Multi-level contact detection to enable efficient polydisperse DEM simulations using LAMMPS

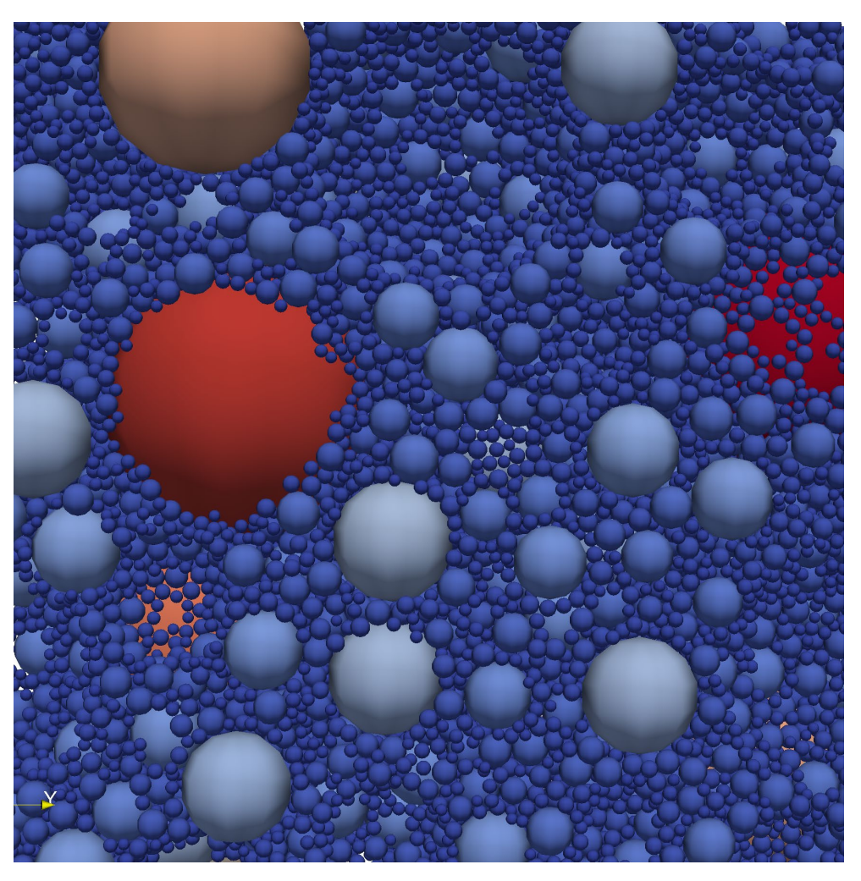
Tom Shire — *School of Engineering, University of Glasgow, Glasgow, UK*

Kevin Stratford — *Edinburgh Parallel Computing Centre (EPCC), The University of Edinburgh, Edinburgh, UK*

Kevin J. Hanley — *Institute for Infrastructure and Environment, School of Engineering, The University of Edinburgh, Edinburgh, UK*

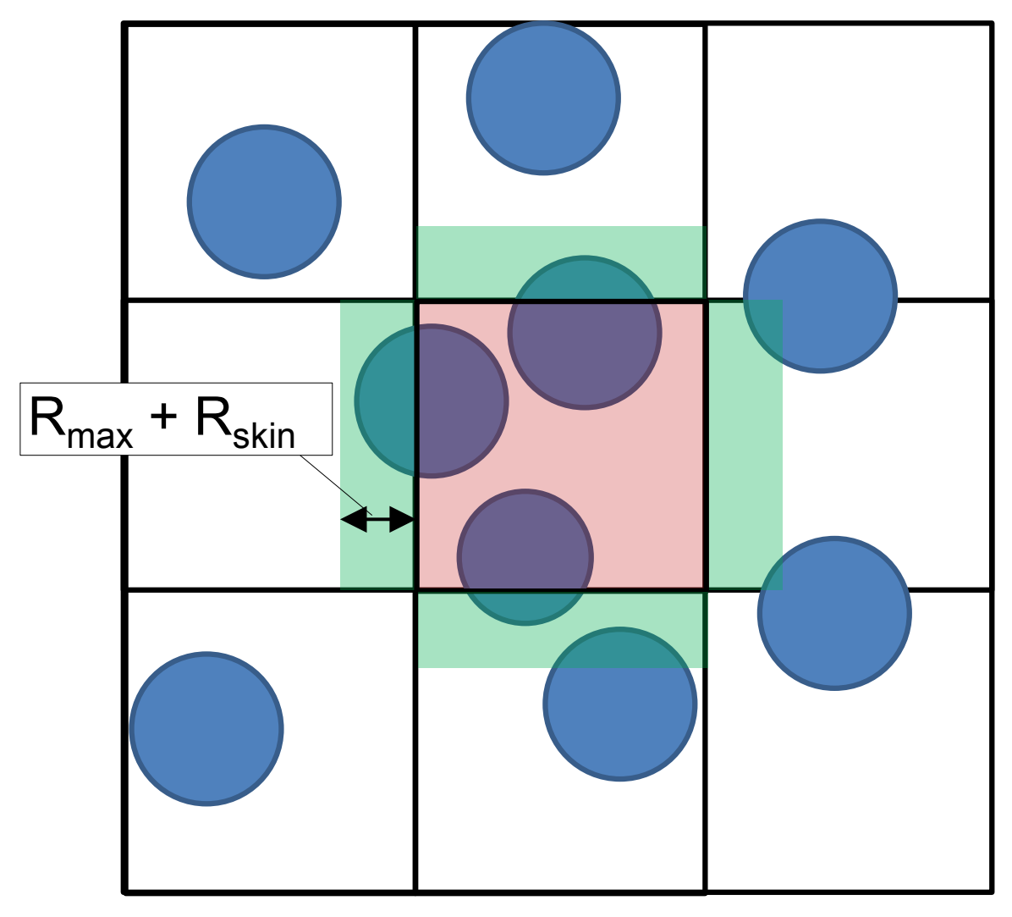
INTRODUCTION

- Polydisperse granular materials occur in many physical and industrial settings.
- Often the range of particle sizes in such systems spans several orders of magnitude (Fig. 1). For example, the granular filter material used to construct the Bennett Dam in Canada contains particles ranging from 0.08 to 75 mm.
- Fig. 2 & 3 summarise a typical method to detect contacts and transfer 'ghost particle' information between processors efficiently for monosized particles.
- When applied to polydisperse systems, these methods become increasingly time-consuming and inefficient as the particle size distribution becomes broader.
- Objective: extend the existing LAMMPS C++ class structure to allow efficient contact detection in cases where a wide disparity in particle sizes exists



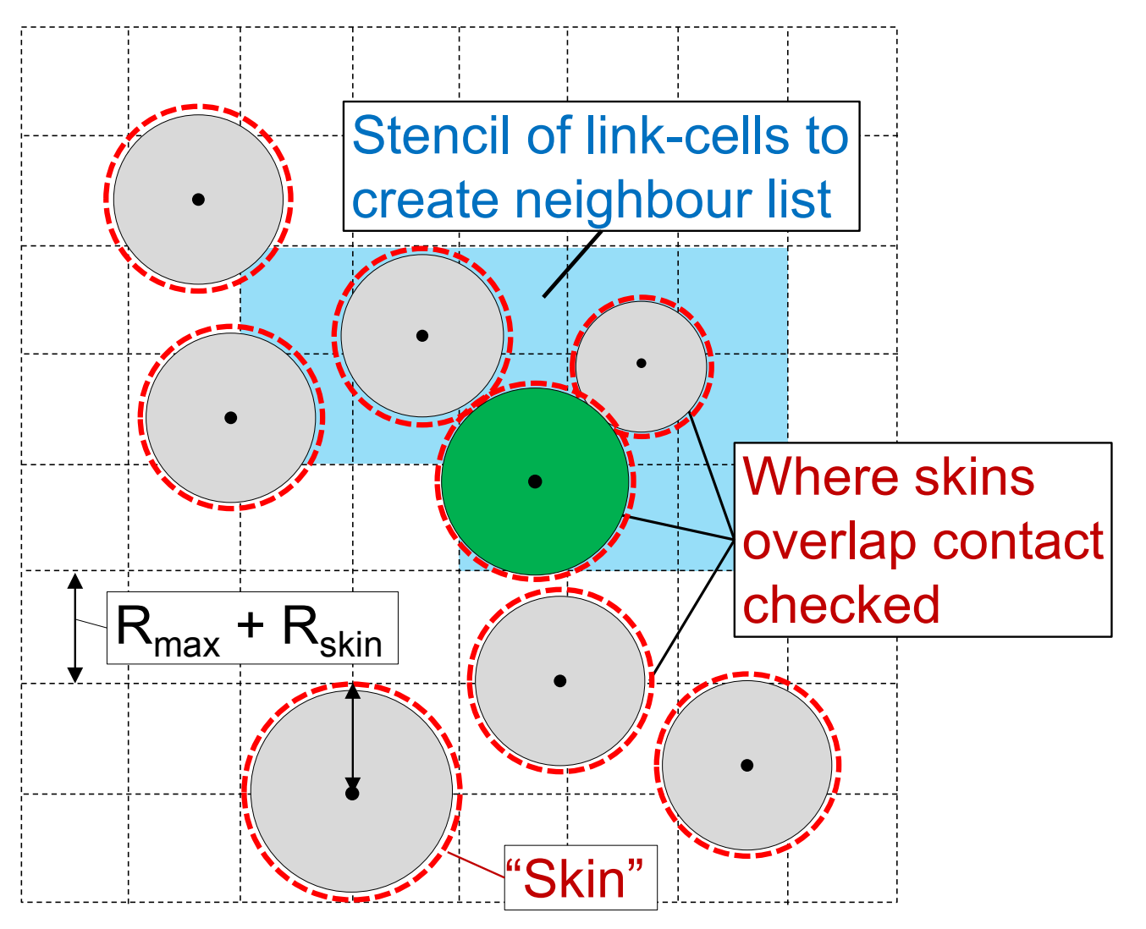
↑ Fig. 1

Example of a highly polydisperse DEM sample



← Fig. 3

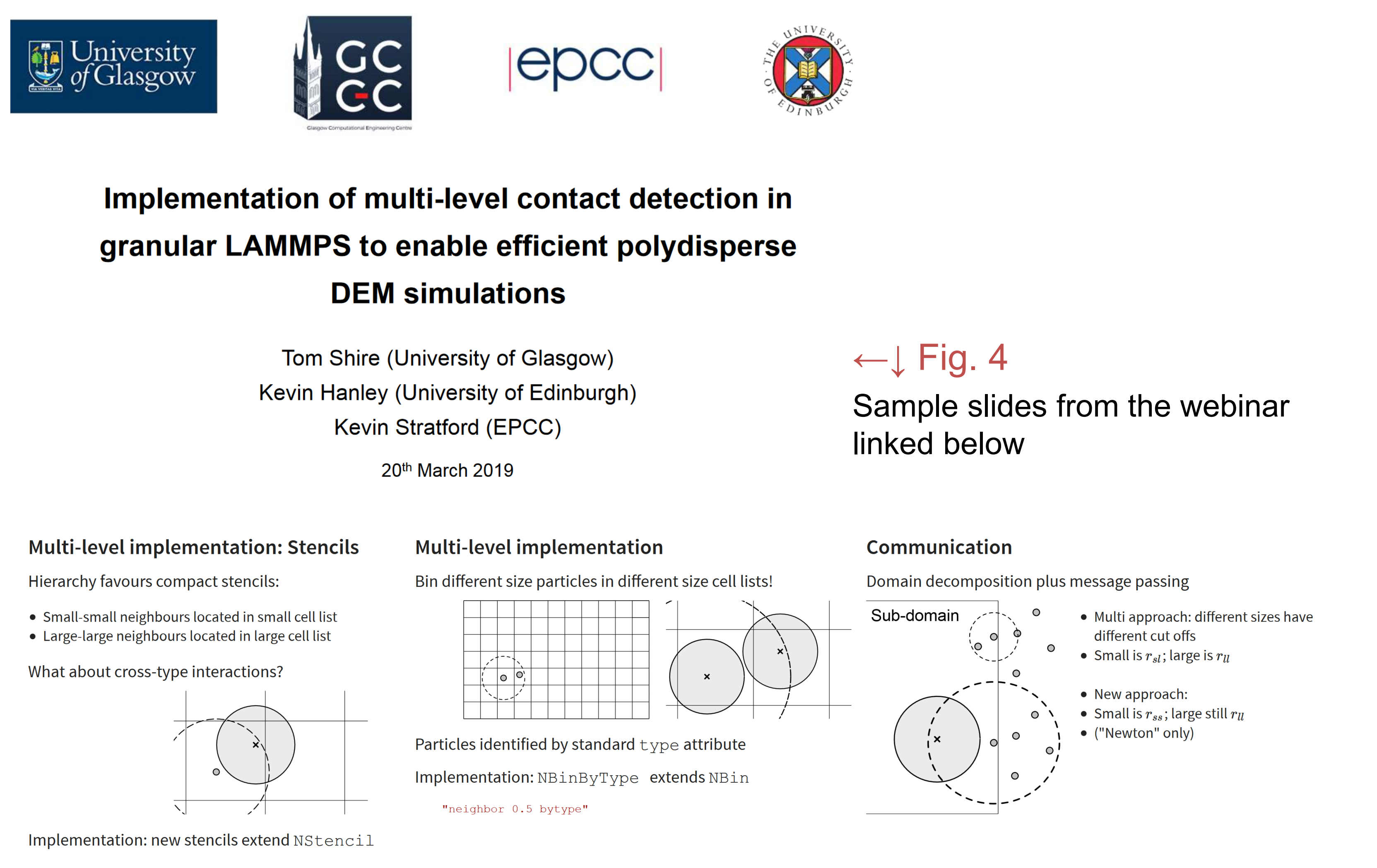
Existing inter-processor communication strategy based on the radius of the largest particle



Stencil of link-cells to create neighbour list
Where skins overlap contact checked
$R_{max} + R_{skin}$
"Skin"

Fig. 2 →

Existing contact detection for Granular pairstyles in LAMMPS. This shows the *bin* neighbour style with a single stencil of link-cells, the sizes of which depend on the radius of the largest particle



Implementation of multi-level contact detection in granular LAMMPS to enable efficient polydisperse DEM simulations

Tom Shire (University of Glasgow)
Kevin Hanley (University of Edinburgh)
Kevin Stratford (EPCC)

20th March 2019

Multi-level implementation: Stencils

Hierarchy favours compact stencils:

- Small-small neighbours located in small cell list
- Large-large neighbours located in large cell list

What about cross-type interactions?

Implementation: new stencils extend `NStencil`

Multi-level implementation

Bin different size particles in different size cell lists!

Particles identified by standard `type` attribute

Implementation: `NBinByType` extends `NBin`

`"neighbor 0.5 bytype"`

Communication

Domain decomposition plus message passing

- Multi approach: different sizes have different cut offs
- Small is r_{sl}; large is r_{ll}

- New approach:
- Small is r_{ss}; large still r_{ll}
- ("Newton" only)

←↓ Fig. 4

Sample slides from the webinar linked below

MULTI NEIGHBOUR STYLE IN LAMMPS

- The *multi* neighbour style is currently available only for non-Granular pairstyles. This study used a test implementation by Dr Ishan Srivastava @ Sandia.
- It is assumed that particle types are chosen which distinguish between particles based on their radius.
- One link-cell list is constructed on the basis of the smallest–smallest interaction cut-off. One stencil of link-cells is needed per particle type.
- Communication of ghost particle information depends on interaction cut-offs, so is more efficient for small particles than the approach in Fig. 3.

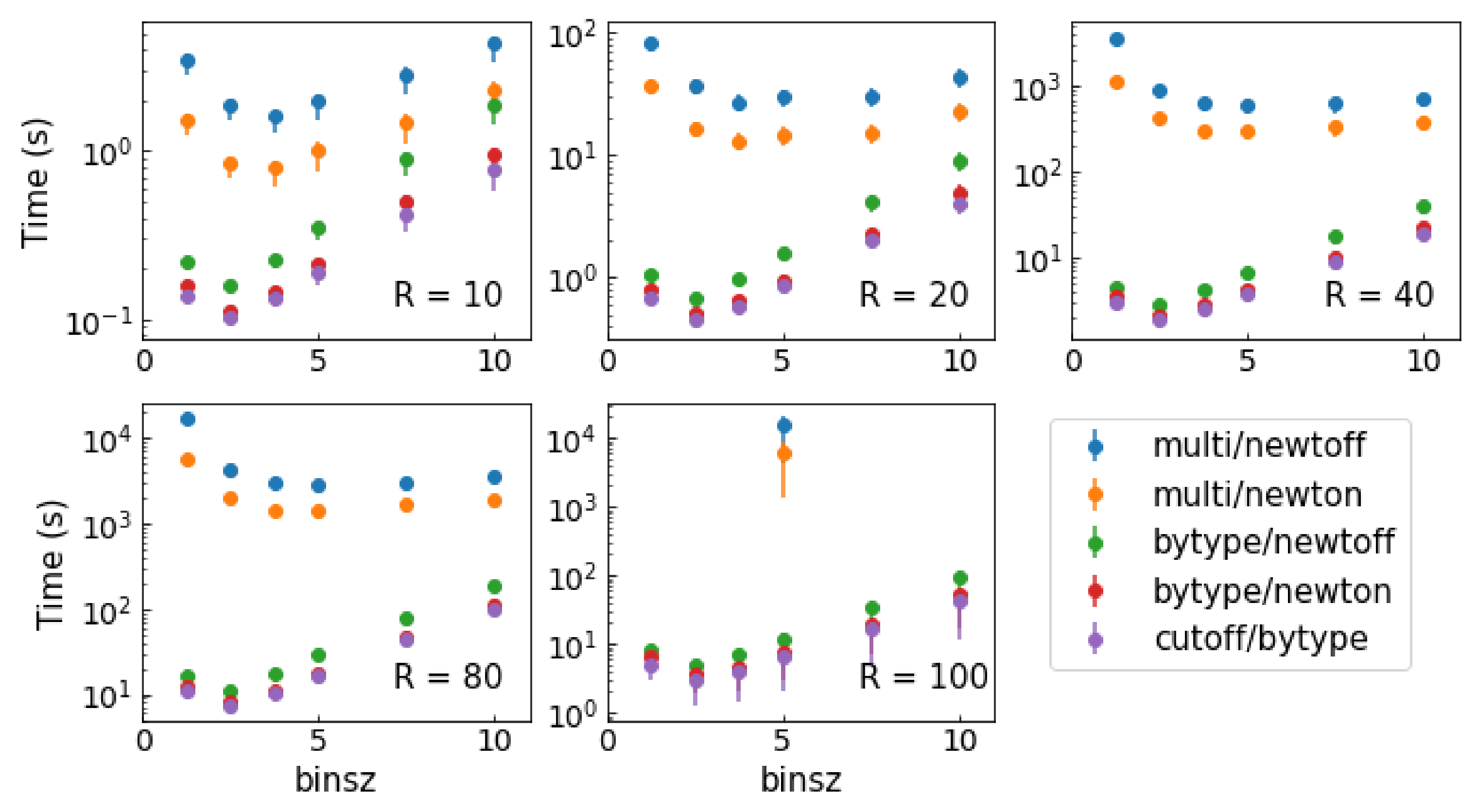
OUR *BYTYPE* IMPLEMENTATION FOR GRANULAR SYSTEMS

- As for *multi*, particle types are allocated based on particle radius.
- However, separate cell lists are instantiated for each type. The smallest particles (only) are binned in the smallest cell list; the largest particles (only) in the largest cell list, etc.
- Interactions between like particles are identified via the appropriate cell list using a compact stencil.
- Interactions between different types can be accomplished efficiently, particularly if *newton on* (an expensive large-to-small search is not necessary).
- Communication more efficient than *multi* for *newton on*.

WEBINAR

- A detailed discussion of the *bytype* implementation was given in a March 2019 webinar for the ARCHER service (UK national HPC facility).
- This 45-minute webinar is available on YouTube:

https://youtu.be/4wrkJR93hXI

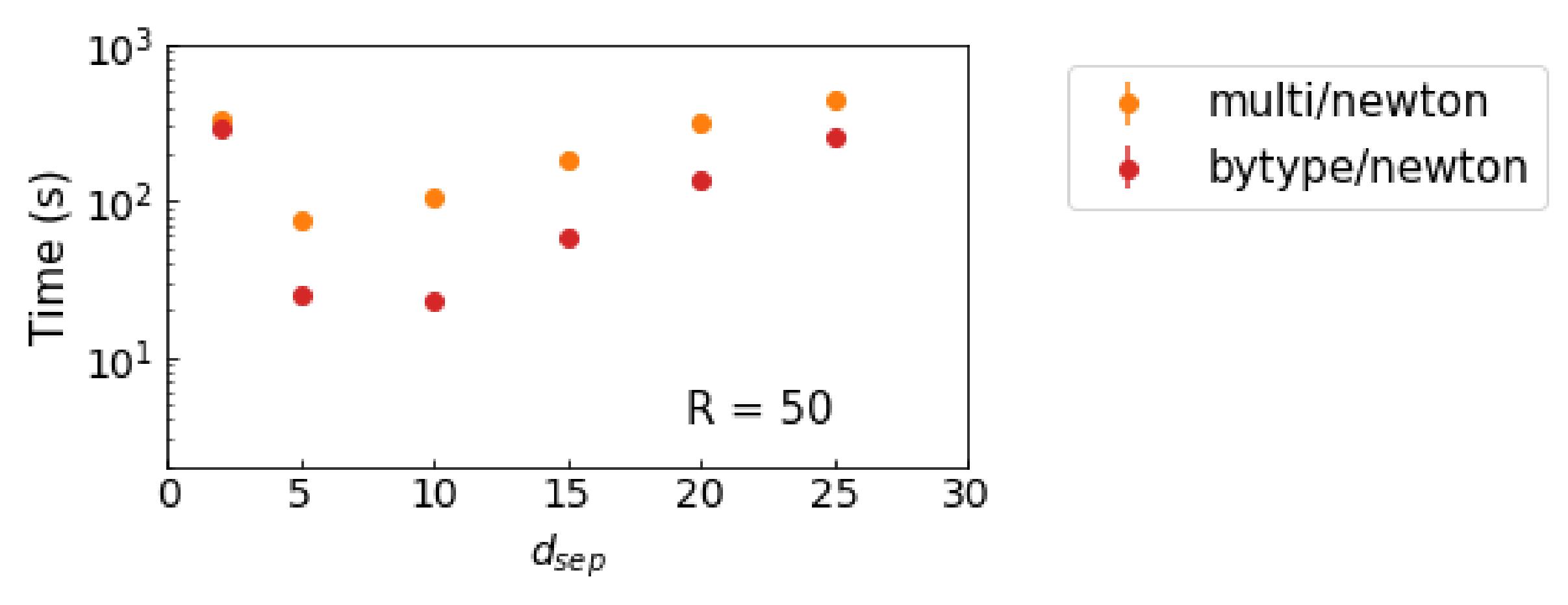


↑ Fig. 5

Comparing simulation times (s) for five contact detection approaches: *multi* with *newton on/off*; *bytype* with *newton on/off* and *bytype* with a reduced communication approach and *newton on* ('*cutoff/bytype*').

SAMPLE RESULTS

- Fig. 5 compares five contact detection approaches using bidisperse systems containing 711k–66M particles in dense configurations with a Hookean pairstyle.
- The times reported by LAMMPS after 100 time-steps are plotted as a function of the cell list bin size (binsz) for five values of R = r_{large}/r_{small}.
- A growing disparity between the *multi* and *bytype* choices can be seen as R increases. In all cases, *newton on* outperforms *newton off*.
- For a polydisperse system containing ~5M particles with r_{large}/r_{small} = 50, *multi* and *bytype* were compared with newton on (Fig. 6). The continuous size distribution was divided into two types at the location d_{sep}: the horizontal axis on Fig. 6.
- Six neighbour list rebuilds took place for each case.
- The time taken for neighbour search is significantly reduced by choosing an appropriate d_{sep} to assign diameters to types.
- The *bytype* method outperforms *multi* by about a factor of 3 in the best case.



↑ Fig. 6

Time (s) for six neighbour list rebuilds for a synthetic polydisperse case against the separation criterion for exactly two types.

CONCLUSIONS & FUTURE WORK

- The implemented *bytype* hierarchical cell list approach allows bidisperse and polydisperse systems to be run with significantly improved efficiency when compared with *multi* (and especially *bin*).
- Initial testing indicates that two types is often optimal. Further investigation is needed to understand how the number of types and the location of d_{sep} affects performance for a highly polydisperse system.
- We hope to implement *bytype* in the main LAMMPS distribution.

This development was funded under the embedded CSE programme of the ARCHER UK National Supercomputing Service (http://www.archer.ac.uk).

Thanks to Drs Ishan Srivastava, Jeremy Lechman and Dan Bolintineanu @ Sandia for their *multi* implementation and helpful comments on this work.