Computer Physics Communications 183 (2012) 363–369

Contents lists available at SciVerse ScienceDirect

Computer Physics Communications

www.elsevier.com/locate/cpc

A molecular dynamics implementation of the 3D Mercedes-Benz water model ☆

T. Hynninen [a,b,*], C.L. Dias [c], A. Mkrtchyan [d], V. Heinonen [b], M. Karttunen [d,e], A.S. Foster [a,b], T. Ala-Nissila [b,f]

[a] *Department of Physics, Tampere University of Technology, P.O. Box 692, FI-33101 Tampere, Finland*
[b] *Department of Applied Physics, Aalto University School of Science, P.O. Box 11100, FI-00076 Aalto, Espoo, Finland*
[c] *Department of Biochemistry, University of Toronto, Toronto, Ontario, Canada M5S 1A8*
[d] *Department of Applied Mathematics, The University of Western Ontario, London, Ontario, Canada N6A 5B7*
[e] *Department of Chemistry, University of Waterloo, Waterloo, Ontario, Canada N2L 3G1*
[f] *Department of Physics, Brown University, Providence, RI 02912-1843, United States*

ARTICLE INFO

Article history:
Received 17 June 2011
Received in revised form 25 September 2011
Accepted 29 September 2011
Available online 14 October 2011

Keywords:
Molecular dynamics
Fortran 90
Mercedes-Benz model
Water

ABSTRACT

The three-dimensional Mercedes-Benz model was recently introduced to account for the structural and thermodynamic properties of water. It treats water molecules as point-like particles with four dangling bonds in tetrahedral coordination, representing H-bonds of water. Its conceptual simplicity renders the model attractive in studies where complex behaviors emerge from H-bond interactions in water, e.g., the hydrophobic effect. A molecular dynamics (MD) implementation of the model is non-trivial and we outline here the mathematical framework of its force-field. Useful routines written in modern Fortran are also provided. This open source code is free and can easily be modified to account for different physical context. The provided code allows both serial and MPI-parallelized execution.

Program summary

Program title: CASHEW (Coarse Approach Simulator for Hydrogen-bonding Effects in Water)
Catalogue identifier: AEKM_v1_0
Program summary URL: http://cpc.cs.qub.ac.uk/summaries/AEKM_v1_0.html
Program obtainable from: CPC Program Library, Queen's University, Belfast, N. Ireland
Licensing provisions: Standard CPC licence, http://cpc.cs.qub.ac.uk/licence/licence.html
No. of lines in distributed program, including test data, etc.: 20 501
No. of bytes in distributed program, including test data, etc.: 551 044
Distribution format: tar.gz
Programming language: Fortran 90
Computer: Program has been tested on desktop workstations and a Cray XT4/XT5 supercomputer.
Operating system: Linux, Unix, OS X
Has the code been vectorized or parallelized?: The code has been parallelized using MPI.
RAM: Depends on size of system, about 5 MB for 1500 molecules.
Classification: 7.7
External routines: A random number generator, Mersenne Twister (http://www.math.sci.hiroshima-u.ac.jp/m-mat/MT/VERSIONS/FORTRAN/mt95.f90), is used. A copy of the code is included in the distribution.
Nature of problem: Molecular dynamics simulation of a new geometric water model.
Solution method: New force-field for water molecules, velocity–Verlet integration, representation of molecules as rigid particles with rotations described using quaternion algebra.
Restrictions: Memory and cpu time limit the size of simulations.
Additional comments: Software web site: https://gitorious.org/cashew/.
Running time: Depends on the size of system. The sample tests provided only take a few seconds.

☆ This paper and its associated computer program are available via the Computer Physics Communications homepage on ScienceDirect (http://www.sciencedirect.com/science/journal/00104655).

* Corresponding author at: Department of Physics, Tampere University of Technology, P.O. Box 692, FI-33101 Tampere, Finland.
E-mail address: teemu.hynninen@tut.fi (T. Hynninen).

0010-4655/$ – see front matter © 2011 Elsevier B.V. All rights reserved.
doi:10.1016/j.cpc.2011.09.008

1. Introduction

Water is among the most abundant and important chemical compounds on Earth. Its properties have crucial implications ranging from the dynamics of the global climate to the chemistry of organic life [1,2]. Not only is water ubiquitous, it is in many ways anomalous among other common fluids: the heat capacity of water is very large, water is denser as a liquid than solid at normal atmospheric pressure, ice tends to melt under high pressure, etc. Many of the peculiar properties of water arise from the way water molecules dominantly interact via highly anisotropic hydrogen bonding (H-bonds, HB), see e.g. [3–5].

There is an abundance of water models of various levels of sophistication and complexity, designed for different types of simulation tasks, for reviews and examples, see e.g., [6–13]. As is typical in numerical simulations, choosing the computational scheme is based on a balance between required accuracy and available resources. In addition, when simplified models are used, one should always employ one that works best for the problem at hand.

A new water model called the three-dimensional (3D) Mercedes-Benz (MB) model was introduced recently [14–16] as an extension to a similar two-dimensional model [17–21]. It is also closely related to another 3D model of similar type [22]. This model treats water molecules as point-like particles with four dangling bonds in tetrahedral coordination, representing the H-bonds of water. It is designed to be a simple model that reproduces the structural and thermodynamic properties of water especially in studies of the hydrophobic effect [15,16].

By treating for H-bonds explicitly and by being able to account naturally for the hydrophobic effect, the MB model is suitable for studies of macromolecules in an aqueous environment as the competition between these forces is the driving mechanism in phenomena such as protein folding [23,24], condensation of DNA [27] and lipoprotein assembly [25,26]. Due to its conceptual simplicity, the 3D MB model is particularly interesting and suitable for investigations of the physics emerging from such systems. While it is relatively easy to implement a 3D MB Monte Carlo code [14,16,22], this is not the case for an MD program [15]. The latter is recommended to simulate dynamic properties of macromolecules in water since Monte Carlo (apart from kinetic MC) cannot account for kinetics. Furthermore, Monte Carlo moves for extended molecules are non-local and computationally inefficient in simulations with explicit solvent as most moves would produce an overlap of the macromolecule with the surrounding water. It is, of course, possible to use methods such as configurational bias Monte Carlo, but one loses the conceptual simplicity as compared to MD.

Mathematically, the 3D MB force field is more complicated than, for instance, models where water molecules are described using partial point charges, such as the TIPnP models. However, the point charge models operate at a different level of detail than the MB model and require the application of Ewald summation or the fast multipole method (FMM) to achieve accurate results. The 3D MB model, on the other hand, operates at the Lennard-Jones level where the details operating at long distances have been coarse-grained out. This gives the advantage of being able to work with a water model that reproduces most of the physical properties of water yet allowing to use Lennard-Jones description for polymers and other molecules, and without the need to use Ewald summation or FMM. The 3D MB model can also be easily made compatible with the even more coarse-grained approaches such as the MARTINI model [28].

Here, we present the MD program CASHEW incorporating the 3D MB model. The acronym stands for "Coarse Approach Simulator for Hydrogen-bonding Effects in Water". Although the MB potential is presented in detail in Ref. [14], in this paper, we derive the force field corresponding to the potential, as implemented in CASHEW. The program offers flexible input and output and ready-to-use machinery for MD simulations of water. Inclusion of polymers and simple molecules at the Lennard-Jones level is straightforward and we have also aimed at making the source code easily readable to allow other developers to implement the model in their programs with relative ease. The latest version, and possible future special versions, of the code is/are available at the listed web site.

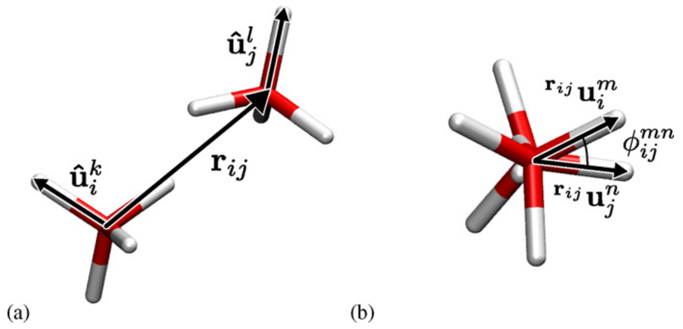

Fig. 1. The Mercedes-Benz molecules are described as a point with four tetrahedrally coordinated unit vectors attached. These vectors represent dangling H-bonds. In principle two of them are donors and the other two are acceptors, though the model does not discriminate between types of the dangling bonds. (a) The vector linking two molecules, i and j, is denoted $\mathbf{r}_{ij}$. The dangling bond vectors of molecule i are denoted $\hat{\mathbf{u}}_i^k$, $k = 1, \ldots, 4$ (and similarly for molecule j). (b) The projections of vectors on the plane perpendicular to $\mathbf{r}_{ij}$ are denoted $^{\mathbf{r}_{ij}}\mathbf{u}$. The dihedral angles ϕ_{ij}^{mn} are defined as angles between the projections of the bond vectors of molecules i and j.

2. Water simulations

2.1. The 3D Mercedes-Benz model

The details of the MB model are explained in the following subsections using the notations defined here and illustrated in Fig. 1: indices i and j are the summation indices denoting MB molecules. The index α refers to the MB molecule for which we are calculating the forces or torques (i.e., the molecule with respect to which we are differentiating). The relative positions of MB-particles i and j are denoted by $\mathbf{r}_{ij} = \mathbf{R}_j - \mathbf{R}_i$, the position of molecule i being $\mathbf{R}_i$ [see Fig. 1(a)]. The dangling hydrogen bond "arms" (HB) of molecule i are written as $\hat{\mathbf{u}}_i^k$, $k = 1, \ldots, 4$. The hat symbol ($\hat{}$) is used for unit vectors, so we write the unit vector pointing from i to j as $\hat{\mathbf{r}}_{ij}$. The indices k and m are always the summation indices for the dangling bond vectors of molecule i (or α); l and n are reserved for molecule j. Projection of a unit vector $\hat{\mathbf{u}}$ on the plane defined as the normal plane of $\mathbf{r}$ is marked $^{\mathbf{r}}\mathbf{u}$. Such projections of the HB vectors are used for defining the dihedral angles of the molecules: The angle between the projections $^{\mathbf{r}_{ij}}\mathbf{u}_i^m$ and $^{\mathbf{r}_{ij}}\mathbf{u}_j^n$ is denoted ϕ_{ij}^{mn} [see Fig. 1(b)].

2.1.1. Mercedes-Benz potential

The 3D Mercedes-Benz potential and its basic properties are introduced in Ref. [14]. The definition of the potential below is the same as the one presented in the above mentioned reference, however, the notation is somewhat different. The physical significance of the various terms is commented here, but for a thorough discussion see Ref. [14].

The potential energy of MB molecules comes from Lennard-Jones (LJ) and hydrogen bond contributions:

$$U = \frac{1}{2}\sum_{\substack{i \\ j\neq i}} U_{ij} = \frac{1}{2}\sum_{\substack{i \\ j\neq i}} U_{ij}^{\mathrm{LJ}} + U_{ij}^{\mathrm{HB}} \tag{1}$$

$$U_{ij}^{\mathrm{LJ}} = 4\varepsilon_{\mathrm{LJ}}\left[\left(\frac{\sigma_{\mathrm{LJ}}}{r_{ij}}\right)^{12} - \left(\frac{\sigma_{\mathrm{LJ}}}{r_{ij}}\right)^{6}\right] \tag{2}$$

$$U_{ij}^{\mathrm{HB}} = \varepsilon_{\mathrm{HB}} \sum_{\substack{k=1\\l=1}}^{4} b_i G_{ij}^{kl} \Phi_{ij}^{kl}. \tag{3}$$

The H-bond potential (3) is a product of a Gaussian factor (G_{ij}^{kl}), a dihedral factor (Φ_{ij}^{kl}), and a bond-order factor (b_i). The Gaussian factor is a product of non-normalized Gaussian functions of the intermolecular distance r_{ij} and the angles between the HB-arms and the vector connecting the molecules [see Fig. 1(a)]. The physical meaning of the factor is to push molecules to an equilibrium distance of R_{HB} and turn the dangling H-bond arms toward each other. The Gaussian factor is defined as

$$\begin{aligned} G_{ij}^{kl} &= g\left(\frac{r_{ij} - R_{\mathrm{HB}}}{\sigma_{\mathrm{HB}}}\right) g\left(\frac{\hat{\mathbf{u}}_i^k \cdot \hat{\mathbf{r}}_{ij} - 1}{\sigma_\theta}\right) g\left(\frac{\hat{\mathbf{u}}_j^l \cdot \hat{\mathbf{r}}_{ji} - 1}{\sigma_\theta}\right) \\ &= g_{ij}^{\mathrm{HB}} g_{ijk}^{\theta} g_{jil}^{\theta} \end{aligned} \tag{4}$$

$$g(x) = \exp\left(-\frac{1}{2}x^2\right). \tag{5}$$

Please see Table 1 for definitions. The dihedral factor is a function of the dihedral angles between dangling bonds m and n of MB molecules i and j, ϕ_{ij}^{nm} [see Fig. 1(b)]. This term is included to prevent the molecules from spinning freely around the $\mathbf{r}_{ij}$ axis.

$$\begin{aligned} \Phi_{ij}^{kl} &= 1 + \frac{\varepsilon_\phi}{2} \sum_{\substack{m\neq k\\n\neq l}} \left(1 + \cos 3\phi_{ij}^{mn}\right) \\ &= 1 + \frac{\varepsilon_\phi}{2} \sum_{\substack{m\neq k\\n\neq l}} \left(1 + 4\cos^3 \phi_{ij}^{mn} - 3\cos \phi_{ij}^{mn}\right) \end{aligned} \tag{6}$$

$$\begin{aligned} \cos \phi_{ij}^{mn} &= \frac{{}^{\mathbf{r}_{ij}}\mathbf{u}_i^m \cdot {}^{\mathbf{r}_{ij}}\mathbf{u}_j^n}{|{}^{\mathbf{r}_{ij}}\mathbf{u}_i^m||{}^{\mathbf{r}_{ij}}\mathbf{u}_j^n|} \\ &= \frac{\hat{\mathbf{u}}_i^m \cdot \hat{\mathbf{u}}_j^n + (\hat{\mathbf{u}}_i^m \cdot \hat{\mathbf{r}}_{ij})(\hat{\mathbf{u}}_j^n \cdot \hat{\mathbf{r}}_{ji})}{[(1 - (\hat{\mathbf{u}}_i^m \cdot \hat{\mathbf{r}}_{ij})^2)(1 - (\hat{\mathbf{u}}_j^n \cdot \hat{\mathbf{r}}_{ji})^2)]^{1/2}} \\ &= \frac{\hat{\mathbf{u}}_i^m \cdot \hat{\mathbf{u}}_j^n + c_{ij}^m c_{ji}^n}{[(1 - (c_{ij}^m)^2)(1 - (c_{ji}^n)^2)]^{1/2}} = \frac{\eta_{ij}^{mn}}{\xi_{ij}^{mn}} \end{aligned} \tag{7}$$

$${}^{\mathbf{r}}\mathbf{u} = \hat{\mathbf{u}} - (\hat{\mathbf{u}} \cdot \hat{\mathbf{r}})\hat{\mathbf{r}} \tag{8}$$

$$c_{ij}^m = \hat{\mathbf{u}}_i^m \cdot \hat{\mathbf{r}}_{ij} \tag{9}$$

$$\eta_{ij}^{mn} = \hat{\mathbf{u}}_i^m \cdot \hat{\mathbf{u}}_j^n + c_{ij}^m c_{ji}^n \tag{10}$$

$$\xi_{ij}^{mn} = \left[\left(1 - \left(c_{ij}^m\right)^2\right)\left(1 - \left(c_{ji}^n\right)^2\right)\right]^{1/2}. \tag{11}$$

Here c_{ij}^m, η_{ij}^{mn} and ξ_{ij}^{mn} are merely shorthands. Finally, the bond-order factor is given by (please see Table 1 for definitions):

$$b_i = \begin{cases} 1 & z_i \leqslant 4 \\ \left(\frac{4}{z_i}\right)^v & z_i > 4 \end{cases} \tag{12}$$

$$z_i = \sum_{q\neq i} f(r_{iq}) \tag{13}$$

$$f(r) = \begin{cases} 1 & r < R_b - D_b \\ \frac{1}{2}\left(1 - \sin \frac{\pi(r - R_b)}{2D_b}\right) & r \in S \\ 0 & r > R_b + D_b \end{cases} \tag{14}$$

where $S = [R_b - D_b, R_b + D_b]$. This factor counts the number of molecules in the immediate neighborhood of molecule i, z_i, and penalizes configurations where more than 4 neighbors are found within a radius of R_b.

The MB potential is characterized by the set of parameters $\{\varepsilon_{\mathrm{LJ}}, \varepsilon_{\mathrm{HB}}, \varepsilon_\phi, \sigma_{\mathrm{LJ}}, \sigma_{\mathrm{HB}}, \sigma_\theta, R_{\mathrm{HB}}, R_b, D_b\}$, as defined above. In addition, parameters such as cutoff radii are needed, as explained in the user manual. The values for these parameters should be defined by the user in the main input file. A set of parameters in reduced units, optimized for bulk water and ice simulations, are given in Table 1, taken from Ref. [14]. The parameters may also be inputted (please see the manual) in real units (energies in eV, lengths in Å) which affects, e.g., the numeric value of the Boltzmann constant.

Table 1
List of suggested values for the physical parameters of the model in reduced units.

Input name	Physical meaning	Symbol	Value
`elj`	Lennard-Jones energy	$\varepsilon_{\mathrm{LJ}}$	0.05
`ehb`	H-bond energy	$\varepsilon_{\mathrm{HB}}$	−1.0
`efi`	Dihedral angle energy	ε_ϕ	−0.01
`rhb`	H-bond equilibrium length	R_{HB}	1.0
`sigmalj`	Lennard-Jones sigma	σ_{LJ}	$1.04/2^{1/6}$
`sigmahb`	H-bond distance part sigma	σ_{HB}	0.1
`sigmath`	H-bond angle part sigma	σ_θ	0.08
`expbond`	Exponent of bond-order coefficient	v	0.5
`rbond`	Bonding distance of bond-order coefficient	R_b	1.3
`dbond`	Bonding cut width of bond-order coefficient	D_b	0.2

CASHEW also allows incorporation of atomic particles in the simulations in addition to the MB molecules. Interactions can be defined for any pairs of elements (types of atomic particles) or elements and MB molecules by choosing and combining from a list of pair-potentials.

2.1.2. Derivation of forces and torques

To extract the force $\mathbf{F}_\alpha$ acting on the MB molecule α, one simply differentiates the potential energy (1) with respect to the position of the molecule—this derivative is denoted here by ∇_α. For instance, differentiating the distance between molecules i and j yields

$$\nabla_\alpha r_{ij} = (\delta_{j\alpha} - \delta_{i\alpha})\hat{\mathbf{r}}_{ij} = \gamma_{ij\alpha}\hat{\mathbf{r}}_{ij}. \tag{15}$$

In Eq. (15) we define another shorthand, $\gamma_{ij\alpha} = \delta_{j\alpha} - \delta_{i\alpha}$, where δ is the Kronecker delta. This factor is +1 if $j = \alpha, i \neq \alpha$, −1 if $i = \alpha$, $j \neq \alpha$, and 0 otherwise. The physical interpretation of the factor is simple: r_{ij} only changes if one moves either particle i or j. We will also need the derivative of the dot product

$$\begin{aligned} \nabla_\alpha c_{ij}^k &= \nabla_\alpha\left(\hat{\mathbf{u}}_i^k \cdot \hat{\mathbf{r}}_{ij}\right) \\ &= \left(\nabla_\alpha \frac{1}{r_{ij}}\right)\left(\hat{\mathbf{u}}_i^k \cdot \mathbf{r}_{ij}\right) + \left(\frac{1}{r_{ij}}\right)\nabla_\alpha\left(\hat{\mathbf{u}}_i^k \cdot \mathbf{r}_{ij}\right) \\ &= -\frac{\gamma_{ij\alpha}}{r_{ij}}\left(\hat{\mathbf{u}}_i^k \cdot \mathbf{r}_{ij}\right)\hat{\mathbf{r}}_{ij} + \frac{\gamma_{ij\alpha}}{r_{ij}}\hat{\mathbf{u}}_i^k \\ &= \frac{\gamma_{ij\alpha}}{r_{ij}} {}^{\mathbf{r}_{ij}}\mathbf{u}_i^k. \end{aligned} \tag{16}$$

Note that $\nabla_\alpha \hat{\mathbf{u}}_i^k = 0$ since we are differentiating with respect to the position of the particle, not its orientation. Using these results, the force acting on molecule α can be calculated

$$\mathbf{F}_\alpha = -\nabla_\alpha U = -\frac{1}{2}\sum_{\substack{i\\j\neq i}} \nabla_\alpha U_{ij}^{\mathrm{LJ}} + \nabla_\alpha U_{ij}^{\mathrm{HB}} = \mathbf{F}_\alpha^{\mathrm{LJ}} + \mathbf{F}_\alpha^{\mathrm{HB}} \tag{17}$$

$$\nabla_\alpha U_{ij}^{\mathrm{LJ}} = 4\varepsilon_{\mathrm{LJ}}\left(6\sigma_{\mathrm{LJ}}^6 r_{ij}^{-7} - 12\sigma_{\mathrm{LJ}}^{12} r_{ij}^{-13}\right)\gamma_{ij\alpha}\hat{\mathbf{r}}_{ij} \tag{18}$$

$$\begin{aligned} \nabla_\alpha U_{ij}^{\mathrm{HB}} = \varepsilon_{\mathrm{HB}} \sum_{k,l=1}^{4} \Big(&(\nabla_\alpha b_i) G_{ij}^{kl} \Phi_{ij}^{kl} + b_i \left(\nabla_\alpha G_{ij}^{kl}\right) \Phi_{ij}^{kl} \\ &+ b_i G_{ij}^{kl} \left(\nabla_\alpha \Phi_{ij}^{kl}\right)\Big) \end{aligned} \tag{19}$$

$$\nabla_\alpha G_{ij}^{kl} = \left(\nabla_\alpha g_{ij}^{\mathrm{HB}}\right) g_{ijk}^{\theta} g_{jil}^{\theta} + g_{ij}^{\mathrm{HB}}\left(\left(\nabla_\alpha g_{ijk}^{\theta}\right) g_{jil}^{\theta} + g_{ijk}^{\theta}\left(\nabla_\alpha g_{jil}^{\theta}\right)\right) \tag{20}$$

$$\nabla_\alpha g_{ij}^{\mathrm{HB}} = -\frac{r_{ij} - R_{\mathrm{HB}}}{\sigma_{\mathrm{HB}}^2} g_{ij}^{\mathrm{HB}} \gamma_{ij\alpha} \hat{\mathbf{r}}_{ij} \tag{21}$$

$$\nabla_\alpha g_{ijk}^{\theta} = -\frac{\hat{\mathbf{u}}_i^k \cdot \hat{\mathbf{r}}_{ij} - 1}{\sigma_\theta^2} g_{ijk}^{\theta} \gamma_{ij\alpha} \frac{{}^{\mathbf{r}_{ij}}\mathbf{u}_i^k}{r_{ij}} \tag{22}$$

$$\nabla_\alpha \Phi_{ij}^{kl} = \frac{\varepsilon_\phi}{2} \sum_{\substack{m\neq k\\ n\neq l}} \left(12\cos^2\phi_{ij}^{mn} - 3\right) \nabla_\alpha \cos\phi_{ij}^{mn} \tag{23}$$

$$\nabla_\alpha \cos\phi_{ij}^{mn} = \frac{\xi_{ij}^{mn} \nabla_\alpha \eta_{ij}^{mn} - \eta_{ij}^{mn} \nabla_\alpha \xi_{ij}^{mn}}{(\xi_{ij}^{mn})^2} \tag{24}$$

$$\nabla_\alpha \eta_{ij}^{mn} = \frac{\gamma_{ij\alpha}}{r_{ij}} \left[c_{ji}^n \, {}^{\mathbf{r}_{ij}}\mathbf{u}_i^m - c_{ij}^m \, {}^{\mathbf{r}_{ij}}\mathbf{u}_j^n\right] \tag{25}$$

$$\nabla_\alpha \xi_{ij}^{mn} = \frac{\gamma_{ij\alpha}}{\xi_{ij}^{mn} r_{ij}} \left[c_{ij}^m \left(1 - (c_{ji}^n)^2\right) {}^{\mathbf{r}_{ij}}\mathbf{u}_i^m - c_{ji}^n \left(1 - (c_{ij}^m)^2\right) {}^{\mathbf{r}_{ij}}\mathbf{u}_j^n\right] \tag{26}$$

$$\nabla_\alpha b_i = \begin{cases} 0 & z_i \leqslant 4 \\ -4^\nu \nu z_i^{-\nu-1} \nabla_\alpha z_i & z_i > 4 \end{cases} \tag{27}$$

$$\nabla_\alpha z_i = \begin{cases} \sum_{q\neq\alpha} \frac{\pi}{4D_b} \cos\frac{\pi(r_{\alpha q} - R_b)}{2D_b} \hat{\mathbf{r}}_{\alpha q} \chi_{r_{\alpha q}\in S} & i = \alpha \\ \frac{\pi}{4D_b} \cos\frac{\pi(r_{\alpha i} - R_b)}{2D_b} \hat{\mathbf{r}}_{\alpha i} \chi_{r_{\alpha i}\in S} & i \neq \alpha. \end{cases} \tag{28}$$

Above the notation $\chi_{x\in A}$ denotes the characteristic function which is 1 when $x \in A$ and 0 otherwise. Collecting all the terms we get [29]

$$\mathbf{F}_\alpha^{\mathrm{LJ}} = 2\varepsilon_{\mathrm{LJ}} \sum_{j\neq\alpha} \left(6\sigma_{\mathrm{LJ}}^6 r_{\alpha j}^{-7} - 12\sigma_{\mathrm{LJ}}^{12} r_{\alpha j}^{-13}\right) \hat{\mathbf{r}}_{\alpha j} \tag{29}$$

$$\mathbf{F}_\alpha^{\mathrm{HB}} = \mathbf{F}_\alpha^{\mathrm{pair}} + \mathbf{F}_\alpha^{\mathrm{many}} \tag{30}$$

$$\begin{aligned} \mathbf{F}_\alpha^{\mathrm{pair}} = {} & -\frac{\varepsilon_{\mathrm{HB}}}{2} \sum_{j\neq\alpha} (b_\alpha + b_j) \sum_{k,l=1}^{4} G_{\alpha j}^{kl} \Bigg[\Phi_{\alpha j}^{kl} \Bigg(-\frac{r_{\alpha j} - R_{\mathrm{HB}}}{\sigma_{\mathrm{HB}}^2} \hat{\mathbf{r}}_{\alpha j} \\ & + \frac{1}{\sigma_\theta^2 r_{\alpha j}} \left[\left(c_{\alpha j}^k - 1\right) {}^{\mathbf{r}_{\alpha j}}\mathbf{u}_\alpha^k + \left(c_{j\alpha}^l - 1\right) {}^{\mathbf{r}_{\alpha j}}\mathbf{u}_j^l\right]\Bigg) \\ & - \frac{\varepsilon_\phi}{2} \sum_{\substack{m\neq k\\ n\neq l}} \frac{(12\cos^2\phi_{\alpha j}^{mn} - 3)}{r_{\alpha j} (\xi_{\alpha j}^{mn})^3} \Big(\Big[c_{j\alpha}^n \left(\xi_{\alpha j}^{mn}\right)^2 \\ & + c_{\alpha j}^m \left(1 - \left(c_{j\alpha}^n\right)^2\right) \eta_{\alpha j}^{mn}\Big] {}^{\mathbf{r}_{\alpha j}}\mathbf{u}_\alpha^m - \Big[c_{\alpha j}^m \left(\xi_{\alpha j}^{mn}\right)^2 \\ & + c_{j\alpha}^n \left(1 - \left(c_{\alpha j}^m\right)^2\right) \eta_{\alpha j}^{mn}\Big] {}^{\mathbf{r}_{\alpha j}}\mathbf{u}_j^n\Big)\Bigg] \end{aligned} \tag{31}$$

$$\begin{aligned} \mathbf{F}_\alpha^{\mathrm{many}} = {} & 4^{\nu-1} \nu \frac{\pi \varepsilon_{\mathrm{HB}}}{2D_b} \sum_{q\neq\alpha} \cos\frac{\pi(r_{\alpha q} - R_b)}{2D_b} \chi_{r_{\alpha q}\in S} \hat{\mathbf{r}}_{\alpha q} \\ & \times \Bigg[z_\alpha^{-\nu-1} \chi_{z_\alpha > 4} \sum_{j\neq\alpha} \sum_{k,l=1}^{4} G_{\alpha j}^{kl} \Phi_{\alpha j}^{kl} \\ & + z_q^{-\nu-1} \chi_{z_q > 4} \sum_{j\neq q} \sum_{k,l=1}^{4} G_{qj}^{kl} \Phi_{qj}^{kl}\Bigg]. \end{aligned} \tag{32}$$

Differentiating the bond-order term b_i generates many-body forces. This happens because the movement of a third molecule q may affect the interaction between molecules i and j by changing the number of molecules in their surroundings. Because of this, simulating a solid (or a gas), where practically no many-body terms appear, is computationally less demanding than simulating a liquid.

Similarly to the forces, the torque $\mathbf{T}_\alpha$ acting on molecule α is obtained by differentiating the potential with respect to rotation of the particle—we denote this derivative by ∇_α^φ. For instance, in a rotation around the y axis, the z component of the unit vector $\hat{\mathbf{u}}$, u_z, changes at a rate

$$\frac{\partial u_z}{\partial \varphi_y} = u_x. \tag{33}$$

In general, the φ_η derivative of the ν component of the vector is

$$\frac{\partial u_\nu}{\partial \varphi_\eta} = \sum_\mu \varepsilon_{\eta\mu\nu} u_\mu \tag{34}$$

where $\varepsilon_{\eta\mu\nu}$ is the Levi-Civita symbol. Using this we may derive the derivative of the dot product $\hat{\mathbf{u}}_i^k \cdot \hat{\mathbf{r}}_{ij}$. Denoting the Cartesian unit vectors as $\hat{\mathbf{e}}_x$, $\hat{\mathbf{e}}_y$, $\hat{\mathbf{e}}_z$, we get

$$\begin{aligned} \nabla_\alpha^\varphi \left(\hat{\mathbf{u}}_i^k \cdot \hat{\mathbf{r}}_{ij}\right) &= \delta_{i\alpha} \sum_\eta \hat{\mathbf{e}}_\eta \frac{\partial}{\partial\varphi_\eta} \left(\hat{\mathbf{u}}_i^k \cdot \hat{\mathbf{r}}_{ij}\right) \\ &= \delta_{i\alpha} \sum_\eta \hat{\mathbf{e}}_\eta \sum_\nu \frac{\partial (\hat{\mathbf{u}}_i^k)_\nu}{\partial\varphi_\eta} \hat{\mathbf{e}}_\nu \cdot \hat{\mathbf{r}}_{ij} \\ &= \delta_{i\alpha} \sum_\eta \hat{\mathbf{e}}_\eta \sum_\nu \sum_\mu \varepsilon_{\eta\mu\nu} \left(\hat{\mathbf{u}}_i^k\right)_\mu (\hat{\mathbf{r}}_{ij})_\nu \\ &= \delta_{i\alpha} \left(\hat{\mathbf{u}}_i^k \times \hat{\mathbf{r}}_{ij}\right). \end{aligned} \tag{35}$$

Note that $\mathbf{r}_{ij}$ is constant with respect to rotations of molecules, and only the differentiation of $\hat{\mathbf{u}}_i^k$ gives non-zero derivatives. Using (35), one can calculate the torque:

$$\mathbf{T}_\alpha = -\nabla_\alpha^\varphi U = -\frac{1}{2} \sum_{\substack{i\\ j\neq i}} \nabla_\alpha^\varphi U_{ij}^{\mathrm{HB}} \tag{36}$$

$$\nabla_\alpha^\varphi U_{ij}^{\mathrm{HB}} = \varepsilon_{\mathrm{HB}} \sum_{k,l=1}^{4} b_i \left[\left(\nabla_\alpha^\varphi G_{ij}^{kl}\right) \Phi_{ij}^{kl} + G_{ij}^{kl} \left(\nabla_\alpha^\varphi \Phi_{ij}^{kl}\right)\right] \tag{37}$$

$$\nabla_\alpha^\varphi G_{ij}^{kl} = g_{ij}^{\mathrm{HB}} \left(\nabla_\alpha^\varphi g_{ijk}^{\theta}\right) g_{jil}^{\theta} + g_{ij}^{\mathrm{HB}} g_{ijk}^{\theta} \left(\nabla_\alpha^\varphi g_{jil}^{\theta}\right) \tag{38}$$

$$\nabla_\alpha^\varphi g_{ijk}^{\theta} = -\frac{\hat{\mathbf{u}}_i^k \cdot \hat{\mathbf{r}}_{ij} - 1}{\sigma_\theta^2} g_{ijk}^{\theta} \delta_{i\alpha} \left(\hat{\mathbf{u}}_i^k \times \hat{\mathbf{r}}_{ij}\right) \tag{39}$$

$$\nabla_\alpha^\varphi \Phi_{ij}^{kl} = \frac{\varepsilon_\phi}{2} \sum_{\substack{m\neq k\\ n\neq l}} \left(12\cos^2\phi_{ij}^{mn} - 3\right) \nabla_\alpha^\varphi \cos\phi_{ij}^{mn} \tag{40}$$

$$\nabla_\alpha^\varphi \cos\phi_{ij}^{mn} = \frac{\xi_{ij}^{mn} \nabla_\alpha^\varphi \eta_{ij}^{mn} - \eta_{ij}^{mn} \nabla_\alpha^\varphi \xi_{ij}^{mn}}{(\xi_{ij}^{mn})^2} \tag{41}$$

$$\begin{aligned} \nabla_\alpha^\varphi \eta_{ij}^{mn} = {} & \gamma_{ij\alpha} \left(\hat{\mathbf{u}}_i^m \times \hat{\mathbf{u}}_j^n\right) - \left[\delta_{i\alpha} c_{ji}^n \left(\hat{\mathbf{u}}_i^m \times \hat{\mathbf{r}}_{ij}\right)\right. \\ & \left. + \delta_{j\alpha} c_{ij}^m \left(\hat{\mathbf{u}}_j^n \times \hat{\mathbf{r}}_{ji}\right)\right] \end{aligned} \tag{42}$$

$$\begin{aligned} \nabla_\alpha^\varphi \xi_{ij}^{mn} = {} & -\frac{\delta_{i\alpha}}{\xi_{ij}^{mn}} \left(1 - \left(c_{ji}^n\right)^2\right) c_{ij}^m \left(\hat{\mathbf{u}}_i^m \times \hat{\mathbf{r}}_{ij}\right) \\ & - \frac{\delta_{j\alpha}}{\xi_{ij}^{mn}} \left(1 - \left(c_{ij}^m\right)^2\right) c_{ji}^n \left(\hat{\mathbf{u}}_j^n \times \hat{\mathbf{r}}_{ji}\right). \end{aligned} \tag{43}$$

Simplifying, the total torque can be written

$$\begin{aligned} \mathbf{T}_\alpha = {} & -\frac{\varepsilon_{\mathrm{HB}}}{2} \sum_{j\neq\alpha} (b_\alpha + b_j) \sum_{k,l=1}^{4} G_{\alpha j}^{kl} \Bigg[\frac{1}{\sigma_\theta^2} \Phi_{\alpha j}^{kl} \left(c_{\alpha j}^k - 1\right) \left(\hat{\mathbf{r}}_{\alpha j} \times \hat{\mathbf{u}}_\alpha^k\right) \\ & + \frac{\varepsilon_\phi}{2} \sum_{\substack{m\neq k\\ n\neq l}} \left(3 - 12\cos^2\phi_{\alpha j}^{mn}\right) \Bigg(\Bigg[\frac{c_{j\alpha}^n}{\xi_{\alpha j}^{mn}} + \frac{c_{\alpha j}^m}{1 - (c_{\alpha j}^m)^2} \cos\phi_{\alpha j}^{mn}\Bigg] \\ & \times \left(\hat{\mathbf{r}}_{\alpha j} \times \hat{\mathbf{u}}_\alpha^m\right) + \frac{1}{\xi_{\alpha j}^{mn}} \left(\hat{\mathbf{u}}_j^n \times \hat{\mathbf{u}}_\alpha^m\right)\Bigg)\Bigg]. \end{aligned} \tag{44}$$

It should also be mentioned that the torques are not symmetric: the interaction between two particles will in general result in different torques for the two. This is due to the asymmetric $\hat{\mathbf{u}} \cdot \hat{\mathbf{r}}$ terms.

2.2. Molecular dynamics

2.2.1. Integration of equations of motion

The positions and velocities of particles are updated using the standard velocity–Verlet algorithm

$$\mathbf{r}_i(t+\Delta t) = \mathbf{r}_i(t) + \mathbf{v}_i(t)\Delta t + \frac{1}{2m_i}\mathbf{F}_i(t)(\Delta t)^2 \tag{45}$$

$$\mathbf{v}_i(t+\Delta t) = \mathbf{v}_i(t) + \frac{1}{2m_i}\big(\mathbf{F}_i(t) + \mathbf{F}_i(t+\Delta t)\big)\Delta t \tag{46}$$

where $\mathbf{r}$ is the position, $\mathbf{v}$ is the velocity, $\mathbf{F}$ is the force, m is mass, t is time and i is the particle index. The orientations of particles are updated using similar integration where the spatial variables are replaced with corresponding angular variables such as angles of rotation, angular velocities, torques, and moments of inertia.

Instead of integrating Newtonian equations of motion, one can also use Langevin dynamics

$$m_i\frac{d^2\mathbf{r}_i}{dt^2} = \mathbf{F}_i - \gamma m_i\frac{d\mathbf{r}_i}{dt} + \mathbf{f}_i \tag{47}$$

where γ is a friction coefficient and $\mathbf{f}_i$ is a random force obeying the fluctuation–dissipation relation $\langle \mathbf{f}_i(t)\cdot\mathbf{f}_j(t')\rangle = 2\gamma k_B m_i T\delta_{ij}\delta(t-t')$ with k_B being the Boltzmann constant and T the temperature. However, this introduces a complication for the velocity update (46), since updating the velocity requires knowledge of the acceleration $\mathbf{F}_i/2m_i$ at a future time $t+\Delta t$, and the Langevin acceleration (47) depends on the velocity $\mathbf{v}_i = d\mathbf{r}_i/dt$. Solving Eq. (47) for the velocity at time $t+\Delta t$ leads to the equations

$$\mathbf{r}_i(t+\Delta t) = \mathbf{r}_i(t) + \mathbf{v}_i(t)\Delta t + \frac{1}{2m_i}\big(\mathbf{F}_i(t) - \gamma m_i\mathbf{v}_i(t) + \mathbf{f}_i\big)(\Delta t)^2 \tag{48}$$

$$\mathbf{v}_i(t+\Delta t) = \frac{1}{1+\frac{1}{2}\gamma\Delta t}\left[\left(1-\frac{1}{2}\gamma\Delta t\right)\mathbf{v}_i(t) + \frac{1}{2m_i}\big(\mathbf{F}_i(t)+\mathbf{F}_i(t+\Delta t)\big)\Delta t + \frac{1}{m_i}\mathbf{f}_i\Delta t\right]. \tag{49}$$

Following the standard velocity Verlet updating scheme, the velocity update (49) is further split in two within the program: the update step $\mathbf{v}_i(t) \to \mathbf{v}_i(t+\frac{1}{2}\Delta t)$ is done first using the force, velocity (friction) and random component at time t, after which the step $\mathbf{v}_i(t+\frac{1}{2}\Delta t) \to \mathbf{v}_i(t+\Delta t)$ is carried out using the force at time $t+\Delta t$ and the scaling factor $(1+\frac{1}{2}\gamma\Delta t)^{-1}$ in Eq. (49) to account for the friction.

2.2.2. Neighbor tracking

As shown in Section 2.1, the forces and torques resulting from the MB potential are rather involved. On the other hand, the potential is quite localized and needs to be evaluated for nearby molecules only, allowing the use of short interaction cutoffs. For efficient pair finding, CASHEW uses both Verlet neighbor lists [30, 31] and cell lists [32].

In addition to a cutoff distance specifying the range of interactions, a neighbor list cutoff radius is used for determining the size of the neighborhood in which neighbor pairs are searched and listed. That is, the program analyzes the intermolecular distances between molecules and creates lists of all molecules which are closer than the neighbor cutoff from each other. Since this cutoff must be larger than the interaction cutoff, all interacting pairs must always be found in this list. Finding interacting neighbors from a predefined list accelerates the summation of pairwise forces to be an $\mathcal{O}(n)$ operation instead of $\mathcal{O}(n^2)$ like a search over all molecules would be. The program then automatically estimates how often the list has to be updated in order to guarantee that all pairs of molecules which are within the interaction cutoff actually are in the neighbor list. The longer the neighbor cutoff radius (with respect to the interaction cutoff), the longer the update interval. On the other hand, the neighbor search itself is slower for a larger cutoff.

In addition, updating the neighbor list by evaluating all the pairs in the system would also be an $\mathcal{O}(n^2)$ operation. To accelerate this, the system is split into smaller subcells in order to first approximately locate particles in the system (an $\mathcal{O}(n)$ operation) and then search for neighbors in the same and adjacent subcells (an $\mathcal{O}(n)$ operation). If the subcells are made to be the size of the neighbor list cutoff radius, all pairs within the cutoff distance will be found in the same or adjacent subcells.

2.3. Quaternion representation of molecular orientations

MB molecules are in principle point-like particles to which four vectors are attached. In CASHEW, they are treated as rigid objects with both a position and an orientation. Defining a 3D position is done simply using vectors. Orientations, however, are represented by quaternions.

An orientation can be defined using a reference configuration and a rotation in 3D space starting from the reference. In CASHEW, the reference orientation for MB molecules is defined so that the H-bond vectors become

$$\hat{\mathbf{u}}^1 = [0,0,1] \tag{50}$$

$$\hat{\mathbf{u}}^2 = [2\sqrt{2}/3, 0, -1/3] \tag{51}$$

$$\hat{\mathbf{u}}^3 = [-\sqrt{2}/3, \sqrt{2/3}, -1/3] \tag{52}$$

$$\hat{\mathbf{u}}^4 = [-\sqrt{2}/3, -\sqrt{2/3}, -1/3]. \tag{53}$$

Rotations from this reference can be characterized, for instance, by a unit vector $[x,y,z]$ (with $x^2+y^2+z^2=1$), defining the axis of rotation, and an angle of rotation ϕ around this axis. This can be summarized in a four-component unit quaternion [33]

$$\mathbf{q} = \cos\phi/2 + x\sin\phi/2i + y\sin\phi/2j + z\sin\phi/2k. \tag{54}$$

It turns out [34] that applying standard quaternion multiplication rules

$$i^2 = j^2 = k^2 = ijk = -1 \tag{55}$$

to rotation quaternions of the form (54), combined rotations are obtained simply as quaternion products $\mathbf{q}_{\mathrm{tot}} = \mathbf{q}_2\mathbf{q}_1$. Handling rotations using quaternions is computationally faster than using matrices, and the method is not susceptible to gimbal lock in the way schemes based on geometric angles are [34].

In the program, orientations are represented by quaternions both internally and externally. In other words, the input and output format for molecular orientations is that specified by Eq. (54). As a rule of thumb, the quaternion $1 = [1,0,0,0]$ is the 'no rotation' orientation, i.e., the reference orientation given by (50)–(53), and the other basis quaternions $i = [0,1,0,0]$, etc. represent rotations by 180° from the reference around the x, y and z axes.

3. Program structure

3.1. Source files

The program is divided to one main program file and supporting modules. The full list of files is

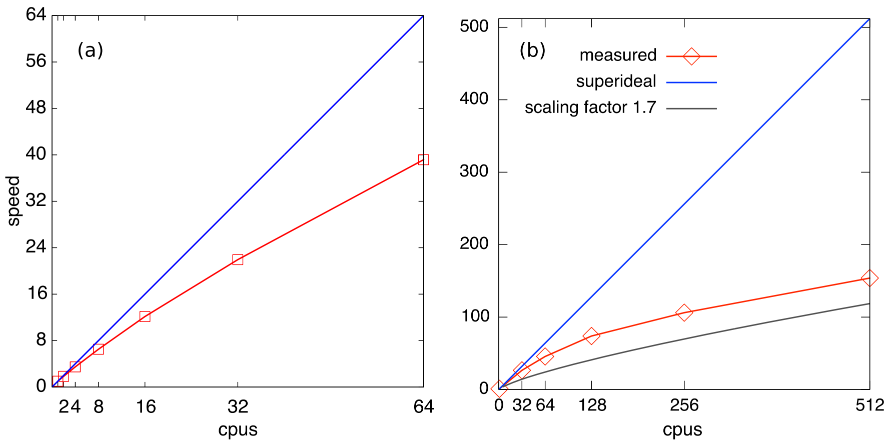

Fig. 2. Parallel scaling for a system of (a) 9720 and (b) 34 200 MB molecules ran for 1000 and 500 time steps, respectively, with little output (red line) vs. the superideal case (doubling the number of processors increases speed by a factor of 2 – blue line). The vertical axis shows the computational speed with respect to a calculation in serial mode. For systems of this size and range, the scaling is adequate: doubling the number of cpus reduces the wall clock time by a factor of more than 1.7 (gray line). According to these tests, one can use about $n_{\text{particles}}/100$ cpus (the number of processors can be any sufficiently small positive integer, not necessarily a power of 2). (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

- `Cashew.F90` – main program, includes MD routines (`PROGRAM cashew`);
- `IO.F90` – module for handling input and output (`MODULE file_handler`);
- `Model.F90` – module defining the physics of the MB model (`MODULE mb_model`);
- `Functions.F90` – module for some commonly used functions (`MODULE functions`);
- `Quaternions.F90` – module for quaternion and vector algebra, representing 3D rotations (`MODULE quaternions`);
- `Parallel.F90` – module for MPI-related variables and functions (`MODULE mpi_mod`);
- `Params.F90` – module for parameters and constants (`MODULE parameters`);
- `Mersenne.F90` – module for random numbers (`MODULE mt95`).

The files contain original source code written by the authors, except the last one, which defines the random number generator. We have used the "Mersenne Twister" random number generator written in Fortran by José Rui Faustino de Sousa and available as open source code (see Program summary). It may be replaced by another random number generator. The generator is invoked by other parts of the program as a module named "`mt95`", and the functions called are named "`genrand_init`" (initialization), "`genrand_real1`" (random real in [0, 1]) and "`genrand_real2`" (random real in [0, 1[).

A makefile is provided for compiling the program. There is a known issue in compiling the input handling module "`IO.F90`" with aggressive compiler optimization in some Fortran compilers, and therefore it is recommended to use only low level compiler optimization for this module.

3.2. Input files

The program is executed from the command line by supplying the name of the simulation as a parameter, e.g., `cashew system` (if the compiled executable is named cashew). The name of the main input file read by the program is formed by attaching the suffix "`.mb`" to the name of the simulation, e.g., "`system.mb`".

The program reads this input file containing all necessary information for setting up and running the simulation. The file should in general be in the directory from which you launch the program. The input is not case sensitive, but it must follow a certain predefined, though flexible, structure. The input must be divided in blocks. A block starts with a tag such as `<particles>` and ends with a tag such as `</particles>`. Anything outside such tags is ignored. The names of the blocks recognized in the input file are

- `<control>` – control parameters, required;
- `<mb-model>` – parameters of the MB model, required;
- `<cell>` – supercell parameters, optional;
- `<elements>` – names and masses of particles other than MB molecules, optional;
- `<particles>` – types and starting positions of particles, required;
- `<velocities>` – initial velocities of particles, optional;
- `<constraints>` – constraints, optional;
- `<potentials>` – atom–atom and MB–atom interactions, optional;
- `<statistics>` – statistics to be recorded, optional.

The detailed syntax of the input is explained in the user manual.

3.3. Output files

The main output file is marked by the suffix `.out` and it contains a summary of input parameters and run progress. Other output files are generated on demand. For instance, the output of detailed statistics can be invoked by including the `<statistics>` block in the input file. The contents of the output files and the interface for enabling their output is explained in the user manual.

3.4. Parallelization and optimization

CASHEW has been written with emphasis on well structured and readable code instead of optimal performance. As is typical in MD programs, most computational effort is spent on calculating the forces and torques acting on the particles. This is done within the routine `calc_forces` in the file `Model.F90`.

The source code includes both a serial and a parallel version, of which the latter is obtained using conditional compiling with the compiler option `-Dmpi`. All the cpu intensive tasks have been MPI-parallelized, and of these the most important is the calculation of forces. The force calculation task is split among processors by distributing molecules. To obtain better load balancing among the cpus, the loads are actively monitored and redistributed where necessary. This is likely not the optimal way to parallelize the problem—for instance distribution of the task by a spatial decomposition is usually a more efficient scheme [35]. Still, according to our tests as shown in Fig. 2, the current simple scheme allows one to use up to $n/100$ processors, where n is the number of particles in the simulation, while maintaining adequate scaling of parallel efficiency. Thus, the code is production-run ready.

In addition to CPU parallelization, the MB model could benefit from GPU acceleration. As the model allows for local computations, it is amenable to GPU implementation as minimal amount of communication with the CPU is required. We are currently investigating an Open CL implementation of the code.

4. Summary and outlook

We present an MD program for simulations of aqueous systems implementing the 3D MB model [16]. This coarse-grained model, and code, reproduce many of the known water anomalies remarkably well [16]. The force field related to the model is derived and the structure of the program is briefly discussed. The program is distributed with a user manual and a tutorial which should allow the user to easily set up and run a simulation using the MB model. The freely available source code together with the provided documentation should enable code developers a straightforward path to implementing the MB model in other programs, if desired. Future versions of CASHEW are planned to include tools to simulate biomolecules in aqueous environment, e.g., proteins.

Acknowledgements

This work has been supported in part by the Academy of Finland through its COMP CoE grant, NSERC of Canada, SharcNet (http://sharcnet.ca), and the Ontario Early Researcher Award. The computational resources of CSC—IT Center for Science, Finland, are acknowledged.

References

[1] M. Chaplin, Nature Rev. Mol. Cell Biol. 7 (2006) 861–866.
[2] P. Ball, ChemPhysChem 9 (2008) 2677–2685.
[3] A. Baranyai, A. Bartok, A. Chialvo, J. Mol. Liq. 134 (2007) 94–98.
[4] I. Brovchenko, A. Oleinikova, ChemPhysChem 9 (2008) 2660–2675.
[5] F. Mallamace, Proc. Natl. Acad. Sci. USA 106 (2009) 15097.
[6] R.A. Lovett, A. Ben-Naim, J. Chem. Phys. 51 (1969) 3108.
[7] W.L. Jorgensen, J. Chandrasekhar, J.D. Madura, R.W. Impey, M.L. Klein, J. Chem. Phys. 79 (1983) 926.
[8] C. Vega, J.L.F. Abascal, M.M. Conde, J.L. Aragones, Faraday Discuss. 141 (2009) 251.
[9] P. Mark, L. Nilsson, J. Phys. Chem. A 105 (2001) 9954–9960.
[10] A. Lyubartsev, M. Karttunen, I. Vattulainen, A. Laaksonen, Soft Mater. 1 (2002) 121–137.
[11] H. Wang, C. Junghans, K. Kremer, Eur. Phys. J. E 28 (2009) 221–229.
[12] A. Lyubartsev, A. Mirzoev, L.J. Chen, A. Laaksonen, Faraday Discuss. 144 (2010) 43–56.
[13] S.O. Yesylevskyy, L.V. Schafer, D. Sengupta, S.J. Marrink, PLoS Comput. Biol. 6 (2010) e1000810.
[14] C.L. Dias, et al., J. Chem. Phys. 131 (2009) 054505.
[15] T. Hynninen, et al., Phys. Rev. Lett. 105 (2010) 086102.
[16] C.L. Dias, et al., J. Chem. Phys. 134 (2011) 065106.
[17] A. Ben-Naim, Water and Aqueous Solutions, Plenum, New York, 1974.
[18] A. Ben-Naim, J. Chem. Phys. 54 (1971) 3682.
[19] K.A. Silverstein, et al., J. Am. Chem. Soc. 120 (2007) 3166.
[20] K.A. Dill, T.M. Truskett, V. Vlachy, B. Hribar-Lee, Ann. Rev. Biophys. Biomol. Str. 34 (2005) 173–199.
[21] C.L. Dias, et al., Phys. Rev. Lett. 100 (2008) 118101.
[22] A. Bizjak, T. Urbic, V. Vlachy, K.A. Dill, Acta Chim. Sloven. 54 (2007) 532–537.
[23] K.A. Dill, Biochemistry 29 (1990) 7133.
[24] J.L. MacCallum, M.S. Moghaddam, H.S. Chan, D.P. Tieleman, Proc. Natl. Acad. Sci. USA 104 (2007) 6206–6210.
[25] A.Y. Shih, A. Arkhipov, P.L. Freddolino, S.G. Sligar, K. Schulten, J. Phys. Chem. B 111 (2007) 11095–11104.
[26] T. Vuorela, A. Catte, P.S. Niemelä, A. Hall, M.T. Hyvönen, S.-J. Marrink, M. Karttunen, I. Vattulainen, PLoS Comput. Biol. 6 (2010) e1000964.
[27] T. Sumi, C. Suzuki, H. Sekino, J. Chem. Phys. 131 (2009) 161103.
[28] S.J. Marrink, H.J. Risselada, S. Yefimov, D.P. Tieleman, A.H. de Vries, J. Phys. Chem. B 111 (2007) 7812–7824.
[29] This form of $\mathbf{F}_\alpha^{\text{many}}$ is nicely symmetric, but computationally it is better to avoid recalculating $G_{ij}^{kl}\Phi_{ij}^{kl}$'s and rearrange the sums so that the innermost sum is over the cosines (index q), which is what CASHEW does.
[30] A.A. Chialvo, P.G. Debenedetti, Comput. Phys. Comm. 60 (1990) 215.
[31] L. Verlet, Phys. Rev. 159 (1967) 98.
[32] M.P. Allen, D.J. Tildesley, Computer Simulation of Liquids, Clarendon Press, Oxford, 1987.
[33] A rotation in 3D space has three degrees of freedom. A unit vector with four components also has three degrees of freedom since normalization to unity sets a restriction on the components.
[34] S.L. Altmann, Rotations, Quaternions, and Double Groups, Clarendon Press, Oxford, 1986.
[35] S. Plimpton, J. Comput. Phys. 117 (1995) 1.