Please cite this article as:
J.M. Tiscar, A. Escrig, G. Mallol, J. Boix and F.A.Gilabert
"DEM-based modelling framework for spray-dried powders in ceramic tiles industry. Part II: Solver implementation".
Powder Technology (2020), DOI: 10.1016/j.powtec.2020.08.095
Received 4 May 2020, Revised 15 August 2020, Accepted 29 August 2020, Available online 6 September 2020.

DEM-based modelling framework for spray-dried powders in ceramic tiles industry. Part II: Solver implementation.

J.M. Tiscar[a,*], A. Escrig[a], G. Mallol[a], J. Boix[a], F.A. Gilabert[b]

[a] *Technological Institute for Ceramics (ITC), Jaume I University (UJI), Castellón; Spain*
[b] *Department of Materials, Textiles and Chemical Engineering (MaTCh), Faculty of Engineering and Architecture (FEA), Ghent University (UGent), Technologiepark Zwijnaarde 46, B-9052, Zwijnaarde, Belgium*

Abstract

In the preceding Part I, a combined experimental-numerical study to characterize fine spray-dried powder used in the ceramic tile pressing process was presented. In the present Part II, the algorithm proposed by (Mazhar, 2011) to solve the numerical simulation of powder dynamics through the Discrete Element Method (DEM) was extended. In this paper, the original algorithm was adapted for efficient use on multi-core CPUs and a single GPU. In both cases, history-dependent contact models were considered. The efficiency of the algorithms was compared among them and with LIGGGHTS, a reference software package for particle simulation using DEM. The results demonstrated a higher performance of the codes developed compared to LIGGGHTS, particularly in demanding scenarios with a large number of particles (more than 1 million) of small size (median diameter in volume less than 1 mm). In particular, the CPU-based algorithm was suitable for simulating the mould filling in ceramic tiles industry.

Keywords: DEM, Parallel computing, GPU computing, multi-threading

*Corresponding author
Email address: juanmiguel.tiscar@itc.uji.es (J.M. Tiscar)

Please cite this article as:
J.M. Tiscar, A. Escrig, G. Mallol, J. Boix and F.A.Gilabert
"DEM-based modelling framework for spray-dried powders in ceramic tiles industry. Part II: Solver implementation".
Powder Technology (2020), DOI: 10.1016/j.powtec.2020.08.095
Received 4 May 2020, Revised 15 August 2020, Accepted 29 August 2020, Available online 6 September 2020.

1. Introduction

Numerical simulation has become a useful tool to optimize and design industrial equipment. Concerning powder processing, the Discrete Element Method (DEM) [1] is widely used to simulate the granular behaviour, however it still involves excessive computational efforts [2–5]. In DEM, the Newton equations for translation and rotation are solved for each particle. This approach allows the use of contact models which reproduce the physical behaviour of the particles, helps to investigate granular flow characteristics and analyses the particle tracking. Different models of contact are present in literature, such as linear [6] and Hertz [7, 8] model. DEM is applied to an extensive range of physical and engineering problems, ranging from the study of agglomerates [9] (micro-scale) to rockfall dynamics [10] (macro-scale). In the context of the simulation of industrial processes, a huge number of particles must be modelled to obtain representative results, and despite the development of techniques for reducing the number of particles [11], simulating an industrial scenario used to involve an important computational effort.

One of the most used software packages in DEM simulations is LIGGGHTS [12], which is an open source software and an improved version of LAMMPS. LAMMPS is another open source software used in the field of molecular dynamics (MD) [13], based on a contact detection algorithm that relies on a uniform grid data structure handled by means of the well-known and widely-used linked-list data structure [14]. Regarding parallel computing, LIGGGHTS uses message-passing interface (MPI) to scale across multiple processing cores and take advantage of linked lists. Although it is a good method to obtain parallel code, some researches have proposed other algorithms in the multi-threaded computing approach [15, 16].

In addition to the CPU computing, nowadays, GPU computing is also used for

Please cite this article as:
J.M. Tiscar, A. Escrig, G. Mallol, J. Boix and F.A.Gilabert
"DEM-based modelling framework for spray-dried powders in ceramic tiles industry. Part II: Solver implementation".
Powder Technology (2020), DOI: 10.1016/j.powtec.2020.08.095
Received 4 May 2020, Revised 15 August 2020, Accepted 29 August 2020, Available online 6 September 2020.

numerical simulations in different scientific fields [17–19]. Several reports have shown a good performance of GPUs versus single CPU computing [20, 21], whilst other researchers have noticed a similar performance for both multi-thread CPU and GPU computing [22].

The main differences between both computing platforms are briefly described below (refer to [23] for further description):

- Number of cores. Whilst the current CPUs consist of 4 or more cores, the GPUs can have hundreds of independent cores. However, the clock frequency of GPU cores is significantly lower than the CPU ones.

- Cache management. The single instruction multiple data (SIMD) model of a GPU sets several cores to run the same code in parallel, so GPUs do not provide a general cache for all memory accesses like CPUs. This design hides the latency of memory accesses, but increases the overhead in the inter-core communication through main memory.

- Algorithm design. The development of scientific applications on GPUs used to be more challenging and more complex than on CPUs [17]. The particular architecture of the GPUs implies using all cores and avoiding inter-core communications (limiting concurrency capabilities) to get a high performance.

There are also algorithms based on a CPU-GPU hybrid computation that demonstrate an opportunity to improve, for example, the simulations of gas-solid two-phase flows [24, 25]. In this context, Mazhar et al. [26] developed a remarkable parallel collision detection algorithm for multibody problems suitable to be carried out in two levels using CPU-GPU hybrid computing. This algorithm overcomes some limitations regarding the use of linked list, such as the process of selecting the optimal

Please cite this article as:
J.M. Tiscar, A. Escrig, G. Mallol, J. Boix and F.A.Gilabert
"DEM-based modelling framework for spray-dried powders in ceramic tiles industry. Part II: Solver implementation".
Powder Technology (2020), DOI: 10.1016/j.powtec.2020.08.095
Received 4 May 2020, Revised 15 August 2020, Accepted 29 August 2020, Available online 6 September 2020.

cell size. However, it is focused on a hybrid computing platform regardless of the physical model underlying the detection. In this work, Mazhar's detection algorithm is extended and specially modified to be independently executed in a multi-core CPU or in a single GPU.

Multi-thread computing is usually faster than the sequential one, specially in DEM simulations, but the decision whether to use a CPU or GPU code for multi-thread computing is not straightforward. The small- and medium-sized industrial powder-related sectors are sometimes reluctant to use large calculation clusters for simulation and modelling activities. At least in the ceramic tile manufacturing sector, the affordability and simplicity of the technology are a crucial factor in the modelling process. Hence the interest in being able to perform DEM simulations on non-specialized or low-cost equipment.

After conducting a literature review, it has been found that many studies simplify the computations to demonstrate the speed-up of the GPU computing versus CPU, although these simplifications could call into question the accuracy and reliability of the model. For example, Govender et al. [27] showed an important speed-up in DEM simulations via GPU computing, but using a history-independent contact model with a special mesh treatment [28]. On the other hand, Radeke et al. [21] claimed that their GPU code is able to simulate the behaviour of one million particles during one physical minute in about four days of wall-clock time. However, their code works on single-precision floating-point format, and a special treatment is needed to achieve a good accuracy [29]. Note that the GPU double-precision capability is significantly slower than the single-precision one [30]. Also the GPU code of Zheng et al. [31] exhibited an increment in efficiency of about 73 speed-up ratio compared with the CPU algorithm. However, they use a threshold value to limit the maximum number of particles in a cell and another threshold value to limit the maximum

Please cite this article as:
J.M. Tiscar, A. Escrig, G. Mallol, J. Boix and F.A.Gilabert
"DEM-based modelling framework for spray-dried powders in ceramic tiles industry. Part II: Solver implementation".
Powder Technology (2020), DOI: 10.1016/j.powtec.2020.08.095
Received 4 May 2020, Revised 15 August 2020, Accepted 29 August 2020, Available online 6 September 2020.

number of contacts of a particle. These considerations allow the use of the shared memory in GPU computing, which increases the performance, and facilitates reading and writing of contact history. As a consequence, the code is limited to simple geometries and narrow particle size distributions. Recently, Gan et al. [32] used a new algorithm [33, 34] to evaluate its performance on a single-GPU, multi-GPU and CPU configurations. They demonstrated that a single GPU outperforms CPU when simulating mono-sized particles. This also occurs for a multi-GPU computing with elliptical particles, which was industrially validated [35]. However, even though history-dependent models can be used, it requires to limit the maximum number of contacts of each particle. In fact, the use of threshold values that limit the number of contacts of each particle is still used in the latest reference dealing with GPU algorithms [36].

In response, opposing studies have also arisen that exemplify the scepticism that exists regarding the vast superiority of GPUs over CPUs in the massive parallelization of scientific computing algorithms. Lee et al. [23] showed how, with regard to scientific calculation, the enormous differences announced between the two processing units are of dubious credibility, while other studies showed that over-simplistic programming can lead to disappointing results [37]. Regarding DEM, Shigeto et al. [22] emphasized that the comparison between the use of both calculation architectures must be realistic and fair, evidencing that the enormous differences between them revealed by some researchers are far from fulfilling both principles. In addition, Washizawa et al. [38] stated that the computational speed of a practical DEM model is significantly slower to run on a GPU than on a CPU.

As it has been seen, there is a lack of clarity comparing CPU and GPU algorithms in equal conditions. In this work, an extension and modification of the Mazhar's detection algorithm is introduced in an in-house developed DEM solver. This proposal

Please cite this article as:
J.M. Tiscar, A. Escrig, G. Mallol, J. Boix and F.A.Gilabert
"DEM-based modelling framework for spray-dried powders in ceramic tiles industry. Part II: Solver implementation".
Powder Technology (2020), DOI: 10.1016/j.powtec.2020.08.095
Received 4 May 2020, Revised 15 August 2020, Accepted 29 August 2020, Available online 6 September 2020.

enables the use of complex physical models and do not require expensive nor technically complex HPC facilities to simulate large particle assemblies, but it can still exploit the multi-core features of conventional desktop or laptop computers. Furthermore, no limiting parameters are required that have to be defined manually at the beginning of the computation. Additionally, the key ideas of this proposal have been extended and integrated in two different versions of the solver, suitable for running on a multi-core CPU and a single dedicated GPU, respectively. Both solvers have been benchmarked against LIGGGHTS.

The present modified solvers still keeps the advantages of the Mazhar's algorithm but it also covers very important aspects:

(i) An efficient layout to cope with the particularities of the physics of the contact model (storage and access to the contact history, regardless of the number of contacts).

(ii) A general treatment for the boundaries, whose handling is similar to that of powder particles. Definition of the calculation domain is never needed.

(iii) An efficient layout allowing for running large-scale simulations in low performance equipments with non-abundant RAM memory.

(iv) A simple layout that avoids the use of manually predefined parameters, which facilitates the usage of the solver without compromising the reliability of the results.

This paper is organized as follows: the process flow for DEM simulations is summarized in Section 2; the extension of Mazhar's algorithm for multi-core CPU and single GPU platforms are explained in Section 3; Section 4 compares the performance of the two algorithms showed in Section 3 with LIGGGHTS, and discusses their applicability; finally, a few conclusions with several final remarks are presented. The

Please cite this article as:
J.M. Tiscar, A. Escrig, G. Mallol, J. Boix and F.A.Gilabert
"DEM-based modelling framework for spray-dried powders in ceramic tiles industry. Part II: Solver implementation".
Powder Technology (2020), DOI: 10.1016/j.powtec.2020.08.095
Received 4 May 2020, Revised 15 August 2020, Accepted 29 August 2020, Available online 6 September 2020.

equipment and programming specifications used in this work are included in the Appendix A.

2. The scheme of the DEM simulations

The general DEM scheme consists in updating the kinematics and dynamics of a system constituted by the so-called "discrete elements". These elements are physical objects that can interact with each other and can represent either individual particles or any other basic geometric entity capable of interaction. The physics governing the contact between the discrete elements depends on the selected constitutive models. As presented in Part I, these models are described by a set of mathematical equations representing the material response under different loads. Although simplifications are always assumed, the main physical features must be retained in order to agree with the experimental observations. The equations of motion incorporating these physically-based contact models are numerically integrated, where the contact forces and transferred moments between elements are continuously computed over time. Therefore, DEM is a very suitable simulation method to accurately describe the complex motion of big assemblies of elements like rocks, stones, powders, granules, seeds, etc. Conversely, DEM is not suitable to study the stress and strain fields inside the elements, unless coupling with other continuous-based numerical method is considered, like the finite element method (FEM).

In the present work, the granules of the spray-dried powder are physical objects modelled as spheres. The boundaries of the setup confining the powder (e.g., rotating drum, mixer, mould feeder, conveying belt, etc.) are meshed using triangles.

Fig. 2 shows a general DEM flowchart. First, all variables are initialized, as the domain parameters, the contact model, the integrator scheme and the mesh movements, amongst others. Second, the geometry and particles are indexed in the

Please cite this article as:
J.M. Tiscar, A. Escrig, G. Mallol, J. Boix and F.A.Gilabert
"DEM-based modelling framework for spray-dried powders in ceramic tiles industry. Part II: Solver implementation".
Powder Technology (2020), DOI: 10.1016/j.powtec.2020.08.095
Received 4 May 2020, Revised 15 August 2020, Accepted 29 August 2020, Available online 6 September 2020.

simulation scope and the data structure is created. Once all parameters are prepared, if the simulation involves GPU computing, the requested information is copied to the GPU.

Subsequently, contacts between objects are detected, the forces are calculated and finally an integrator scheme is applied in order to update the particle positions. Afterwards, if these stages are executed on the GPU, the relevant information is copied back to the CPU. These operations are repeated iteratively until the final time is achieved. Furthermore, if the confining boundaries experience any type of motion, all their geometrical elements must be updated at the beginning of the next time increment. Generally speaking, the three basic stages in a DEM solver are:

(i) The collision detection algorithm that foresees and registers all the contacts between the elements.

(ii) The calculation of forces.

(iii) The numerical integration scheme that updates all positions.

Likewise, the collision detection process is typically the most demanding computational stage, followed by the force calculation and finally by the integration scheme [39]. The simplest searching algorithms to detect pair of elements under interaction usually lead to a computational complexity of $\mathcal{O}(n^2)$, meaning that the amount of time and memory scales as the number of elements is squared [40].

However, the most efficient and sequential collision detection algorithms can lead to a computational complexity of $\mathcal{O}(n)$. These algorithms are based on performing the so called "space subdivision" process, consisting in dividing the physical space in small "cells" or "bins". This process greatly reduces the amount of evaluations because only the element pairs inside neighbouring cells are checked. This subdivision can be performed by using uniform bins, quadtrees, octrees or by any other topo-

Please cite this article as:
J.M. Tiscar, A. Escrig, G. Mallol, J. Boix and F.A.Gilabert
"DEM-based modelling framework for spray-dried powders in ceramic tiles industry. Part II: Solver implementation".
Powder Technology (2020), DOI: 10.1016/j.powtec.2020.08.095
Received 4 May 2020, Revised 15 August 2020, Accepted 29 August 2020, Available online 6 September 2020.

logical arrangement. The evenly distributed bin approach is commonly used due to its good ratio performance to implementation cost. Having said that, the main differences between algorithms rely on the structure and management of the data [41]. Moreover, this subdivision is fully independent of the computational platform.

With the aim of reducing the computational cost even more, all these algorithms can be implemented taking advantage of the parallelism of modern multi-core platforms. Here it is important to remark that computational complexity $\mathcal{O}(n)$ is unaltered, as n is an invariant.

This work proposes a set of algorithms focused on reducing the computational cost by means of parallelizing the collision detection and force computation stages. The algorithms are an extension of Mazhar's algorithm for multi-core CPU and single GPU platforms. The description of how the initialization and the integration stages are parallelized, as well as the boundary movements, is omitted in this paper due to their minor impact on the final performance of the algorithms. Finally, note that this paper presents two independent algorithms for CPU and GPU computing, but not for a hybrid computing like the original proposal [26].

3. Algorithms for DEM computing

Traditionally, contact detection for DEM simulations has been performed using a two-phase approach based on a linked-list structure [14]. A linked list is a linear data structure where the elements are not stored at contiguous memory locations. It is a dynamic data structure which enables a fast insertion and deletion data, and efficient memory utilization. This structure is used by most DEM software, such as LIGGGHTS [12]. Despite its efficiency, this strategy has two major drawbacks. Firstly, all cells of the domain have to be evaluated even though they do not contain

Please cite this article as:
J.M. Tiscar, A. Escrig, G. Mallol, J. Boix and F.A.Gilabert
"DEM-based modelling framework for spray-dried powders in ceramic tiles industry. Part II: Solver implementation".
Powder Technology (2020), DOI: 10.1016/j.powtec.2020.08.095
Received 4 May 2020, Revised 15 August 2020, Accepted 29 August 2020, Available online 6 September 2020.

objects. If the domain is big, it can lead to important memory limitations. Secondly, for the contact evaluation, searching is limited to all objects in the given cell and in the neighbouring cells (2 cells in 1D, 8 cells in 2D and 26 in 3D). Therefore, the minimum size of a cell has to be the same as the largest element in the system to ensure a correct evaluation. It involves an important limitation if the elements are very different in size, because it worsens load balance on multi-core platform. Poor load balancing can be avoided by increasing the cell size, however, decreasing the number of cells might imply an increase in the order of complexity of the algorithm. Consequently, a pre-analysis step is usually required to run efficient DEM computations [42].

In order to avoid these disadvantages, in this work a strategy inspired by that used by Mazhar et al. [26] has been developed. This new proposal extends the Mazhar's strategy, describing how to track the previous contact history, required in some constitutive models, and reducing dramatically the memory usage. The latter is specially relevant to simulate big assemblies of discrete elements (>1 million of elements).

3.1. Boundary treatment

As stated above, in this work the boundaries are meshed using triangles. This triangles are, in turn, decomposed into simpler elements, also considered as discrete elements (or boundary elements). Consequently, the boundary becomes part of the data structure of the model, consisting of granule, face, edge and vertex type elements. Figure 3(a) exemplifies a mesh. Note that while all faces must be part of the data structure of the model, not all the edge and vertex elements need to take part on it. Only the outer edges and vertices are likely to be in single contact with the granules (figure 4). On the one hand, the outer edges are considered to be those that

Please cite this article as:
J.M. Tiscar, A. Escrig, G. Mallol, J. Boix and F.A.Gilabert
"DEM-based modelling framework for spray-dried powders in ceramic tiles industry. Part II: Solver implementation".
Powder Technology (2020), DOI: 10.1016/j.powtec.2020.08.095
Received 4 May 2020, Revised 15 August 2020, Accepted 29 August 2020, Available online 6 September 2020.

belong to several faces, as long as one of them is not coplanar with the others, or those that belong only to one face. On the other hand, outer vertices are only those that belong to non-collinear edges.

Since a granule can be in contact with several geometric elements at the same time, it is necessary to establish rules to avoid multiple contacts arising from the same boundary. Therefore, when detecting a contact with a boundary element, it is necessary to validate the contact to ensure its uniqueness. The checking rules are defined as follows:

1. A granule is in contact with a face element. It is validated without any additional condition.
2. A granule is in contact with an edge. It is validated only if the granule is not in contact with a face containing that edge.
3. A granule is in contact with a vertex. It is validated only if the granule is not in contact with any edge or face connected to that vertex.

The proposed methodology is inspired by the procedure established by Su et al. [43] to perform simulations of particles flow in arbitrary complex geometries. Figure 4 shows the three possible scenarios of a granule in contact with the boundary. In scenario 1, the centroid of the granule can be projected into the face and the granule-face contact is detected. In scenario 2, the centroid of the granule cannot be projected into the face, so the granule-face contact is not identified. Thus, a granule-edge contact is detected. In scenario 3, the centroid of the granule cannot be projected neither into the face nor into the edge, so that only the granule-vertex contact can be detected. The checking rules stated above must be applied for each detection.

Please cite this article as:
J.M. Tiscar, A. Escrig, G. Mallol, J. Boix and F.A.Gilabert
"DEM-based modelling framework for spray-dried powders in ceramic tiles industry. Part II: Solver implementation".
Powder Technology (2020), DOI: 10.1016/j.powtec.2020.08.095
Received 4 May 2020, Revised 15 August 2020, Accepted 29 August 2020, Available online 6 September 2020.

3.2. DEM Algorithm on a CPU

The following algorithm is based on Mazhar's algorithm [26], whose memory management is excellent. It is focused on the sorting, which is extremely fast in GPU computing, specially using the radix sort. A similar strategy can be addressed to the multi-core CPU computing introducing some variations in the original proposal.

On the other hand, the original strategy only considers sphere-sphere interactions because the domain is spherically decomposed. How to track the contact history is not discussed. In the following, a general algorithm for DEM simulations focused on multi-core CPU computing is explained together with the tracking of the contact history. The latter feature enables the use of realistic contact models, such as those which consider the tangential micro-slip [44] during the contact.

Stage 1. Indexation. This stage begins by identifying all the intersections produced between the elements and the cells surrounding them. Figure 3(b) shows a geometry and the cells surrounding that geometry, where only the cells intersecting those elements are indexed. The size of the cells is defined as twice the mean diameter of all the granular elements.

For each element, the maximum and minimum points that delimit its position in the domain are determined by means of a bounding box and the cells owning those points are identified. Figure 5 shows an example of the bounding box and the bounding points of a granular and edge element.

The number of cells (N_{cells}) intercepting any granular element is given by:

$$N_{cells} = (I_{max} - I_{min} + 1)(J_{max} - J_{min} + 1)(K_{max} - K_{min} + 1) \tag{1}$$

where $\{I_{min},\ J_{min},\ K_{min}\}$ and $\{I_{max},\ J_{max},\ K_{max}\}$ are the cell coordinates of the bounding points of that element (this terna is hereinafter denoted by $Coord$). In the 2D case presented in figure 5, the granular element occupies 4 cells and the edge

Please cite this article as:
J.M. Tiscar, A. Escrig, G. Mallol, J. Boix and F.A.Gilabert
"DEM-based modelling framework for spray-dried powders in ceramic tiles industry. Part II: Solver implementation".
Powder Technology (2020), DOI: 10.1016/j.powtec.2020.08.095
Received 4 May 2020, Revised 15 August 2020, Accepted 29 August 2020, Available online 6 September 2020.

element occupies 3 cells. Following this construction, it can be noted that a vertex element only occupies 1 cell. It is important to remark that faces (or edges) are, in general, arbitrarily oriented, thus generating a bounding box that may span a big number of cells, as equation 1 states. Obviously, this situation is highly inefficient because most of the cells in the bounding box may be empty. To overcome this drawback, this work adopts the algorithms proposed by the references [45] and [46] to efficiently index the cells surrounding edges and faces, respectively. The algorithm to index the cells surrounding edges registers only the cells visited by a 3D line with endpoints on the grid. The algorithm to surround the faces by cells scans the original bounding that encloses the face and then applies a filter equation. This filter evaluates the distance between the cell and the face, wherein only the cells intersecting the face are labelled and stored.

In order to parallelise this indexation stage, the supporting arrays A1 and A2 are defined. A1 has a size equal to the number of elements, and its function is to store the number of cells occupied by each element, as figure 6 shows. Array A2 is used to save the cumulative sum of A1, wherein each entry contains the memory shift of the corresponding location in A1. Counting the number of cells occupied by each element and the creation of A2 can be done in parallel without risk of race conditions[1], since the number of elements in the simulation is known at the beginning of this stage.

Stage 2. Grouping the neighbouring contacts. The next step is to put in contiguous cells all those elements that may potentially be in contact. This is done in the same way as in Stage 1, but instead of counting the number of cells, the cell

[1]A race condition is an undesirable situation that occurs when two or more execution threads perform operations at the same time over shared resource, and the results may change depending on the order of execution.

Please cite this article as:
J.M. Tiscar, A. Escrig, G. Mallol, J. Boix and F.A.Gilabert
"DEM-based modelling framework for spray-dried powders in ceramic tiles industry. Part II: Solver implementation".
Powder Technology (2020), DOI: 10.1016/j.powtec.2020.08.095
Received 4 May 2020, Revised 15 August 2020, Accepted 29 August 2020, Available online 6 September 2020.

information is stored. For this purpose, a new array A3 is created. A3 includes the *Coord* variables that contains a given element as well as the element ID (ID_Elem). Figure 7 shows this operation, where it can be seen that thanks to the memory shifts saved by A2, this process can be fully parallelised.

Finally, A3 is sorted according to *Coord*, so that all elements occupying the same cell are arranged contiguously. It is worth mentioning that two elements occupying the same cell is a necessary condition but not sufficient to set up contact.

Stage 3. Contact identification. The main differences between the proposed algorithm and the one proposed by Mazhar start at this stage. After sorting, A3 is divided into as many chunks as threads can be simultaneously launched in the CPU. Then, each thread identifies the real contacts in the cells of its chunk.

Figure 8 exemplifies the contact identification with 3 parallel threads: all positions of A3 corresponding to the same cell are coloured in the same colour, and the ID_Elem is the number that appears in each position of the array. The different line styles correspond to the contact evaluations made by processing threads 1, 2 and 3, respectively. On the other hand, the colour of the arrows indicates the number of iterations performed by a thread for each position in A3. For example, focusing on the first chunk (thread 1), a forward evaluation with 4 iterations is performed as follows:

1. Checking contacts between elements 1-4, 1-10 and 1-41.
2. Checking contacts between elements 4-10 and 4-41.
3. Checking contact 10-41.
4. Checking contact 3-5, 3-7, 3-32 and 3-33.

It can be noted that this methodology might lead to the identification of a particular contact in different cells. Nevertheless, the contact uniqueness in the cells can be

Please cite this article as:
J.M. Tiscar, A. Escrig, G. Mallol, J. Boix and F.A.Gilabert
"DEM-based modelling framework for spray-dried powders in ceramic tiles industry. Part II: Solver implementation".
Powder Technology (2020), DOI: 10.1016/j.powtec.2020.08.095
Received 4 May 2020, Revised 15 August 2020, Accepted 29 August 2020, Available online 6 September 2020.

ensured by means of the collision centre, a concept suggested by Mazhar for the granule-granule contact. This idea is here extended to deal with the granule-vertex, granule-edge and granule-face contacts. In such cases, the centre of collision coincides with the projection of the centre of the granule on the corresponding element.

Stage 4. Forces and moments calculation. This stage can be performed efficiently and without race conditions by incorporating a critical section[2] in the code after the contact checking.

Algorithm 1 exemplifies this procedure for two granular elements, although it is also applicable to evaluate the contact between other element types. In Lines 3-5 the forces and moments are computed. These magnitudes are subsequently added to each element in Lines 14, 15, 18 and 19. The procedure is free from race conditions by means of a locking mechanism, as showed in Lines 13, 16, 17 and 20. Once the elements in contact are known, the application of the physical model is direct as long as the previous contact history is not required. However, most of the physical models need to retrieve the previous contact history to calculate the current forces and moments [47].

Algorithm 2 describes how the previous contact history can be extracted if it exists. Once the contact is identified, the contact history is retrieved from a contact list that each granule is associated with. First, in Lines 7-12 the lowest indexed granular element that constitutes the contact is checked to see if it contains information from a previous contact with the second element. If so, this information is returned in Line 10. Otherwise, a new contact is created in Line 14 and appended in Line 15 to the contact list of the lowest indexed granule element. The used contacts are marked

[2]A critical section is a shared resource of a computer that can only be accessed by one execution thread at a time.

Please cite this article as:
J.M. Tiscar, A. Escrig, G. Mallol, J. Boix and F.A.Gilabert
"DEM-based modelling framework for spray-dried powders in ceramic tiles industry. Part II: Solver implementation".
Powder Technology (2020), DOI: 10.1016/j.powtec.2020.08.095
Received 4 May 2020, Revised 15 August 2020, Accepted 29 August 2020, Available online 6 September 2020.

as active in Lines 9 and 14 to track the previous history used in each iteration. Once the forces and moments are computed, the contact history is rewritten with the new information generated so that it can be retrieved in the next iteration. The process can be boosted by using a stable sorting algorithm in A3, which maintains the original relative order of the elements. This ensures that the elements are sorted not only according to the cell coordinates but also by their identifiers (figure 8). Finally, all contacts that are not marked as active are removed from the contact lists.

3.3. DEM Algorithm on a GPU

The code developed to run on a single GPU is inevitably more complex than that for the CPU due to restrictions in programming language, memory and communication management. On the one hand, the maximum amount of RAM available on a GPU is typically less than that available on a CPU. Therefore, developing an effective GPU-oriented code enforces an efficient and wise memory usage. On the other hand, while the GPU has a large number of cores (allowing massive parallelization of code), the performance of these cores is limited to run relatively simple code with a balanced workload to avoid the so-called "branch divergence" [48–50]. This happens when threads executed in a same block (or warp[3]) run through different instructions controlled by conditional statements. This situation must be avoided as non-equally conditioned branch split can unbalance the workflow.

The following describes the algorithm used for DEM computing on GPU architecture, paying attention to the differences with respect to the CPU-based scheme.

Stage 1. Indexation. After transferring all the necessary information (geometry, global values, tags, etc.) from the system to the GPU memory, this step performs

[3]A warp is a set of 32 threads within a thread block such that all the threads in a warp execute the same instruction.

Please cite this article as:
J.M. Tiscar, A. Escrig, G. Mallol, J. Boix and F.A.Gilabert
"DEM-based modelling framework for spray-dried powders in ceramic tiles industry. Part II: Solver implementation".
Powder Technology (2020), DOI: 10.1016/j.powtec.2020.08.095
Received 4 May 2020, Revised 15 August 2020, Accepted 29 August 2020, Available online 6 September 2020.

the same actions as *Stage 1* in the CPU-based architecture.

Stage 2. Grouping of neighbouring contacts. This stage is performed in the same way as *Stage 2* for the CPU architecture.

Stage 3. Contact identification. This stage starts after grouping array A3 (obtained in *Stage 2*) using a stable sorting algorithm to reduce the branch divergence in the next steps. In order to improve the understanding of the following parts of the algorithm, Figure 9 contains a flowchart of the complementary arrays created from A3 and their usage. Arrays A4, A5 and A6 facilitate the efficient GPU storage and kernel executions. Arrays A7 and A7p store the contacts, while A8, A9 and A10 make it possible to store the force and moments of each contact without the use of atomic operations. Note that atomic operations, depending on its implementation mode, might lead to significant bottlenecks in the performance of a GPU algorithm [51].

To analyse A3 in parallel, as many execution threads as cells are executed. This strategy, different from the one performed in the section 3.2, reduces the branch divergence and helps to balance the workload on the GPU cores. It requires creating an array A4, which contains the cells offsets of A3.

Figure 10 sketches the layout of A4 and algorithm 3 shows its construction via pseudocode. The construction of A4 proceeds as follows. In Line 3, A4 is defined and allocated using the same size as A3 with all values initialized to *0xffff*. This hexadecimal number is the largest unsigned integer value of 32 bits possible on x86 architectures. In Lines 9 and 10, every position of A3 is analysed by separated threads where two conditions are checked:

(i) Whether that element is the first one in its cell.

(ii) Whether that cell contains more elements.

If (i) and (ii) are fulfilled, the index of its position is assigned to the same location in

Please cite this article as:
J.M. Tiscar, A. Escrig, G. Mallol, J. Boix and F.A.Gilabert
"DEM-based modelling framework for spray-dried powders in ceramic tiles industry. Part II: Solver implementation".
Powder Technology (2020), DOI: 10.1016/j.powtec.2020.08.095
Received 4 May 2020, Revised 15 August 2020, Accepted 29 August 2020, Available online 6 September 2020.

A4 as line 11 shows. Finally, in Line 14 array A4 is sorted. This allows for keeping on the left side all values other than *0xfff* and whose value is the position of the first element of each cell in array A3 that may contain contacts. The finalized array A4 can be seen in figure 10.

The next step is to identify the contacts in parallel. First, the number of contacts in each cell is identified and counted to perform the memory allocation of the contacts history. Secondly, the number of contacts in each cell is identified again, and their history is stored in the previously allocated memory space. It is noticeable that in the CPU algorithm the contacts are only identified once, since a dynamic memory allocation can be done on-the-fly.

The counting procedure is outlined in figure 11 and described via pseudocode in algorithm 4 as follows. In Line 4 an intermediate array (A5) is defined and allocated with a size equal to the number of cells with different values of *0xfff* in A4 (hereinafter called active cells). In Line 6, as many processing threads as number of active cells in A4 are executed. Each thread counts the number of contacts of its assigned cell by brute force, as described in Lines 9-20. Finally, in Line 21 the total number of contacts of the cell is stored in A5.

Similar to the construction process of A2 from A1 (see figure 6), the array A6 is created to store the accumulated sum of A5. Once the number of contacts is known, a new array A7 is allocated, where the contact history can be written without race conditions. A7 is filled by re-identifying the contacts and using A5 and A6, similarly as performed with A3 (see figure 7).

Previous contact history can be efficiently obtained by an extra step after writing the array A7. To do that, two arrays are kept in memory: A7 for the current instant (t) and A7p for the previous instant (t-Δt). The only requirement is to know the position of every contact at both instants, as long as both exist.

Please cite this article as:
J.M. Tiscar, A. Escrig, G. Mallol, J. Boix and F.A.Gilabert
"DEM-based modelling framework for spray-dried powders in ceramic tiles industry. Part II: Solver implementation".
Powder Technology (2020), DOI: 10.1016/j.powtec.2020.08.095
Received 4 May 2020, Revised 15 August 2020, Accepted 29 August 2020, Available online 6 September 2020.

Figure 12 and algorithm 5 show the process of linking A7 to A7p with a sketch and pseudocode, respectively. In Line 5, a parallel thread is executed for each component of array A7. In Line 7, every thread has to identify whether the current contact existed at t-Δt. If so, the cell coordinates containing this contact history (*Old_Coord*) are retrieved, otherwise, a null value is returned. This checking is needed because it might happen that the cell that evaluates the contact changes with the movement of the elements. Line 8 checks if the current contact is new, in which case its history is initialized in Line 9.

If the current contact exists in A7p, *Old_Coord* is now known and the cell where the contact occurred is searched in A7p using a binary search, as shown in Line 12. The binary search, of $\mathcal{O}(\log n)$, is possible because A7 is naturally ordered by cell coordinates. After finding the cell, the contact is found using a linear search. In Lines 13-22 a backward linear search is described. Line 15 checks whether the elements in contact are those desired. If so, Line 16 updates the history in A7. If not, Line 19 reduces the array index. If after completing the backward linear search the contact history has not been found, Lines 22-30 perform a forward linear search in the same way.

Finally, note that the branch divergence can be reduced by evaluating the contacts in a staggered manner (first the granule-granule contacts, second the granule-face contacts...).

Stage 3. Forces and momentums assignment. This part also requires extra stages compared to the CPU code, due to the peculiarities of the GPU, which imply maximizing code parallelization while minimizing code concurrency. Figure 13 sketches this assignment and algorithm 6 describes the procedure via pseudocode. Arrays A8, A9 and A10 are created and allocated with a size equal to twice the size of A7 and initialized to *0xfff* as shown in Lines 4-7. In Line 8, as many processing threads as

Please cite this article as:
J.M. Tiscar, A. Escrig, G. Mallol, J. Boix and F.A.Gilabert
"DEM-based modelling framework for spray-dried powders in ceramic tiles industry. Part II: Solver implementation".
Powder Technology (2020), DOI: 10.1016/j.powtec.2020.08.095
Received 4 May 2020, Revised 15 August 2020, Accepted 29 August 2020, Available online 6 September 2020.

number of contacts are executed. In Line 9, the forces and moments of each contact are computed by the corresponding thread. For the first element in contact, its identifier, force and moment are set in the corresponding positions of A8, A9 and A10 as observed in Lines 10-12. For the second element in contact, Line 13 checks whether it is a granule element. If so, its identifier, force and moment are set in Lines 14-16.

In Line 19, A9 and A10 are sorted by key, using as key A8. Then, A9 and A10 are reduced in Line 20, and the position of the first geometrical element in A8 is checked in Line 21. After these operations, A9 and A10 contain the total forces and moments for each granule element. Finally, in Lines 22-25 the new forces and moments are transferred to the granule elements without race conditions and in parallel.

4. Performance Analysis

The algorithms previously described were implemented on both multi-core CPU and single GPU platforms, and were analysed together with the software LIGGGHTS. The equipment and programming specifications used for the analysis are described in Appendix A. The three DEM codes were compared among them to analyse the effect of number of granules and its polydispersity in their performance. The capabilities of each code were evaluated simulating a mould filling, which is a crucial stage in the ceramic tile forming process. Basically, it involves pouring in a mould, via a transportation system, a certain amount of powder (usually a spray-dried powder).

The study consisted in filling a small mould (figure 14). The system was composed of three objects: a hopper, a feeder and a mould. In total, eight different Grain Size Distributions (GSD) were tested with a median granule size in volume (d_{50}) of 3 and 0.5 mm and a geometric standard deviation (σ_{geo}) of 1, 1.2, 1.4 and 2, respectively. In the ceramic tile industry, tiles manufactured in a mould of this size are considered to be small-sized tiles, so the results obtained here can be relatively extrapolated to the

Please cite this article as:
J.M. Tiscar, A. Escrig, G. Mallol, J. Boix and F.A.Gilabert
"DEM-based modelling framework for spray-dried powders in ceramic tiles industry. Part II: Solver implementation".
Powder Technology (2020), DOI: 10.1016/j.powtec.2020.08.095
Received 4 May 2020, Revised 15 August 2020, Accepted 29 August 2020, Available online 6 September 2020.

industrial mould filling. Table 1 summarises all simulations performed in this section. Each simulation was labelled depending on its group, polydispersity and the code used. The simulations were split in two groups: group A includes the simulations with a d_{50} of 3 mm, and group B those with a d_{50} of 0.5 mm. Three DEM codes were tested: the code developed to be used in a CPU platform, the one to be used in a GPU platform and the code of LIGGGHTS executed in a CPU platform. In each group, and for each code, four simulations were executed with different levels of polydispersity. Table 1 also includes the number of granules and the timestep used in each simulation.

4.1. Simulation conditions

Figure 15 depicts the three different phases of the mould filling simulations, taking simulation A4 as example. Initially the powder is poured into the feeder while the hopper is moving backwards (figure 15(a)). Following, the feeder moves forward depositing the powder into the mould (figure 15(b)). Finally, the feeder returns to its original position (figure 15(c)).

The number of granules varied between $8.7 \cdot 10^4$ and $1.3 \cdot 10^6$, depending on the simulation. The physical model used was the Linear Spring-Dashpot (LSD) model [52]. Imposing properties that require very low time steps in integration does not make sense given the comparative nature of the analysis. However, similar values for granule density and stiffness to those calibrated in [52] were used. All properties and parameters were kept constant, independently of the code used. However, depending on the granular distribution used, the time step was revised. On the other hand, LIGGGHTS incorporates the LSD model in a different way from the codes developed [12], so in order to impose a similar behaviour in all codes, it was decided to eliminate the rolling resistance and the viscous damping. Table 2 shows the

Please cite this article as:
J.M. Tiscar, A. Escrig, G. Mallol, J. Boix and F.A.Gilabert
"DEM-based modelling framework for spray-dried powders in ceramic tiles industry. Part II: Solver implementation".
Powder Technology (2020), DOI: 10.1016/j.powtec.2020.08.095
Received 4 May 2020, Revised 15 August 2020, Accepted 29 August 2020, Available online 6 September 2020.

properties of the granular material used in all cases in this study. In addition, the default cell size of LIGGGHTS was used, which coincides with the maximum size of the largest granule.

Powder distribution was generated previously in order to fill the hopper by gravity. As a consequence, all initial conditions in the mould filling simulations contained the particles deposited on the hooper, which reduced the physical time of the simulations.

4.2. Results

Performance results conducted are described below. The performance was evaluated by measuring the total time of each simulation using the time command provided by Linux. The results are detailed in two separate groups as given in table 1, and discussed later.

4.2.1. Group A

Figure 16 compares the average elapsed time per iteration as function of the number of contacts in different simulations with powder distributions of a $d_{50} = 3$ mm. Note that large differences are observed in the number of granules used for each experiment due to the different degrees of dispersion. While for a GSD with $\sigma_{geo} = 2$ the hopper was completely filled with 88000 granules, for $\sigma_{geo} = 1$ a total number of 504000 granules was required. All plots of figure 16 are equivalent and the number of granules and its dispersion does not change the fact that the LIGGGHTS code is the one that works best, followed by the CPU and GPU code. The order of complexity of the algorithms is, for each case, linear with respect to the number of granules.

The differences observed in calculation speed of each simulation are mainly attributed to the large differences in the number of granules in each experiment. Like-

Please cite this article as:
J.M. Tiscar, A. Escrig, G. Mallol, J. Boix and F.A.Gilabert
"DEM-based modelling framework for spray-dried powders in ceramic tiles industry. Part II: Solver implementation".
Powder Technology (2020), DOI: 10.1016/j.powtec.2020.08.095
Received 4 May 2020, Revised 15 August 2020, Accepted 29 August 2020, Available online 6 September 2020.

wise, the differences between the three algorithms are reduced with the decrease in the number of granules.

Regarding to the GPU code, in figure 16(a) the maximum difference observed between the GPU code and the LIGGGHTS one is approximately 300 ms, while in figure 16(d) the difference is only about 20ms. As discussed in Section 3.3, issues such as code divergence, memory allocation, and too complex threads penalize the GPU code.

4.2.2. Group B

Figure 17 illustrates the average elapsed time per iteration for a given number of contacts in different simulations with powder distributions of $d_{50} = 0.5$ mm. These GSD are similar to the true distribution of spray-dried powder used in the ceramic tile industry [53]. The results shown here are hence of great relevance.

In these cases, the d_{50} is so small that in none of the simulations of group B it was computationally possible to fill the hopper completely, because of the time that would be required to complete the calculations (more than 10 million granules would be involved). Consequently, a number of granules around 1 million was considered sufficient to evaluate the code. This enabled all the calculations to be completed in a reasonable time. It is important to mention that the differences in the number of granules used in these simulations were negligible. This statement was confirmed by carrying out complementary performance analyses using exactly the same number of granules (1 million).

As in group A, the order of complexity of the algorithms is, for each case, linear with respect to the number of granules. Figure 17 shows that code developed for CPUs outperforms the code developed for GPUs, even more so than the LIGGGHTS code. These differences are reduced by increasing σ_{geo}. It is remarkable that the

Please cite this article as:
J.M. Tiscar, A. Escrig, G. Mallol, J. Boix and F.A.Gilabert
"DEM-based modelling framework for spray-dried powders in ceramic tiles industry. Part II: Solver implementation".
Powder Technology (2020), DOI: 10.1016/j.powtec.2020.08.095
Received 4 May 2020, Revised 15 August 2020, Accepted 29 August 2020, Available online 6 September 2020.

simulation with a value of $\sigma_{\text{geo}} = 1$ (figure 17(a)) could only be done with the CPU code, since the RAM memory needed to run the simulation on the other two codes was larger than that available. Table 3 provides the maximum RAM consumption of each simulation shown in figure 17. It highlights the limitation of using a single GPU in DEM simulation, due to the amount of RAM available. It also demonstrates the bigger amount of RAM required by the LIGGGHTS code, since equipment with which the simulations have been performed has 32GB of RAM (table A.1). Regarding the other simulations (figures 17(b), 17(c), 17(d)), it was possible to perform them without any trouble.

Memory availability is specially important in the LIGGGHTS code, since it is required to explicitly define a calculation domain and store information of all the cells present in it. This implies that the smaller the size of the cells and the larger the domain, the more memory is required. In contrast, in the CPU and GPU codes, the impact is much smaller, as the domain is not fully indexed, and the size of the cells only influences the construction of the contact identification array A3 (figure 7).

On the other hand, as σ_{geo} increases, the default cell size of LIGGGHTS increases. This involves two phenomena: (i) RAM consumption decreases. (ii) The number of contacts per cell increases, so the resolution tends to $\mathcal{O}(n^2)$. Here, the increment leads to a drastic reduction in memory consumption, and a substantial improvement in the speed of the algorithm is observed. Therefore, the impact of memory management on LIGGGHTS performance is greater than the impact of increasing σ_{geo}, even if it means an increment of the number of evaluations of possible contacts per cell.

This pattern is shown in figure 18, where the slope of the trend lines drawn in figure 17 is plotted as a function of the value of σ_{geo}. The impact of σ_{geo} on the scalability of the codes developed is very low. However, the impact on LIGGGHTS is very important.

Please cite this article as:
J.M. Tiscar, A. Escrig, G. Mallol, J. Boix and F.A.Gilabert
"DEM-based modelling framework for spray-dried powders in ceramic tiles industry. Part II: Solver implementation".
Powder Technology (2020), DOI: 10.1016/j.powtec.2020.08.095
Received 4 May 2020, Revised 15 August 2020, Accepted 29 August 2020, Available online 6 September 2020.

568 *4.2.3. Discussion*

569 Simulations performance exhibits large differences depending on the number of
570 contacts involved. While in the simulations of the group A the LIGGGHTS code
571 is superior, in the simulations of the group B the CPU and GPU codes overper-
572 form LIGGGHTS. RAM consumption seems to be decisive for LIGGGHTS code
573 performance, as observed in table 3, where the enormous difference in memory con-
574 sumption of the algorithms is evidenced. Although this indicator alone does not
575 determine the quality of the algorithm, excessive memory use can expose problems
576 in its management, such as cache conflicts [54] and fragmentation [55].

577 With respect to GPU code, the results generally show much lower memory con-
578 sumption than LIGGGHTS code, but on the order of 4 times higher than CPU code.
579 This is due to the singular data structure used in GPU code, which needs to store
580 and replicate information in memory due to the limitations of the architecture. Note
581 here the huge advantage of the simulation algorithm developed over the one used
582 by LIGGGHTS, since with the available RAM in the GPU (5Gb), the possibilities
583 of the LIGGGHTS resolution scheme in this architecture would be really restrictive.
584 In both developed codes, the reduction of memory with the increased dispersion is
585 attributable to the reduction in the size of the array A3.

586 Finally, most of the results shown in figure 17 have hysteresis, specially the
587 LIGGGHTS and GPU code. This indicates that for the same number of contacts
588 there are parts of the calculation where the algorithm performance is different. The
589 explanation for this hysteresis is shown in figure 19. This figure shows the typical
590 evolution of the number of contacts in the simulations of the group B. In all the sub-
591 figures there are two differentiated areas where the number of contacts is extremely
592 high, corresponding to the filling of the feeder and to the moment when all the gran-

Please cite this article as:
J.M. Tiscar, A. Escrig, G. Mallol, J. Boix and F.A.Gilabert
"DEM-based modelling framework for spray-dried powders in ceramic tiles industry. Part II: Solver implementation".
Powder Technology (2020), DOI: 10.1016/j.powtec.2020.08.095
Received 4 May 2020, Revised 15 August 2020, Accepted 29 August 2020, Available online 6 September 2020.

ules are on the mould. In figure 19(a) the different regions of the curve are associated with the instant of filling shown in figure 15.

The first zone (between iteration $80 \cdot 10^3$ and $300 \cdot 10^3$ for figure 19(a)) corresponds to the values of the upper part of the hysteresis in figures 17(b), 17(c) and 17(d). The second zone (between iteration $400 \cdot 10^3$ and $600 \cdot 10^3$ for the figure 19(a)) corresponds to the lower part of the hysteresis. The explanation of hysteresis may be related to memory management. The continuous increase of the number of contacts in the first zone necessarily involves a continuous memory allocation, while in the second zone the information can simply be reallocated. Therefore, in the second part of the simulation there would be no such over-cost and there is a greater probability of memory access optimization. This behaviour is more difficult to observe in the CPU code. In the first place, in this code the contacts are not stored in a single container, so the optimization in the memory access is not so evident. Secondly, the CPU code has a much lower RAM consumption than the other two, so the problems associated with memory management are less visible.

As a general summary, figure 20 shows the average calculation time per iteration as a function of the number of granules. Each algorithm and all the simulations carried out have been included with the exception of figure 19(a), whose execution was, considering our hardware, only feasible with the CPU code.

Considering all simulations, the order of complexity of the CPU and GPU codes remains linear, while the LIGGGHTS code shows a fairly exponential one. For scenarios with a low number of granules ($< 0.5 \cdot 10^6$) the LIGGGHTS code performs better than those developed. In scenarios with a higher number of granules ($> 0.5 \cdot 10^6$), the CPU code outperforms all others. GPU code is also faster than LIGGGHTS when the number of granules is greater than $0.9 \cdot 10^6$ and small ($d_{50} = 0.5\,\text{mm}$).

Finally, it is worth mentioning that the performance of the GPU code is not

Please cite this article as:
J.M. Tiscar, A. Escrig, G. Mallol, J. Boix and F.A.Gilabert
"DEM-based modelling framework for spray-dried powders in ceramic tiles industry. Part II: Solver implementation".
Powder Technology (2020), DOI: 10.1016/j.powtec.2020.08.095
Received 4 May 2020, Revised 15 August 2020, Accepted 29 August 2020, Available online 6 September 2020.

as expected. On the one hand, although the parallelization capacity of the GPU architecture is excellent, the data has to be processed carefully. A lot of intermediate stages are required to perform efficiently DEM simulations on GPU, hence the code results less clean and comprehensible than the CPU one. On the other hand, although it can be minimized, DEM simulations involve some divergence and may cause memory bank conflicts in its execution. This is the case during the heterogeneous contact evaluation (different element types in contact have different mathematical treatment, the elements to evaluate are not necessarily adjacent in memory, etc.) and the searching of the previous contact history, amongst others. Unfortunately, GPU computing has an important performance penalty in those situations [56, 57].

On the other hand, in all cases the CPU code is faster than the GPU code. This differences are more noticeable as the number of granules increases. As a last remark, it is important to note that the conclusions drawn in figure 20 are subject to the cases tested in this work, and the software and hardware conditions used. Therefore, the conclusions are not necessarily extrapolated to other scenarios (for example, to scenarios where, despite the number of granules, the number of cells in the domain is not large enough to significantly increase RAM consumption) and other computer equipment.

5. Conclusions

This paper presents two algorithms that overcome the restrictions of the traditional DEM resolution algorithm (based on a linked-list approach). The former is aimed to be used on a multi-core CPU, and the latter in a single GPU. Both algorithms make an efficient use of the available RAM on the system, and are not limited by the domain calculation size. The proposed algorithms are a modified and extended version of the algorithm proposed by Mazhar et al. [26]. Both codes have

Please cite this article as:
J.M. Tiscar, A. Escrig, G. Mallol, J. Boix and F.A.Gilabert
"DEM-based modelling framework for spray-dried powders in ceramic tiles industry. Part II: Solver implementation".
Powder Technology (2020), DOI: 10.1016/j.powtec.2020.08.095
Received 4 May 2020, Revised 15 August 2020, Accepted 29 August 2020, Available online 6 September 2020.

been compared with the reference software LIGGGHTS by means of simulating a mould filling as performed in ceramic tile manufacturing.

Regarding to the LIGGGHTS code, a direct relationship between the memory consumption of LIGGGHTS (table 3) and its performance has been observed. Indeed, the high computational cost of LIGGGHTS code in some situations is attributed to management issues that can occur when RAM memory consumption is so high. Although it has not been studied, it would be possible to increase the cell size in the LIGGGHTS code in those simulations with higher memory consumption. This would mean decreasing RAM memory consumption, although it would also imply an increase in the number of granules in each cell (with the increase in the number of evaluations that this involves). A previous study should be perform to identify the optimal cell size.

With respect to the GPU code, the following conclusions have been reached:

- RAM limitation is a significant handicap of using a single GPU for DEM simulations. This drawback might be overcome by using a multi-GPU platform or a CPU-GPU hybrid computation.

- Contact identification is much more complex to do on a GPU than on a CPU, involving extra steps and making the code much less maintainable and readable.

- Despite using a high-performance graphics card (Kepler K20c), the developed code for a single GPU performs significantly worse than the two CPU-based codes analysed. It must be highlighted that the calculations always used double-precision floating-point format and the granular system was formed by a polydisperse size assembly of particles, where contact history is accurately tracked for an arbitrary number of contacts. These aspects penalize tremendously the GPU performance.

Please cite this article as:
J.M. Tiscar, A. Escrig, G. Mallol, J. Boix and F.A.Gilabert
"DEM-based modelling framework for spray-dried powders in ceramic tiles industry. Part II: Solver implementation".
Powder Technology (2020), DOI: 10.1016/j.powtec.2020.08.095
Received 4 May 2020, Revised 15 August 2020, Accepted 29 August 2020, Available online 6 September 2020.

The conclusions reached concerning the CPU code are as follows:

- CPU code has proven to be the most versatile and balanced of the three codes.
- It is efficient for simulating both large and small quantities of granules.
- Its performance is markedly superior to the other codes when the number of granules to be simulated is high, as in industrial scenarios.
- Its low memory consumption allows to run demanding simulations in conventional computers without a high memory capacity.

According to the findings of this work, the CPU-based solver developed in this research will be used in Part III of this collection to validate and study the spray-dried powder model in a real filling system commonly used by the ceramic industry. The validation process will be performed by comparing the model predictions with the experimental results obtained from a prototype.

As final remark, it has been proven that the GPU architecture does not necessarily outperforms the CPU one when implementing DEM. This findings agree with the results of Lee et al. [23], Shigeto et al. [22] and Washizawa et al. [38]. However, Gan et al.[32] also showed that the use of a single GPU does seem adequate to simulate a maximum of about 1 million mono-sized and/or ellipsoidal particles.

In the author's opinion, the computational cost of simulating fine powders by using spheres in DEM is not necessarily lower on a single GPU than on a multi-core CPU. The extension of Mazhar's algorithm presented in this paper is a general-purpose algorithm that handles efficiently typical spray-dried powders used in ceramic tile industry. This extended algorithm enables the use of complex models that require double precision calculations and tracking of the contact history. Furthermore, the purposed code architecture does not rely on predefined threshold values

Please cite this article as:
J.M. Tiscar, A. Escrig, G. Mallol, J. Boix and F.A.Gilabert
"DEM-based modelling framework for spray-dried powders in ceramic tiles industry. Part II: Solver implementation".
Powder Technology (2020), DOI: 10.1016/j.powtec.2020.08.095
Received 4 May 2020, Revised 15 August 2020, Accepted 29 August 2020, Available online 6 September 2020.

693 to setup the optimum functioning of the solver. If a high-performance equipment is
694 not available and a solver that requires predefined values to adjust its functioning
695 is not desired, this research suggests that there are not substantial reasons to use a
696 single GPU computing platform to run powder simulations with DEM.

697 6. Acknowledgements

698 The authors of this paper wish to thank MACER S.L and the CDTI Ministry of
699 Science and Innovation of Spain, as part of the project with the reference number:
700 IDI20140989. J.M. Tiscar wishes to thank the Conselleria de Cultura, Educació i
701 Ciència de la Generalitat Valenciana for its financial support in the conducted study
702 with a PhD scholarship, through the programme VALi+D for researchers training.

703 Appendix A. Equipment and programming specifications

704 Table A.1 shows the hardware and software specifications for the simulations.
705 The total numbers of CPU parallel threads used in the simulations performed with
706 the CPU and LIGGGHTS code was 8. Furthermore, the domain in LIGGGHTS
707 was divided along the z-axis into 8 portions. This ensured that no thread was
708 idle. Regarding the GPU code, the simulations were performed launching parallel
709 thread blocks of 256 threads each of them. The number of blocks used depended
710 on the parallel threads required. All calculations were performed in double-precision
711 floating-point format.

712 The codes were written taking advantage of an object-oriented programming
713 (OOP) approach and mechanisms of Run-Time Type Information (RTTI) when pos-
714 sible. OOP is a programming language model that organizes software design around
715 data, or objects, rather than functions and logic. On the other hand, RTTI is a

Please cite this article as:
J.M. Tiscar, A. Escrig, G. Mallol, J. Boix and F.A.Gilabert
"DEM-based modelling framework for spray-dried powders in ceramic tiles industry. Part II: Solver implementation".
Powder Technology (2020), DOI: 10.1016/j.powtec.2020.08.095
Received 4 May 2020, Revised 15 August 2020, Accepted 29 August 2020, Available online 6 September 2020.

mechanism that exposes information about an object's data type at runtime. It provides a way to write cleaner code at the expense of an overhead. Unfortunately, GPU programming still has much work to do with respect to the implementation capabilities for OOP and RTTI. Although a procedural programming approach may be more appropriate to perform High Performance Computing (HPC) [58], OOP involves a natural implementation structure (easy to debug and maintain) which does not compromise excessively the efficiency and feasibility of the code.

References

[1] P. A. Cundall, O. D. L. Strack, A discrete numerical model for granular assemblies, 1979.

[2] Z. Zhang, L. Liu, Y. Yuan, A. Yu, A simulation study of the effects of dynamic variables on the packing of spheres, Powder Technology 116 (2001) 23–32.

[3] C. Bierwisch, T. Kraft, H. Riedel, M. Moseler, Die filling optimization using three-dimensional discrete element modeling, Powder Technology 196 (2009) 169–179.

[4] Y. Guo, C.-Y. Wu, K. Kafui, C. Thornton, 3D DEM/CFD analysis of size-induced segregation during die filling, Powder Technology 206 (2011) 177–188.

[5] Y. Tsunazawa, Y. Shigeto, C. Tokoro, M. Sakai, Numerical simulation of industrial die filling using the discrete element method, Chemical Engineering Science 138 (2015) 791–809.

[6] S. Luding, Collisions & Contacts between two particles, Physics of dry granular media - NATO ASI Series E350 1 (1998) 285.

Please cite this article as:
J.M. Tiscar, A. Escrig, G. Mallol, J. Boix and F.A.Gilabert
"DEM-based modelling framework for spray-dried powders in ceramic tiles industry. Part II: Solver implementation".
Powder Technology (2020), DOI: 10.1016/j.powtec.2020.08.095
Received 4 May 2020, Revised 15 August 2020, Accepted 29 August 2020, Available online 6 September 2020.

[7] H. Hertz, On the contact of elastic solids, Journal für die reine und angewandte Mathematik 92 (1881) 156–171.

[8] K. L. Johnson, Contact Mechanics, Cambridge University Press, 1985. Cambridge Books Online.

[9] S. Kozhar, M. Dosta, S. Antonyuk, S. Heinrich, U. Bröckel, DEM simulations of amorphous irregular shaped micrometer-sized titania agglomerates at compression, Advanced Powder Technology 26 (2015) 767–777.

[10] B. An, D. D. Tannant, Discrete element method contact model for dynamic simulation of inelastic rock impact, Computers & Geosciences 33 (2007) 513 – 521.

[11] M. Sakai, S. Koshizuka, Large-scale discrete element modeling in pneumatic conveying, Chemical Engineering Science 64 (2009) 533–539.

[12] CFDEM® Project, LIGGGHTS – Open-Source Discrete Element Simulations of Granular Materials Based on LAMMPS. Documentation. Versión 3.X, `https://www.cfdem.com/media/DEM/docu/Manual.html`, 2020.

[13] LAMMPS molecular dynamics package, `http://lammps.sandia.gov`, 2016.

[14] M. P. Allen, D. J. Tildesley, Computer Simulation of Liquids, Clarendon Press, 1989.

[15] J.-A. Ferrez, T. M. Liebling, High-Performance Computing and Networking: 9th International Conference, HPCN Europe 2001 Amsterdam, The Netherlands, June 25–27, 2001 Proceedings, Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 211–220.

Please cite this article as:
J.M. Tiscar, A. Escrig, G. Mallol, J. Boix and F.A.Gilabert
"DEM-based modelling framework for spray-dried powders in ceramic tiles industry. Part II: Solver implementation".
Powder Technology (2020), DOI: 10.1016/j.powtec.2020.08.095
Received 4 May 2020, Revised 15 August 2020, Accepted 29 August 2020, Available online 6 September 2020.

[16] H. Nguyen, Chapter 32. Broad-Phase Collision Detection with CUDA, Addison-Wesley Professional, first edition, 2007.

[17] R. Couturier, Designing scientific applications on GPUs, CRC Press, 2013.

[18] L. Genovese, B. Videau, D. Caliste, J.-F. Méhaut, S. Goedecker, T. Deutsch, Electronic Structure Calculations on Graphics Processing Units: From Quantum Chemistry to Condensed Matter Physics, Wiley-Blackwell, 2016.

[19] J. R. Cheng, M. Gen, Accelerating genetic algorithms with GPU computing: A selective overview, Computers & Industrial Engineering 128 (2019) 514 – 525.

[20] M. Durand, P. Marin, F. Faure, B. Raffin, DEM-based simulation of concrete structures on GPU, European Journal of Environmental and Civil Engineering 16 (2012) 1102–1114.

[21] C. A. Radeke, B. J. Glasser, J. G. Khinast, Large-scale powder mixer simulations using massively parallel GPUarchitectures, Chemical Engineering Science 65 (2010) 6435 – 6442.

[22] Y. Shigeto, M. Sakai, Parallel computing of discrete element method on multi-core processors, Particuology 9 (2011) 398–405.

[23] V. Lee, C. Kim, J. Chhugani, M. Deisher, D. Kim, et al., Debunking the 100X GPU vs. CPU myth: an evaluation of throughput computing on CPU and GPU, ACM SIGARCH Computer Architecture News 38 (2010) 451–460.

[24] M. Xu, F. Chen, X. Liu, W. Ge, J. Li, Discrete particle simulation of gas-solid two-phase flows with multi-scale CPU-GPU hybrid computation, Chemical Engineering Journal 207-208 (2012) 746 – 757. 22nd International Symposium on Chemical Reaction Engineering (ISCRE 22).

Please cite this article as:
J.M. Tiscar, A. Escrig, G. Mallol, J. Boix and F.A.Gilabert
"DEM-based modelling framework for spray-dried powders in ceramic tiles industry. Part II: Solver implementation".
Powder Technology (2020), DOI: 10.1016/j.powtec.2020.08.095
Received 4 May 2020, Revised 15 August 2020, Accepted 29 August 2020, Available online 6 September 2020.

[25] Y. He, F. Muller, A. Hassanpour, A. E. Bayly, A CPU-GPU cross-platform coupled CFD-DEM approach for complex particle-fluid flows, Chemical Engineering Science 223 (2020) 115712.

[26] H. Mazhar, T. Heyn, D. Negrut, A scalable parallel method for large collision detection problems, Multibody System Dynamics 26 (2011) 37–55.

[27] N. Govender, D. Wilke, S. Kok, A large scale discrete element framework for NVIDIA GPUs, in: NVIDIA GTC 2015, California.

[28] N. Govender, R. K. Rajamani, S. Kok, D. N. Wilke, Discrete element simulation of mill charge in 3D using the BLAZE-DEM GPU framework, Minerals Engineering 79 (2015) 152–168.

[29] K. J. Bowers, D. E. Chow, H. Xu, R. O. Dror, M. P. Eastwood, B. A. Gregersen, J. L. Klepeis, I. Kolossvary, M. A. Moraes, F. D. Sacerdoti, J. K. Salmon, Y. Shan, D. E. Shaw, Scalable algorithms for molecular dynamics simulations on commodity clusters, in: SC '06: Proceedings of the 2006 ACM/IEEE Conference on Supercomputing, pp. 43–43.

[30] NVIDIA Corporation, NVIDIA CUDA Programming Guide version 7.5, `http://docs.nvidia.com/cuda/pdf/CUDA_C_Programming_Guide.pdf`, 2015. Accessed: 2016-05-04.

[31] J. Zheng, X. An, M. Huang, GPU-based parallel algorithm for particle contact detection and its application in self-compacting concrete flow simulations, Computers and Structures 112-113 (2012) 193–204.

[32] J. Gan, Z. Zhou, A. Yu, A GPU-based DEM approach for modelling of particulate systems, Powder Technology 301 (2016) 1172 – 1182.

Please cite this article as:
J.M. Tiscar, A. Escrig, G. Mallol, J. Boix and F.A.Gilabert
"DEM-based modelling framework for spray-dried powders in ceramic tiles industry. Part II: Solver implementation".
Powder Technology (2020), DOI: 10.1016/j.powtec.2020.08.095
Received 4 May 2020, Revised 15 August 2020, Accepted 29 August 2020, Available online 6 September 2020.

[33] D. Nishiura, H. Sakaguchi, Parallel-vector algorithms for particle simulations on shared-memory multiprocessors, Journal of Computational Physics 230 (2011) 1923–1938.

[34] D. Nishiura, M. Furuichi, H. Sakaguchi, Computational performance of a smoothed particle hydrodynamics simulation for shared-memory parallel computing, Computer Physics Communications 194 (2015) 18 – 32.

[35] J. Gan, T. Evans, A. Yu, Application of GPU-DEM simulation on large-scale granular handling and processing in ironmaking related industries, Powder Technology 361 (2020) 258 – 273.

[36] S. Ji, L. Liu, High Performance Algorithm and Computing Analysis Software of DEM Based on GPU Parallel Algorithm, Springer Singapore, Singapore, pp. 211–234.

[37] J. Fang, H. Sips, L. Zhang, C. Xu, Y. Che, et al., Test-driving Intel® Xeon Phi, in: Proceedings of the 5th ACM/SPEC International Conference on Performance Engineering, ACM, pp. 137–148.

[38] T. Washizawa, Y. Nakahara, Parallel Computing of Discrete Element Method on GPU, Applied Mathematics 4 (2013) 242–247.

[39] J. K. Hahn, Realistic animation of rigid bodies, Proceedings of the 15th Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH 1988 22 (1988) 299–308.

[40] B. Mirtich, Efficient algorithms for two-phase collision detection, Practical Motion Planning in Robotics Current Approaches and Future Directions (1997) 203–223.

Please cite this article as:
J.M. Tiscar, A. Escrig, G. Mallol, J. Boix and F.A.Gilabert
"DEM-based modelling framework for spray-dried powders in ceramic tiles industry. Part II: Solver implementation".
Powder Technology (2020), DOI: 10.1016/j.powtec.2020.08.095
Received 4 May 2020, Revised 15 August 2020, Accepted 29 August 2020, Available online 6 September 2020.

[41] H. Samet, An Overview of Quadtrees, Octrees, and Related Hierarchical Data Structures, Theoretical Foundations of Computer Graphics and CAD (1988) 51–68.

[42] H. Mio, A. Shimosaka, Y. Shirakawa, J. Hidaka, Cell optimization for fast contact detection in the discrete element method algorithm, Advanced Powder Technology 18 (2007) 441 – 453.

[43] J. Su, Z. Gu, X. Y. Xu, Discrete element simulation of particle flow in arbitrarily complex geometries, Chemical Engineering Science 66 (2011) 6069–6088.

[44] A. Di Renzo, F. P. Di Maio, Comparison of contact-force models for the simulation of collisions in DEM-based granular flow codes, Chemical Engineering Science 59 (2004) 525–541.

[45] D. Cohen, Voxel traversal along a 3D line , Graphics gems IV 4 (1994) 366.

[46] F. Dachille, A. Kaufman, Incremental Triangle Voxelization, in: Proceedings of the Graphics Interface 2000 Conference, May 15-17, Montréal, Québec, Canada, pp. 205–212.

[47] B. Peng, L. Wang, M. Abbas, C. Druta, Discrete Element Method (DEM) Contact Models Applied to Pavement Simulation, Ph.D. thesis, 2014.

[48] P. Xiang, Y. Yang, H. Zhou, Warp-level divergence in gpus: Characterization, impact, and mitigation, in: 2014 IEEE 20th International Symposium on High Performance Computer Architecture (HPCA), pp. 284–295.

[49] L. Zhang, S. Quigley, A. Chan, A fast scalable implementation of the two-dimensional triangular Discrete Element Method on a GPU platform, Adv. Eng. Softw. 60-61 (2013) 70–80.

Please cite this article as:
J.M. Tiscar, A. Escrig, G. Mallol, J. Boix and F.A.Gilabert
"DEM-based modelling framework for spray-dried powders in ceramic tiles industry. Part II: Solver implementation".
Powder Technology (2020), DOI: 10.1016/j.powtec.2020.08.095
Received 4 May 2020, Revised 15 August 2020, Accepted 29 August 2020, Available online 6 September 2020.

[50] M. Chimeh, P. Richmond, Simulating heterogeneous behaviours in complex systems on GPUs, Simul. Model. Pract. Th. (2018).

[51] T. Liu, N. Wolfe, H. Lin, C. D. Carothers, X. G. Xu, Performance study of atomic tally methods for gpu-accelerated monte carlo dose calculation, in: 20th Topical Meeting of the Radiation Protection and Shielding Division of the American Nuclear Society.

[52] J. M. Tiscar, A. Escrig, G. Mallol, J. Boix, F. A. Gilabert, DEM-based modelling framework for spray-dried powders in ceramic tiles industry. Part I: Calibration procedure, Powder Technology 356 (2019) 818–831.

[53] J. Amorós, V. Bagan, M. Orts, A. Escardino, La operación de prensado en la fabricación de pavimentos por monococción. I. Influencia de la naturaleza del polvo de prensas sobre las propiedades de las piezas en crudo, Bol. Soc. Esp. Ceram. Vidr 27 (1988) 273–282.

[54] Y. Afek, D. Dice, A. Morrison, Cache index-aware memory allocation, ACM SIGPLAN Notices 46 (2011) 55–64.

[55] C. Walls, Dynamic Memory Allocation and Fragmentation in C & C++, in: Embedded World 2018, Nuremberg.

[56] P. Xiang, Y. Yang, H. Zhou, Warp-level divergence in gpus: Characterization, impact, and mitigation, in: High Performance Computer Architecture (HPCA), 2014 IEEE 20th International Symposium on, IEEE, pp. 284–295.

[57] A. R. Brodtkorb, T. R. Hagen, M. L. Sætra, Graphics processing unit (gpu) programming strategies and trends in gpu computing, Journal of Parallel and Distributed Computing 73 (2013) 4 – 13. Metaheuristics on GPUs.

Please cite this article as:
J.M. Tiscar, A. Escrig, G. Mallol, J. Boix and F.A.Gilabert
"DEM-based modelling framework for spray-dried powders in ceramic tiles industry. Part II: Solver implementation".
Powder Technology (2020), DOI: 10.1016/j.powtec.2020.08.095
Received 4 May 2020, Revised 15 August 2020, Accepted 29 August 2020, Available online 6 September 2020.

[58] R. Balevičius, a. Džiugys, R. Kačianauskas, a. Maknickas, K. Vislavičius, Investigation of performance of programming approaches and languages used for numerical simulation of granular material by the discrete element method, Computer Physics Communications 175 (2006) 404–415.

Please cite this article as:
J.M. Tiscar, A. Escrig, G. Mallol, J. Boix and F.A.Gilabert
"DEM-based modelling framework for spray-dried powders in ceramic tiles industry. Part II: Solver implementation".
Powder Technology (2020), DOI: 10.1016/j.powtec.2020.08.095
Received 4 May 2020, Revised 15 August 2020, Accepted 29 August 2020, Available online 6 September 2020.

Tables

Table 1. Total of simulations in the performance analysis.

Label			GSD		Number of granules	$\Delta t(\mu s)$	Group
CPU	GPU	LIGGGHTS	d_{50}(mm)	σ_{geo}			
A1-CPU	A1-GPU	A1-LIGGGHTS	3	1	504000	50	A
A2-CPU	A2-GPU	A2-LIGGGHTS	3	1.2	356000	50	
A3-CPU	A3-GPU	A3-LIGGGHTS	3	1.4	224000	50	
A4-CPU	A4-GPU	A4-LIGGGHTS	3	2	88000	50	
B1-CPU	B1-GPU	B1-LIGGGHTS	0.5	1	1316000	10	B
B2-CPU	B2-GPU	B2-LIGGGHTS	0.5	1.2	1122000	10	
B3-CPU	B3-GPU	B3-LIGGGHTS	0.5	1.4	946000	3	
B4-CPU	B4-GPU	B4-LIGGGHTS	0.5	2	1071100	3	

Please cite this article as:
J.M. Tiscar, A. Escrig, G. Mallol, J. Boix and F.A.Gilabert
"DEM-based modelling framework for spray-dried powders in ceramic tiles industry. Part II: Solver implementation".
Powder Technology (2020), DOI: 10.1016/j.powtec.2020.08.095
Received 4 May 2020, Revised 15 August 2020, Accepted 29 August 2020, Available online 6 September 2020.

Table 2. Simulation parameters in the performance analysis.

Contact **Granule-Granule**	
Granule density, ρ_{g} (kg/m^3)	2000
Normal stiffness, k_{n} (N/m)	100
Tangential stiffness, k_{s} (N/m)	100
Rolling stiffness, k_{r} (N/m)	0
Damping normal, γ_{n} (kg·m/s)	0
Damping tangential, γ_{s} (kg·m/s)	0
Sliding friction, $\mu_{\mathrm{s,g-g}}$	0.5
Rolling friction, $\mu_{\mathrm{r,g-g}}$	0
Contact **Granule-Boundary**	
Granule density, ρ_{g} (kg/m^3)	2000
Normal stiffness, k_{n} (N/m)	200
Tangential stiffness, k_{s} (N/m)	200
Rolling stiffness, k_{r} (N/m)	0
Damping normal, γ_{n} (kg·m/s)	0
Damping tangential, γ_{s} (kg·m/s)	0
Sliding friction, $\mu_{\mathrm{s,g-s}}$	0.5
Rolling friction, $\mu_{\mathrm{r,g-s}}$	0
Hopper velocity (m/s)	0.1
Feeder velocity (m/s)	0.5

Please cite this article as:
J.M. Tiscar, A. Escrig, G. Mallol, J. Boix and F.A.Gilabert
"DEM-based modelling framework for spray-dried powders in ceramic tiles industry. Part II: Solver implementation".
Powder Technology (2020), DOI: 10.1016/j.powtec.2020.08.095
Received 4 May 2020, Revised 15 August 2020, Accepted 29 August 2020, Available online 6 September 2020.

Table 3. Maximum RAM consumption (MB), in figure 17.

Group B $d_{50} = 0.5$ mm	CPU	LIGGGHTS	GPU
(a)	1070	-	-
(b)	1029	28534	4275
(c)	882	12175	3356
(d)	664	2835	2922

Please cite this article as:
J.M. Tiscar, A. Escrig, G. Mallol, J. Boix and F.A.Gilabert
"DEM-based modelling framework for spray-dried powders in ceramic tiles industry. Part II: Solver implementation".
Powder Technology (2020), DOI: 10.1016/j.powtec.2020.08.095
Received 4 May 2020, Revised 15 August 2020, Accepted 29 August 2020, Available online 6 September 2020.

Table A.1. Hardware and software specifications for the simulations.

Specifications	
Operating System	Ubuntu 14.04
Processor	Intel® Xeon® E3-1234 v3 • Frequency: 3.4Ghz • Bandwidth: 25.6Gb/s • Cores: 4 • Threads: 8
RAM	32 Gb
Graphic Card	NVIDIA® Kepler K20c • Connection: PCI Express x16 • RAM: 5Gb • Bandwidth: 208Gb/s • Processor cores: 2496
CPU code parallelization	Shared memory parallel programming • Library: Intel® TBB
Compiling tools	• Programming Language: C++ • CPU: g++-4.8.4 with flag -Ofast • GPU: nvcc-7.5 with flag -Ofast • CUDA Toolkit 7.5

Please cite this article as:
J.M. Tiscar, A. Escrig, G. Mallol, J. Boix and F.A.Gilabert
"DEM-based modelling framework for spray-dried powders in ceramic tiles industry. Part II: Solver implementation".
Powder Technology (2020), DOI: 10.1016/j.powtec.2020.08.095
Received 4 May 2020, Revised 15 August 2020, Accepted 29 August 2020, Available online 6 September 2020.

Figures

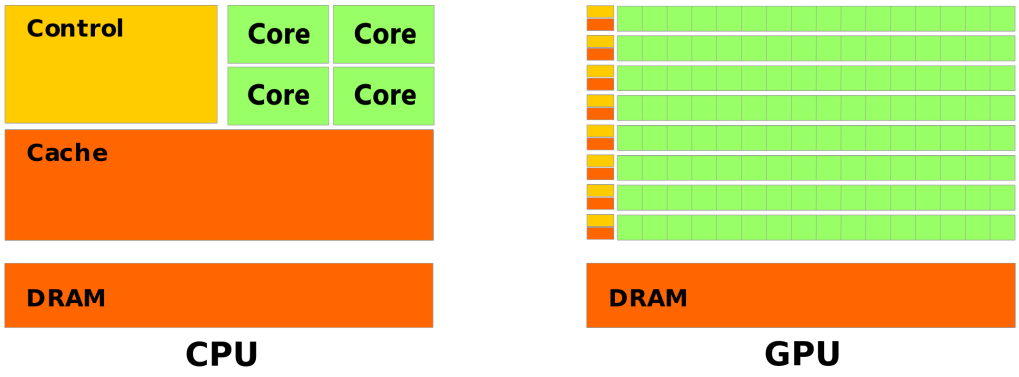


Fig. 1. Basic architecture differences between CPU and GPU.

Please cite this article as:
J.M. Tiscar, A. Escrig, G. Mallol, J. Boix and F.A.Gilabert
"DEM-based modelling framework for spray-dried powders in ceramic tiles industry. Part II: Solver implementation".
Powder Technology (2020), DOI: 10.1016/j.powtec.2020.08.095
Received 4 May 2020, Revised 15 August 2020, Accepted 29 August 2020, Available online 6 September 2020.

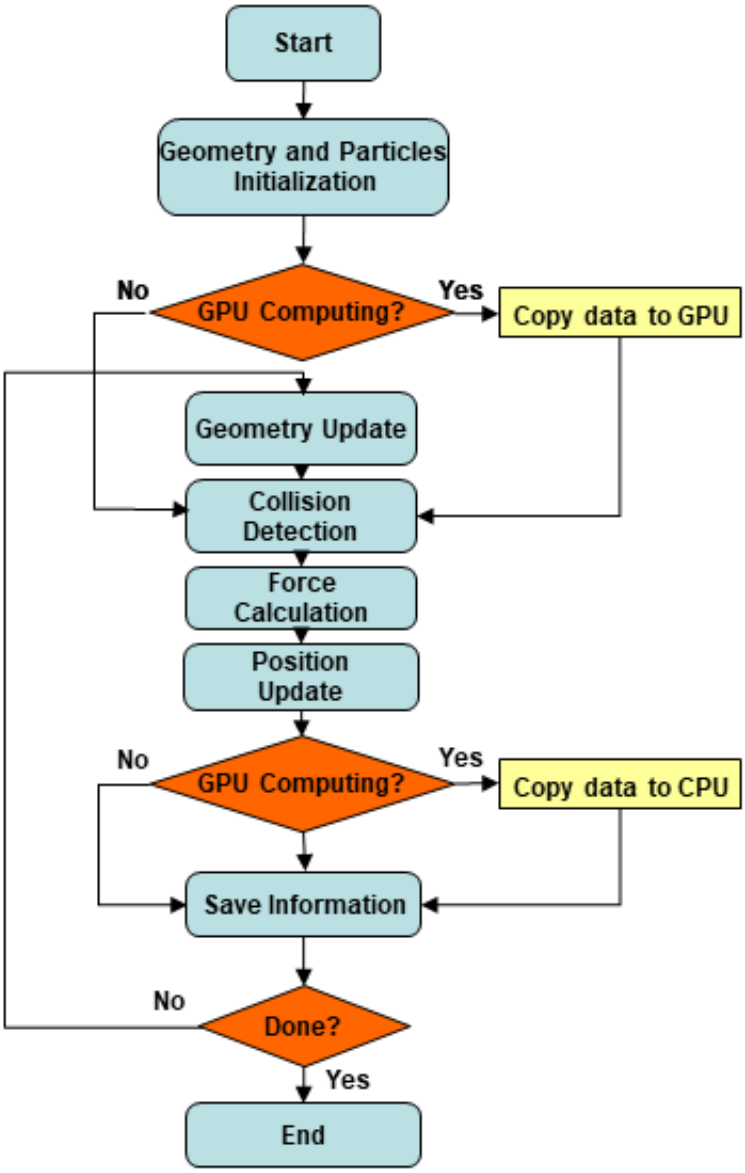
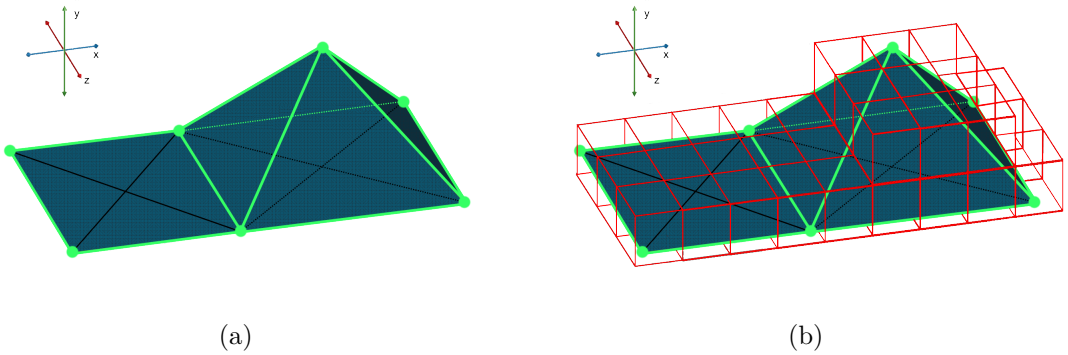


Fig. 2. The basic scheme of DEM simulations

Please cite this article as:
J.M. Tiscar, A. Escrig, G. Mallol, J. Boix and F.A.Gilabert
"DEM-based modelling framework for spray-dried powders in ceramic tiles industry. Part II: Solver implementation".
Powder Technology (2020), DOI: 10.1016/j.powtec.2020.08.095
Received 4 May 2020, Revised 15 August 2020, Accepted 29 August 2020, Available online 6 September 2020.



(a) (b)

Fig. 3. Example of a geometry composed of a triangular mesh. Although all faces are considered, only the outer edges and vertices (green) are introduced into the model. (a) without grid. (b) with grid.

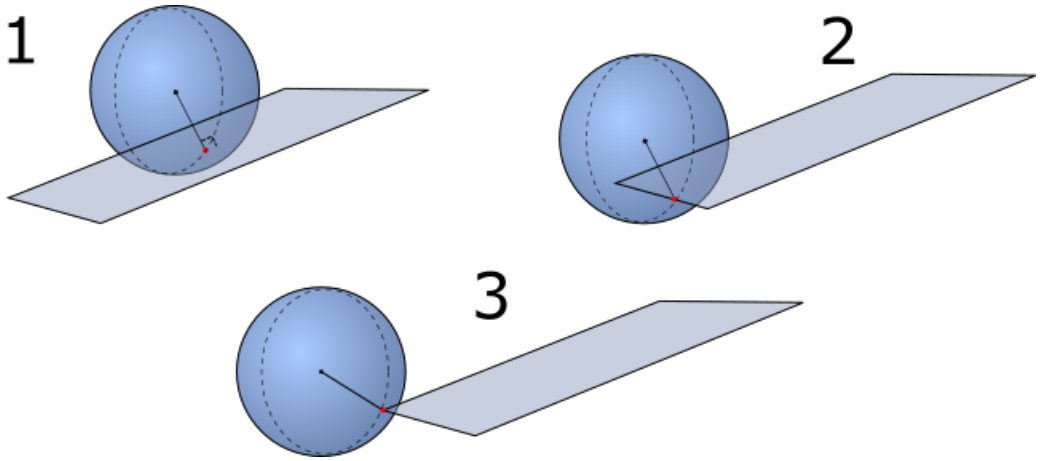


Fig. 4. Types of contact scenarios between the granules and the geometric elements.

Please cite this article as:
J.M. Tiscar, A. Escrig, G. Mallol, J. Boix and F.A.Gilabert
"DEM-based modelling framework for spray-dried powders in ceramic tiles industry. Part II: Solver implementation".
Powder Technology (2020), DOI: 10.1016/j.powtec.2020.08.095
Received 4 May 2020, Revised 15 August 2020, Accepted 29 August 2020, Available online 6 September 2020.

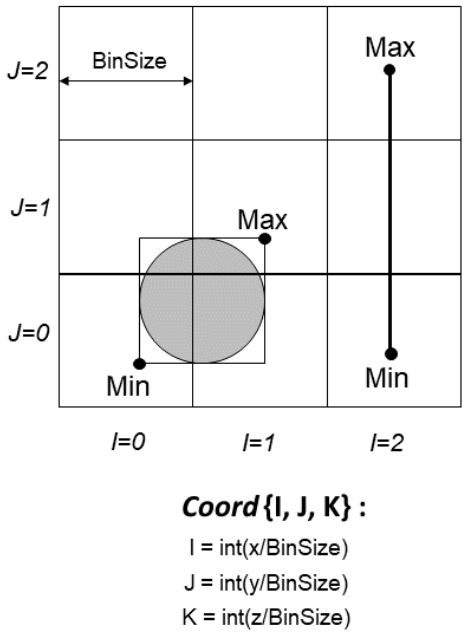


Fig. 5. Bounding box of different objects. A uniform spatial subdivision is used to normalize the coordinates.

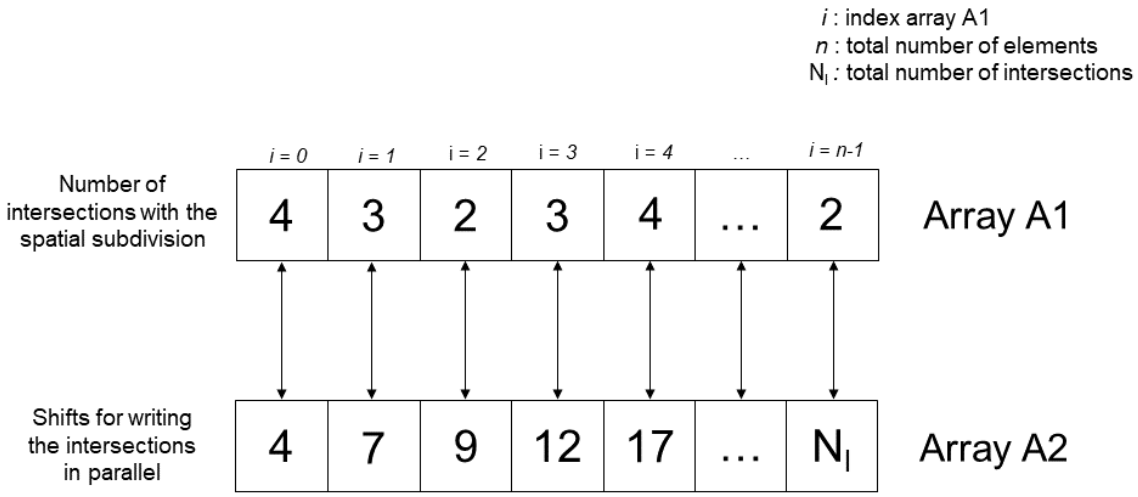


Fig. 6. Examples of Array A1 and A2. A1 results of intersection counting. A2 results of the partial summation performed on A1.

Please cite this article as:
J.M. Tiscar, A. Escrig, G. Mallol, J. Boix and F.A.Gilabert
"DEM-based modelling framework for spray-dried powders in ceramic tiles industry. Part II: Solver implementation".
Powder Technology (2020), DOI: 10.1016/j.powtec.2020.08.095
Received 4 May 2020, Revised 15 August 2020, Accepted 29 August 2020, Available online 6 September 2020.

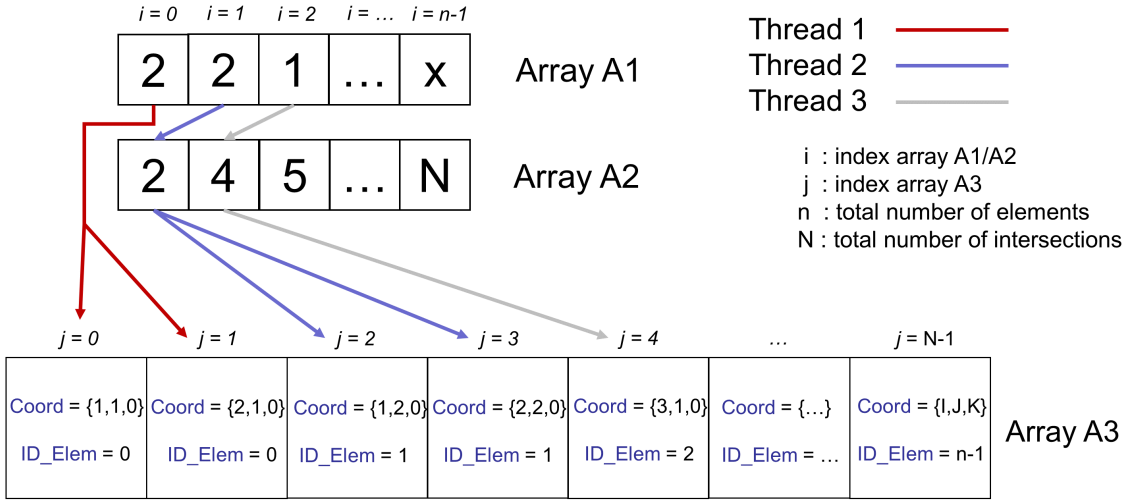


Fig. 7. Construction of array A3 in parallel, from arrays A1 and A2.

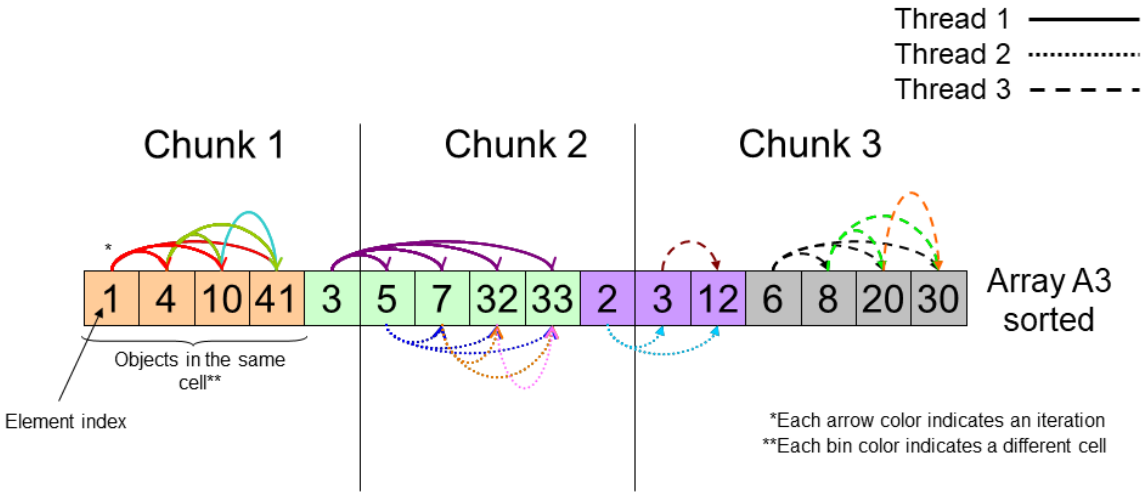


Fig. 8. (Color online) Schematic of how the contact evaluation works in array A3 sorted.

Please cite this article as:
J.M. Tiscar, A. Escrig, G. Mallol, J. Boix and F.A.Gilabert
"DEM-based modelling framework for spray-dried powders in ceramic tiles industry. Part II: Solver implementation".
Powder Technology (2020), DOI: 10.1016/j.powtec.2020.08.095
Received 4 May 2020, Revised 15 August 2020, Accepted 29 August 2020, Available online 6 September 2020.

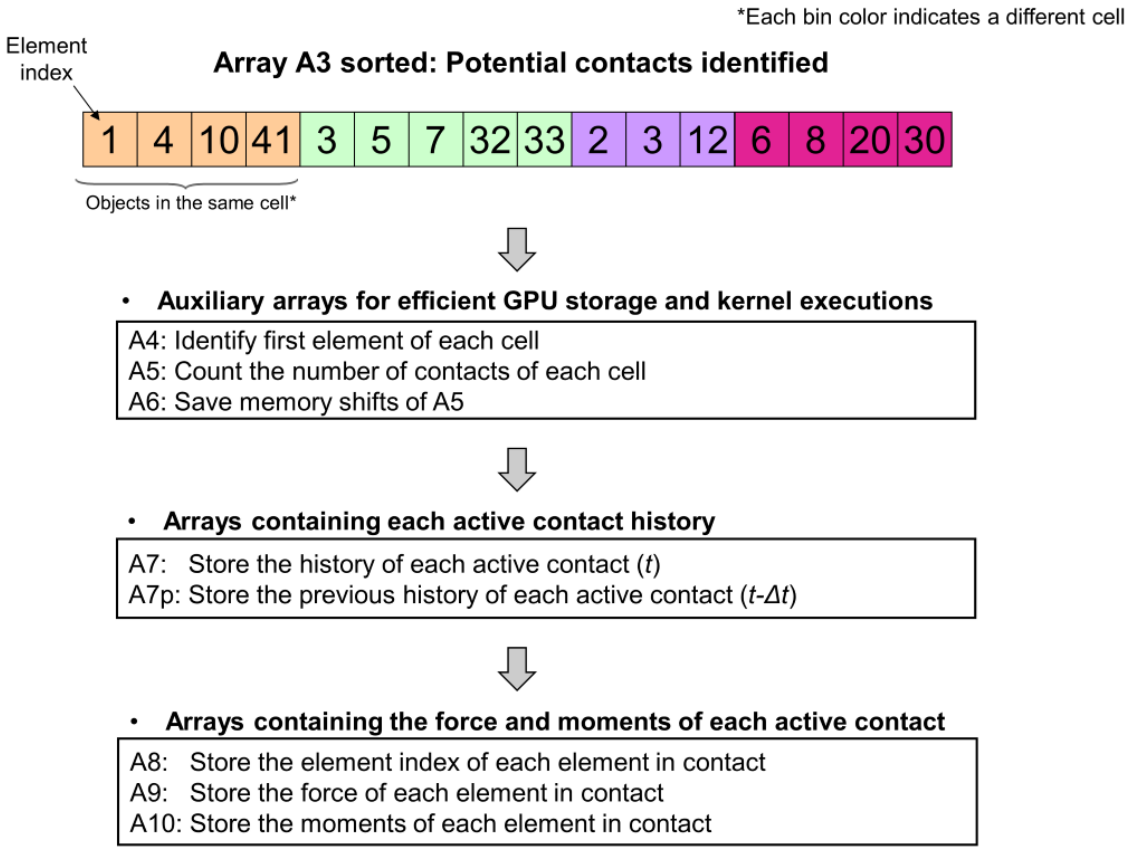


Fig. 9. Flowchart of the arrays created from A3, during the DEM algorithm on a GPU.

Please cite this article as:
J.M. Tiscar, A. Escrig, G. Mallol, J. Boix and F.A.Gilabert
"DEM-based modelling framework for spray-dried powders in ceramic tiles industry. Part II: Solver implementation".
Powder Technology (2020), DOI: 10.1016/j.powtec.2020.08.095
Received 4 May 2020, Revised 15 August 2020, Accepted 29 August 2020, Available online 6 September 2020.

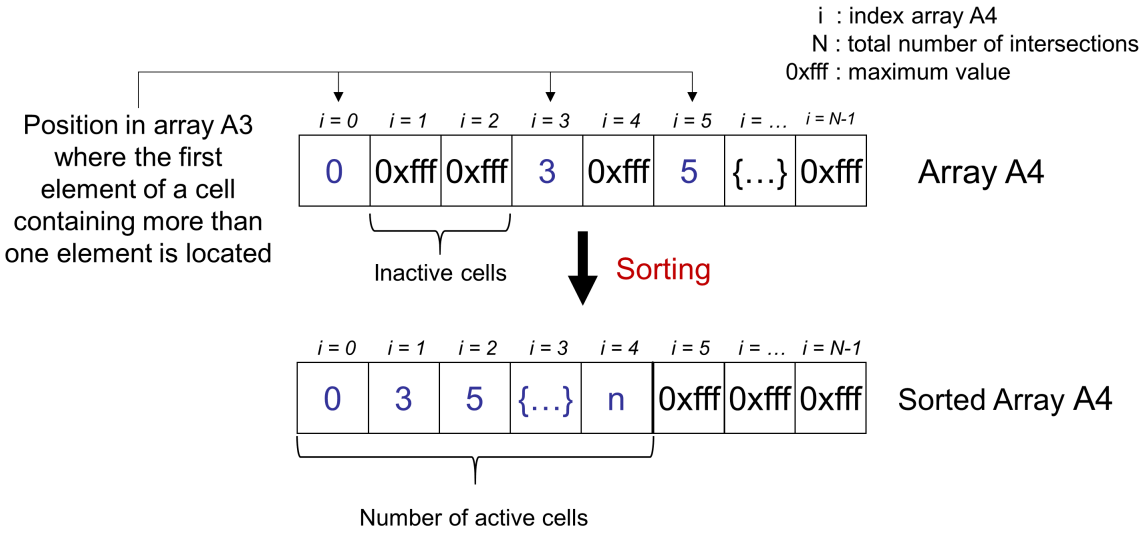


Fig. 10. Array A4 for the parallel evaluation of contacts.

Please cite this article as:
J.M. Tiscar, A. Escrig, G. Mallol, J. Boix and F.A.Gilabert
"DEM-based modelling framework for spray-dried powders in ceramic tiles industry. Part II: Solver implementation".
Powder Technology (2020), DOI: 10.1016/j.powtec.2020.08.095
Received 4 May 2020, Revised 15 August 2020, Accepted 29 August 2020, Available online 6 September 2020.

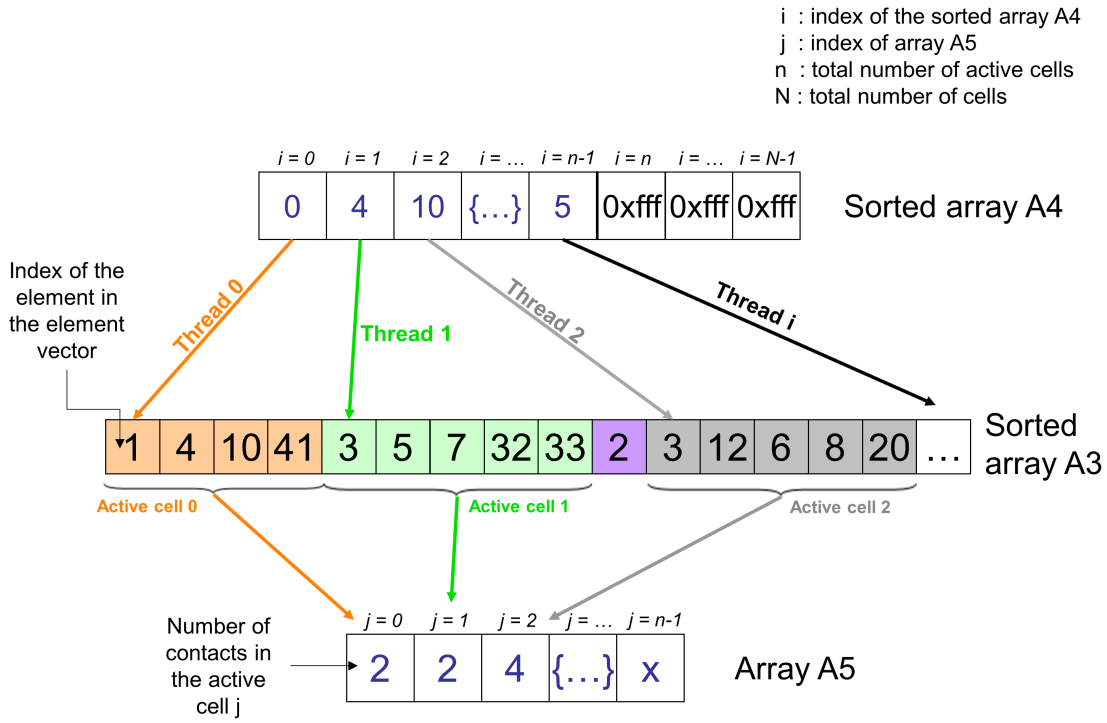


Fig. 11. Contact counting in the GPU algorithm.

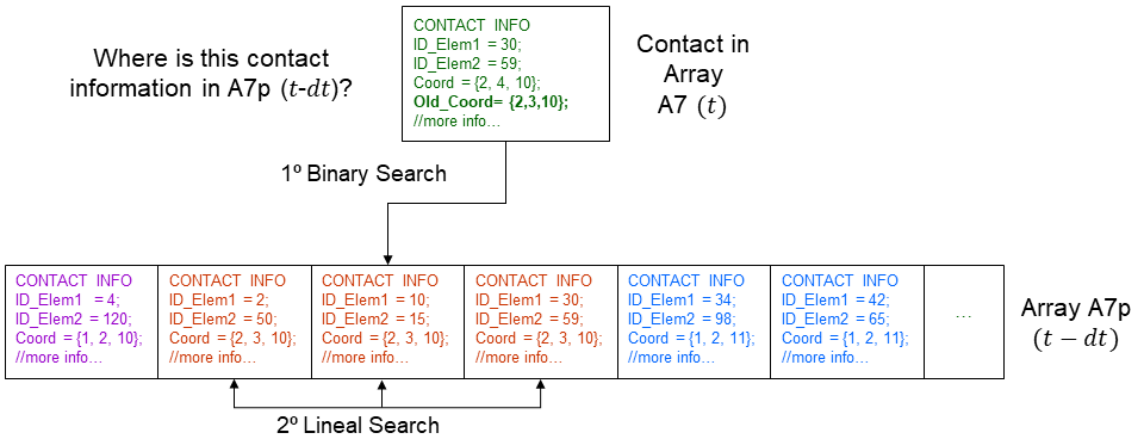


Fig. 12. Searching for a previous contact history.

Please cite this article as:
J.M. Tiscar, A. Escrig, G. Mallol, J. Boix and F.A.Gilabert
"DEM-based modelling framework for spray-dried powders in ceramic tiles industry. Part II: Solver implementation".
Powder Technology (2020), DOI: 10.1016/j.powtec.2020.08.095
Received 4 May 2020, Revised 15 August 2020, Accepted 29 August 2020, Available online 6 September 2020.

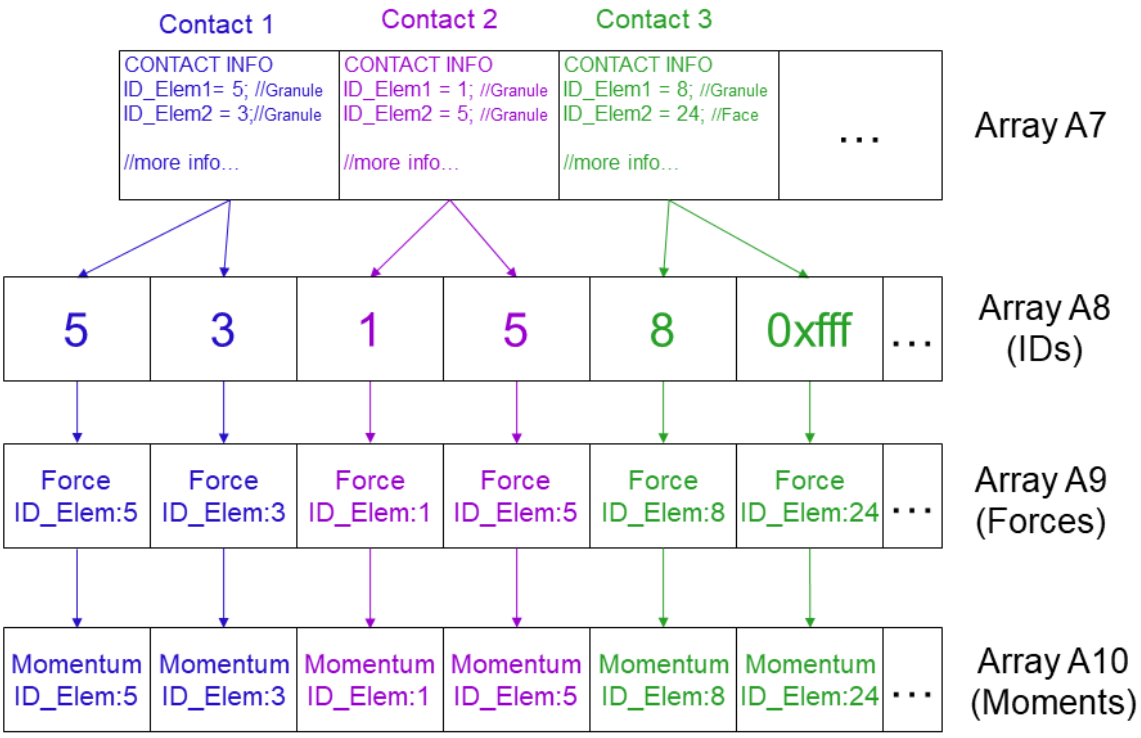


Fig. 13. Construction diagram of arrays A8, A9 and A10.

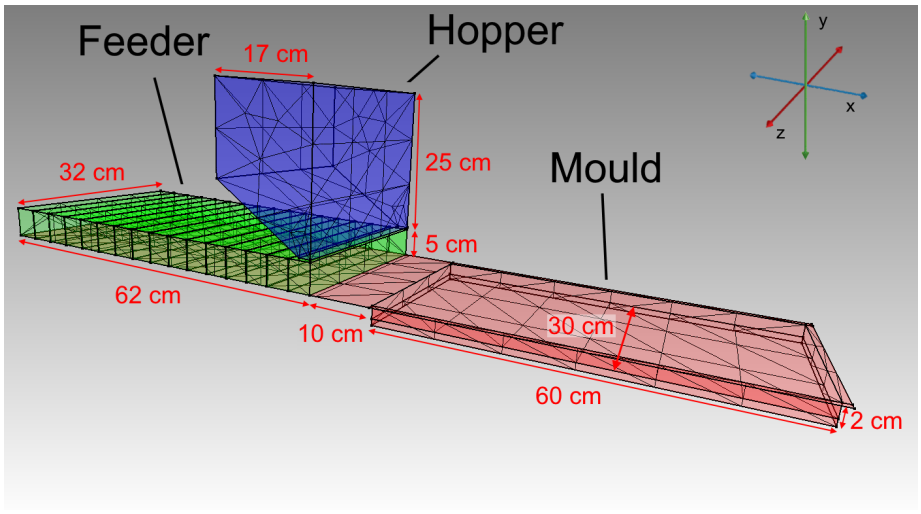


Fig. 14. Mould filling system used in the performance analysis.

Please cite this article as:
J.M. Tiscar, A. Escrig, G. Mallol, J. Boix and F.A.Gilabert
"DEM-based modelling framework for spray-dried powders in ceramic tiles industry. Part II: Solver implementation".
Powder Technology (2020), DOI: 10.1016/j.powtec.2020.08.095
Received 4 May 2020, Revised 15 August 2020, Accepted 29 August 2020, Available online 6 September 2020.

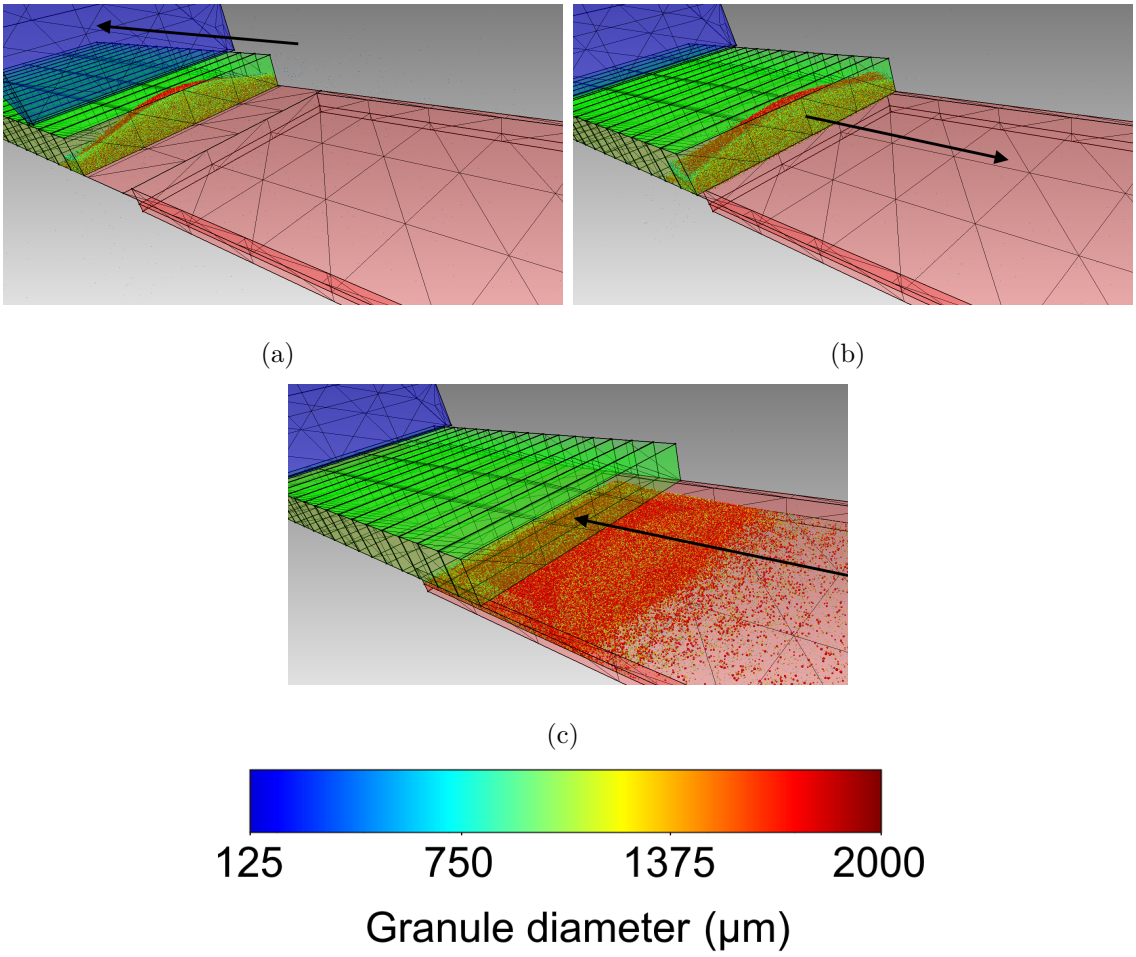
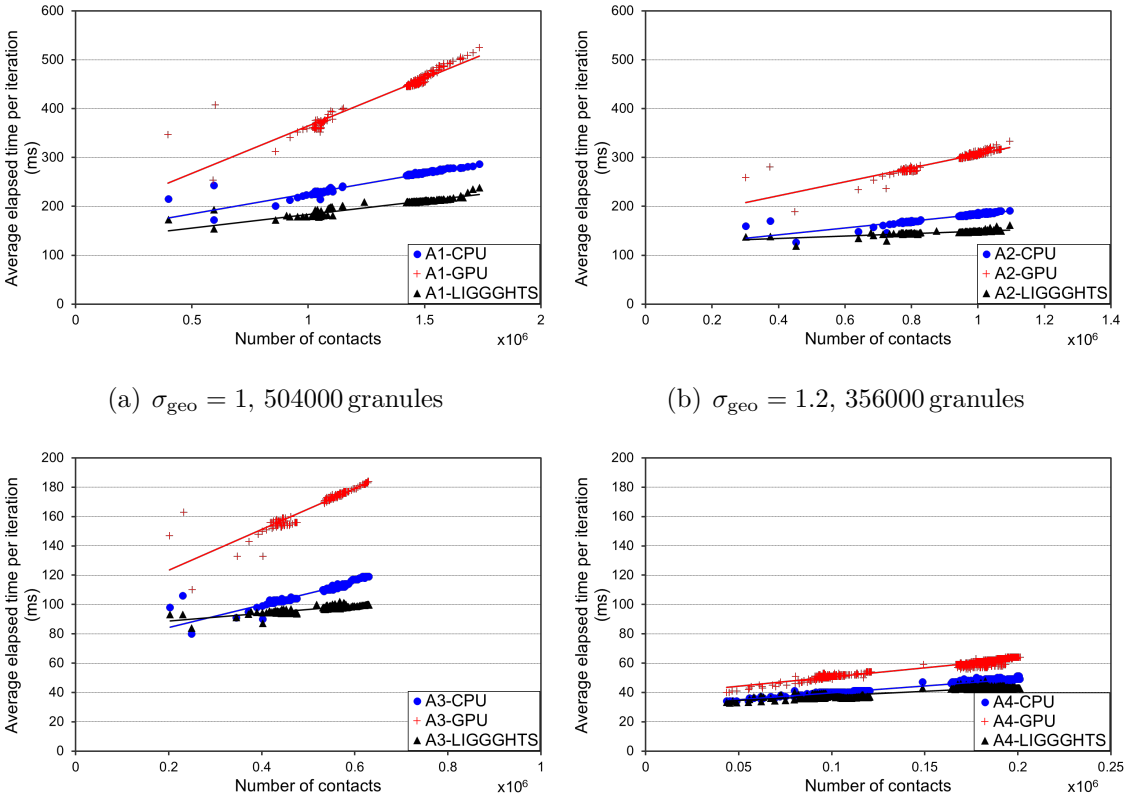


Fig. 15. Sequence of mould filling in the ceramic tile forming process. Simulation A4.

Please cite this article as:
J.M. Tiscar, A. Escrig, G. Mallol, J. Boix and F.A.Gilabert
"DEM-based modelling framework for spray-dried powders in ceramic tiles industry. Part II: Solver implementation".
Powder Technology (2020), DOI: 10.1016/j.powtec.2020.08.095
Received 4 May 2020, Revised 15 August 2020, Accepted 29 August 2020, Available online 6 September 2020.



(a) $\sigma_{\text{geo}} = 1$, 504000 granules

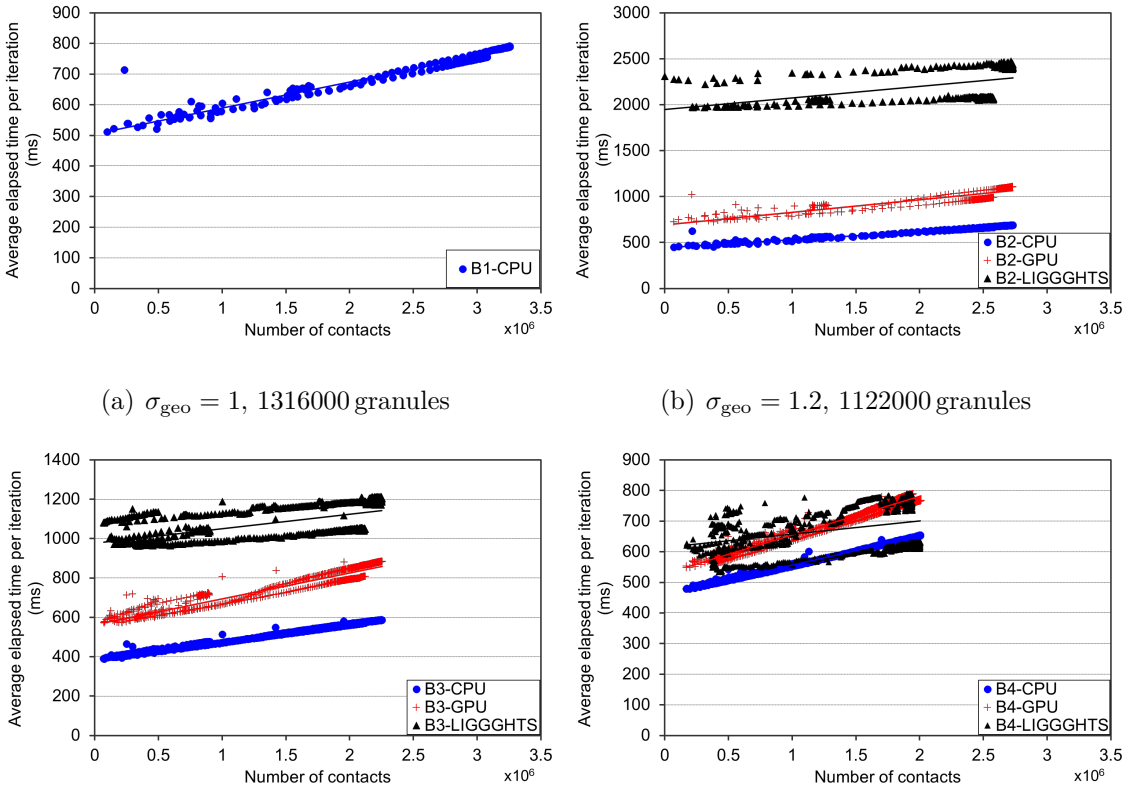
(b) $\sigma_{\text{geo}} = 1.2$, 356000 granules

(c) $\sigma_{\text{geo}} = 1.4$, 224000 granules

(d) $\sigma_{\text{geo}} = 2$, 88000 granules

Fig. 16. Performance analysis of the group A ($d_{50} = 3\,mm$).

Please cite this article as:
J.M. Tiscar, A. Escrig, G. Mallol, J. Boix and F.A.Gilabert
"DEM-based modelling framework for spray-dried powders in ceramic tiles industry. Part II: Solver implementation".
Powder Technology (2020), DOI: 10.1016/j.powtec.2020.08.095
Received 4 May 2020, Revised 15 August 2020, Accepted 29 August 2020, Available online 6 September 2020.



(a) $\sigma_{\text{geo}} = 1$, 1316000 granules

(b) $\sigma_{\text{geo}} = 1.2$, 1122000 granules

(c) $\sigma_{\text{geo}} = 1.4$, 946000 granules

(d) $\sigma_{\text{geo}} = 2$, 1071100 granules

Fig. 17. Performance analysis of the group B ($d_{50} = 0.5$ mm).

Please cite this article as:
J.M. Tiscar, A. Escrig, G. Mallol, J. Boix and F.A.Gilabert
"DEM-based modelling framework for spray-dried powders in ceramic tiles industry. Part II: Solver implementation".
Powder Technology (2020), DOI: 10.1016/j.powtec.2020.08.095
Received 4 May 2020, Revised 15 August 2020, Accepted 29 August 2020, Available online 6 September 2020.

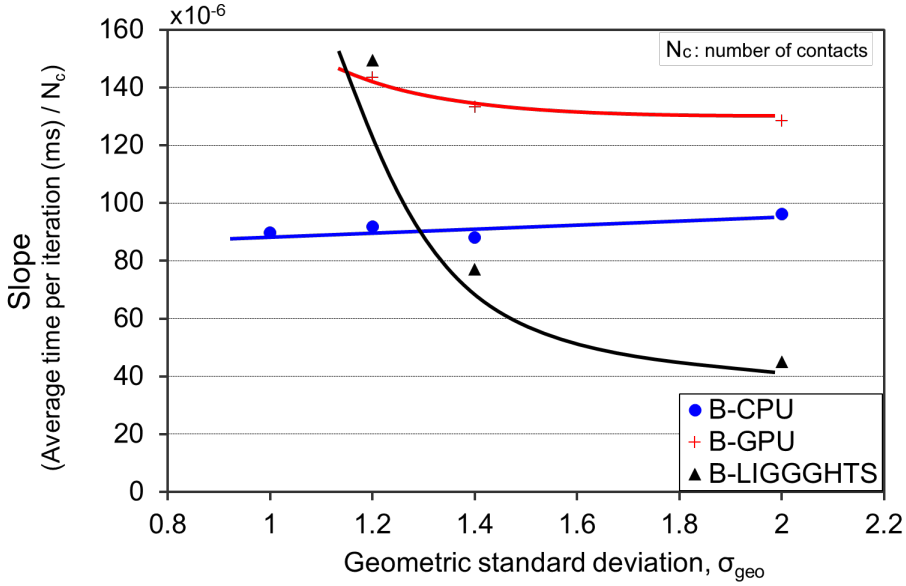
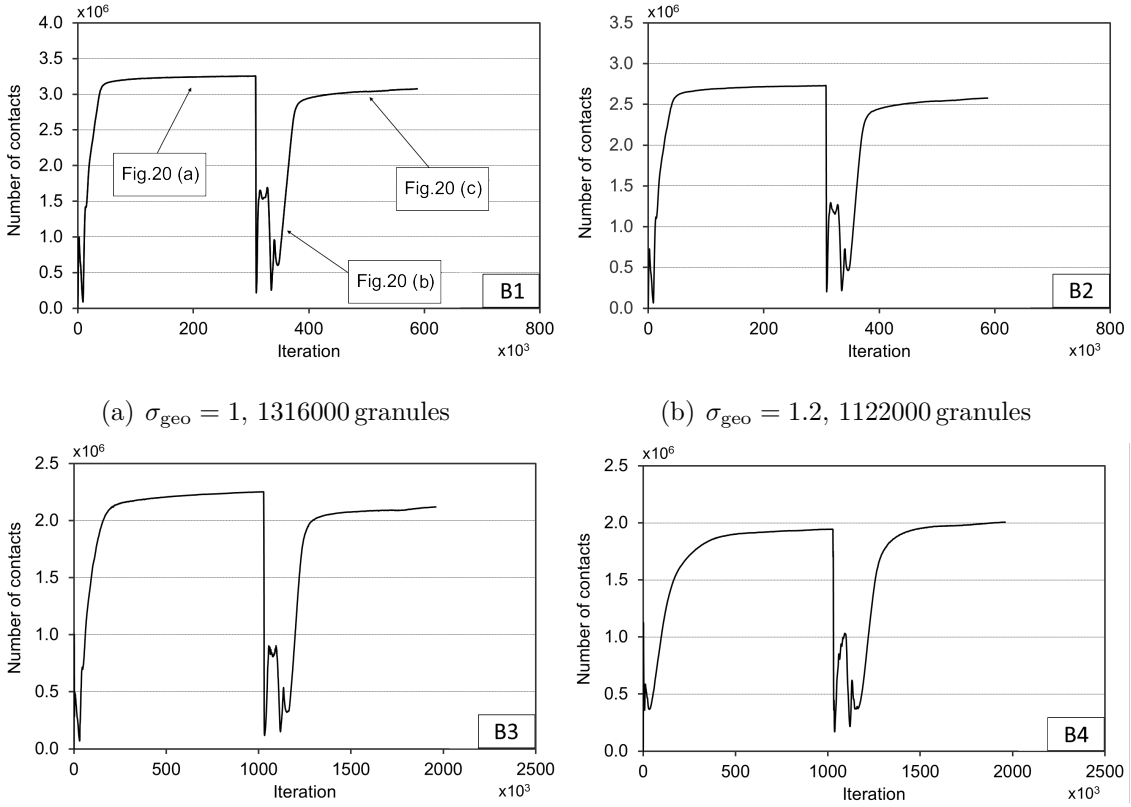


Fig. 18. The impact of σ_{geo} on the scalability based on figure 17.

Please cite this article as:
J.M. Tiscar, A. Escrig, G. Mallol, J. Boix and F.A.Gilabert
"DEM-based modelling framework for spray-dried powders in ceramic tiles industry. Part II: Solver implementation".
Powder Technology (2020), DOI: 10.1016/j.powtec.2020.08.095
Received 4 May 2020, Revised 15 August 2020, Accepted 29 August 2020, Available online 6 September 2020.



(a) $\sigma_{\text{geo}} = 1$, 1316000 granules

(b) $\sigma_{\text{geo}} = 1.2$, 1122000 granules

(c) $\sigma_{\text{geo}} = 1.4$, 946000 granules

(d) $\sigma_{\text{geo}} = 2$, 1071100 granules

Fig. 19. Evolution of the number of contacts in the group B simulations.

Please cite this article as:
J.M. Tiscar, A. Escrig, G. Mallol, J. Boix and F.A.Gilabert
"DEM-based modelling framework for spray-dried powders in ceramic tiles industry. Part II: Solver implementation".
Powder Technology (2020), DOI: 10.1016/j.powtec.2020.08.095
Received 4 May 2020, Revised 15 August 2020, Accepted 29 August 2020, Available online 6 September 2020.

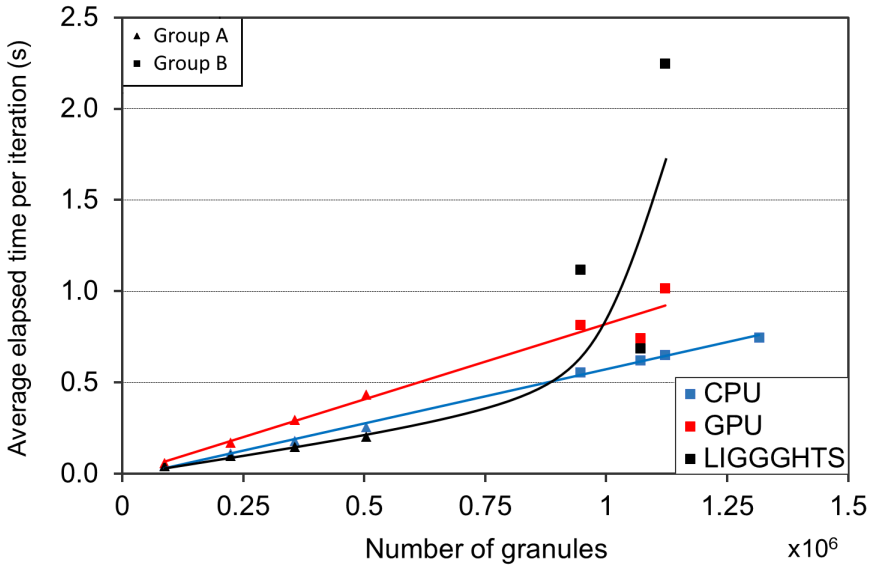


Fig. 20. Representation of the average calculation time per iteration as a function of the number of granules.

Please cite this article as:
J.M. Tiscar, A. Escrig, G. Mallol, J. Boix and F.A.Gilabert
"DEM-based modelling framework for spray-dried powders in ceramic tiles industry. Part II: Solver implementation".
Powder Technology (2020), DOI: 10.1016/j.powtec.2020.08.095
Received 4 May 2020, Revised 15 August 2020, Accepted 29 August 2020, Available online 6 September 2020.

881 # Algorithms

Please cite this article as:
J.M. Tiscar, A. Escrig, G. Mallol, J. Boix and F.A.Gilabert
"DEM-based modelling framework for spray-dried powders in ceramic tiles industry. Part II: Solver implementation".
Powder Technology (2020), DOI: 10.1016/j.powtec.2020.08.095
Received 4 May 2020, Revised 15 August 2020, Accepted 29 August 2020, Available online 6 September 2020.

Algorithm 1 Compute and assign forces and moments in parallel

Require:

G_i ▷ Granule i abstraction

G_j ▷ Granule j abstraction

Initialize:

$\boldsymbol{F} = \{F_x, F_y, F_z\}$ ▷ 3D Force

$\boldsymbol{M}_i = \{M_x, M_y, M_z\}$ ▷ 3D Moment on G_i

$\boldsymbol{M}_j = \{M_x, M_y, M_z\}$ ▷ 3D Moment on G_j

if not contact between granules **then**

    return

end if

▷ Computation depending on the constitutive model

$\boldsymbol{F} \leftarrow$ Compute force between G_i and G_j

$\boldsymbol{M}_i \leftarrow$ Compute moment on G_i

$\boldsymbol{M}_j \leftarrow$ Compute moment on G_j

{**Start critical section**}

Lock G_i ▷ Mutual exclusion event

Add $\boldsymbol{F}$ to G_i

Add $\boldsymbol{M}_i$ to G_i

Unlock G_i

Lock G_j ▷ Mutual exclusion event

Add $\boldsymbol{F}$ to G_j

Add $\boldsymbol{M}_j$ to G_j

Unlock G_j

{**End critical section**}

Please cite this article as:
J.M. Tiscar, A. Escrig, G. Mallol, J. Boix and F.A.Gilabert
"DEM-based modelling framework for spray-dried powders in ceramic tiles industry. Part II: Solver implementation".
Powder Technology (2020), DOI: 10.1016/j.powtec.2020.08.095
Received 4 May 2020, Revised 15 August 2020, Accepted 29 August 2020, Available online 6 September 2020.

Algorithm 2 Return contact history

Require:

G_i ▷ Granule i abstraction

G_j ▷ Granule j abstraction

$\boldsymbol{CList}_i$ ▷ G_i contact list

$i, j \in \mathbb{N} \,|\, i < j$

{**Start critical section**}

Lock G_i ▷ Mutual exclusion event

for each Contact in $\boldsymbol{CList}_i$ **do**

    if Contact contains G_j **then**

        Set Contact as **active**

        return Contact

    end if

end for

Initialize $\mathrm{C}_{\mathrm{new}}$ ▷ New contact history

Set $\mathrm{C}_{\mathrm{new}}$ as **active**

Append $\mathrm{C}_{\mathrm{new}}$ to $\boldsymbol{CList}_i$

return $\boldsymbol{CList}_i$[end] ▷ Unlock is auto-called

{**End critical section**}

Please cite this article as:
J.M. Tiscar, A. Escrig, G. Mallol, J. Boix and F.A.Gilabert
"DEM-based modelling framework for spray-dried powders in ceramic tiles industry. Part II: Solver implementation".
Powder Technology (2020), DOI: 10.1016/j.powtec.2020.08.095
Received 4 May 2020, Revised 15 August 2020, Accepted 29 August 2020, Available online 6 September 2020.

Algorithm 3 Construction of array A4

Require:

1: $\boldsymbol{A3}$ ▷ Array A3

2: n ▷ size of A3

Initialize:

3: $\boldsymbol{A4} \leftarrow$ **Array**(size : n)

4: Initialize all $\boldsymbol{A4}$ values to *0xfff* ▷ *0xfff* is the largest unsigned 32-bit integer on x86 architecture

5: **if** $\boldsymbol{A3}[0].Coord == \boldsymbol{A3}[1].Coord$ **then**

6: $\quad \boldsymbol{A4}[0] \leftarrow 0$

7: **end if**

8: **for in parallel:** $i \leftarrow 0$ to $n-2$ **do**

9: $\quad$ **if** !($\boldsymbol{A3}[i\text{-}1].Coord == \boldsymbol{A3}[i].Coord$) &&

10: $\quad\quad (\boldsymbol{A3}[i].Coord == \boldsymbol{A3}[i+1].Coord)$ **then**

11: $\quad\quad \boldsymbol{A4}[i] \leftarrow i$

12: $\quad$ **end if**

13: **end for**

14: Sort($\boldsymbol{A4}$) ▷ Stable sorting

Please cite this article as:
J.M. Tiscar, A. Escrig, G. Mallol, J. Boix and F.A.Gilabert
"DEM-based modelling framework for spray-dried powders in ceramic tiles industry. Part II: Solver implementation".
Powder Technology (2020), DOI: 10.1016/j.powtec.2020.08.095
Received 4 May 2020, Revised 15 August 2020, Accepted 29 August 2020, Available online 6 September 2020.

Algorithm 4 Counting Contacts

Require:

1: $\boldsymbol{A3}$ ▷ Array A3
2: $\boldsymbol{A4}$ ▷ Array A4
3: n ▷ number of active cells in A4

Initialize:

4: $\boldsymbol{A5} \leftarrow$ **Array**(size : n)
5: Initialize all $\boldsymbol{A4}$ values to *0*

6: **for in parallel:** $i \leftarrow 0$ to $n-1$ **do**
7: NContacts $\leftarrow 0$ ▷ Number of contacts
8: $j \leftarrow 0$
9: **do**
10: $k \leftarrow j+1$
11: **while** $\boldsymbol{A3}[j].Coord == \boldsymbol{A3}[k].Coord$ **do**
12: E1 $\leftarrow \boldsymbol{A3}[j].Elem$ ▷ Element in $\boldsymbol{A3}[j]$
13: E2 $\leftarrow \boldsymbol{A3}[k].Elem$ ▷ Element in $\boldsymbol{A3}[k]$
14: **if** E1 and E2 are in contact **then**
15: NContacts $\leftarrow$ NContacts + 1
16: $k++$
17: **end if**
18: **end while**
19: $j++$
20: **while** $\boldsymbol{A3}[j].Coord == \boldsymbol{A3}[j+1].Coord$
21: $\boldsymbol{A5}[i] \leftarrow$ NContacts
22: **end for**

Please cite this article as:
J.M. Tiscar, A. Escrig, G. Mallol, J. Boix and F.A.Gilabert
"DEM-based modelling framework for spray-dried powders in ceramic tiles industry. Part II: Solver implementation".
Powder Technology (2020), DOI: 10.1016/j.powtec.2020.08.095
Received 4 May 2020, Revised 15 August 2020, Accepted 29 August 2020, Available online 6 September 2020.

Algorithm 5 Update array A7 from A7p

Require:

A7 ▷ Current contact Array A7 (t)
A7p ▷ Previous contact Array A7 (t-Δt)
n ▷ size of A7
np ▷ size of A7p

for in parallel: $i \leftarrow 0$ to $n-1$ **do**
  Contact $\leftarrow$ ***A7***[i]
  Old_Coord $\leftarrow$ Cell coordinates where Contact occurs in the previous step
  if Old_Coord == **null then**
    Initialize ***A7***[i] ▷ No contact occurred
    return
  end if
  pos $\leftarrow$ **BinarySearch**(***A7p***, Old_Coord)

▷ Backward linear search

  l $\leftarrow$ pos
  do
    if ***A7***[i] and ***A7p***[l] are the same contact **then**
      Update ***A7***[i] with ***A7p***[l]
      return
    end if
    l$--$
    if lft < 0 **then break end if**
  while ***A7p***[pos].$Coord$ == ***A7p***[l].$Coord$

▷ Forward linear search

  r $\leftarrow$ pos+1
  do
    if ***A7***[i] and ***A7p***[r] are the same contact **then**
      Update ***A7***[i] with ***A7p***[r]
      return
    end if
    r++
    if r $> np-1$ **then break end if**
  while ***A7p***[pos].$Coord$ == ***A7p***[r].$Coord$
end for

Please cite this article as:
J.M. Tiscar, A. Escrig, G. Mallol, J. Boix and F.A.Gilabert
"DEM-based modelling framework for spray-dried powders in ceramic tiles industry. Part II: Solver implementation".
Powder Technology (2020), DOI: 10.1016/j.powtec.2020.08.095
Received 4 May 2020, Revised 15 August 2020, Accepted 29 August 2020, Available online 6 September 2020.

Algorithm 6 Assign forces and moments to the elements

Require:

A7 ▷ Contact Array A7

ElemLst ▷ Elements list

n ▷ size of A7

Initialize:

A8 ← **Array**(size : $2n$) ▷ Array of IDs

A9 ← **Array**(size : $2n$) ▷ Array of Forces

A10 ← **Array**(size : $2n$) ▷ Array of Moments

Initialize all **A8,A9,A10** values to *0xfff*

for in parallel: $i \leftarrow 0$ to $n-1$ **do**

  Compute force and moments generated in ***A7***$[i]$

  A8$[2i] \leftarrow$ ***A7***$[i].ID\_Elem1$

  A9$[2i] \leftarrow$ ***A7***$[i].F1$ ▷ Force in El.1

  A10$[2i] \leftarrow$ ***A7***$[i].M1$ ▷ Moment in El.1

  if ***A7***$[i].ID\_Elem2$ is a granule **then**

    A8$[2i+1] \leftarrow$ ***A7***$[i].ID\_Elem2$

    A9$[2i+1] \leftarrow$ ***A7***$[i].F2$ ▷ Force in El.2

    A10$[2i+1] \leftarrow$ ***A7***$[i].M2$ ▷ Moment in El.2

  end if

end for

SortByKey(***A8***,{***A9***,***A10***}) ▷ Sort an associative array {***A9***,***A10***}, according to the key ***A8***

ReduceByKey(***A8***,{***A9***,***A10***}) ▷ Aggregate the values of ***A9*** and ***A10*** according to the reduction function applied on ***A8***

$fpos \leftarrow$ first position in ***A9*** containing *0xfff*

for in parallel: $i \leftarrow 0$ to $fpos-1$ **do**

  ElemLst[***A8***[i]].$Force \leftarrow$ ***A9***[i]

  ElemLst[***A8***[i]].$Moment \leftarrow$ ***A10***[i]

end for

▷ Abbreviation: El., Element

Please cite this article as:
J.M. Tiscar, A. Escrig, G. Mallol, J. Boix and F.A.Gilabert
"DEM-based modelling framework for spray-dried powders in ceramic tiles industry. Part II: Solver implementation".
Powder Technology (2020), DOI: 10.1016/j.powtec.2020.08.095
Received 4 May 2020, Revised 15 August 2020, Accepted 29 August 2020, Available online 6 September 2020.

Research highlights

► A novel GPU and CPU-based DEM algorithm is developed.

► Any general-purpose contact model can be used without restrictions in the contact searching.

► The developed algorithms are fairly compared with LIGGGHTS.

►Developed CPU-based DEM algorithm outperforms LIGGGHTS in demanding filling scenarios.

►GPU-based DEM algorithm executed on a single GPU does not enhance the CPU-based algorithm.

Please cite this article as:
J.M. Tiscar, A. Escrig, G. Mallol, J. Boix and F.A.Gilabert
"DEM-based modelling framework for spray-dried powders in ceramic tiles industry. Part II: Solver implementation".
Powder Technology (2020), DOI: 10.1016/j.powtec.2020.08.095
Received 4 May 2020, Revised 15 August 2020, Accepted 29 August 2020, Available online 6 September 2020.

Credit Author Statement

J. M. Tiscar: Conceptualization, Methodology, Software, Investigation, Writing – Original Draft;

A. Escrig: Software, Validation.

G. Mallol: Funding acquisition, Resources.

J. Boix: Project administration, Supervision, Funding acquisition.

F. A. Gilabert: Conceptualization, Software, Formal analysis, Supervision, Funding acquisition, Writing – Review & Editing