Simulation box size probability profiles for a mixture of 40 coarse grained proteins obtained by simultaneous tempering in osmotic pressure, p, and salt concentration [24]. *Bottom*: Exchange events as a function of MC steps.

3.2 *Particle Swap Moves*

Since moves in MC need not follow real-time dynamics, particles can be swapped to mimic equilibrium processes, for example of the type $\mathrm{AB} \rightleftharpoons \mathrm{A} + \mathrm{B}$ with dissociation constant, K_d. Describing one of the particles, B, solely by its chemical potential, or activity we construct a swap move with the following trial energy,

$$\begin{aligned} \beta\Delta U_{\mathrm{AB}\to\mathrm{A}} &= -\ln([\mathrm{AB}]/[\mathrm{A}]) \\ &= (p\mathrm{B} - pK_d)\ln 10 + \delta u \end{aligned} \tag{6}$$

where pB is the minus logarithm of the activity of species B and δu is the energy difference due to interactions [26, 27]. In Faunus, this equilibrium analysis is implemented as part of the Hamiltonian, and by simply probing the particle types in the system, the free energy state is established. Multiple equilibrium processes can compete for binding sites, allowing coupled equilibrium processes to be solved during simulation. While an analytical solution obviously exists for the ideal case, the numerical MC approach has the distinct advantage that it can be extended to any complexity and explicitly capture intermolecular interactions, i.e. activity coefficients.

We have previously used this implicit titration scheme for constant pH studies of proteins [28] and for more exotic Hofmeister related effects such as thiocyanate and iodide binding to amino acid motifs [29]. The latter was done by defining not only proton binding sites, but also specific thiocyanate binding to the protein backbone and hydrophobic sites using experimental dissociation constants – see Table 4. Therefore, in this model the protein charge distribution is no longer a mere function of pH, but also of the salt concentration and *type*. With the latter we mean the chemical specificity that for example distinguish thiocyanate and chloride anions.

Table 4. Binding constants of protons and thiocyante anions to amino acid motifs [29, 30]. These enters the MC acceptance criteria for particle swap moves.

Residue	pK_{d,H^+}	pK_{d,SCN^-}
Asp	4.0	
Glu	4.4	
His	6.3	
Tyr	9.6	
Lys	10.4	
Cys	10.8	
Arg	12.0	
Ctr	2.6	
Ntr	7.5	
Ala		0.82
Ile		0.82
Leu		0.82
Met		0.82
Phe		0.82
Pro		0.82
Trp		0.82
Val		0.82
backbone		0.60
backbone (Pro,Gly)		1.30

Using a coarse grained protein representation with two beads per amino acid and implicit solvent, we have calculated the orientationally averaged interaction free energy between two γ-crystallin proteins, see Fig. 4. This shows that the proteins attract each other more in a thiocyanate solution at low pH, while at high pH chloride causes more attraction. This "Hofmeister reversal" coincides with experimentally measured osmotic second virial coefficients [31] and can be explained using electrostatic arguments based on the protein net charge arising from both protons and bound anions.

Specific ion binding to amino acid motifs is intimately connected with solvent-solvent as well as solvent-solute interactions and a full description requires an atomistic level model. In the present case, the MC swap move neatly brings in this information – obtained either from atomistic simulations or experiment – and thus allows for studies of length scales otherwise impossible with an all atom model.

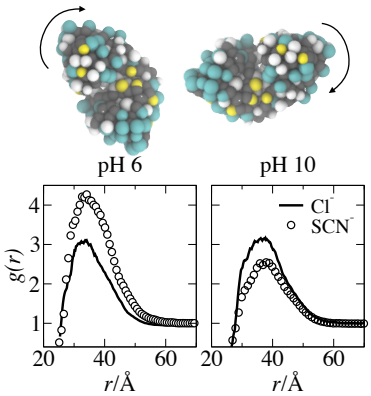


Figure 4. Radial distribution function, $g(r)$, between two γ-crystallin proteins in aqueous sodium chloride and sodium thiocyanate solutions. Note the Hofmeister reversal when changing pH. More information can be found in ref. [29].

4 Conclusions and Outlook

A main consideration when developing Faunus was to allow for straightforward implementation of new functionality, achieved through an object oriented approach with minimal dependencies between objects. This flexibility is herein illustrated by inclusion of arbitrary particle types, energy functions and Monte Carlo moves. We find that generic programming is particularly

useful for obtaining a good balance between flexibility, performance and an intuitive API. This, however, at the cost of longer compile times and limited support for scripting interfaces.

Near future development will focus on efficient energy summation methods and advanced moves to improve sampling. The latter is a key advantages of MC over MD and may well compensate for the difficulties of parallelising the inherently sequential Markov chain beyond trivial multiple walkers.

5 Availability

Faunus is released under the GNU Public License and is available at `http://github.com/mlund/faunus`. Early versions previously described [32] are hosted at `http://sourceforge.net/projects/faunus`.

Acknowledgements

The authors are grateful for financial support from Södra's Research Foundation; the Swedish Research Council; Organizing Molecular Matter, eSSENCE, and LUNARC in Lund. RV acknowledges EU 7th Framework Program (contract No. 286154 - SYLICA) and the European Regional Development Fund (CZ.1.05/1.1.00/02.0068 CEITEC).

References

[1] M. Troyer, B. Ammon, and E. Heeb, *Parallel Object Oriented Monte Carlo Simulations*, in *Computing in Object-Oriented Parallel Environments*, , D. Caromel, R. Oldehoeft, eds., Springer Berlin Heidelberg, 1998, pp. 191–198.
[2] K. Hinsen, *The molecular modeling toolkit: A new approach to molecular simulations*, Journal of Computational Chemistry 21 (2000), pp. 79–85.
[3] G. Brown, H. Lee, and T. Schulthess, *C++ and Generic Programming for Rapid Development of Monte Carlo Simulations*, in *Computer Simulation Studies in Condensed-Matter Physics XVI*, , D. Landau, S. Lewis, eds., Springer Berlin Heidelberg, 2006, pp. 157–168.
[4] J.D. Halverson, T. Brandes, O. Lenz, A. Arnold, S. Bevc, V. Starchenko, K. Kremer, T. Stuehn, and D. Reith, *ESPResSo++: A modern multiscale simulation package for soft matter systems*, Computer Physics Communications 184 (2013), pp. 1129–1149.
[5] P. Eastman, M.S. Friedrichs, J.D. Chodera, R.J. Radmer, C.M. Bruns, J.P. Ku, K.A. Beauchamp, T.J. Lane, L.P. Wang, D. Shukla, and al.et , *OpenMM 4: A Reusable, Extensible, Hardware Independent Library for High Performance Molecular Simulation*, Journal of Chemical Theory and Computation 9 (2013), pp. 461–469.
[6] N. Metropolis, A.W. Rosenbluth, M.N. Rosenbluth, A.H. Teller, and E. Teller, *Equation of State Calculations by Fast Computing Machines*, J. Chem. Phys. 21 (1953), p. 1087.
[7] D. Frenkel and B. Smit, *Understanding Molecular Simulation: From Algorithms to Applications* 2nd ed., Academic Press, 2001.
[8] A. Alexandrescu, *Modern C++ Design: Generic Programming and Design Patterns Applied* Addison-Wesley Professional, 2001.
[9] T.A. Andrea, W.C. Swope, and H.C. Andersen, *The role of long ranged forces in determining the structure and properties of liquid water)*, J. Chem. Phys. 79 (1983), p. 4576.
[10] E. Suli and D.F. Mayers, *An introduction to Numerical Analysis* Cambride University Press, 2008.
[11] I.R. Cooke and M. Deserno, *Solvent-free model for self-assembling fluid bilayer membranes: Stabilization of the fluid phase based on broad attractive tail potentials*, J. Chem. Phys. 123 (2005), p. 224710.
[12] G. Guennebaud, B. Jacoband others, Eigen v3; http://eigen.tuxfamily.org.
[13] W.H. Stockmayer, *Second Virial Coefficients of Polar Gases*, The Journal of Chemical Physics 9 (1941), p. 398.
[14] A. Holt and G. Karlström, *Inclusion of the quadrupole moment when describing polarization. The effect of the dipole-quadrupole polarizability*, Journal of Computational Chemistry 29 (2008), pp. 2033–2038.
[15] J.W. Ponder, C. Wu, P. Ren, V.S. Pande, J.D. Chodera, M.J. Schnieders, I. Haque, D.L. Mobley, D.S. Lambrecht, R.A. DiStasio, and al.et , *Current Status of the AMOEBA Polarizable Force Field*, The Journal of Physical Chemistry B 114 (2010), pp. 2549–2564.
[16] C. Kramer, P. Gedeck, and M. Meuwly, *Multipole-Based Force Fields from ab Initio Interaction Energies and the Need for Jointly Refitting All Intermolecular Parameters*, Journal of Chemical Theory and Computation 9 (2013), pp. 1499–1511.
[17] L. Onsager, *The Effects of Shape on the Interaction of Colloidal Particles*, Annals of the New York Academy of Sciences 51 (1949), pp. 627–659.
[18] P. Bolhuis and D. Frenkel, *Tracing the phase boundaries of hard spherocylinders*, The Journal of Chemical Physics 106 (1997), p. 666.
[19] R. Vcha and D. Frenkel, *Relation between Molecular Shape and the Morphology of Self-Assembling Aggregates: A Simulation Study*, Biophys. J. 101 (2011), pp. 1432–1439.

[20] R.H. Swendsen and J.S. Wang, *Replica Monte Carlo Simulation of Spin-Glasses*, Physical Review Letters 57 (1986), pp. 2607–2609.
[21] D.J. Earl and M.W. Deem, *Parallel tempering: theory, applications, and new perspectives.*, Phys. Chem. Chem. Phys. 7 (2005), pp. 3910–6.
[22] E. Rosta and G. Hummer, *Error and efficiency of replica exchange molecular dynamics simulations*, J. Chem. Phys. 131 (2009), p. 165102.
[23] J.D. Chodera and M.R. Shirts, *Replica exchange and expanded ensemble simulations as Gibbs sampling: Simple improvements for enhanced mixing*, The Journal of Chemical Physics 135 (2011), p. 194110.
[24] A. Kurut, B.A. Persson, T. Akesson, J. Forsman, and M. Lund, *Anisotropic Interactions in Protein Mixtures: Self Assembly and Phase Behavior in Aqueous Solution*, The Journal of Physical Chemistry Letters 3 (2012), pp. 731–734.
[25] M. Doxastakis, V.G. Mavrantzas, and D.N. Theodorou, *Atomistic Monte Carlo simulation of cis-1,4 Polyisoprene Melts. II. Parallel Tempering End-Bridging Monte Carlo Simulations*, J. Chem. Phys. 115 (2001), pp. 11352–11361.
[26] A.P. Sassi, S. Beltrn, H.H. Hooper, H.W. Blanch, J. Prausnitz, and R.A. Siegel, *Monte Carlo simulations of hydrophobic weak polyelectrolytes: Titration properties and pH-induced structural transitions for polymers containing weak electrolytes*, The Journal of Chemical Physics 97 (1992), p. 8767.
[27] M. Ullner, B. Jönsson, and P.O. Widmark, *Conformational properties and apparent dissociation constants of titrating polyelectrolytes: Monte Carlo simulation and scaling arguments*, The Journal of Chemical Physics 100 (1994), p. 3365.
[28] A.A.R. Teixeira, M. Lund, and F.L.B. Silvada , *Fast Proton Titration Scheme for Multiscale Modeling of Protein Solutions*, Journal of Chemical Theory and Computation 6 (2010), pp. 3259–3266.
[29] A. Kurut and M. Lund, *Solution electrostatics beyond pH: a coarse grained approach to ion specific interactions between macromolecules*, Faraday Discussions 160 (2013), p. 271.
[30] K.B. Rembert, J. Paterov, J. Heyda, C. Hilty, P. Jungwirth, and P.S. Cremer, *Molecular Mechanisms of Ion-Specific Effects on Proteins*, Journal of the American Chemical Society 134 (2012), pp. 10039–10046.
[31] S. Finet, F. Skouri-Panet, M. Casselyn, F. Bonneté, and a. Tardieu, *The Hofmeister effect as seen by SAXS in protein solutions*, Current Opinion in Colloid & Interface Science 9 (2004), pp. 112–116.
[32] M. Lund, M. Trulsson, and B. Persson, *Faunus: An object oriented framework for molecular simulation*, Source Code for Biology and Medicine 3 (2008), p. 1.