A Parallel Grid-based Approach for Multiscale Non-Spherical Geometry Impact Dynamics

Konstantinos Krestenitis[1]

January 30, 2017

[1]School of Engineering and Computing Sciences, University of Durham, DH1 3LE, Durham
konstantinos.krestenitis@durham.ac.uk
Mechanics Research Group

Supervised by
Dr Tobias Weinzierl
Dr Tomasz Koziara (former)
Professor Jon Trevelyan

1 Introduction

We present a Discrete Element Method (DEM) contact detection code that simulates rigid non-spherical particles on manycore shared memory machines and distributed memory computers. DEM is used to study granular particles in fields like soil mechanics. We rely on triangulated particle meshes to model surfaces. Spherical or multi-sphere models currently are state-of-the-art - due to a lack of well-suited software and runtime demands. Non-spherical particles promise to facilitate more accurate physics than sphere-based approaches [1]. The focus on triangles for rigid contact mechanics facilitates memory layouts allowing vectorised computation [2, 40, 41]. It is vital to yield high performance on current and upcoming processor architectures to enable engineers to simulate more realistic materials.

It is important to investigate the problem of finding the minimum distance between triangles because of the changes in the computational hardware [9]. The central processing unit architecture today and the future upcoming hardware support Single Instruction Multiple Data (SIMD)[2] parallelism which allow data level parallelism. These speed-ups are enabled because of new instruction sets and wide vector register. Furthermore, shared memory parallelism at the node level as well as distributed memory for supercomputers can provide significant speed-ups. It is vital to extract resources available on current and upcoming hardware, the speed-ups enable more triangles to be used to describe surfaces. Consequently, our objective is to enable engineers to use an algorithm that is capable to handle the maximum amount of triangles per time step to do better engineering and science.

In the present work we propose a hybrid method that combines the advantages of two optimised triangle-to-triangle distance computation methods. It benefits from both performance and robustness of an iterative Newton-penalty and a brute force solver. We exploit shared memory parallelism and SIMD (Single Instruction Data) to perform contact detection at the node level. On distributed memory, we use spatial domain decomposition whilst with the Message Passing Interface (MPI), we exploit asynchronous non-blocking communication to overlap data exchange with computation.

The state-of-the-art large scale DEM work, to the best of my knowledge, relies on sphere-based or multi-sphere particles [21, 22]. Large scale applications of spherical particles can be found in the field of molecular dynamics. Large scale non-spherical particle based DEM simulations in literature [20, 29, 28, 30] often describe particles as convex polyhedra and contact detection is resolved with GJK (Gilbert-Johnson-Keerthi) [14] variants which require interpenetration and access to all vertices of the particle to detect contact on the surface. Interpenetration introduces contact point clustering and the problem of divergence [38, 37, 41]. In addition, convex polyhedra based contact detection methods require undeterministic memory access to all particle simplices [14] which is contradictory to the aligned memory access imposed by SIMD for data locality. Triangle-based contact detection retain locality for vectorisation as well as increases accuracy of contact point generation [41]. Furthermore, the method can be extended to run on shared memory manycore and distributed memory machines.

The remainder of the paper is structured as follows. In Section 2, we describe the serial DEM algorithm. In Section 3 we review two triangle-to-triangle distance algorithms.

We create a new hybrid shared memory method that benefits from fast convergence and robustness. Next, in Section 3, we propose a distributed memory algorithm for contact detection that use asynchronous communication to overlap computation over communication. Section 4, discusses the future research directions.

2 Algorithmic Overview

- Completed since last report: Implemented serial DEM simulation with triangle-to-triangle contact detection using spring-dashpot contact forces, explicit time step.
- New concepts: New shared memory hybrid method for computation of triangle-to-triangle distance.
- Open questions: Best-case design of distributed memory implementation is not clear yet, real-world experiments are missing.

In contact mechanics, fluid-stucture interaction or other fields, it is an essential task to compute the distance between geometries to determine contact. Contact detection also is the most expensive algorithmic step [39, 57]. We present a Discrete Element Method (DEM) code that simulate particles that interact with spring-based contact. Non-spherical particles promise to facilitate more accurate physics than sphere-based approaches [1, 20]. The contact detection routine determines contact based on the distance between the triangles of every particle against all other. If two triangles are closer than a prescribed threshold, they are considered to contact each other. The use of triangles for rigid body contact dynamics rather than arbitrary polygons simplifies geometric checks and facilitates memory layouts that allow vectorised computation [13, 32, 41, 42].

The serial DEM Algorithm 1 demonstrates the execution of contact detection, an $O(n^2)$ operation. In line 3 and line 4 the nested for loop indicate the complexity of iterating through all triangle pairs. In line 5 TTD(i, j) invokes a Triangle-to-Triangle Distance (TTD) algorithm that determines the distance between triangle i and triangle j. Based on the boundary layer margin contact model [41] to avoid penetration, we check if distance between Triangle i and j is smaller than a set margin (line 6). If the parent particle of triangle i and j is not the same then this indicate a contact point between the two particles (line 7). The contact point information at line 13 and line 14 is used to derive the forces. The forces accelerate the particles that are then integrated by an explicit time stepping scheme such as explicit Euler.

Algorithm 1 DEM Serial Simulation Pseudo code

```
1 FOR time = 0; time < simulation time; time+=step
2   //contact detection
3     FOR i = 0 to N triangles
4       FOR j = i+1 to N triangles
5         distance = TTD(i,j)
6         IF (distance < margin) AND ParticleID(i) != ParticleID(j)
7           contact(PID(i)).add(point, normal)
8         ENDIF
9       ENDFOR
10    ENDFOR
11  //force derivation
12    FOR z = 0 to NB particles
13      FOR k = 0 to contacts(z).size()
14          force = granular(velocity(z), position(z), contacts(z).getcontact(k))
15      ENDFOR
16    ENDFOR
17  //explicit time stepping
18 ENDFOR
```

Algorithm 2 Spring-dashpot force algorithm

```
0 FUNCTION force = penaltyForce(normal, relativeVelocity, depth, massA, massB)
1  mass = 1.0 / ((1/massA) + (1/massB));
2  velocitymagnitude = DOT(relativeVelocity, normal);
3  magnitude = SPRING*depth + DAMPER*sqrt(SPRING*mass)*velocitymagnitude;
4  force = magnitude*normal;
5 ENDFUNCTION
```

Algorithm 2 shows the spring-based force derivation approach [38, 53, 57, 59]. To define contact points without penetration, we use the boundary layer margins to extend the surface boundary. Margin size is set based on velocities and time step size. When in contact, the margin overlap is used as the interpenetration depth. The contact point is at the middlepoint of the distance and normal direction is either side of the distance. At line 0, given the contact point normal, relative linear velocities, penetration depth and the mass of the two bodies we derive the interaction force to update the position of particle triangles on every time step.

3 Domain Decomposition for Contact Detection

- Completed since last report: Implemented domain decomposition for distributed memory load balancing using library, MPI blocking communication for data migration.
- New concepts: Overlapping process communication for contact detection with ghost data. Asynchronous communication waiting time significant as number of ranks increase.
- Open questions: multilayer grid/spacetree implementation for DEM

We further scale the computation of triangle-to-triangle contact with the Message Passing Interface (MPI) [16] for distributed memory computation. Our DEM-based contact detection is inspired by molecular dynamics simulations, where millions of spheres are simulated. State-of-the-art MD computation is decomposed into smaller computational sub-domains and communication exchanges necessary boundary information. Similar computational stages are applied by us in the DEM-based contact detection algorithm. We study synchronous and asynchronous modes of data exchange and how two communication strategies can be exploited to increase performance and minimize computation.

The distributed memory DEM contact detection algorithm is performed in three stages. In stage one the computational domain is divided into equally balanced sub-domains. In stage two, data migration assigns the sub-domains to MPI instances. In stage three, we compute the distances to determine contact between all triangles. In stage four we accumulate the forces and in stage five we perform the explicit time step integration.

To scale the performance of contact detection simulation, the computational workload has to be processed in parallel by splitting the domain into sub-domains and computation deployment is based on it, making the decomposition an important stage. Decomposition affects load balancing and the communication patterns of the simulation. Communication is essential when sub-domains are inter-dependent since new boundaries are introduced with decomposition. It is important to decompose evenly whilst also reducing sub-domain communication. There are three types of decomposition that are widely used in different contexts [11, 47] that can apply in DEM. Decomposition by particle, decomposition by force and spatial decomposition.

In domain decomposition by particle [37], the domain is divided such that all sub-domains hold equal number of particles. For N particles we split the domain to N parts and we assign them to P processors (N/P). The sub-domain boundary is always located at the space between particles. In this method an all-to-all communication is required to detect contact points because they are always located at the sub-domain boundary. The global communication is a major disadvantage for small particles as each processor communicates with all other processors, staggering the overall parallel processing. Another disadvantage is the non-equal splitting (N/P) as some particles may require more computation than others (i.e. more triangles). The main advantage of particle-based decomposition is the implementation simplicity [47].

An alternative decomposition approach found in literature is force-based decomposition that is often applied in molecular dynamics (MD) and smoother particle hydrodynamic (SPH) simulations [4, 19, 22, 47, 48, 50]. The method formulates a interaction force matrix that solves the contact detection and the forces as a group of linear equations. The matrix is decomposed into small blocks and shared on each process to solve in parallel. Force based decomposition assume short-range interactions and thus a dense force matrix to solve. In force based methods, the advantage is that computation is decomposed without any spatial information from the particles. The major disadvantage in DEM applications is the assumption of dense and uniform sparse force matrices, the assumption can be wrong leading to imbalanced domains on run-time.

According to N-body simulation literature [11, 15, 58, 60] the state-of-the-art method for supercomputing applications is the spatial decomposition. Our decomposition is based on the spatial position of vertices using Recursive Coordinate Bisection (RCB) [3]. In RCB, the computational domain is first divided into two regions by a cutting plane orthogonal to one of the coordinate axes so that half the work load is in each of the sub-regions. The splitting direction is determined by computing in which coordinate direction the set of objects is most elongated, based upon the geometric locations of the objects. Each triangle vertex thus is owned and persistently stored exclusively on one process/sub-domain for the whole duration of the timestep.

Spatial decomposition implicitly creates logical boundaries that split space and computation (c.f. Figure **??**). The contact detection complexity of N trianglesthat belong to n_i particles without boundary boxes is $\sum_{i=0}^{n} n_i = \mathrm{O}(n^2)$ in big-O notation (Algorithm 1). Contrary, with equally spaced boundary boxes/cells, if C is the maximum number of particles then there are 26 neighboring cells and $max(n_i)$ is the maximum number of triangles per particle i which result to $n_1 \times 26 \times C \times max(n_i)$ triangles per particle i in the local cell neighborhood. For all N triangles in the domain there are $N \times 26 \times C \times max(n_i)$ per neighborhood. Boundary boxes due to spatial decomposition reduce the overall computational complexity to $O(n)$. Similarly, in MD simulations a cut-off range set the range of interaction fields per particle to remove redundant computation. Equally spaced boundary boxes rely on octree-based variants data structures that are great for recursive eight cell space subdivision. Our decomposition relies on non-uniform recursive subdivisions that rely on kd-tree-based data structure decomposition [6] (i.e. RCB) and the number of neighbours can be arbitrarily many as they are not equally spaced, but they are equally vertex-sized.

The number of neighbours for both methods may have an implication on communication patterns between processes. It is an open question what are the performance implications on uniform spaced octree-based decomposition versus the non-uniform kd-tree based decomposition. It is an area of investigation since on an octree-based approach information about the level of refinement is known a priori by the sub-domain boundary size, which is an interesting application for multiscale simulations.

When spatial decomposition finishes we migrate the data to the processors (Algorithm 6 line 2) with blocking synchronous communication. At each time step the triangles migrate according to the DEM kinematics. In addition to migration, a local area data exchange is required to communicate the boundaries of the sub-domains that cut triangles at the boundary.

4 Conclusion and Future Work

We implement a new fast but also robust hybrid algorithm that exploits modern CPU architectures. In addition, we study different schemes of domain decomposition in literature. I implement the state-of-the-art spatial domain decomposition and load balancing using Zoltan. Manual migration has been implemented using blocking MPI communication. We implement an asynchronous communication for data exchange of ghost data and compare two possible strategies. Finally based on benchmark measurements we pick the best performing scheme that overlaps computation, increases performance and decreases data exchange volume.

The next stage of my research is to investigate contact detection strategies for large scale supercomputing on a multiscale setting using adaptive multiscale grids. I'm also interested in vectorized memory layout implications on MPI buffered communication. Lastly, the source code will be generalized to form a open source library that any simulation tool can use.

According to the project plan of the previous report, the work packages have been completed as projected.

4.1 Acknowledgments

This work has been sponsored by EPSRC (Engineering and Physical Sciences Research Council) and EDF Energy as part of an ICASE studentship. This work also made use of the facilities of N8 HPC provided and funded by the N8 consortium and EPSRC (Grant No. N8HPC_DUR_TW_PEANO). The Centre is co-ordinated by the Universities of Leeds and Manchester. We also thank University of Durham for the supercomputing resources and technical assistance. All underlying software is open source and available at: https://github.com/KonstantinosKr/delta.

4.2 Work Packages

References

[1] Fernando Alonso-Marroquin, Alvaro Ramirez-Gomez, Carlos Gonzalez-Montellano, Nigel Balaam, Dorian A H Hanaor, E. A. Flores-Johnson, Yixiang Gan, Shumiao Chen, and Luming Shen. Experimental and numerical determination of mechanical properties of polygonal wood particles and their flow analysis in silos. *Granular Matter*, 15(6):811–826, 2013.

[2] Mauricio Alvarez, Esther Salami, Alex Ramirez, and Mateo Valero. Performance impact of unaligned memory operations in SIMD extensions for video codec applications. *ISPASS 2007: IEEE International Symposium on Performance Analysis of Systems and Software*, pages 62–71, 2007.

[3] E. G. Boman, U. V. Catalyurek, C. Chevalier, and K. D. Devine. The Zoltan and Isorropia parallel toolkits for combinatorial scientific computing: Partitioning, ordering, and coloring. *Scientific Programming*, 20(2):129–150, 2012.

[4] Kevin J. Bowers, Ron O. Dror, and David E. Shaw. Zonal methods for the parallel execution of range-limited N-body simulations. *Journal of Computational Physics*, 221(1):303–329, 2007.

[5] Robert Bridson, Ronald Fedkiw, and John Anderson. Robust treatment of collisions, contact and friction for cloth animation. *ACM Transactions on Graphics*, 21(3):594–603, 2002.

[6] Russell A Brown. Building a Balanced k-d Tree in O (kn log n) Time. 4(1):50–68, 2015.

[7] T.S. Coffey, C.T. Kelley, and D.E. Keyes. Pseduo-Transient Continuation and Differential-Algebraic Equations. *SIAM Journal of Scientific Computing*, pages 1–13, 2002.

[8] Leonardo Dagum and Ramesh Menon. Openmp: An industry-standard api for shared-memory programming. *IEEE Comput. Sci. Eng.*, 5(1):46–55, January 1998.

[9] J. Dongarra, P. Beckman, T. Moore, P. Aerts, G. Aloisio, J.-C. Andre, D. Barkai, J.-Y. Berthou, T. Boku, B. Braunschweig, F. Cappello, B. Chapman, Xuebin Chi, a. Choudhary, S. Dosanjh, T. Dunning, S. Fiore, a. Geist, B. Gropp, R. Harrison, M. Hereld, M. Heroux, a. Hoisie, K. Hotta, Zhong Jin, Y. Ishikawa, F. Johnson, S. Kale, R. Kenway, D. Keyes, B. Kramer, J. Labarta, a. Lichnewsky, T. Lippert, B. Lucas, B. Maccabe, S. Matsuoka, P. Messina, P. Michielse, B. Mohr, M. S. Mueller, W. E. Nagel, H. Nakashima, M. E. Papka, D. Reed, M. Sato, E. Seidel, J. Shalf, D. Skinner, M. Snir, T. Sterling, R. Stevens, F. Streitz, B. Sugar, S. Sumimoto, W. Tang, J. Taylor, R. Thakur, a. Trefethen, M. Valero, a. van der Steen, J. Vetter, P. Williams, R. Wisniewski, and K. Yelick. The International Exascale Software Project roadmap. *International Journal of High Performance Computing Applications*, 25(1):3–60, 2011.

[10] David Eberly. Distance between point and triangle in 3D. *Magic Software...*, pages 1–6, 1999.

[11] Wolfgang Eckhardt. Efficient HPC Implementations for Large-Scale Molecular Simulation in Process Engineering. 2014.

[12] Wolfgang Eckhardt, Robert Glas, Denys Korzh, and Stefan Wallner. On-the-fly memory compression for multibody algorithms.

[13] Alexandre E. Eichenberger, Peng Wu, and Kevin O'Brien. Vectorization for SIMD architectures with alignment constraints. *ACM SIGPLAN Notices*, 39(6):82, 2004.

[14] C Ericson. The Gilbert-Johnson-Keerthi algorithm, 2005.

[15] Florian Fleissner and Peter Eberhard. Parallel Load Balanced Particle Simulation with Hierarchical Particle Grouping Strategies. pages 33–44.

[16] Message P Forum. Mpi: A message-passing interface standard. Technical report, Knoxville, TN, USA, 1994.

[17] Franz Franchetti and Markus Püschel. Generating SIMD vectorized permutations. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 4959 LNCS:116–131, 2008.

[18] Robert M Freund. Algorithms for Constrained Optimization A . 1 Penalty and Barrier Methods. *Methods*, (23):55–58, 1968.

[19] Evghenii Gaburov and Yuri Cavecchi. XeonPhi Meets Astrophysical Fluid Dynamics. pages 1–7.

[20] L. Girolami, V. Hergault, G. Vinay, and A. Wachs. A three-dimensional discrete-grain model for the simulation of dam-break rectangular collapses: Comparison between numerical results and experiments. *Granular Matter*, 14(3):381–392, 2012.

[21] Pedro Gonnet. QuickSched: Task-based parallelism with dependencies and conflicts. pages 1–25, 2013.

[22] Pedro Gonnet. Efficient and Scalable Algorithms for Smoothed Particle Hydrodynamics on Hybrid Shared/Distributed-Memory Architectures. *SIAM Journal on Scientific Computing*, 37(1):C95–C121, 2015.

[23] P C Hansen. The L-Curve and its Use in the Numerical Treatment of Inverse Problems. *in Computational Inverse Problems in Electrocardiology, ed. P. Johnston, Advances in Computational Bioengineering*, 4:119–142, 2000.

[24] David Harmon, Etienne Vouga, Breannan Smith, Rasmus Tamstorf, and Eitan Grinspun. Asynchronous contact mechanics. *Communications of the ACM*, 55(3):102, 2012.

[25] David Harmon, Etienne Vouga, Rasmus Tamstorf, and Eitan Grinspun. Robust treatment of simultaneous collisions. *ACM Transactions on Graphics*, 27(3):1, 2008.

[26] Mark Hoemmen. Generating random numbers in parallel. pages 1–8, 2007.

[27] Kaixi Hou, Hao Wang, Wu-chun Feng, and Virginia Tech. ASPaS : A Framework for Automatic SIMDization of Parallel Sorting on x86-based Many-core Processors Categories and Subject Descriptors. pages 383–392.

[28] K. Iglberger and U. Rüde. Massively parallel granular flow simulations with non-spherical particles. *Computer Science - Research and Development*, 25(1-2):105–113, 2010.

[29] Klaus Iglberger and Ulrich Rüde. Massively parallel rigid body dynamics simulations. *Computer Science - Research and Development*, 23(3-4):159–167, 2009.

[30] Klaus Iglberger and Ulrich Rüde. Large-scale rigid body simulations. *Multibody System Dynamics*, 25(1):81–95, 2011.

[31] Adrian Jensen, Kirk Fraser, and George Laird. Improving the Precision of Discrete Element Simulations through Calibration Models. *13th International LS-DYNA Users Conference*, pages 1–12, 2014.

[32] Ben Juurlink. Performance Impact of Misaligned Accesses in SIMD Extensions. *Computer Engineering*, pages 334–342.

[33] Å Kaczmarczyk, Tomasz Koziara, and Cj Pearce. Corotational formulation for 3d solids. An analysis of geometrically nonlinear foam deformation. *arXiv preprint arXiv:1110.5321*, pages 1–23, 2011.

[34] Nils Karajan, Zhidong Han, Hailong Teng, and Jason Wang. On the Parameter Estimation for the Discrete-Element Method in LS-DYNA ®. *13 th International LS-DYNA Users Conference*, pages 1–9, 2014.

[35] Tero Karras. Maximizing Parallelism in the Construction of BVHs , Octrees , and k-d Trees. *EGGH-HPG'12 Proceedings of the Fourth ACM SIGGRAPH / Eurographics conference on High-Performance Graphics*, pages 33–37, 2012.

[36] Danny M. Kaufman, Timothy Edmunds, and Dinesh K. Pai. Fast frictional dynamics for rigid bodies. *ACM Transactions on Graphics*, 24(3):946, 2005.

[37] T. Koziara and N. Bicani. A distributed memory parallel multibody Contact Dynamics code. *International Journal for Numerical Methods in Engineering*, 87(1-5):437–456, 2011.

[38] Tomasz Koziara and Nenad Bićanić. Semismooth Newton method for frictional contact between pseudo-rigid bodies. *Computer Methods in Applied Mechanics and Engineering*, 197(33-40):2763–2777, 2008.

[39] Tomasz Koziara and Nenad Bićanić. Simple and efficient integration of rigid rotations suitable for constraint solvers. *International Journal for Numerical Methods in Engineering*, 81(9):1073–1092, 2010.

[40] Tomasz Koziara, Collision Detection, Sweep-plane Approach, Spatial Hashing, and Priority Search Tree. Sweep-plane approach to bounding box. *Complas VIII*, pages 1–4, 2005.

[41] Konstantinos Krestenitis and Tomasz Koziara. Calculating the Minimum Distance Between Two Triangles on SIMD Hardware. Number April, pages 8–11. 23th UK Conference on Computational Mechanics (ACME-UK). Swansea, UK., 2015.

[42] Konstantinos Krestenitis, Tobias Weinzierl, and Tomasz Koziara. A Multiscale DEM Contact Detection Code using Triangles for Non-Spherical Particle Simulations. Number April, pages 1–4. 24th UK Conference on Computational Mechanics (ACME-UK). Cardiff, UK., 2016.

[43] Shin-jae Lee, Minsoo Jeon, Dongseung Kim, and Andrew Sohn. Partitioned Parallel Radix Sort 1. *Journal of Parallel and Distributed Computing*, 668(October 2000):656–668, 2002.

[44] Xiang Ni, Laxmikant V. Kale, and Rasmus Tamstorf. Scalable Asynchronous Contact Mechanics Using Charm++. *2015 IEEE International Parallel and Distributed Processing Symposium*, pages 677–686, 2015.

[45] Igor P. Omelyan. Algorithm for numerical integration of the rigid-body equations of motion. *Physical Review E*, 58(1):1169–1172, 1998.

[46] Eric J R Parteli. DEM simulation of particles of complex shapes using the multisphere method: Application for additive manufacturing. *AIP Conference Proceedings*, 1542:185–188, 2013.

[47] Steve Plimpton. Fast Parallel Algorithms for Short – Range Molecular Dynamics. *Jounal of Computational Physics*, 117(June 1994):1–42, 1995.

[48] Chris H Rycroft, Terttaliisa Lind, Salih Güntay, and Abdel Dehbi. Granular flow in pebble bed reactors : dust generation and scaling. pages 447–455, 2012.

[49] Nadathur Satish, Mark Harris, and Michael Garland. Designing Efficient Sorting Algorithms for Manycore GPUs. *Proceedings of 23rd IEEE International Parallel and Distributed Processing Symposium*, pages 1–10, 2009.

[50] David E. Shaw. A fast, scalable method for the parallel evaluation of distance-limited pairwise particle interactions. *Journal of Computational Chemistry*, 26(13):1318–1328, 2005.

[51] Du Shen, Qi Luo, Denys Poshyvanyk, and Mark Grechanik. Automating performance bottleneck detection using search-based application profiling. *Proceedings of the 2015 International Symposium on Software Testing and Analysis - ISSTA 2015*, pages 270–281, 2015.

[52] Alice E Smith and David W Coit. Penalty functions. *Handbook of Evolutionary Computation*, 97(1):C5, 1995.

[53] Jerome M. Solberg and Panayiotis Papadopoulos. Impact of an elastic pseudo-rigid body on a rigid foundation. *International Journal of Engineering Science*, 38(6):589–603, 2000.

[54] Xinmin Tian, Hideki Saito, Serguei V. Preis, Eric N. Garcia, Sergey S. Kozhukhov, Matt Masten, Aleksei G. Cherkasov, and Nikolay Panchenko. Effective SIMD Vectorization for Intel Xeon Phi Coprocessors. *Scientific Programming*, 2015, 2015.

[55] J. Treibig, G. Hager, and G. Wellein. Likwid: A lightweight performance-oriented tool suite for x86 multicore environments. In *Proceedings of PSTI2010, the First International Workshop on Parallel Software Tools and Tool Infrastructures*, San Diego CA, 2010.

[56] Oren Tropp, Ayellet Tal, and Ilan Shimshoni. A fast triangle to triangle intersection test for collision detection. *Computer Animation and Virtual Worlds*, 17(5):527–535, 2006.

[57] Anthony Wachs, Laurence Girolami, Guillaume Vinay, and Gilles Ferrer. Grains3D, a flexible DEM approach for particles of arbitrary convex shape - Part I: Numerical model and validations. *Powder Technology*, 224:374–389, 2012.

[58] Anthony Wachs and Andriarimina Daniel Rakotonirina. A MPI / domain decomposition strategy for large-scale simulations of granular media made of particles of arbitrary shape. 130(2011):69360, 2012.

[59] J R Williams and R O Connor. Discrete Element Simulation and the Contact Problem. *Methods in Engineering*, 6(June 1996):279–304, 1999.

[60] Afra Zomorodian and Herbert Edelsbrunner. Fast Software for Box Intersections. *International Journal of Computational Geometry & Applications*, 12(01n02):143–172, 2002.