Available online at www.sciencedirect.com
ScienceDirect
Computer Physics Communications 179 (2008) 320–329
Computer Physics Communications
www.elsevier.com/locate/cpc

Accurate and efficient methods for modeling colloidal mixtures in an explicit solvent using molecular dynamics

Pieter J. in 't Veld [1], Steven J. Plimpton *, Gary S. Grest

Sandia National Laboratories, Albuquerque, NM 87185, USA

Received 6 December 2007; accepted 8 March 2008

Available online 19 March 2008

Abstract

Most simulations of colloidal suspensions treat the solvent implicitly or as a continuum. However as particle size decreases to the nanometer scale, this approximation fails and one needs to treat the solvent explicitly. Due to the large number of smaller solvent particles, such simulations are computationally challenging. Additionally, as the ratio of nanoparticle size to solvent size increases, commonly-used molecular dynamics algorithms for neighbor finding and parallel communication become inefficient. Here we present modified algorithms that enable fast single processor performance and reasonable parallel scalability for mixtures with a wide range of particle size ratios. The methods developed are applicable for any system with widely varying force distance cutoffs, independent of particle sizes and independent of the interaction potential. As a demonstration of the new algorithm's effectiveness, we present results for the pair correlation function and diffusion constant for mixtures where colloidal particles interact via integrated potentials. In these systems, with nanoparticles 20 times larger than the surrounding solvent particles, our parallel molecular dynamics code runs more than 100 times faster using the new algorithms.

PACS: 61.20.Ja; 61.46.-w; 61.25.H-

Keywords: Molecular dynamics; Colloid mixtures; Neighbor algorithm

1. Introduction

The properties of colloidal suspensions have been well studied both experimentally and theoretically for many years [1]. The simplest such system is the near-hard sphere colloidal liquid in which the interaction between colloids can be neglected except for their excluded volume interaction. Theoretically this is the most basic system to model since the colloidal particles can be treated as hard spheres. Experimentally such systems are obtained by adding a short chain surfactant coating to the colloidal particles to avoid flocculation and keep the colloidal particles dispersed. For typical colloidal particles in the micron size range this coating increases the radii of the particles by only a few percent and is usually ignored in analyzing their properties. However as the size of the colloidal particles decrease to the nanoscale, the details of the interaction and the fact that the solvent molecules are comparable in size to the nanoparticle play an important role in the properties of the suspension.

Computer simulation methods have been widely used to determine both the structure and dynamical properties of colloidal suspensions. The shear viscosity η and diffusion constant D as a function of packing fraction of the particles have been widely studied since they provide a useful test of theoretical modeling and more specifically the role of long range, hydrodynamic interactions [2]. Due to limitations of computational resources most previous simulations of colloids have modeled the solvent implicitly. The most common approach, which neglects hydrodynamics interactions, has been to simply treat the colloids as Brownian particles coupled to a heat bath [3–5]. Stokesian dynamics [6] and related methods [7,8] have been successfully used to include hydrodynamic interactions with an implicit solvent. This is, at present, the most elegant way of treating the hydrodynamics of spherical particles. However it is also a computationally expensive technique, although advances have been

* Corresponding author.
E-mail address: sjplimp@sandia.gov (S.J. Plimpton).
[1] Current address: BASF Aktiengesellschaft, Ludwigshafen, Germany.

0010-4655/$ – see front matter © 2008 Published by Elsevier B.V.
doi:10.1016/j.cpc.2008.03.005

recently made to achieve greater efficiency and flexibility in terms of boundary conditions [9,10]. Currently, these methods are limited to spherical or mildly aspherical particles. Finite element approaches to modeling the solvent as a continuum surrounding spherical or irregularly shaped particles have also been proposed [20] which potentially allow for great generality, but are also typically limited in use due to the computational cost.

Alternatively, one can treat the background solvent in a coarse-grained particle fashion. One such successful approach for particle-fluid suspensions is the lattice Boltzmann method [11]. Another is to treat the background solvent as dissipative particles (DPD) [12–15] or stochastic rotation dynamics (SRD) particles [16–19]. Each of these particle-based methods introduces an effective coarse-graining length scale that is smaller than the colloids but much larger than the natural length scales of a microscopic solvent. These approaches thus rely on the fact that the colloidal particle is much larger than the solvent and there is therefore a clear separation of time and length scales. For nanoparticles in the 2–20 nm size, treating the background solvent as a continuum or via coarse-grained particles may not always be adequate. For example, it does not account for local packing of the solvent around the nanoparticles, which can increase the effective radii of the nanoparticles and strongly affect the properties of the suspension. Thus it is important to be able to model nanoparticle suspensions with an explicit solvent to address even such simple questions as (1) how large should the nanoparticles be to treat the solvent as a continuum or (2) how strongly do changes in the nanoparticle/nanoparticle or nanoparticle/solvent interactions affect rheology.

As noted above, the most prevalent model of colloidal particles is to treat them as hard spheres. Thus the first approach one might consider for modeling colloidal particles in an explicit solvent is to treat all interactions (colloid/colloid, colloid/solvent and solvent/solvent) as hard-sphere like. However hard sphere mixtures strongly phase separate even for relatively small differences in size [21], which makes this approach unsuitable. What is missing is an attractive component of the interaction between the colloidal or nanoparticle and the solvent, which is critical to enabling solvation of the particles. This can be done in a variety of ways. One approach, which is particularly useful for aspherical nanoparticles, is to treat each nanoparticle as a collection of atoms [22–26]. Interactions between a pair of particles are the sum of all the pairwise interactions between their constituent atoms. This has the advantage that it is straightforward to construct nanoparticles of varying shape [26], but can quickly become computationally intractable as even small nanoparticles can contain hundreds to thousands of atoms.

A more computationally efficient approach is to treat the colloidal particles as large particles interacting with an effective potential which depends only on the distance between their centers. The simplest such effective potential is a Lennard–Jones (LJ) interaction shifted to the surface of the colloidal particle [27–29]. However this potential does not capture the true interaction between particles, since the range over which the interactions are important relative to thermal fluctuations becomes increasingly small with increasing particle size. A more realistic approach is to treat each colloidal particle as composed of a uniform distribution of atoms, similar to treating them as a collection of atoms, except that, since the atoms are uniformly distributed, an effective pair potential can be derived analytically by integrating over the colloidal particle volume [30,31]. This can be done for both colloid/colloid and colloid/solvent interactions. The interaction between two colloidal particles now requires a single pairwise calculation, though the ability to freely choose particle shapes is typically restricted to symmetrical shapes such as spheres or ellipsoids. In this paper we focus on spherical particles, though the algorithms presented can be more generally applied. Also since the colloidal particles are in the range of 5 to 20 times the size of the solvent, we refer to them in the remainder of the paper as nanoparticles.

Even using integrated potentials for interactions between nanoparticles, a number of computational issues must be addressed before large-scale simulations can be performed in a fast, efficient manner. Many molecular dynamics codes, such as our parallel simulator LAMMPS [32], are not efficient for systems with particles of widely disparate size or widely disparate force distance cutoffs. This is primarily because neighbor searches and inter-processor communications are typically based on the largest cutoff in the system. For atomic systems in which force distance cutoffs are (nearly) the same for all atom types this is not an issue. However if a nanoparticle/nanoparticle cutoff is many times larger than a solvent/solvent cutoff, the need to efficiently find neighbors and communicate the minimal amount of needed information to neighboring processors becomes critical. The importance of these issues increases with the disparity in particle size or cutoff lengths.

In the next section we describe a prototypical integrated interaction potential for nanoparticles in an explicit solvent. In Section 3 we discuss new neighbor-list construction and communication algorithms suitable for such systems. In Section 4 we highlight performance data to illustrate the effectiveness of the new algorithms in our LAMMPS molecular dynamics code. In Section 5, pair correlation function and diffusion constant D data for systems with nanoparticle/solvent size ratios of 10:1 and 20:1 are presented. In Section 6 we briefly summarize our results and discuss additional potential applications of the new algorithms.

2. Potentials

Potential discussions start with elementary Lennard–Jones (solvent) pair interactions, which are described by

$$U_{\mathrm{LJ}}(r_{12}) = 4\epsilon_{\mathrm{ss}}\left[\left(\frac{\sigma_s}{r_{12}}\right)^{12} - \left(\frac{\sigma_s}{r_{12}}\right)^{6}\right], \tag{1}$$

where ϵ_{ss} and σ_s are the Lennard–Jones (LJ) parameters for energy and distance and $r_{12} = |\mathbf{r}_1 - \mathbf{r}_2|$ is the center-to-center distance between two solvent atoms. For efficiency, a cutoff radius r_c is typically used, beyond which the interaction is zero. The interaction between a LJ solvent atom and an integrated nanoparticle consisting of LJ particles of size σ_n and number

density ρ_1 can be obtained by integrating such a particle over a sphere of radius a_1 to yield

$$U_{\rm ns}(r_{12}) = \frac{2a_1^3\sigma_{\rm ns}^3 A_{\rm ns}}{9(a_1^2 - r_{12}^2)^3} \times \left[1 - \frac{(5a_1^6 + 45a_1^4 r_{12}^2 + 63a_1^2 r_{12}^4 + 15r_{12}^6)\sigma_{\rm ns}^6}{15(r_{12} - a_1)^6(r_{12} + a_1)^6}\right], \tag{2}$$

with $A_{\rm ns} = 24\pi\rho_1\sigma_{\rm ns}^3\epsilon_{\rm ns}$ and $\sigma_{\rm ns} = (\sigma_s + \sigma_n)/2$. Here $\epsilon_{\rm ns}$ is the interaction between a solvent atom and an atom in the nanoparticle. Note that the mixed potential diverges when r_{12} approaches a_1.

The interaction between two nanoparticles is derived by integrating the remaining LJ particle over a sphere of radius a_2 [30], to give

$$\begin{aligned} U_A(r_{12}) &= -\frac{A_{\rm nn}}{6} \times \left[\frac{2a_1a_2}{r_{12}^2 - (a_1 + a_2)^2} + \frac{2a_1a_2}{r_{12}^2 - (a_1 - a_2)^2} + \ln\left(\frac{r_{12}^2 - (a_1 + a_2)^2}{r_{12}^2 - (a_1 - a_2)^2}\right)\right], \\ U_R(r_{12}) &= \frac{A_{\rm nn}}{37800}\frac{\sigma_n^6}{r_{12}} \times \left[\frac{r_{12}^2 - 7r_{12}(a_1 + a_2) + 6(a_1^2 + 7a_1a_2 + a_2^2)}{(r_{12} - a_1 - a_2)^7} + \frac{r_{12}^2 + 7r_{12}(a_1 + a_2) + 6(a_1^2 + 7a_1a_2 + a_2^2)}{(r_{12} + a_1 + a_2)^7} \right. \\ &\quad \left. - \frac{r_{12}^2 + 7r_{12}(a_1 - a_2) + 6(a_1^2 - 7a_1a_2 + a_2^2)}{(r_{12} + a_1 - a_2)^7} - \frac{r_{12}^2 - 7r_{12}(a_1 - a_2) + 6(a_1^2 - 7a_1a_2 + a_2^2)}{(r_{12} - a_1 + a_2)^7}\right], \\ U_{\rm nn}(r_{12}) &= U_A(r_{12}) + U_R(r_{12}), \end{aligned} \tag{3}$$

where $A_{\rm nn} = 4\pi^2\rho_1^2\sigma_n^6\epsilon_{\rm nn}$ is the Hamaker constant. The attractive term $U_A(r_{12})$ is the standard attractive interaction between colloidal particles, first derived by Hamaker [35]. The nanoparticle radius a_2 has the same meaning as the previously defined nanoparticle radius a_1, thus creating an infinite potential when interparticle distance r_{12} goes to $a_1 + a_2$. Note that unlike most interaction potentials, $U_{\rm nn}(r_{12})$ and $U_{\rm ns}(r_{12})$ depend both on the size of the atoms making up the nanoparticle σ_n and on the radii of the nanoparticles themselves, namely a_1 and a_2. To reduce the number of parameters we now set $\sigma_n = \sigma_s = \sigma$ for the remainder of the paper.

Depending on the value of the Hamaker constant, nanoparticles will either disperse in the solvent or aggregate. For nanoparticles made of the same LJ atoms as the solvent ($\epsilon_{\rm nn} = \epsilon_{\rm ns} = \epsilon_{\rm ss}$ and $\sigma_n = \sigma_s$) at density $\rho_1\sigma_n^3 = 1.0$, $A_{\rm nn} = 4\pi^2\epsilon_{\rm ss}$ and $A_{\rm ns} = 24\pi\epsilon_{\rm ss}$. As shown by Grest et al. [36] this corresponds to a very strongly interacting system which results in a colloidal gel. However, by separately varying Hamaker constants $A_{\rm nn}$ and $A_{\rm ns}$ it is possible to control the relative strength of the nanoparticle/nanoparticle and nanoparticle/solvent interaction and to study the interface of the various interactions. For example, in most nanoparticle suspensions, the nanoparticles are coated with short surfactants to avoid flocculation. This can be modeled by reducing $A_{\rm nn}$.

In the following sections we consider two model systems, one with $A_{\rm nn} = 4\pi^2\epsilon_{\rm ss}$, $A_{\rm ns} = 24\pi\epsilon_{\rm ss}$ and the other with $A_{\rm nn} = \epsilon_{\rm ss}$, $A_{\rm ns} = 12\epsilon_{\rm ss}$. The former corresponds to a colloidal gel while the latter corresponds to a hard-sphere like colloidal suspension. By hard-sphere like, we mean the nanoparticle/nanoparticle interactions are weak but the nanoparticle/solvent interactions are strong enough so that the nanoparticles remain in solution. A strictly hard-sphere system with only repulsive interactions (truncated at the minimum of the potential energy) is expected to phase separate as the size disparity in nano and solvent particles increases. Note that including the attractive tail of the nanoparticle potential requires a very long cutoff on the solvent length scale. Consider the usual motivation for a LJ cutoff at $\sim 2.5\sigma$, where the radial distribution function $g(r)$ approaches unity. Applied to nanoparticle/nanoparticle interactions for particles of size 10σ, this requires a cutoff of $\sim 25\sigma$. This wide disparity in cutoff distances motivates the need for new neighbor-finding and inter-processor communication algorithms, which we discuss next.

3. Computational issues

The algorithms described here are those implemented in our parallel MD code LAMMPS [32], but the basic ideas are common to many MD codes. For example, neighbor lists and some form of binning are often used to efficiently find pairwise interactions. A Verlet neighbor list [37] stores all atom pairs separated by a distance less than a neighbor cutoff $\Delta^N = \Delta^F + s$ where Δ^F is the force cutoff and s is a skin distance. The list is typically stored on a per-atom basis, with each atom i storing a list of its neighbors, and the i, j pair stored only once, either by atom i or j. Constructed once every few time steps, the list can be scanned to quickly find pairs separated by a distance $r_{ij} < \Delta^F$ on the current time step, and energies and forces computed for those pairs.

The list is valid until the separation of some atom pair shrinks by the skin distance at which point the list is rebuilt. Thus a safe, albeit conservative, criterion for rebuilding the list is when any atom has moved half the skin distance. The larger the skin distance, the more timesteps the list can be used, but at the cost of storing more pairs whose distance exceeds Δ^F. For simulations of LJ liquids with a force cutoff of 2.5σ, a skin distance of 0.3σ is typical, and the Verlet list is recomputed every 10–20 time steps.

An efficient algorithm for constructing a Verlet list is to sort atoms spatially to avoid examining all pairs [38]. A regular grid with an integer number of grid cells or bins in each dimension overlays the simulation domain, as in Fig. 1 for a 2D example. When the neighbor list is constructed, each atom is first assigned to a bin, so that atoms within a particular bin can be looped over. The Verlet list for atom i can then be built, by only checking distances to atoms in bins near atom i's bin.

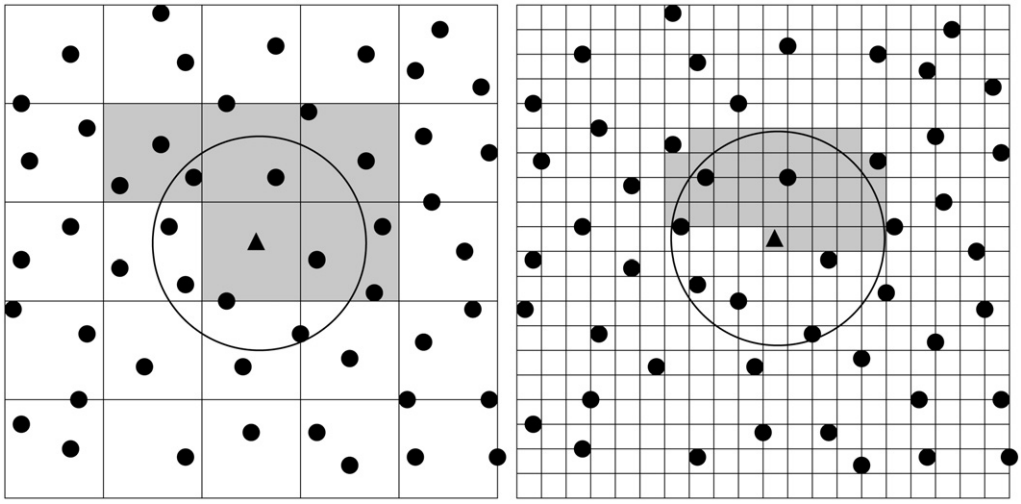

Fig. 1. Binning atoms on a grid for efficient neighbor finding. The bins on the left are the size of the neighbor cutoff distance, shown as a circle centered on the triangular particle. The bins on the right are roughly 1/4 the cutoff size.

Which bins must be included depends on the bin size. A typical choice is to make each bin edge length b_x, b_y, b_z as small as possible while satisfying the integer constraint and requiring $b_x, b_y, b_z \geqslant \Delta^N$. Then, as in the left side of Fig. 1 where the circle has radius Δ^N, the triangular atom need only check 9 bins, its own and 8 neighbors. In 3D, 27 bins are checked. If each i, j pair is stored only once (either with atom i or j), only half the surrounding bins need to be checked by each atom, which are the 5 shaded bins in 2D or 14 in 3D.

Smaller bins can be used, resulting in a larger number of bins to check for each atom. The right side of Fig. 1 shades the relevant bins for a bin size of approximately $\Delta^N/4$. Now, bins near the corners of the square of side-length $2\Delta^N$ can be ignored, since they are entirely outside the cutoff. In the limit of tiny bins, only the optimal spherical volume of radius Δ^N surrounding an atom is checked for possible neighbors. However, checking a large number of bins for a few neighboring atoms incurs overhead. In practice, the optimal bin size is roughly $\Delta^N/2$, which is what is used in LAMMPS. To quickly identify the appropriate bins surrounding atom i, the "stencil" of surrounding bins can be pre-computed and stored as a list of array offsets, e.g., $[-4, +1]$ for the shaded bin at the lower-left corner on the right of the figure. The offset list is then looped over for each central atom. Periodic boundary conditions can be enforced as needed, if the stencil wraps around the simulation box boundary.

The preceding algorithm is efficient because atom densities are roughly uniform in MD simulations of liquids and solids. Assigning atoms to bins is an $O(N)$ operation, meaning it can be performed in a time proportional to the number of atoms. The cost for finding the neighbors of a single atom is a constant $O(1)$, although the prefactor varies with the cube of the cutoff distance. Thus the entire pairwise computation (binning + Verlet list construction + calculation of energies and forces) is $O(N)$, which is optimal for MD with short-range forces. In a typical simulation, the cost of computing pairwise forces requires 80% of the CPU time, neighbor list construction is 15%,

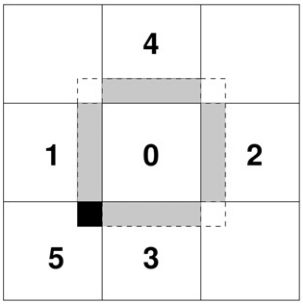

Fig. 2. Communication pattern for 2D grid of processors. Processor 0 receives messages from processors 1–4 to extend its owned sub-domain with ghost atoms within a cutoff distance away.

and various other operations (time integration, thermodynamic computations, output) are 5% of the CPU cost.

The same ideas are used in parallel MD codes such as LAMMPS, where a spatial decomposition partitions the work across processors of a distributed-memory machine [32]. If the processors are arrayed logically as a 3D grid mapped to the full simulation box, then each processor owns a small brick-shaped sub-domain of the box and the atoms within it. To compute forces and neighbor lists for the atoms it owns, each processor communicates with neighboring processors in the logical grid, and acquires coordinates of nearby "ghost" atoms owned by other processors. Once this has been done, Fig. 1 can now represent the sub-domain owned by a single processor. One or more layers of grid cells at the outer boundaries of the diagram contain ghost atoms and the inner grid cells contain owned atoms. Thus the same binning and stencil algorithms described above can be used in parallel to build a distributed Verlet list where each processor finds and stores neighbors for only its owned atoms.

The inter-processor communication required to support this spatial-decomposition is diagrammed in Fig. 2, again for a 2D example. The central processor 0 is surrounded by 4 neighbors (1–4) each of which owns one of the 9 squares of the overall simulation domain. Processor 0 needs to acquire all ghost atoms

in the expanded dotted square that surrounds it. This can be done in 4 send/receive communication stages, with processors 1–4 sending atoms in their shaded regions to processor 0. These include the corner atoms (dark small square) since processor 3 can send them to processor 0 after receiving them from processor 5 in an earlier stage. If all processors communicate in this pattern together, each acquires its needed ghost atoms simultaneously. The analogous operation in 3D requires 6 communication stages.

A thickness of Δ^N for the shaded rectangles (or slabs in 3D) is sufficient to enable a processor to build neighbor lists and compute forces for its owned atoms. Since the thickness is a constant, independent of the number of atoms, for large systems the amount of atom information communicated (and hence the communication cost in a parallel MD simulation) scales as $(N/P)^{2/3}$, where P is the number of processors. This is effectively the surface-to-volume ratio of the layer of slabs surrounding the sub-domain volume. Such scaling has enabled simulations of millions to billions of atoms on large parallel machines at parallel efficiencies of 90% or more [39,40].

More sophisticated parallel algorithms can be used which reduce the required thickness by a factor of two [41–43]. These can reduce communication cost significantly when Δ^N is large compared to the sub-domain size, e.g., when a large number of processors is used to simulate a small number of atoms. The ideas we present next are applicable to these strategies as well; here we limit the discussion to the simpler case of slabs of Δ^N thickness.

We now examine how these algorithms perform for mixtures with a large disparity in particle sizes and/or force cutoffs. Consider a system with nano and solvent particles of diameter d_1 and σ and associated force cutoffs $\Delta^F_{\rm nn} > \Delta^F_{\rm ns} > \Delta^F_{\rm ss}$ for the 3 types of pairwise interactions. If $d_1/\sigma = M$, then as discussed in the previous section, the ratio of force cutoffs $\Delta^F_{\rm nn}/\Delta^F_{\rm ss}$ is also typically M. Thus for colloidal particles in a LJ solvent, the colloid–colloid interactions are very long range on the solvent length scale. A mixture with $M = 10$ would require $\Delta^F_{\rm nn} = 25\sigma$, $\Delta^F_{\rm ns} \simeq 12\sigma$, and $\Delta^F_{\rm ss} = 2.5\sigma$.

For such a system, the Verlet list should now store neighbors based on the respective cutoffs. If atom i is a solvent particle and atom j is nanoparticle, the i, j pair should only appear in the list if $r_{ij} < \Delta^F_{\rm ns} + s = \Delta^N_{\rm ns}$, and similarly for $\Delta^N_{\rm ss}$ and $\Delta^N_{\rm nn}$, where s is again the skin distance. If this is the case, the cost for computing forces from the list is optimal, just as it was in the mono-disperse system. Fig. 3, drawn roughly to scale, illustrates how the Verlet list would be constructed using the binning algorithm described previously for a mixture of nano and solvent particles with $d_1 = 10\sigma$. As is typical in MD codes, including LAMMPS, the bin size is chosen based on the maximum cutoff for any pairwise interaction, or $\frac{1}{2}\Delta^N = 12.65\sigma$ for this example where $\Delta^F_{\rm nn} = 25\sigma$ and $s = 0.3\sigma$.

The figure shows the corresponding 5×5 stencil of neighboring bins (in 2D), half of which must be searched for each particle in the central bin to find interactions at distances up to $\Delta^N_{\rm nn}$, shown by the circle. If the particle in the central bin is a nanoparticle, distances to a huge number of solvent particles will be computed which are further away than $\Delta^N_{\rm ns}$. If the particle in the central bin is a solvent particle, distances to an even larger number of solvent particles will be computed where $r_{ij} > \Delta^N_{\rm ss}$ as well as to large particles where $r_{ij} > \Delta^N_{\rm ns}$. These searches through neighboring bins and distance computations represent wasted effort which, as we show in the next section, dominate the run-time of a simulation if the standard algorithms are used. The effect is exacerbated as M grows for larger nanoparticles or if a bin size equal to the maximum cutoff is used (25.3σ in this case).

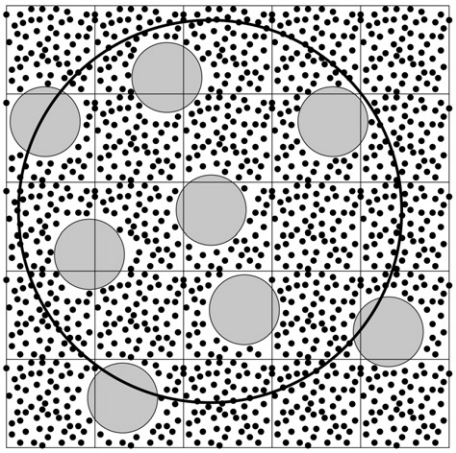

Fig. 3. Stencil of surrounding bins needed to find neighbors of nanoparticles and solvent particles when the bin size is based on the cutoff length for nanoparticle-nanoparticle interactions (circle).

Clearly, for this system, it is better to choose a bin size based on the minimum cutoff distance $\Delta^N_{\rm ss}$, since the majority of interactions are between pairs of solvent particles. However, the overhead due to stencils containing too many bins must still be addressed. This is done via the following algorithm, which has a one-time pre-computed setup phase and an execution phase, computed each time a Verlet list is built.

For the setup phase, the bin size is set to half the minimum cutoff distance, or $\frac{1}{2}\Delta^N_{\rm ss} = 1.4\sigma$ in this example. For each particle type, a stencil is pre-computed, based on this bin size, which extends to the largest cutoff of the particle type with all other types. For nanoparticles the stencil extends to $\Delta^N_{\rm nn} = 25.3\sigma$. For solvent particles the stencil extends to $\Delta^N_{\rm ns} = 12.3\sigma$. For each bin in each stencil, two distances are also pre-computed and stored. $D_{\rm min}$ and $D_{\rm max}$ are the minimum and maximum distances from any location in the central bin to any location in the stencil bin, i.e. the distance between their nearest and furthest pairs of corners.

Fig. 4, drawn roughly to scale, illustrates these stencils for a central solvent particle (left) and central nanoparticle (right). The bins in the stencils are shaded by the pre-computed distances. On the left, the inner shaded bins are those whose $D_{\rm min} < \Delta^N_{\rm ss} = 2.8\sigma$. The stencil extends to $\Delta^N_{\rm ns}$ or 12.3σ and the outer shaded bins are those whose $D_{\rm max} > \Delta^N_{\rm ns}$. Note that for clarity, bins in the figure are shaded if they are intersected by the cutoff circle centered on one point in the central bin. In actuality, the criterion is distance from any point inside the central bin, which would shade a few more bins.

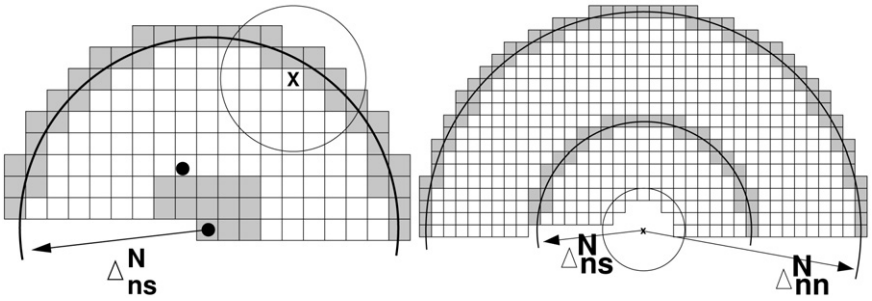

Fig. 4. Stencils of surrounding bins needed to find neighbors of solvent (left) and nanoparticles (right) when the bin size is based on the cutoff length for solvent–solvent interactions. Bin shading is discussed in the text.

On the right of the figure, the bin and particle sizes are the same as on the left, but the length scale of the diagram is expanded. The inner shaded bins are those whose $D_{\max} > \Delta^N_{\rm ns} = 12.3\sigma$ and $D_{\min} < \Delta^N_{\rm ns}$. The outer shaded bins are those whose $D_{\max} > \Delta_{\rm nn} = 25.3\sigma$. Note that, as shown on the right, if a nanoparticle particle excludes all particles from its effective volume, bins closer than some minimum distance can be excluded from the stencil altogether. For the colloidal potential described in the previous section, a minimum distance of d_1 can be used since the pairwise energy is infinite at that separation.

Now for the execution phase. All particles (nano and solvent) are sorted into the small bins. To build the Verlet list for a solvent particle, the stencil on the left of Fig. 4 is used. As the bins of the stencil are looped over, every particle within them is found. If the found particle is a solvent particle and the bin's $D_{\min} > \Delta^N_{\rm ss}$, then the particle is skipped; it's distance to the central particle need not be computed. If the found particle is a nanoparticle and the bin's $D_{\max} < \Delta^N_{\rm ns}$, the particle is added to the Verlet list. Again, it's distance to the central particle need not be computed. If neither of these criteria is met, the distance to the particle is computed and the usual test against $\Delta^N_{\rm ss}$ or $\Delta^N_{\rm ns}$ is performed. Note that while the stencil contains a large number of bins, relatively few distances between pairs of particles are actually computed. Distances to solvent particles are only computed for the inner set of shaded bins; others like the 2nd solvent particle in the figure are discarded without a distance check. Distances to nanoparticles are only computed for the outer set of shaded bins; others like the nanoparticle in the figure are included in the Verlet list without a distance check.

If the central particle is a nanoparticle, the stencil on the right side of Fig. 4 is looped over in a similar way. Distances are only computed for solvent particles in the inner shaded bins and for large nanoparticles in the outer shaded bins. Particles in the unshaded bins can be discarded or added to the Verlet list without a distance check.

This algorithm can easily be extended to mixtures with intermediate size particles and cutoffs. One stencil per particle type (size) is required. The pre-computed bin-to-bin distances effectively partition each stencil into a number of active "rings" of bins (or shells in 3D) equal to the number of particle types. Distances between two particles are only computed when the bin of a found particle is within the ring corresponding to the found particle's type.

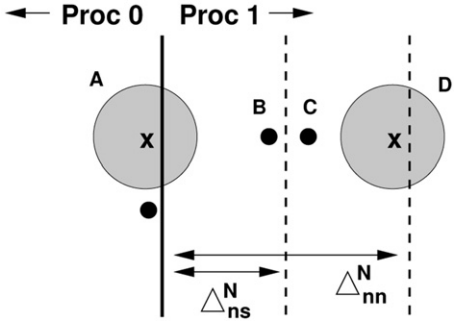

Fig. 5. Processor 1 particles needed by processor 0 to compute its neighbor lists and force interactions. Nanoparticle D within a cutoff distance $\Delta^N_{\rm nn}$ is needed as is solvent particle B, but solvent particle C beyond $\Delta^N_{\rm ns}$ is not needed, since particles A and C do not interact.

Also note that if bin sizes never change, the setup phase of the algorithm need only be computed once. However a changing simulation box volume typically induces a change in bin size (due to the integer constraint of fitting bins to the periodic box), in which case the setup must be re-computed. For example, when LAMMPS runs an NPT simulation, bin and stencil properties are re-computed each time neighbor lists are built based on the current box dimensions, which is a small amount of additional overhead.

The preceding strategy for building Verlet lists also motivates a simple enhancement to the communication algorithm described previously. As with neighbor bins, the standard implementation in MD codes, including LAMMPS, is to communicate all particles within a slab of thickness equal to the maximum neighbor cutoff Δ^N. Fig. 5 illustrates the inefficiency that results for mixtures with disparate size particles and/or cutoffs. Processor 0 receives a slab of particles from Processor 1. Nanoparticles are needed at a distance up to the largest cutoff $\Delta^N_{\rm nn}$, since the A, D pair must be added to the Verlet list. However, solvent particles such as B are only needed up to the $\Delta^N_{\rm ns}$ cutoff, since the A, C pair will not be in the Verlet list. Thus the enhanced communication algorithm uses a type-specific cutoff when deciding which particles to send to a neighboring processor. This cutoff is the maximum neighbor cutoff of a particular type (nano or solvent particle) with all other types. As illustrated in the next section, the communication savings can also be considerable, since the majority of sent particles are solvent, and now only about half as many are sent. As before, this idea can easily be extended to mixtures with intermediate size particles and cutoffs.

Table 1
CPU time in seconds for 100 time steps of MD runs of various nanoparticle systems on different numbers of processors using colloidal gel parameters (see text)

d_1/σ	N_{procs}	ϕ_v	N_n	N	Old/old	New/old	New/new
5:1	27	0.12	700	199200	17.8	6.84	5.44
					(3.59/10.2/3.75)	(3.51/1.08/1.96)	(3.16/0.926/1.16)
		0.25	1400	142736	7.62	4.04	3.12
					(1.99/3.75/1.68)	(1.91/0.569/1.34)	(1.75/0.513/0.723)
		0.39	2100	87032	3.03	2.04	1.63
					(0.805/1.10/1.02)	(0.777/0.259/0.884)	(0.731/0.243/0.575)
10:1	64	0.12	250	671623	471	19.2	11.2
					(5.91/371/93.0)	(6.23/3.41/8.46)	(5.42/2.32/3.10)
		0.25	500	558553	284	15.9	8.17
					(4.69/242/36.0)	(4.76/2.39/7.83)	(4.14/1.69/2.07)
		0.39	750	444651	165	11.3	5.63
					(3.28/138/23.4)	(3.26/1.41/5.80)	(2.85/1.09/1.46)
20:1	125	0.12	111	2331840	5630	88.6	38.6
					(13.4/4140/1470)	(16.0/19.5/47.4)	(14.9/12.9/9.97)
		0.25	222	2003131	4390	73.7	31.2
					(10.6/2960/1410)	(12.2/14.1/42.3)	(11.3/9.63/9.47)
		0.39	333	1673369	2750	56.9	22.0
					(8.13/1880/864)	(9.03/9.06/34.7)	(8.34/6.47/6.62)

d_1/σ = size ratio of nanoparticle to solvent particles, ϕ_v = volume fraction of N_n nanoparticles, N = total particles. Old/old timings are for the original neighboring and communication algorithms, new/old are for the new neighbor algorithm and old communication, new/new are new neighbor and communication timings. The parenthesized values are time spent in pairwise computation, neighbor list construction, and communication.

4. Performance

The algorithms of the preceding section were tested on systems of monodispered ($a_1 = a_2$) nanoparticle suspensions (large nanoparticles in a background LJ solvent) with different size ratios of nanoparticle to solvent particles (5:1, 10:1, 20:1) and at different volume fractions $\phi_v = \frac{\pi}{6}\sigma_b^3 N_n / V$, where N_n is the number of nanoparticles and V is the simulation box volume.

The initial state of each system was generated by randomly placing N_n nanoparticles in a box with volume V at the desired volume fraction. To remove any overlaps, dynamics was run for a short time using a soft potential. Then a standard LJ potential was used, cutoff at $2^{1/6}\sigma$ (repulsion only). Initially $d_1 < \sigma$ was used, and it was increased over time until the particles were full size with diameter d_1. Separately, a simulation of pure LJ solvent particles at a density $\rho\sigma^3 = 0.6$ was equilibrated at temperature $T = \epsilon_{\text{ss}}/k_B$ in a box the same size as the nanoparticle-only simulation. The two systems were merged by removing solvent particles that overlapped the nanoparticles. The merged system was then equilibrated in an NPT ensemble at a small positive pressure $P = 0.1\epsilon_{\text{ss}}/\sigma^3$ at $T = \epsilon_{\text{ss}}/k_B$ to ensure a homogeneous liquid suspension phase.

The timing statistics were generated by running further dynamics in an NVE ensemble using a velocity-Verlet algorithm coupled weakly to a heat bath [44,45] with damping constant $\Gamma = 0.01\tau^{-1}$. An integration time step of $\delta t = 0.005\tau$ was used, where $\tau = \sigma(m/\epsilon_{\text{ss}})^{1/2}$ and m is the mass of the solvent atoms. The Hamaker constants for nanoparticle–nanoparticle and nanoparticle–solvent interactions were set to $A_{\text{nn}} = 4\pi^2\epsilon_{\text{ss}}$, $A_{\text{ns}} = 24\pi\epsilon_{\text{ss}}$. The interactions between LJ atoms were cutoff at 3.0σ, between nanoparticles at $5.0a_1$ and between LJ solvent atoms and nanoparticles at $a_1 + 4.0\sigma$. Thus for the 20:1 system, the three cutoffs were 3.0, 50.0, and 14.0σ, respectively. The mass of the nanoparticles was set to $m_p = \frac{1}{6}\pi d_1^3 \rho_1 m$ with $\rho_1\sigma^3 = 1.0$.

Table 1 lists the time in CPU seconds for 100 time steps (averaged over longer runs) for each of the systems on varying numbers of processors. The runs were performed on a large Linux cluster (512 dual processor nodes) built with 3.4 GHz Intel EM64T processors and a Myrinet communication network with 230 Mb/sec and 9 μsec bandwidth and latency performance for point-to-point MPI message passing. A neighbor skin of 1.0σ was used which triggered neighbor list re-builds roughly every 20 time steps. Each entry in the table lists a total time and also a breakdown for the portion spent in pairwise computation, neighbor list construction, and inter-processor communication (of ghost atoms). The three sub-totals sum to less than the total due to other tasks (e.g., time integration) and load-imbalance (unequal numbers of particles per processor).

These timings evidence a dramatic reduction in neighbor list construction cost (new/old column versus old/old column) when the new algorithm is used. The payoff increases for larger nanoparticles and for smaller volume fractions. The new/old timings also show a speed-up in communication versus the old/old timings, though this is likely due to better load balance once the neighbor cost improves (the communication operation has to wait when computations are imbalanced).

The new/old timings indicate the old communication algorithm can become a bottleneck when neighbor list construction is fast. In the new/new column the new communication algorithm is turned on. Now the pair cost (energy and force computation) is the largest component in all of the simulations, though the new/new timings are not as dominated by pairwise costs as typical mono-disperse LJ simulations. The overall impact of the two new algorithms is substantial. It is a factor of nearly 2× even for the 5:1 system at high volume fractions. For the 20:1

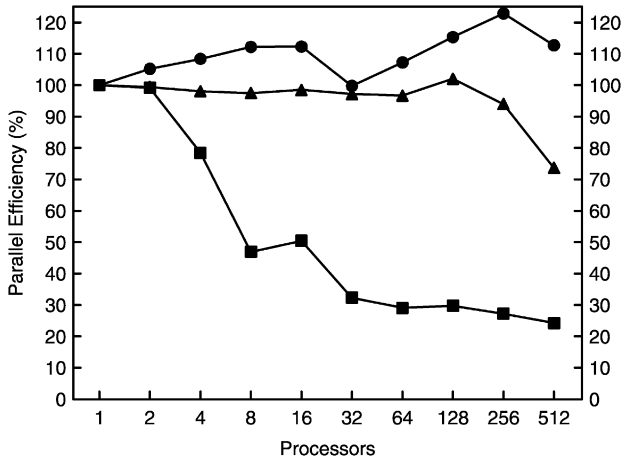

Fig. 6. Parallel scalability for a system with 5:1 nanoparticle-to-solvent size ratio (squares), 10:1 (triangles), and 20:1 (circles) using hard-sphere like parameters (see text). Efficiency is defined as the one-processor time to perform each calculation divided by the P-processor time, multiplied by 100/P.

system the overall speed-up is 125× for the 0.39 volume fraction system and 146× for the 0.12 volume fraction system.

To test parallel scalability of the new algorithms, another series of benchmarks was performed, where each system was run on varying numbers of processors. The simulation details, including pressure, temperature, and interaction cutoffs, were the same as above, except that the Hamaker constants were set to $A_{\rm nn} = \epsilon_{\rm ss}$, $A_{\rm ns} = 12\epsilon_{\rm ss}$ to model a hard-sphere like system. As before, three test cases were constructed for nanoparticle-to-solvent size ratios of 5:1, 10:1, 20:1, all at a volume fraction $\phi_v = 0.25$. The three systems had 74,801 particles (1400 nanoparticles, remainder solvent), 437,267 particles (500 nanoparticles), and 1,777,162 particles (222 nanoparticles), respectively. These runs were performed on a Cray XT3 with 2.4 GHz dual-core Opteron processors and a custom interconnect providing 2 Gb/sec and 5 μsec bandwidth and latency performance for point-to-point MPI message passing. Fig. 6 plots the parallel scalability for each system as a function of processor (core) count.

For each system the one-processor timing is plotted as 100% efficient. These were 39.0, 307, and 3140 seconds for 100 time steps of the 5:1, 10:1, and 20:1 systems, respectively. As described in the figure caption, timings on more processors are plotted such that a horizontal line for any system would represent perfect scalability. Overall, the scalability is good for all three systems. Even the 5:1 system runs roughly 100× faster on 512 processors than it does on 1 processor. For this small size ratio, declining scalability on large numbers of processors is likely due to load-imbalance. For example, on 512 processors, each processor owns 3 nanoparticles on average, but the difference in the number of owned solvent particles would be dramatic if a processor owned 1 nanoparticle versus 5.

For the systems with 10:1 and 20:1 ratios, load-imbalance may also be a factor. But the scalability is boosted by cache effects, due to running a smaller problem on each processor as the processor count increases. Thus the 20:1 system runs for 100 time steps in 5.45 seconds on 512 processors, 576 times faster than the one-processor timing.

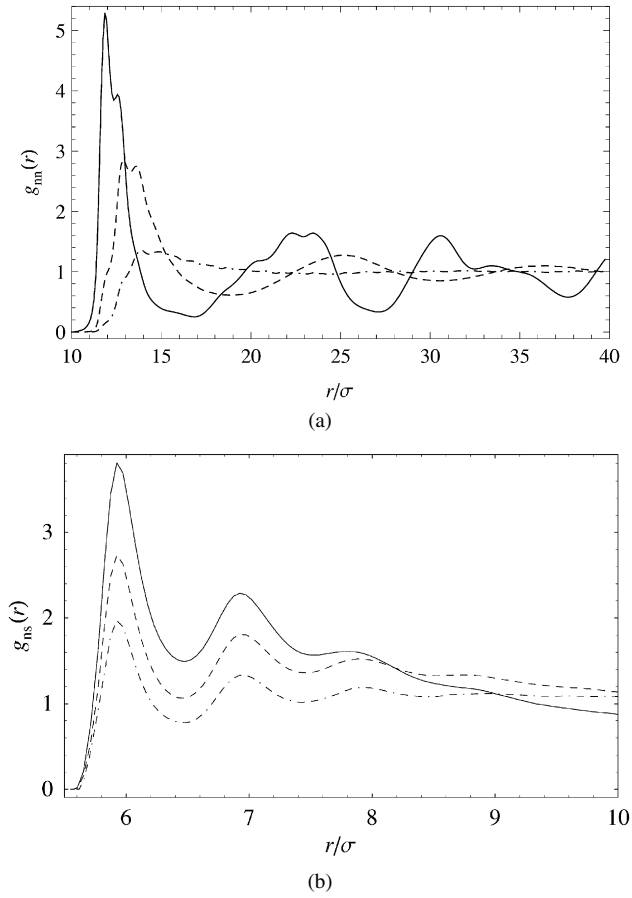

Fig. 7. Pair correlation functions of (a) nanoparticle–nanoparticle and (b) nanoparticle–solvent interactions at nanoparticle concentrations of $\phi_v = 0.39$ (solid), 0.25 (dash), and 0.08 (dash–dash–dot) with diameter $d_1 = 10\sigma$.

For comparison purposes, constant NVE runs of all-solvent LJ systems were performed on the same machine for matching system sizes: 75,000, 440,000, and 1.78 million particles, with a solvent density of $\rho\sigma^3 = 0.6$, similar to its initial value in the mixture systems. As in the mixture model, the LJ cutoff was set to 3.0σ, and a neighbor skin of 1.0σ was used. The all-solvent simulations ran for 100 timesteps on a single processor in 21.0, 122, and 493 secs. The same systems ran on 512 processors in 0.0880, 0.294, and 1.07 seconds for parallel efficiencies of 47%, 81%, and 90%, respectively.

5. Results

After equilibration at constant pressure $P = 0.1\epsilon_{\rm ss}/\sigma^3$, we carried out simulations at constant volume to determine the pair correlations and diffusion of nanoparticles in an explicit solvent. Here we present results for $A_{\rm nn} = \epsilon_{\rm ss}$ and $A_{\rm ns} = 12\epsilon_{\rm ss}$, which corresponds to weakly interacting nanoparticles. Results for more strongly interacting nanoparticles are presented elsewhere [36]. The nanoparticle–nanoparticle $g_{\rm nn}(r)$ and nanoparticle–solvent $g_{\rm ns}(r)$ pair correlations functions are shown in Fig. 7 for $d_1 = 10\sigma$ for three volume fractions ϕ_v. The pair correlation between nanoparticles is close to what one expects for soft, repulsive particles. In this case the nanopar-

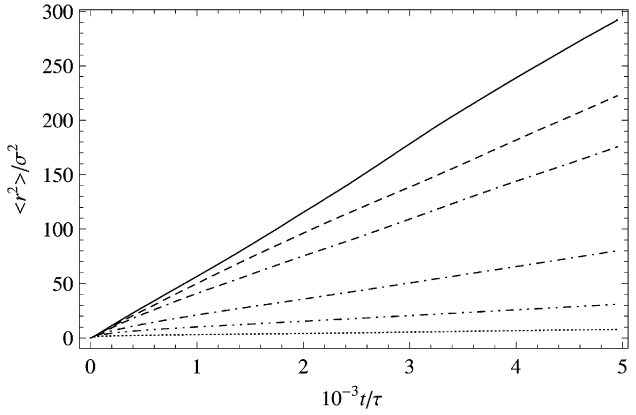

Fig. 8. Mean square displacement $\langle r^2 \rangle$ as a function of time for nanoparticles of diameter $d_1 = 10\sigma$ for volume fractions $\phi_v = 0.08$, 0.12, 0.17, 0.25, 0.31, and 0.39 (in order of top to bottom line).

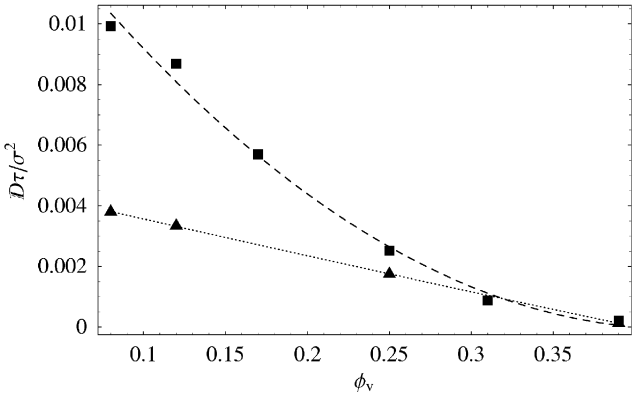

Fig. 9. Diffusion constant D versus concentration ϕ_v for nanoparticles of diameter $d_1 = 10\sigma$ (square) and 20σ (triangle). Lines are added as a guide to the eye.

ticles are very weakly interacting, essentially hard sphere-like, as the temperature is much greater than the liquid/vapor critical temperature T_c for pure nanoparticles [34]. Reducing the nanoparticle–solvent interaction further, for example to $A_{ns} = 6.0\epsilon_{ss}$, leads to phase separation, similar to what is observed for hard sphere mixtures of disparate size [21].

One of the advantages of the new algorithms is that we can follow the dynamics of the nanoparticles over long times. As an example Figs. 8 and 9 show results for the mean squared displacement as a function of time for nanoparticles of size $d_1 = 10\sigma$. From the slope of these lines, we can determine the diffusion constant $D = \langle r^2 \rangle / 6t$, which is shown in Fig. 9 for $d_1 = 10$ and 20σ. For the highest volume fractions we extend the runs to times longer than shown in Fig. 8 to obtain the diffusion constant. In the dilute limit, $\phi_v \to 0$, the ratio of the diffusion constant for the two cases approaches the expected value of roughly 2, in agreement with the Stokes–Einstein relation where diffusivity is inversely proportional to particle diameter. As the volume fraction increases, the diffusion constant for the two sizes approach each other. More detailed results, comparing the diffusion constant with the viscosity for these nanoparticle suspensions will be presented elsewhere [46].

6. Conclusions

We have described new algorithms for neighbor list construction and inter-processor communication in MD codes which are improvements over standard algorithms (both in serial and parallel) for systems in which there is a large disparity in particle size and/or force distance cutoffs. The new neighbor algorithm achieves its speed-up in two ways. First, it uses smaller bins (based on the minimum cutoff rather than the maximum) to search for nearby atoms. Second, it uses specialized stencils (list of bins) to minimize the number of particle pairs for which distances are computed. The latter is actually more critical to performance than the former, as illustrated by the following test.

We ran one of the benchmark problems from Table 1 using the old neighbor algorithm with bin sizes based on the minimum cutoff (3σ in this case) rather than the maximum cutoff. For the 10:1 system at $\phi_v = 0.39$ running on 64 processors, the neighbor list time shrank from 138 secs to 57.3 secs. But with the new algorithm the neighbor time is 1.41 secs (new/old column in table), indicating the majority of the speed-up comes from the selectivity of the stencils.

We have also presented performance and simulation results for two specific models of nanoparticles in explicit solvent, colloidal gel and hard-sphere like systems. Other hybrid models may also benefit from these algorithms. For example, for mixtures with 20:1 particle size ratios, both models used cutoffs ranging from 3σ for solvent/solvent interactions to 50σ for nanoparticle/nanoparticle interactions. As discussed at the end of Section 2, for strictly hard-sphere mixtures, the integrated potentials of Section 2 could be truncated at their minima, yielding shorter cutoffs. Alternatively, a LJ potential shifted to the nanoparticle radius could be used to model nanoparticle/nanoparticle and nanoparticle/solvent interactions. While the disparity in cutoffs would now not be as large, the algorithms discussed here would still have a significant performance impact.

The new algorithms have been implemented as options in LAMMPS, an open-source parallel MD code, available for download from the LAMMPS WWW site [33].

Acknowledgements

Sandia is a multiprogram laboratory operated by Sandia Corporation, a Lockheed Martin Company, for the United States Department of Energy under Contract No. DE-AC04-94AL85000.

References

[1] W.B. Russel, D.A. Saville, W.R. Schowalter, Colloidal Dispersions, first ed., Cambridge University Press, Cambridge, 1989.
[2] D.M. Heyes, H. Sigurgeirsson, J. Rheo. 48 (2003) 223.
[3] D.L. Ermak, J.A. McCammon, J. Chem. Phys. 69 (1978) 1352.
[4] P. Strating, Phys. Rev. E 59 (1999) 2175.
[5] D.R. Foss, J.F. Brady, J. Fluid Mech. 407 (2000) 167.
[6] J.F. Brady, G. Bossis, Annu. Rev. Fluid Mech. 20 (1988) 111.
[7] R.C. Ball, J.R. Melrose Jr., Adv. Colloid Interface Sci. 59 (1995) 19.

[8] A.A. Catherall, J.R. Melrose Jr., R.C. Ball, J. Rheo. 44 (2000) 629.
[9] A. Sierou, J.F. Brady, J. Fluid Mech. 448 (2001) 115.
[10] J.P. Hernandez-Ortiz, J.J. de Pablo, M.D. Graham, Phys. Rev. Lett. 98 (2007) 140602.
[11] A.J.C. Ladd, R. Verberg, J. Stat. Phys. 104 (2001) 1191.
[12] P.J. Hoogerbrugge, J.M.V.A. Koelman, Europhys. Lett. 199 (1992) 155.
[13] P. Espanol, Phys. Rev. E 57 (1998) 2930.
[14] W. Dzwinel, K. Boryczko, D.A. Yuen, J. Colloid Interface Sci. 258 (2003) 163.
[15] V. Pryamitsyn, V. Ganesan, J. Chem. Phys. 122 (2005) 104906.
[16] A. Malevanets, R. Kapral, J. Chem. Phys. 110 (1999) 8605.
[17] A. Malevanets, R. Kapral, J. Chem. Phys. 112 (2000) 7260.
[18] M. Hecht, J. Harting, T. Ihle, H.J. Herrmann, Phys. Rev. E 72 (2005) 011408.
[19] J.T. Padding, A.A. Louis, Phys. Rev. E 74 (2006) 031402.
[20] R. Glowinski, T.W. Pan, D.D. Joseph, J. Periaux, J. Comput. Phys. 169 (2001) 363.
[21] M. Dijkstra, R. van Roij, R. Evans, Phys. Rev. E 59 (1999) 5744.
[22] F.W. Starr, T.B. Schoder, S.C. Glotzer, Macromolecules 35 (2002) 4481.
[23] F.W. Starr, J.F. Douglas, S.C. Glotzer, J. Chem. Phys. 119 (2003) 1777.
[24] G.D. Smith, D. Bedrov, L. Li, O. Byutner, J. Chem. Phys. 117 (2002) 9478.
[25] S. Sinsawalt, K.L. Anderson, R.A. Vaia, B.L. Farmer, J. Polym. Sci. B Polym. Phys. 41 (2003) 3272.
[26] S.T. Knauert, J.F. Douglas, F.W. Starr, J. Polm. Sci. B Polym. Phys. 45 (2007) 1882.
[27] C. Powell, N. Fenwick, F. Bresme, N. Quirke, Colloids Surf. A 206 (2002) 241.
[28] D. Gersappe, Phys. Rev. Lett. 89 (2002) 058301.
[29] S.R. Challa, F. Van Swol, Phys. Rev. E 73 (2006) 016306.
[30] R. Everaers, M.R. Ejtehadi, Phys. Rev. E 67 (2003) 041710.
[31] T. Desai, P. Keblinski, S.K. Kumar, J. Chem. Phys. 122 (2005) 134910.
[32] S. Plimpton, J. Comp. Phys. 117 (1995) 1.
[33] LAMMPS molecular dynamics package WWW site: lammps.sandia.gov.
[34] M.A. Horsch, P.J. in 't Veld, J.B. Lechman, G.S. Grest, J. Chem. Phys., submitted for publication.
[35] H.C. Hamaker, Physica 4 (1937) 1058.
[36] G.S. Grest, P.J. in 't Veld, J.B. Lechman, in: 5th International Conference on Complex Systems, AIP Conf. Proceeding, 2007.
[37] L. Verlet, Phys. Rev. 159 (1967) 98.
[38] R.W. Hockney, S.P. Goel, J.W. Eastwood, J. Comp. Phys. 14 (1974) 148.
[39] R.H. Gee, N. Lacevic, L.E. Fried, Nature Materials 5 (2006) 39.
[40] K. Kadau, T.C. Germann, P.S. Lomdahl, Inter. J. Mod. Phys. C 17 (2006) 1755.
[41] S.J. Plimpton, et al., J. Parallel Distrib. Comput. 50 (1998) 104.
[42] B.G. Fitch, et al., Blue matter: Approaching the limits of concurrency for classical molecular dynamics, in: Proc. 2006 ACM/IEEE Conf. on Supercomputing SC'06, IEEE Computer Society Press, 2006.
[43] K.J. Bowers, R.O. Dror, D.E. Shaw, J. Chem. Phys. 124 (2006) 184109.
[44] G.S. Grest, K. Kremer, Phys. Rev. A 33 (1986) 3628.
[45] K. Kremer, G.S. Grest, J. Chem. Phys. 92 (1990) 5057.
[46] P.J. in 't Veld, M.K. Petersen, G.S. Grest, in preparation.