Implementation of multi-level contact detection in LAMMPS

Kevin Stratford, EPCC
Tom Shire, University of Glasgow,
Kevin Hanley, University of Edinburgh

October 2018

AN ARCHER ECSE PROJECT

Contents

1 Introduction

This report concerns the implementation of an improved contact detection mechanism in the molecular dynamics code LAMMPS [1]. The specific target is granular media simulation (also referred to as discrete element modelling, DEM), where "contact detection" may generally be taken to mean the search procedure required to identify pair-wise frictional interactions between neighbouring particles of finite diameter. Existing methods in LAMMPS [4] allow efficient contact detection where the particles all have the same size (a mono-disperse system). However, in cases where two or more particle sizes are present (bidisperse and polydisperse systems, respectively), the existing methods become increasingly time-consuming as the disparity in the size of the smallest and largest particle in the system grows.

The solution presented in this work involves the extension of the existing LAMMPS C++ class structure to allow efficient neighbour detection in cases where a wide disparity in particle sizes exists. Such systems are of widespread interest in a range of areas including suspensions and powders, and occur in common industrial materials such as concrete, where typical particle size ratios can reach at least 100:1. This report contains a short background of the relevant areas of contact detection, a description of the improved approach, and a number of benchmarks demonstrating improved performance for various systems.

2 Background

The standard approach used to prevent an $O(N^2)$ complexity when searching for neighbouring pairs in a set of N particles dates to the earliest days of molecular dynamics [2], and is variously referred to as the cell list or link cell method. This takes a three-dimensional system and partitions it into cubes of equal size. Each of the particles in the system, with position $\mathbf{x}_i$, can then be assigned to a cell based on that position. Some appropriate integer addressing mechanism may then be employed to enumerate the particles belonging only to a given cell, or cells.

Consider a system of like particles having a short-range spherically symmetric interaction cut-off distance r_c. If the width of the cells is chosen to be at least $2r_c$, then a given particle i is able to identify all possible pair-wise interactions with neighbours j by examining its own cell and the 26 immediately adjoining cells in three dimensions. This process is of complexity $O(N)$. The examination of the aggregate of 27 cells will encounter more pairs than those strictly required (having $r_{ij} < r_c$). A further refinement is then often introduced in which a subset of neighbouring particles form a neighbour list following Verlet [3]. Such a neighbour list usually introduces a cut-off criterion modified by a small skin distance s; this provides scope for particles to move a short distance before the neighbour list has to be reconstructed. In this case the cells would be chosen to have width at least $r_n = 2r_c + s$; in practice, the number of cells is usually maximised consistent with this condition.

Consider now a granular system with two types of particle having radii r_s and r_l (for small and large). This introduces three different interaction cut-off ranges: r_c^{ss}, r_c^{sl}, and r_c^{ll}. In principle, one should choose the cell width to be $r_n = 2r_c^{ll} + s$ to ensure all large–large interactions are captured. However, the resulting cell size will necessarily drag into the search very many small–large and small–small pairs well beyond their respective cut-offs. This is inefficient, and is a picture that only gets worse as the ratio r_l/r_s increases.

2.1 The LAMMPS multi approach

LAMMPS implements the standard cell list approach as outlined above; in a system having particles with different interaction cut-offs, the cell width would be chosen on the basis of the largest interaction cut-off detected. To address the resulting inefficiency, LAMMPS also offers the "multi" approach [4]. The approach relies on a feature of LAMMPS which allows atoms to be classified via a type property (an integer), which is usually assigned for each atom by the user in the initial configuration of atomic positions, velocities, and so on.

Consider again the example of a bidisperse system with two particle sizes r_s and r_l. The type property in the initial configuration should then be set to discriminate particles on the basis of radius. Instead of constructing the cell list on the basis of the largest interaction cut-off r_c^{ll}, in the multi approach the cell list is determined on the basis of the smallest present: r_c^{ss}. Each particle position $\mathbf{x}_i$ can be located in the single cell list, which can then be used as the basis for identifying interactions at the three different cut offs. This is conveniently organised using different stencils.

2.1.1 Stencils: "Newton off" and "Newton on"

A stencil is simply a set of cells in the cell list used to identify potential pairs i, j for a given interaction range. An appropriate set of stencils for the bidisperse case is illustrated in Figure 1A–C. Given a central particle i which is small, potential neighbours j may be identified via a compact [5] stencil consisting of the central cell and its immediate neighbours (shaded area in Figure 1A). For larger neighbours, a larger stencil is required to capture the appropriate interaction cut-off r_c^{sl} (Figure 1B). If the central particle is a large particle, a larger stencil is still required to ensure large–large pairs are captured (Figure 1C). In practice, the LAMMPS multi approach uses exactly one stencil per type: if the central particle is small, the intermediate stencil is used (1B) and if the central particle is large, the large stencil is used (1C).

The stencils shown in Figure 1A–C are symmetric about the central cell. Such a stencil is appropriate for what is referred to as the "Newton off" option in LAMMPS. The use of Newton's third law in the pair interaction implementation allows a single computation of the interaction to be performed. The third law $\mathbf{F}_{ij} = -\mathbf{F}_{ji}$ then allows a force to be assigned to both i and j with the correct sign. The "Newton off" option means Newton's third law is not used and interactions must be computed twice [6]; this means pairs must be stored twice (in neighbour lists of both i and j).

With the "Newton on" option, the stencil can be reduced: see Figure 1D–F. Cells are searched in a one-sided fashion, including the central cell, which allows pairs to be identified exactly once. This provides a saving in the search procedure of around a factor of two.

2.1.2 Parallelism and communication

LAMMPS implements a standard approach of domain decomposition with message passing via the message passing interface (MPI) in parallel. This means that particles are "owned" by a given MPI task dependent on their position. In order that all relevant interactions may be located, a given local domain must obtain information on particles in adjoining regions. Communication of ghost particles is performed to fulfil this requirement.

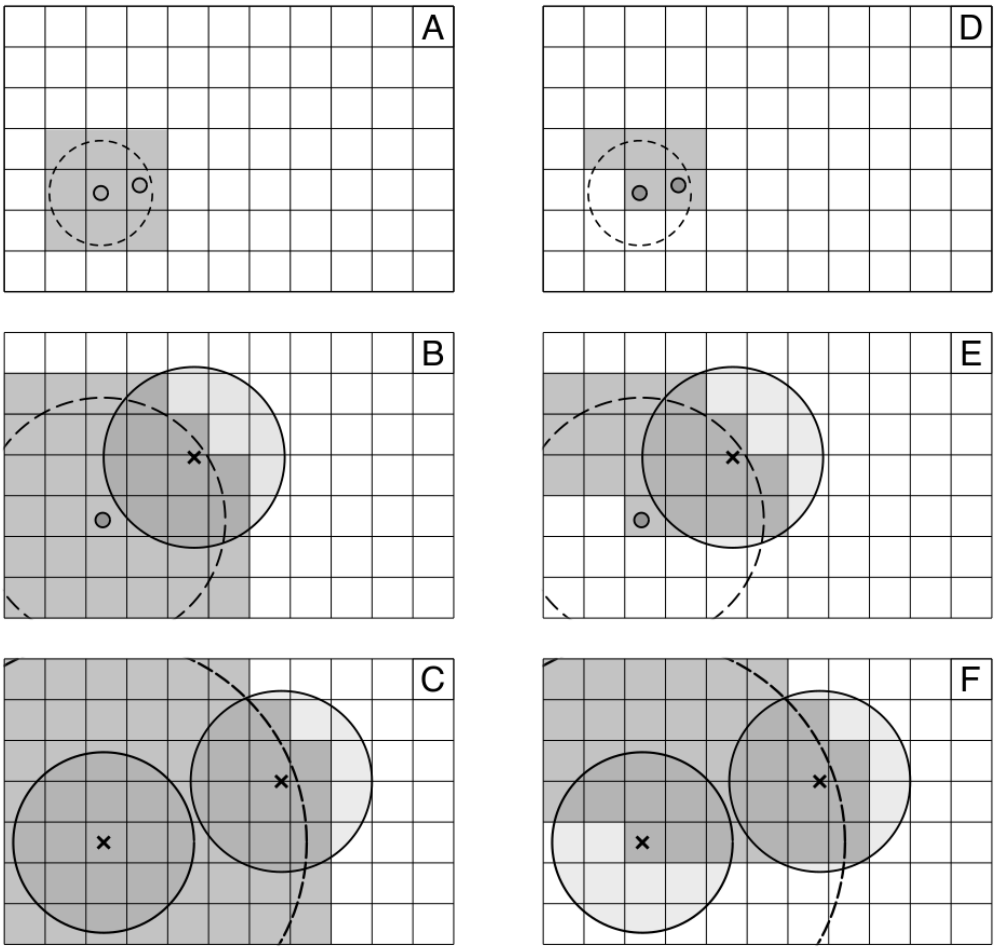

Figure 1: A two-dimensional schematic representation of stencils in the case of "Newton off" (A–C) and "Newton on" (D–F) approaches based on a small cell list (gridded lines). (A) A neighbour of the lower left particle within the relevant interaction cut-off (dashed line) may be identified by searching a compact stencil (shared area) centred on the particle's own cell . (B) Identification of small–large interactions requires an intermediate stencil, and (C) large–large interactions involve a large stencil. The analogous Newton stencils (D–F) involve only cells "on one side" (and the central cell). The generalisation to three dimensions is straightforward.

The number of particles communicated at each local domain boundary is related to the interaction cut-off distance in a way similar to that used to determine the width of the cells in the cell list. In the most simple picture, one would identify all particles within a distance r_c of the boundary as ghost particles requiring communication. In the bidisperse case, this distance would again be the largest relevant cut-off r_c^{ll}. The multi option introduces a more efficient variant of this where the ghost cut-off is r_c^{sl} for small particles, and r_c^{ll} for large particles. This is sufficient to allow identification of all potential pairs.

In parallel, the stencils extend beyond the local domain as far as necessary. Particles are assigned to cells in the usual way dependent on position. In the "Newton off" case, owned particles search for potential neighbours i, j; this means an interaction spanning a sub-domain boundary will be

recorded once by the MPI task owning i and once by the MPI task owning j. The relevant interaction can then be computed on each MPI task. In the "Newton on" case, the situation is slightly different: each interaction i, j is identified exactly once on only one MPI task and computed exactly once. In cases where i and j are owned by different MPI tasks, the resulting force must be communicated to the MPI task which has not registered the pair. This is referred to as a reverse communication in LAMMPS. The "Newton on" option therefore incurs a additional communication overhead compared with "Newton off" again of around a factor of two. There may, therefore, be some trade off between decreased computation and increased communication in moving between the different stencils.

3 A type-based approach

The fundamental difference between the new approach and those described above is to instantiate separate cell lists for each type of particle. This follows a hierarchical picture [7]. In the bidisperse case, this means one cell list with cells of width determined by the properties of the small particles alone (i.e., r_c^{ss}), and another having cells of width determined by properties of the large particles alone (r_c^{ll}). Small particles (only) are then binned in the small cell list and large particles (only) in the large cell list. Interactions between like particles may then be identified via the appropriate cell list using a compact stencil in either case (see Figure 2).

It remains to identify potential pairs i, j of different types. This is approached by locating the first particle i in the cell list of the other type j and then using an appropriate stencil in the j-type cell list to perform a search. The exact method is slightly different depending on whether a "Newton on" or a "Newton off" stencil is required. For "Newton on", a one-way search is used which identifies potential small–large pairs by considering only owned small particles, and using a symmetric stencil in the large cell list to examine large neighbours (see Figure 2B). No large-to-small search is required. This ensures all small–large pairs are located exactly once on one MPI task. For "Newton off", a large-to-small search is required for owned particles. This requires locating large particles in the small cell list and searching with a stencil appropriate for the small–large interaction limit r_c^{sl} (cf. Figure 1B).

3.1 Communication

The time spent in communication is proportional to the number of particles which have to be transferred between neighbouring processes. In the new approach, a significant efficiency saving can be made when the "Newton on" option is in operation compared with the multi approach. Potential small–small pairs may be located on the basis of r_c^{ss}. In addition, as all potential small–large pairs are located by examining owned small particles, no small ghost particles beyond r_c^{ss} are required. The ghost cut-off distance for large particles is unchanged at r_c^{ll}, and potential large–large pairs are identified as before.

The reduction in the ghost cut-off distance for small particles from r_c^{sl} to r_c^{ss} represents an increasingly large volume as R becomes larger. The number of small particles requiring communication can therefore be decreased significantly. Note there is no similar efficiency saving to be obtained in the case of "Newton off" as potential small–large pairs must be located from both owned small particles and owned large particles.

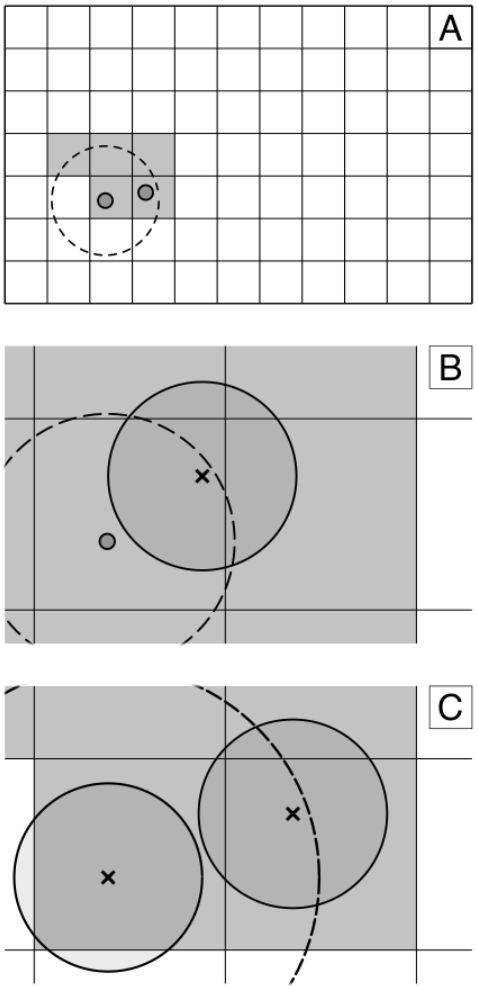

Figure 2: A schematic showing the relevant stencils in the two-level picture in the case of the "Newton on" option. (A) Interactions between two small particles (shaded circles) within the relevant cut-off distance (dashed circle) are captured via a standard Newton stencil (shaded) in the small cell list (small squares). (B) A small–large interaction is captured by a "Newton off" stencil containing all 8 neighbouring cells in two dimensions in the large cell list (large squares; not shown in their entirety). The search for small–large pairs only takes place for small particles; there is no large-to-small search. This ensures each interaction is identified exactly once on the process which owns the relevant small particle. (C) Large–large interactions are captured by a Newton stencil in the large cell list. The picture for "Newton off" is different in that a large-to-small search is required to ensure that potential pairs are identified on all MPI ranks. This uses an extended "Newton off" stencil in the small cell list cf. Figure 1B.

3.2 Polydisperse systems

At this point only bidisperse systems have been considered. However, the generalisation to a polydisperse system is, in principle, straightforward. For each type, a corresponding cell list is instantiated. For each pair of types, an appropriate stencil is generated; this may be an "empty" stencil in the large-to-small direction in the case of "Newton on", indicating no search is required. Potential pairs are then identified by examining each of the cell lists in turn.

The key question is: how to assign particles of different sizes to a discrete set of types to obtain an efficient overall procedure? Previous studies [8] suggest this is strongly dependent on the particle size distribution for the problem at hand, and no prescriptive approach is readily available to decide the question without testing. Some findings for polydisperse cases are presented in Section 4.2.

3.3 Implementation

The LAMMPS code [9] is written in C++ (the implementation is sometimes referred to as "C with classes") and provides a very wide range of functionality.

There are, broadly, three steps involved in the construction of a neighbour list in LAMMPS:

1. Allocation of a bin structure for each type, and the assignment of every particle to the appropriate bin based on the particle type and particle position $\mathbf{x}_i$;
2. Allocation of the appropriate stencils for each bin structure to allow identification of the relevant neighbours depending on the two particle types involved;
3. Construction of a neighbour list of pairs for each owned atom from the current owned and ghost particles together with a search of the bin structures based on the appropriate stencil or stencils.

These steps are embodied in three LAMMPS base classes, respectively, `NBin`, `NStencil`, and `NPair`. The existing bin and multi approaches in LAMMPS are implemented by extending the base classes in the appropriate way, and allowing the user to select the appropriate choice at run time. (Computation of the final interaction is dependent only on the final neighbour list, and occurs in a relevant implementation of the `Pair` class.) Likewise, the new approach is effected by appropriate extensions of these three base classes so that the whole approach sits naturally within the existing LAMMPS structure. A number of these classes are now described in turn. Some further implementations are required to allow particular features of LAMMPS to operate; a full list is provided in the Appendix.

3.3.1 `NBin` class

A single class `NBinBytype` extending `NBin` is provided to implement the cell lists for each type. The implementation involves a simple addition to `NBin` data structures:

```
int ** atom2bin_type;   // Per type bin assignment
int ** binhead_type;    // Per type index of first atom each bin
int ** bins_type;       // Per type index of next atom in same bin
```

These provide, respectively, a mechanism to map particles to bins, and a mechanism to provide a list of particles in a given bin.

In generating a bin structure for each type, an operational bin size is required for each type. By default, each type uses one half of the current neighbour cut-off distance for that type. This, in turn, is determined for the relevant pair interaction (for granular pair styles, this usually means the maximum diameter present for particles of that type) plus the skin distance. Enough bins are generated to span local sub-domains in parallel, along with the extra extent required to accommodate ghost atoms. The bins adapt dynamically to any change in system size.

To allow particles to be located in bins of any type, an additional virtual function is added to the `NBin` class:

```
virtual int coord2bin(double * x, int type);
```

This locates three-dimensional position $\mathbf{x}$ in the bin structure for the given type, and returns a one-dimensional bin index.

3.3.2 `NStencil` classes

Two classes extending `NStencil` are provided: `NStencilHalfBytype3dNewtoff` for the "Newton off" case and `NStencilHalf3dBytypeNewton` for the "Newton on" case. Both classes require a $T \times T$ matrix of stencils, where T is the number of types. These are provided by the data structures:

```
int ** nstencil_type;  // No. of points in [itype][jtype] stencil
int *** stencil_type;  // Stencil for [itype][jtype] interaction
```

Stencils for like types (on the diagonal in the matrix) are the same as those seen in Figure 2 (A and C). For particles of any given type, a compact stencil can be used to identify potential neighbours of the same type. For off-diagonal stencils, required for identifying potential pairs of differing types, the situation is different for "Newton off" and "Newton on".

First consider "Newton off". For a given particle i, potential larger neighbours can be identified by locating i in the larger stencil, and then using the compact stencil in the larger bin structure centred on $\mathbf{x_i}$. The cell size for the larger bin structure must be large enough to capture small–large interactions ($r_c^{sl} < r_c^{ll}$). To find potential smaller neighbours, particle i can be located in the smaller bin structure, and then make use of the small–large stencil (cf. Figure 1B). For "Newton on", the pair search is based on a hierarchical approach where owned small particles find larger neighbours, but not vice-versa. This can be achieved by locating smaller particle i in a larger bin structure and then using a symmetric stencil (cf "Newton off"). No large-to-small stencils are required (entries in the stencil matrix are only required on one side of the diagonal).

Determination of number of stencil points is related to the type–type cutoff distance for like particles (`cutneighsq`), and `cuttypesq` for unalike stencils. Stencils are reconstructed following changes in the corresponding bin structure.

3.3.3 `NPair` classes

Finally, two classes extending `NPair` are provided: `NPairHalfSizeBytypeNewtoff` and `NPairHalfSizeBytypeNewton`. These implement the neighbour search and construction of the neighbour list for each relevant atom. These additional classes generalise the multi implementations to search through the matrix of stencils. Schematically:

```
    // For each owned atom i of type itype ...
    ibin = atom2bin_type[itype][i];
    for (int ktype = 1; ktype <= atom->ntypes; ktype++) {
      if (itype == ktype) kbin = ibin;
      else                kbin = coord2bin_type(x[i], ktype);
      ns = nstencil_type[itype][ktype];
      s  = stencil_type[itype][ktype];
      for (k = 0; k < ns; k++) {
        int js = binhead_type[ktype][kbin + s[k]];
        for (j = js; j >=0; j = bins_type[ktype][j]) {
           // identify potential pairs i.j ...
```

The exact details of the pair search depend on a number of details which are similar to those of other `NPair` implementations.

3.4 Additional changes

Logic is added in the `Neighbor` class to instantiate the appropriate child classes from their corresponding factory methods depending on the user input. Logic is also provided in `Comm` to set the appropriate ghost cut-off distances in the case that the user selects the reduced communication option associated with "Newton on".

4 Results

4.1 Synthetic benchmark: the bidisperse case

In the spirit of Pieter in t'Veld *et al.* [4], a number of different systems involving different values of $R = r_l/r_s$ are now examined to compare the performance of the existing LAMMPS `multi` approach with the new `bytype` approach. In each case, a bidisperse system has been generated by running two separate LAMMPS simulations of the same system size L^3. The first is for particles r_l initialised in random positions to a solid volume fraction of 0.12. The second is for particles r_s initialised at a solid volume fraction of 0.36. Each system is run with a soft repulsive potential until no solid body overlaps remain. The final positions are then combined, eliminating any small particles overlapping the position of the large particles. This forms a single bidisperse system for each R: various parameters are detailed in Table 1 below.

Radius large	Radius small	Number large	Number small	Box L	MPI tasks	Rebuilds
10.0	1.0	250	711,657	205.9	2x3x4	6
20.0	1.0	111	2,337,678	314.15	2x3x4	8
40.0	1.0	56	9,624,306	500.14	2x3x4	8
80.0	1.0	28	42,900,051	793.92	3x3x4	8
100.0	1.0	25	66,193,461	955.61	6x4x4	7

Each of the bidisperse systems is run with `gran/hooke/history/multi` for 100 time steps. The skin distance $s = 0.5$ in all cases. All cases report the same interaction energies when run in either `multi` or `bytype` modes. Time breakdowns are recorded in each case via the LAMMPS command

```
timer     full sync
```

This allows an explicit estimate of load imbalance to be provided.

4.1.1 Comparison

Figure 3 shows the time reported by LAMMPS as a function of the cell list `binsz` in the range 1.25–10.0 for a number of different values of $R = r_l/r_s$. (Note the smallest `binsz` corresponds to the default choice in the case that neighbour style `multi` is used.) A similar pattern of results is observed in each case. First, as a function of `binsz`, a minimum is observed typically at 3–4 times the default for the `multi` approach, and at twice the default in the `bytype` method. In the `multi` case, this may be understood as moving from a regime where there are very many small cells in the stencils, each of which must be searched. As the cell size is increased, the number of cells is reduced to a point which is optimal. However, if the cell size is too large,

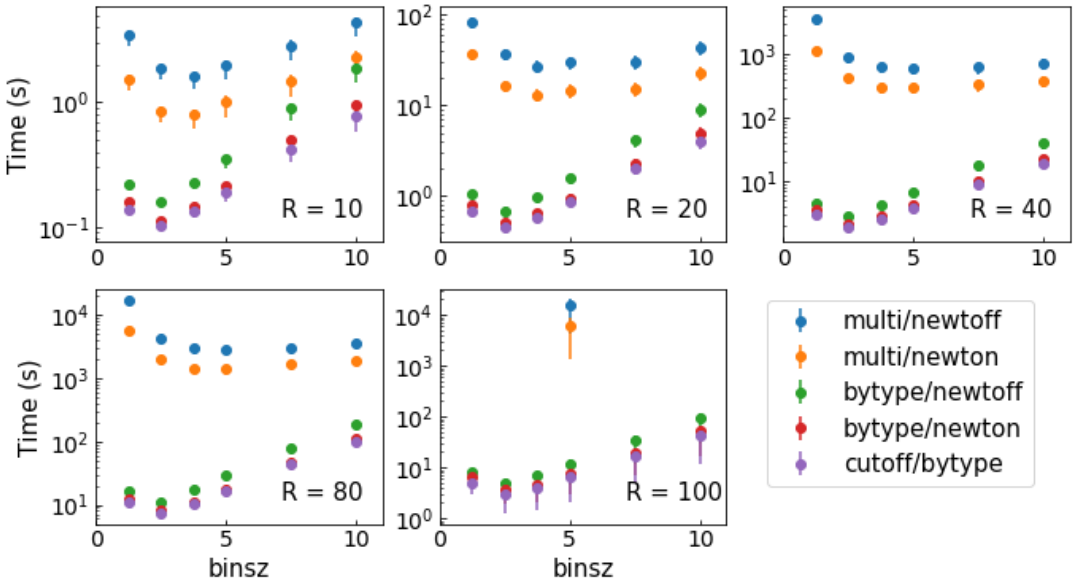

Figure 3: Comparison of pair search methods as a function of the cell list bin size. All the systems are bidisperse and the panels show different values of $R = r_l/r_s$. The markers for each panel are, top–bottom in all cases, in the same order as the legend. Times are all in seconds as reported by LAMMPS ("Neigh" in the breakdown). Figures are based on an average of four repeats ($R = 10, 20, 40$), or three repeats ($R = 80, 100$); the exception is for the multi method at $R = 100$ where only one indicative run was made. The bars show the maximum and minimum of the times as reported by LAMMPS, and so provide some measure of imbalance. Note the different vertical scales in each case.

additional (unnecessary) particles will be brought into the search, increasing the time required. In the `bytype` case, the like–like stencils are always compact, so there is no very marked decrease above 1.25. The minimum is always at twice the default, which is what might be expected if setting the cell list bin size on the basis of the interaction cut-off.

For a given value of R, Figure 3 shows the different neighbour styles appear in the same order: `multi/newtoff` is roughly twice as expensive in the search as `multi/newton` as the stencils have approximately twice as many bins. The same is true for `bytype`, although to a lesser extent as the 'cross-type' stencil is "Newton-off"; here, the time for the `newton` choice is typically 70-80% of the time for the `newtoff` option. For `bytype` combined with the reduced communication options, the reduction in the number of small ghost particles is reflected in the neighbour search time as fewer unnecessary ghost particles are involved (communication time is considered in the following paragraph). Finally, a growing disparity between the `multi` and `bytype` choices can be seen as the value of R increases. At $R = 100$, it almost becomes impractical to run the `multi` method.

From the same set of runs, Figure 4 shows a comparison of communication and imbalance times as reported by LAMMPS. (Note that $R = 100$ is not shown in the same panel as it does not use the same number of MPI tasks, and is therefore not directly comparable.) Communication time shows the difference between `multi/newton` and `multi/newtoff`: this is related to the extra cost of reverse communication in `newton`. The reduced communication option with `bytype/newton` shows the reduction in the number of small ghost particles in each case.

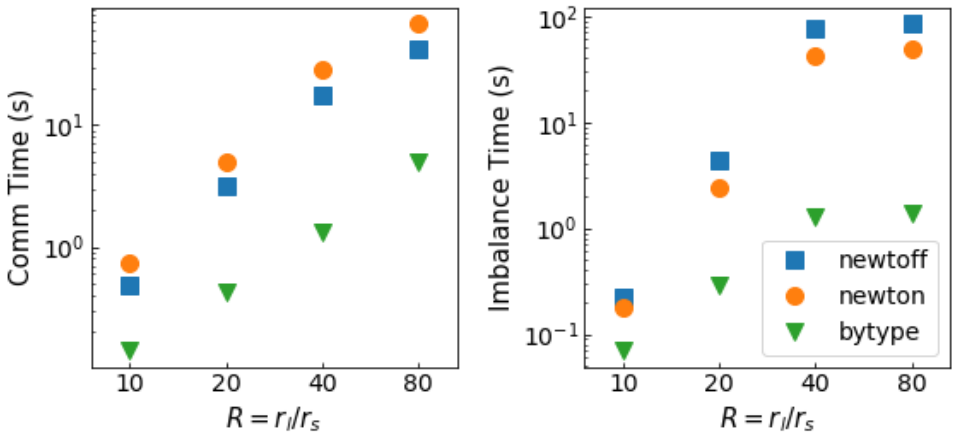

Figure 4: Communication and imbalance times as a function of particle size ratio R for the bidisperse synthetic benchmark. Times are in seconds and are as reported by LAMMPS ("Comm" and "Sync" in the breakdown). Results are shown for `multi/newtoff`, `multi/newton`, and `bytype/newton` with reduced communication. Results for communication time are entirely repeatable with the synchronised timer: statistical errors are less than the size of the markers.

The imbalance time in Figure 4 becomes increasingly adverse as R becomes larger (here at fixed sub-domain size). This cost is reduced significantly in the `bytype` picture. The exact values seen here are likely to be dependent on the exact distribution of large particles in the system.

4.1.2 Parallel scaling

An overall comparison of the `bytype` approach in terms of parallel scaling is now presented in Figure 5. Here, `multi` is absented as uncompetitive on the strength of the results of the previous section. These tests are for the same systems as those detailed in Table 1, but times are recorded without synchronisation (as would be usual in a production run). The times are the "Loop time" as reported in the LAMMPS standard output. In Figure 5 it can be seen that the reduced communication option is important to improve scaling. Broadly, the scaling is efficient to larger numbers of cores for larger problems (as would be expected).

4.2 Synthetic benchmark: polydisperse cases

4.2.1 Dunkirk sand

Many different particle size distributions may be imagined if one moves away from the bidisperse situation. This makes exhaustive examination of performance as a function of the distribution impossible in the most general case. Instead, in this section, one real example is considered as relevant to a commonly studied granular system, specifically, sand. The size distribution in question is shown in Figure 6, [11]. The size distribution used is broadly bidisperse, but contains particles over a wide range of sizes from about 2 microns to 925 microns. There is a sharp peak

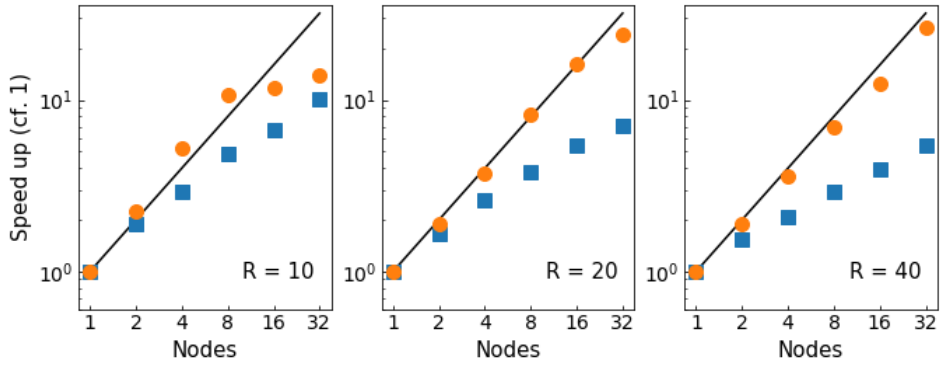

Figure 5: Parallel scaling in the bidisperse case for $R = 10, 20, 40$. Each panel shows `bytype/newton` with (circles) and without (squares) reduced communication. Each node of ARCHER has 24 cores and runs 24 MPI tasks. Ideal scaling in each case is denoted by the black line. Note the vertical scale is a log scale.

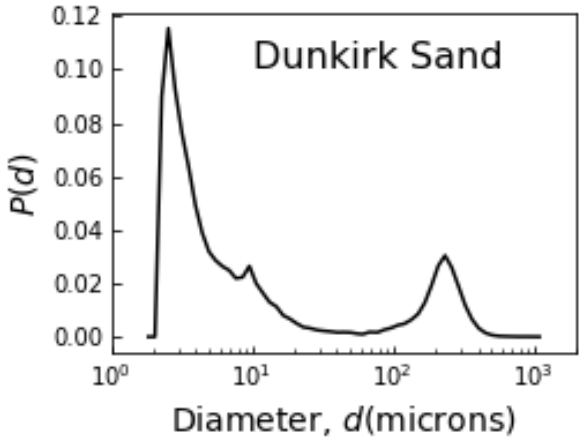

Figure 6: The particle size distribution (by number) for Dunkirk sand [11] as the basis for a synthetic polydisperse benchmark.

in the distribution at around 2.5 microns, and a broader peak at around 250–350 microns. The simulation of a system with such a wide distribution of particle sizes would be a suitable aim for an efficient hierarchical scheme.

However, here, in order that the results for the polydisperse case are broadly comparable to those of the bidisperse case appearing in the previous section, some adjustments are appropriate. The real distribution is scaled so that $R = d_l/d_s$ can be controlled to match the values in the bidisperse case. Here d_l and d_s are now the largest and smallest particle sizes in the scaled distribution. The units are also adjusted so that the smallest particle has $d_s \approx 1.0$ in Lennard-Jones units. For selected system sizes, particles are generated with random diameter from the appropriate range so that the distribution matches that seen in Figure 6. Particles are placed at random in a non-overlapping initial configuration. The following parameters were used for size distributions at $R = 10$, 20 and 40:

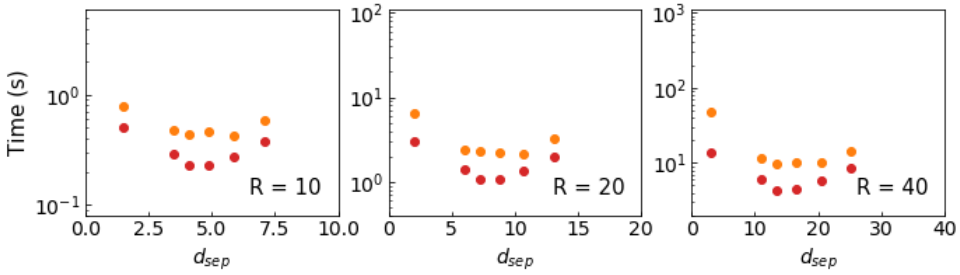

Figure 7: For synthetic polydisperse cases, time (seconds) for the relevant number of neighbour list rebuilds as a function of the separation criteria for two types d_{sep}. Runs have been performed by `multi/newton` and `bytype/newton`, the latter being the faster in all cases. The vertical scales are set to be comparable to the corresponding plot seen in Figure 3.

Radius large	Radius small	Number total	Box L	MPI tasks	Rebuilds
10.0	1.0	712,333	205.9	2x3x4	6
20.0	1.0	2,336,380	506	2x3x4	8
40.0	1.0	9,591,137	1475	2x3x4	8

For each case, exactly two types are used, but the criterion for assigning diameters to a given type is varied. The default bin sizes are adopted in all cases. Times for the neighbour list rebuild process are shown in Figure 7. It can be seen that, in each case, the best time is recorded when d_{sep} is somewhat less than $d_l/2$. This corresponds to a cut-off of a relatively small number of the largest particles in the size distribution (clearly to the right of the second peak in the distribution in Figure 6).

Figure 7 also shows that the `bytype` method is around twice as fast as the `multi` method at the optimal d_{sep}. With a little caution, the results may also be compared with those seen in the bidisperse case for the relevant value of R seen in Figure 3. Such a comparison reveals that while the `bytype` method is a little more expensive, the `multi` method is significantly more efficient. This suggests that the stencil approach deals with the smoothly varying range of diameters well; the bidisperse case is something of a worst case where many small stencils are a poor way to span to separation of scales R in the problem. (The caution is that the systems are not directly comparable in that they do not have quite the same average numbers of neighbours in each case.)

4.2.2 Exponential configuration

As a second example we take a polydisperse configuration with a continuous size distribution, consisting predominantly of small particles (by number), with a maximum size ratio $R = 50$. The smallest particle has diameter 1 micron and the underlying particle size distribution is a simple exponential one; a cummulative size distribution is shown in Figure 8.

While the distribution is of a slightly simpler form than that seen for Dunkirk sand, it is illustrative of a case with a higher density, and hence a higher average number of neighbours per particle. Exactly two types are used, but the separation criterion d_{sep} is again varied. In each case, the default bin size is used, that is, effectively $d_{sep}/2$. The results are shown in Figure 9.

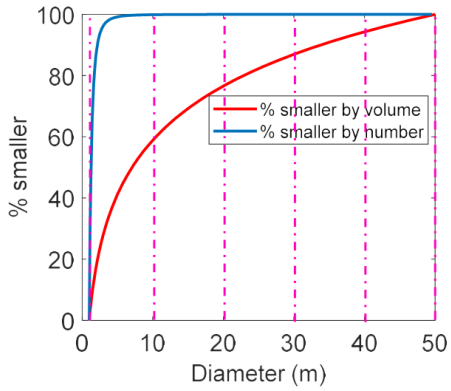

Figure 8: A cummulative particle size distribution for the simulations presented in Section 4.2.2. Particle sizes are in microns.

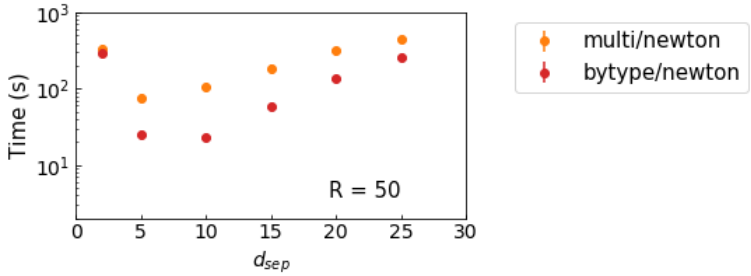

Figure 9: Synthetic polydisperse case with the particle size distribution shown in Figure 8. The time for six neighbour list rebuilds is shown (seconds) against the separation criterion for exactly two types. Benchmarks are repeated for `multi/newton` and `bytype/newton` methods. The problem has 5,040,560 particles in system size $L = 368.77$ and generates 6 neighbour list rebuilds in each case.

The `bytype` method outperforms the `multi` method by about a factor of 3 in the best case. The new method also appears slightly more forgiving in the exact choice of the criterion used to assign the two types (there is a broader minimum as a function of d_{sep}).

The two polydisperse cases show the same trend: a significant reduction in the time taken for the neighbour search if an appropriate separation criterion d_{sep} is selected to assign diameters to types. This length affects a number of things related to the bin structure. First, the cut ideally selects a vast majority of the particles in the smaller type. If this is the case, then a good bin size and compact stencil should lead to good performance. If the cut-off d_{sep} is too small, the bin size in the smaller type is likely to be too small on the scale of characteristic particle separation and hence inefficient [10]. At the same time, this will leave a significant number of particles in the larger type, which must cover a wide range in diameter. This would account for the steep penalty seen in Figure 9 below $d_{sep} = 5$. The risk of selecting a d_{sep} appears less abrupt, but is ultimately as serious. If d_{sep} is too large, the small type covers a wide range and the default bin size may be too large (compared with the position of the maximum in the distribution) for efficiency.

Clearly details may depend on the exact form of the particle distribution, and exploratory

benchmarks may be required to check acceptable performance.

There is no evidence that any increase in the number of types beyond two improves the neighbour list rebuild time for any of the cases investigated here. It is conjectured that an increased number of types might only become necessary if the particle distribution is very wide (100:1 and above, say), and flat. This might mandate more types to allow both bin size and stencil extent to be matched efficiently.

5 Summary

This work has described the implementation of a hierarchical cell list in LAMMPS. This allows bidisperse and polydisperse systems to be run with significantly improved efficiency when compared with the available methods. Parallel scaling may be improved by the use of a related reduced communication option.

Acknowledgements:

This work was funded under the embedded CSE programme of the ARCHER UK National Supercomputing Service (http://www.archer.ac.uk). We are grateful to Ishan Srivastava for sharing implementations of various classes not in the public release of LAMMPS at the time of writing, along with Jeremy Lechman and Steve Plimpton at Sandia for useful discussions.

Appendix

LAMMPS input

The `bytype` implementation is selected via the LAMMPS neighbor style, e.g.:

```
neighbor 0.5 bytype
```

where 0.5 here is the skin distance. By default, the bin size is one half the cut-off distance for the type. One can specify the bin size for the smallest type (only) explicitly via the `binsize` parameter of the `neigh_modify` command. The bin size for other types is not changed from the default.

Reduced communciation in the case of `bytype/newton` may be selected via

```
comm_modify mode multi cutoff/bytype
```

The `cutoff/bytype` must appear in conjunction with the `mode multi` option. It is stressed that the `cutoff/bytype` option would be erroneous with `bytype/newtoff`. This is not checked at run time.

These are the only commands affected by this work.

ARCHER protocol

All reported benchmarks are compiled on ARCHER with Intel programming environment:

```
 1) PrgEnv-intel/5.2.82
 2) intel/17.0.3.191
 3) cray-mpich/7.5.5
```

Compiler options are:

```
-g -O3 -restrict -DLMP_INTEL_NO_TBB
```

All benchmark runs are fully occupied nodes:

```
aprun -N 24 -n ...
```

New and updated classes

New classes (files):

```
nbin_bytype
nstencil_full_bytype_3d
nstencil_half_bytype_2d_newton
nstencil_half_bytype_3d_newtoff
nstencil_half_bytype_3d_newton
npair_full_bytype
npair_half_bytype_newton
npair_half_size_bytype_newtoff
npair_half_size_bytype_newton
```

Relevant additions have been made to:

```
nbin
nstencil
neighbour
comm_brick
```

Notes and References

[1] S.J. Plimpton, Fast Parallel Algorithms for Short-Range Molecular Dynamics, *J. Comp. Phys.* **117**, 1–19 (1995)

[2] B.J. Alder and T.E. Wainwright, Studies in Molecular Dynamics. I. General Method, *J. Chem. Phys*. **31**, 459–466 (1959).

[3] L. Verlet, Computer "Experiments" on Classical Fluids. I. Thermodynamical Properties of Lennard-Jones Molecules, *Phys. Rev.* **159**, 98–103 (1967).

[4] P.J. in t'Veld, S.J. Plimpton, G.S. Grest, Accurate and efficient methods for modelling colloidal mixtures in a explicit solvent using molecular dynamics, *Comp. Phys. Comm*. **179** 320–329 (2008).

[5] The term "compact" here is used to indicate stencils of extent three or five points in each dimension. Anything larger is considered "extended".

[6] This is not exactly what LAMMPS does in practice. For "half" `NPair` implementations, each pair i, j is stored twice only if i and j are owned by separate MPI tasks. An additional condition is added in the interaction implementation to test the situation.

[7] V. Ogarko and S Luding, A fast multilevel algorthm for contact detection of arbitrarily polydisperse objects, *Comp. Phys. Comm.* **183** 931–936 (2012).

[8] D. Krijgsman, V. Ogarko, and S. Luding, Optimal parameters for a hierarchical grid data structure for contact detection in arbitrarily polydisperse particle systems, *Comp. Part. Mech.*, **1** 357–372 (2014).

[9] https://github.com/lammps/lammps (2018). This work is based on the May 2018 stable release.

[10] If the average number of particles per bin falls below unity, empty bins are more common: this is likely to be inefficient. However, for the user wishing to know where to place the criterion d_{sep}, the average number of particles per bin is not an immediately accessible quantity to base a decision on.

[11] X. Huang, K.J. Hanley, C. O'Sullivan, C.-Y. Kwok, Exploring the influence of interparticle friction on critical state behaviour using DEM, *Int. J. Numer. Anal. Meth. Geomech.* **38**(12) 1276–1297 (2014).