A review of recent work on the Discrete Particle Method at the University of Twente : An introduction to the open-source package MercuryDPM

Anthony Thornton, Dinant Krijgsman, Rudi Fransen, Sebastian Gonzalez, Ate te Voortwis, Stefan Luding, Onno Bokhove, Thomas Weinhart
University of Twente, The Netherlands

In this paper we review some recent advances in DEM (DPM) modelling undertaken at the University of Twente. We introduce the new open-source package MercuryDPM that we have been developing over the last few years.

MercuryDPM is an object-oriented program with a simple C++ implementation and includes: support for moving and complex walls, such as polyhedra or screw-threads; state-of-the-art granular contact models; multi-species support; specialised classes, allowing the easy implementation of common geometries like chutes, hoppers, etc.; common handler interfaces for particles, walls and boundaries (so all type of objects are changed using the same interfaces); restarting; large self-test suite and numerous simple demos; visualisation support, both internal and using Visual Molecular Dynamics.

Additionally to these features, MercuryDPM has two major components that, to the best of our knowledge, cannot be found in other DPM packages. Firstly, it uses a novel advanced contact detection method that is able of dealing with multiple distinct granular components with sizes ranging over many orders of magnitude: the hierarchical grid. We explain how this algorithm works and demonstrate the speedup gained over the traditional linked cell approach. This algorithm has lower complexity for poly-dispersed flows which means for the first time large simulations with extremely wide size distributions are feasible.

Secondly, we present a novel way to extract continuum fields from discrete particle systems that is applicable to mixtures as well as boundaries and interfaces. The particle data is coarse grained in a way that is by construction compatible with the continuum equations of mass-, momentum-, and energy balance. Boundary interaction forces are taken into account in a self- consistent way and thus allow the construction of a continuous stress field even within one particles radius of the boundaries. The method does not require temporal averaging and thus can be used to investigate time-dependent flows as well as static and steady situations. This coarse-graining method is available from MercuryDPM either as a post-processing tool or it can be run in real time. In real-time mode, it not only reduces the data which has to be stored but also allows boundary conditions etc. to be updated depending on the current macroscopic state of the system, e.g. allowing the creation of a pressure-release wall.

Finally, we illustrate these tools and a selection of other features of MercuryDPM via various problems including size-driven segregation in chute flow, rotating drums, and screw-conveyer.

GENERAL INTRODUCTION

MercuryDPM is a code for performing discrete particle simulations. Often the method used in these packages is referred to as the discrete element method (DEM), which was originally designed for geotechnical applications [3]. However, as MercuryDPM is designed for simulating particles with emphasis on contact models [7], optimised contact detection for highly different particle sizes [8], and in-code coarse-graining (in contrast to post-processing) [14], we prefer the name discrete particle method (DPM). The code was originally developed for granular chute flows [10, 11, 13], and has since been extended to many other granular applications, including the geophysical modelling of cinder cone creation, indentation of nanoparticles, vibrated granular systems,etc. Despite its granular heritage it is designed in a flexible way so it could can be adapted to include other features such as long-range interactions, non-spherical particles, etc.

WHY A NEW SIMULATION CODE?

There are many open-source particle simulation packages, so the question does arise of why another? The originally concept was to develop a code that could be used along side our (University of Twnente) existing continuum solver hpGEM [9] (see `http://einder.ewi.utwente.nl/hpGEM/` for details). The aim was that the coupled code could be used to approach problems using various multi-scale computational methods.

Additionally also at the University of Twente, a novel contact detection method, the hierarchical grid, that is quicker than existing methods for poly-dispersed flows (and still the same speed for mono-dispersed) had been developed [8]. So the idea of a new simulation code that had three core design aims was born:

1. It should be easy to use with minimal C++ knowledge.
2. It should be built around the new hierarchical grid detection method;
3. It should be able to generate accurate continuum fields that could be used with/along side continuum solvers.

FEATURES

Since it was first started it has evolved and gained many novel features. The main features include:

1. **The hierarchical grid:** The neighbourhood search algorithm to effectively compute interaction forces, even for highly poly-dispersed particles.
2. **Built-in coarse-graining statistical package:** It has an in-built advanced statistics package to extract continuum fields such as density, velocity, structure and stress tensors, either during the computation or as a post-processing step.
3. **Access to continuum fields in real time:** Continuum field can be evaluated at run-time, which means it can respond to its current macroscopic state. An illustrative example of using this would be a pressure-release wall, i.e., a wall whose motion is determined by the macroscopic pressure created via by particle collisions and moves such that its pressure (not position) is controlled.
4. **Contact laws for granular materials:** Many granular contact force models are implemented, including elastic (linear or Hertzian), plastic, cohesive, temperature/pressure/time-dependent (sintering) temperature/pressure/time-dependent, and frictional (sliding/rolling/torsion) forces.,.
5. **Simple C++ implementation:** MercuryDPM consists of a series of C++ classes that are flexible, but easy to use. This allows the user to generate advanced applications with only a few lines of code.
6. **Handlers:** The code has handlers for particles, walls and boundaries. Thus, each object type has a common interface, even though individual objects can have completely different properties. It also makes it easier for the user to create new objects.
7. **Complex walls:** The code not only supports simple flat walls, but also axially symmetric, polyhedral and helical screw walls are all available by default. Additionally, due to the handler interface it is easy for more advanced users to define new types of walls themselves.
8. **Specialised classes:** Many specialised classes exist that reduce the amount of code required by the user to develop standard geometries and applications. Examples include chute flows, vertically vibrated walls and rotating drums.
9. **Species:** Particles and walls each have a unique species, which is hidden for basic use of the code; however, this feature can be enabled by a single function call. Different particle properties for each species and different interaction forces for each pair of species can then be defined, allowing the simulation of mixtures.
10. **Self-test suite and demos:** MercuryDPM comes with a large number of (over 100) self-tests and demo codes. These serve two purposes: 1) they allow us to constantly test both new and old features so we can keep bugs to a minimum; 2) secondly, they serve as good example, for new users, of how to perform different tasks.
11. **Simple restarting:** Every time a code is run, and at intervals during the computation, restart files are generated. Codes can be restarted without recompilation simply by calling the executable again with the restart file name as an argument. Also the restart files are complete in the sense they contain all the information about the problem. Therefore small changes can be made (e.g. the individual particle density or coefficient of restitution) and the simulation can be rerun without the need for recompilation of the code.
12. **Visualisation:** The particles output can be visualised easily using the free package VMD (visual molecular dynamics, http://www.ks.uiuc.edu/Research/vmd/) as well as the in-house visualisation tool xballs.

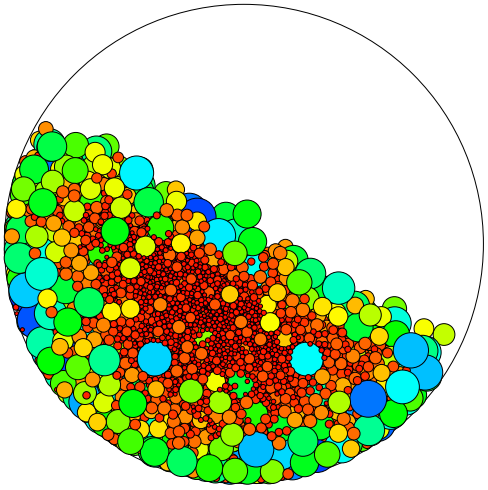

Figure 1: Poly-dispersed segregation in a rotating drum. Colour denotes particle size.

13. **Parallel:** There is currently a parallel-distributed version of the code under development using MPI and this version should be publicly available shortly.

INTRODUCTION TO THE HIERARCHAL GRID

Traditionally, particle simulations codes use a linked list system for contact detection [1, 5]. The method divides the domain into small cells whose size equals the diameter of the largest particle and exploits the fact that all possible contact can only be from particles in neighbouring cells. This method has a complexity of order N for mono-dispersed flows, where N is the total number of particles. This means that if you double the number of particles you are simulating the total computational time doubles. However, for poly-dispersed flows this nice scaling is lost and in the extreme limit of one very large particle and the rest small particles, the complexity becomes order N^2. Due the hierarchical grid contact detection algorithm that forms MercuryDPM's heart this problem is avoided within MercuryDPM. Therefore highly poly-dispersed and wide-size distributions can be easily tackled for the first time, in an open-source environment.

The hierarchal grid consists of a number of regular grids with different cell sizes. The grids are ordered such that the cell size in each level is smaller than the level above (note, the grid cell size in higher levels are not necessarily integer multiples of the lowest grid cell size). The algorithm consist of two phases: building the grid (mapping) and contact detection.

In the building phase every particles is placed on the highest grid level, that the cell size is below the particle's diameter. Note, the grid is not rebuilt every time step, but only when enough time-steps have passed that it is possible for a particle to have fully moved out of its original cell.

The contact detection phase has two steps. Firstly, contacts within the currently level are checked for using the classical linked list methods. Next the second cross level search step, where each particle potential contacts at levels lower than the level of insertion are checked for. This implies that the particle will be checked only against the smaller ones, thus avoiding double checks for the same pair of particles. The net effect of this is a vast reduction in the number of checks undertaken. Details of the algorithm and the speed up gained for poly-dispersed size-distributions is discussed in more detail in [8] and a consideration for bi-dispersed in [12].

Here, we demonstrate the power of the hierarchal grid using the example of poly-dispersed particles in a rotating drum. One of the key reasons that poly-dispersed flow has not been investigated in the past is the computational cost. Figure 1 shows an example of one of these poly-dispersed simulations. In this simulation every particle is of a different size, with a uniform volume distribution. The colour represents the size of the particles, with red the smallest and blue the largest. The ratio of the smallest to largest particle is ten to one in this simulation. The image is taken after two revolutions of the drum and a strong segregation pattern can be observed with the small particles located in the centre of the drum.

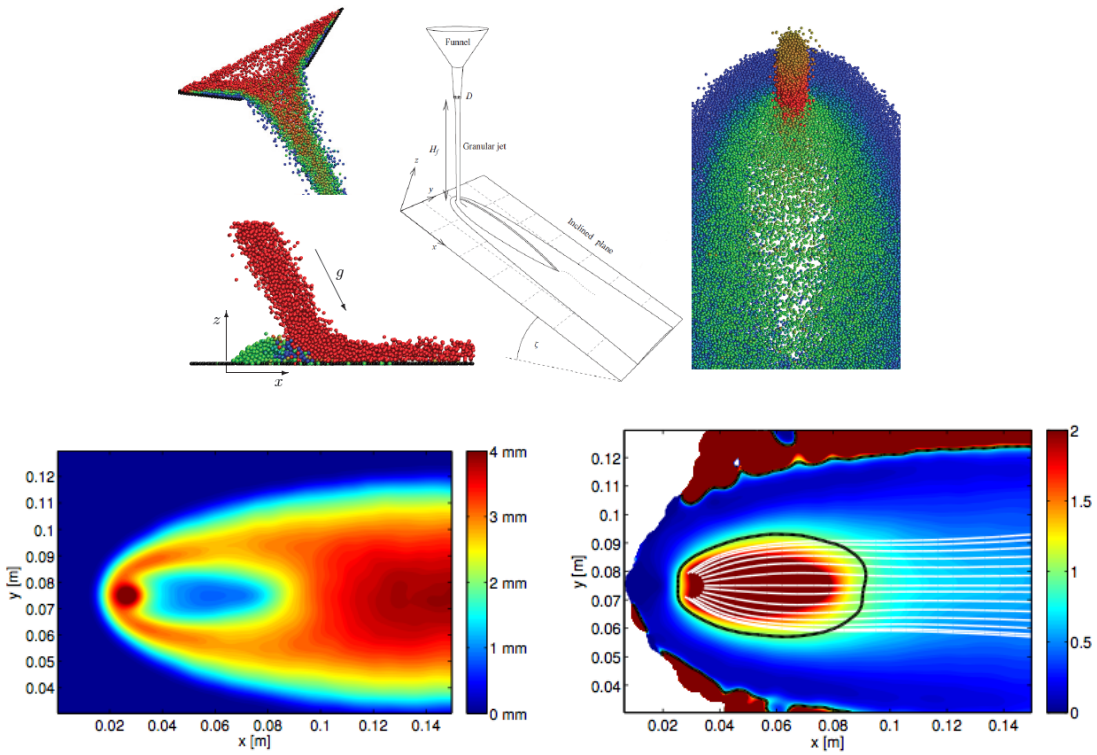

Figure 2: Top : Left shows the flow in the hopper, bottom left the impact region, middle of schematic of the original experiment taken from [6], right the top view of the full particle simulation (≈500k particles). Black particles indicate fixed particles, all other colours indicate speed, with blue low and red high speed. Bottom : Course grained macroscopic fields creating using the MercuryDPM coarse-graining packaged. Left shows the height of the flow in millimetres and right the local Froude number of the flow, the white lines indicate velocity streamlines. The black line indicates the location of a hydraulic jump/shock.

INTRODUCTION TO COARSE GRAINING

To obtain macroscopic fields, we use the coarse-graining statistical methods as described in [2, 4], extended to incorporate external boundary forces [14]. For this statistical method a coarse-graining function, $\mathcal{W}$, has to be defined that spatially smears the discrete data. MercuryDPM has many predefined coarse-graining functions and common choices are Gaussian or Lucy functions.

The method has several advantages over other methods because: (i) the fields produced automatically satisfy the equations of continuum mechanics, even near the flow base; (ii) it is neither assumed that the particles are rigid nor spherical; and, (iii) the results are even valid for single particles as no averaging over groups of particles is required. The only assumptions are that each particle pair has a single contact region i.e., the particle shapes are convex), the contact area can be replaced by a contact point i.e., the particles are not too soft), and that collisions are not instantaneous.

Here we will give a quick overview of the key ideas. Vectorial and tensorial components are denoted by Greek letters in order to distinguish them from the Latin particle indices i, j. Bold vector notation will be used when convenient. Assume a system given by N_p particles. From statistical mechanics, the microscopic mass density of the flow, ρ^{mic}, at a point $\boldsymbol{r}$ at time t is defined by

$$\rho^{\text{mic}}(\boldsymbol{r},t) = \sum_{i=1}^{N_p} m_i \delta\left(\boldsymbol{r} - \boldsymbol{r}_i(t)\right), \tag{1}$$

where $\delta(\boldsymbol{r})$ is the Dirac delta function and m_i is the mass of particle i. The following definition of the macroscopic density of the flow is used

$$\rho(\boldsymbol{r},t) = \sum_{i=1}^{N_p} m_i \mathcal{W}\left(\boldsymbol{r} - \boldsymbol{r}_i(t)\right), \tag{2}$$

thus replacing the Dirac delta function in (1) by an integrable 'coarse-graining' function $\mathcal{W}$ whose integral over space is unity.

Next we will demonstrate how to obtain other fields of interest with the simple equation of momentum vector field. The coarse grained momentum density $\boldsymbol{p}(\boldsymbol{r},t)$ is defined by

$$p_\alpha(\boldsymbol{r},t) = \sum_{i=1}^{N_p} m_i v_{i\alpha} \mathcal{W}(\boldsymbol{r}-\boldsymbol{r}_i), \tag{3}$$

where the $v_{i\alpha}$'s are the velocity components of particle i. The macroscopic velocity field $\boldsymbol{V}(\boldsymbol{r},t)$ is then defined as the ratio of momentum and density fields, $V_\alpha(\boldsymbol{r},t) = p_\alpha(\boldsymbol{r},t)/\rho(\boldsymbol{r},t)$. It is straightforward to confirm that equations (2) and (3) lead to the continuity equation

$$\frac{\partial \rho}{\partial t} + \frac{\partial p_\alpha}{\partial r_\alpha} = 0, \tag{4}$$

with the Einstein summation convention for Greek letters.

Using the coarse-gaining method it is possible, but more involved, to produce formulae for other macroscopic fields. Common fields like kinetic stress, contact stress, fabric, temperature, displacement, traction, collisional heat flux, density, flow height, etc. are already available from the coarse-graining tool box included as part of MercuryDPM. For the interested reader formulaes for the other fields can be found [2, 4, 13, 14].

We will illustrate the use of coarse-graining package via the simulation of a granular jet impacting an inclined plane using MercuryDPM. This problem was first investigated both experimentally and via the continuum approach by Johnson and Gray [6]. The problem setup, snapshots, and examples of the results of the coarse-graining package for the jet problem are shown in Figure 2. To obtain the height of the flow we assume that the density of the flow is constant over height and that the flow is steady and uniform enough to have a lithostatic stress profile. Thus, the height can be defined using the depth-averaged stress and density. Once the height is known, a depth-averaged velocity and the Froude number can be defined. A Froude number larger than unity denotes supercritical flow, otherwise the flow is subcritical. This allows us to determine the location of the shock (black line in right panel of Figure 2).

COMPLEX WALL SUPPORT

The final feature of MercuryDPM which we will illustrate in detail is the helical screw. This both highlights the flexibility of the versatile handlers, and is a common feature in many industrial apparatuses. The difficulty of these simulations lies in the interaction between the screw and the particles. The approach that is used in most similar particle simulation packages is to triangulate the screw and do collision detection between the particles and small segments of the screw. The major disadvantage of this method is that for accuracy the single screw element has to be divided into a large number of triangles. All possible combinations of these triangles with the particles have to be checked for contacts, resulting in high computational costs. To circumvent this high computational cost, in MercuryDPM, the screw is modelled as a single parametric surface.

Figure 3 shows a screw feeder simulation, where the screw is positioned in a box, with a circular tube attached at the front end. The purpose of the feeder is to push the particles from the container into the tube to possibly feed another machine. This type of setup in industry is often used to give consistent doses of cohesive powders. For industrial apparatus simulations are able to provide detailed information on the flow inside the machinery, which are difficult to obtain from experiments. With this detailed information one is able to investigate the optimisation of these processes.

IF YOU ARE INTERESTED IN MERCURY-DPM

If you would like more information about the code, this can be found at the MercuryDPM website `http://www.MercuryDPM.org/`. Alternatively, you can obtain updates and information about the code by joining the mailing list. To do so, simply send a mail to listserv@lists.utwente.nl with subject: `subscribe` and body: `MERCURY-USER <your full name>`. This is a low volume mailing list and typically you will receive no more than one e-mail a month. The code itself is available from a public svn repository and details of how to obtain and install the code can also be found on the website.

CODE DEVELOPMENT ACKNOWLEDGEMENT

MercuryDPM was started by Anthony Thornton and Thomas Weinhart, and is currently actively developed by Thomas Weinhart, Anthony Thornton and Dinant Krijgsman, with input from Stefan Luding and Onno

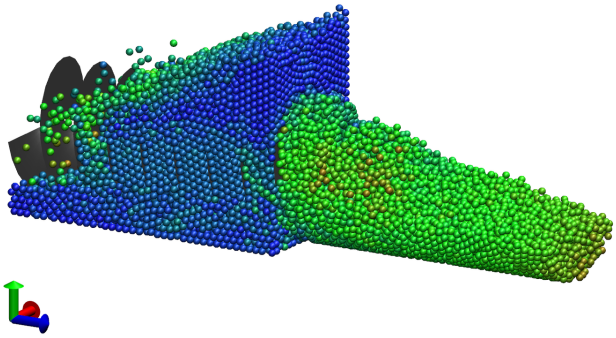

Figure 3: Snapshot of the screw feeder simulation coloured by particle velocity. The screw pushes the particles out of the box into the tube. Colours indicate particle speed. Visualisation in VMD

Bokhove. Stefan Luding provided a complete working particle simulation code with a state-of-the-art set of contact models [2] that has been used as both a validation tool and as a reference guide for various features of MercuryDPM. Additionally he has provided a great amount of theoretical and technical support in the area of (advanced) contact laws for granular materials and coarse graining. Rudi Fransen developed the current support for visualising the output of the code using VMD [11]. Ate te Voortwis is currently working on the parallel distribution of MercuryDPM.

The development of MercuryDPM has benefited from financial support provided by grants obtained primarily obtained by Stefan Luding and Onno Bokhove. A full list of the grants that have (in part) supported the development is: (1) the late Institute of Mechanics, Processes and Control, Twente (IMPACT) as part of the research program Superdispersed multiphase flows; (2) STW project 11039.STW (3) NWO VICI grant 10828; (4) DFG project SPP1482 B12; (5) FOM project 07PGM27 (6) STW MuST project 10120. All of whom we would like to thank for the essential financial support they have provided to this project.

References

[1] M. P. Allen and D. J. Tildesley. *Computer Simulations of Liquids*. Clarendon Press, Oxford, 1989.

[2] M. Babic. Average balance equations for granular materials. *Int. J. Eng. Science*, 35(5):523–548, 1997.

[3] P. A. Cundall and O. D. L. Strack. A discrete numerical model for granular assemblies. *Geotechnique*, 29(47–65), 1979.

[4] I. Goldhirsch. Stress, stress asymmetry and couple stress: from discrete particles to continuous fields. *Granular Matter*, 12(3):239–252, 2010.

[5] R. W. Hockney and J.W. Eastwood. *Computer Simulation Using Particles*. McGraw-Hill, New York,, 1981.

[6] C.G. Johnson and J.M.N.T Gray. Granular jets and hydraulic jumps on an inclined plane. *Journal of Fluid Mechanics*, 675:87–116, 2011.

[7] S. Luding. Cohesive, frictional powders: contact models for tension. *Granular Matter*, 10(4):235–246, 2008.

[8] V. Ogarko and S. Luding. A fast multilevel algorithm for contact detection of arbitrarily polydisperse objects. *Comp. Phys. Comm.*, 183:932–936, 2012.

[9] L. Pesch, A. Bell, W. E. H. Sollie, V. R. Ambati, O. Bokhove, and J. J. W. Vegt. hpgem a software framework for discontinous galerkin finite element methods. *ACM Transactions on Mathematical Software*, 33(4), 2007.

[10] A. R. Thornton, T. Weinhart, S. Luding, and O. Bokhove. Friction dependence of shallow granular flows from discrete particle simulations. *EPJ E*, 35:127., 2012.

[11] A. R. Thornton, T. Weinhart, S. Luding, and O. Bokhove. Modelling of particle size segregation: Calibration using the discrete particle method. *Int. J. Mod. Phy. C.*, 23(1240014), 2012.

[12] A. R. Thornton, T. Weinhart, V. Ogarko, and S. Luding. Multi-scale modeling of multi-component granular materials. journal computer methods in materials science. *Computer Methods in Materials Science*, 13(2):1–16, 2013.

[13] T. Weinhart, A.R. Thornton, S. Luding, and O. Bokhove. Closure relations for shallow granular flows from particle simulations. *Granular Matter*, 14:531–552, 2012.

[14] T. Weinhart, A.R. Thornton, S. Luding, and O. Bokhove. From discrete particles to continuum fields near a boundary. *Granular Matter*, 14:289–294, 2012.