PARALLEL SOLUTION TO MULTI-SCALE, MULTI-DIMENSIONAL COUPLED DEM-PBM MODEL FOR HIGH SHEAR GRANULATION USING HIGH PERFORMANCE COMPUTING

By

CHAITANYA SAMPAT

A thesis submitted to the

School of Graduate Studies

Rutgers, the state university of New Jersey

In partial fulfillment of the requirements

For the degree of

Master of Science

Graduate program in Chemical and Biochemical Engineering

Written under the direction of

Rohit Ramachandran

And Approved by

_____

_____

_____

New Brunswick, New Jersey

JANUARY, 2019

ABSTRACT OF THE THESIS

PARALLEL SOLUTION TO MULTI-SCALE, MULTI-DIMENSIONAL HIGH SHEAR GRANULATION COUPLED DEM-PBM MODEL USING HIGH PERFORMANCE COMPUTING

By CHAITANYA SAMPAT

Thesis Director:

Rohit Ramachandran

Particulate processes are prevalent in the pharmaceutical industry but, the physics underlying these processes are complicated. The particle-level models used to describe these systems require large amounts of computation to solve which makes simulations slow. Another approach to model these systems is less accurate bulk model which does not capture all the particle level data but is more efficient to simulate and requires lesser computational power. A quicker and a more accurate way to model such systems is to use a multi-scale model i.e. use the particle-scale data into a bulk model. Using a coupled model does not decrease the time taken by each individual component of the simulation, thus there is also a need to increase the speed of the separate simulations.

In this work, a unidirectional multi-scale model was used to model the high shear wet granulation process. A multi-dimensional population balance model (PBM) was developed with a mechanistic kernel, which obtained collision data from the discrete element modeling (DEM) simulation. The PBM was run in parallel using MPI + OMP hybrid technique. The DEM simulations were performed on LIGGGHTS, which runs in parallel using MPI. Speedup of about 14 times was obtained for the PBM simulations and around 12 for the DEM simulations. This coupling was performed using the radical pilot for scaling studies from 1 to 128 cores for the

PBM and up to 256 cores for the DEM. A further improvement to the PBM code was also done by developing a code in CUDA C++ such that it could utilize up to 1024 cores on the NVIDIA graphical processor units (GPU). Using this developed framework, the granulation process has been modeled accurately much faster than existing approaches in literature.

Acknowledgements

I would like to thank my advisor Dr. Rohit Ramachandran for giving me an opportunity to work in their group and particularly on this project. I would like to thank him for mentoring me and providing me support all through my graduate school journey. I would like to thank National Science Foundation (NSF) for funding this project. I would also like thank Dr. Marianthi Ierapetritou and Dr. Shantenu Jha, co-PIs on this NSF project. I would also like to acknowledge The Research in Advanced Distributed Cyberinfrastructure and Applications Laboratory (RADICAL) from the department of electrical and computer engineering, Rutgers for providing the Radical Pilot software.

I would like to thank Yukteshwar Baranwal, Anik Chaturbedi, Subhodh Karkala and Franklin Bettencourt for their help in development of various components of the project and providing their valuable inputs to the project. Special thanks to the pharmaceutical the particulate systems lab and team members.

Lastly, I would like to thank my parents, my family and my friends for supporting me through each and every step of my life.

I would also like to thank the National Science Foundation (NSF) for funding this project through the grant number: 1547171. Computational resources were provided by NSF XRAC award TG-MCB090174.

Table of Contents

List of Tables

List of Illustrations

Chapter 1: Introduction

1.1 Particulate processes

Nearly half of all industrial chemical production rely on processing of particulate systems to obtain their final products(Seville, Tüzün and Clift, 2012). These processes account for about 70% of the products of industries like detergents, aerosols, fertilizers, and pharmaceuticals (Litster, 2016). Particulate processes are widely popular as the manufactured products have great advantages over liquid formulations such as better chemical and physical stability, better handling and lower transportation costs. Despite the prevalence of these particulate processes, the underlying physics of these processes is poorly understood (Rogers, Hashemi and Ierapetritou, 2013). As a result industries that rely on these processes, especially pharmaceuticals, have to use expensive heuristic studies and inefficient operational protocols with high recycle ratios to meet strict regulatory standards (Ramachandran et al., 2009). This can lead to increase in costs and delay in release of new products. What makes these processes so challenging to design is that there are none or few governing equations to accurately predict their behavior (Sen et al., 2012).

Particulate processes are dominated by chaotic micro-scale phenomena that are a result of numerous particle-particle interactions within these systems. These small scale phenomenon develop into complex bulk behavior of these processes. To successfully predict the bulk behavior of these systems, a model needs to capture the particle-particle interactions and emergent meso-scale phenomena. Recently, the Discrete Element Method (DEM) (Cundall and Strack, 1979) simulations have been employed in pharmaceutical process modeling to obtain particle-level data, which helps describe the bulk properties of the particulate system accurately (Hancock and Ketterhagen, 2011). DEM uses Newton's equations of motion to model the forces on each particle in the system and it's interactions with the system geometry

and other particles. This enables DEM to model the small scale phenomena that determines the bulk behavior of particulate system. Since DEM includes large amount of interactions and force computations, it usually takes a large amount of time to run. Thus, there is a need for a more efficient simulation technique for these particles. Another approach employed for such systems is the population balance model (PBM), which is more computationally efficient but, lacks the accuracy of the DEM.

PBM takes into account the changes in internal or spatial particle properties. This model lacks sensitivity to design parameters such as equipment geometry. It is semi-mechanistic in nature, meaning they use population averages and probability to capture bulk behavior but still seek to capture some of the micro-phenomena of particle-particle interactions using correlations or empirically developed kernels to approximate those interactions. Even though PBM is much faster than DEM, a detailed PBM can still take a significant amount of time to solve which prevents them from being used as widely as they could be in academia and industries (Barrasso, Walia and Ramachandran, 2013). Though highly detailed PBMs take into account populating, averaging and contain semi-mechanistic kernels, these PBMs still can have difficulties in capturing the micro-scale phenomena that are crucial to predicting accurate dynamics of particulate processes.

Thus, to complement the limitations of each of these modeling techniques they are coupled to provide a more accurate model. The typical work-flow of such a PBM-DEM coupled model involves using a short DEM simulation to capture the particle-particle level interactions of the system and then this physics is fed into the PBM, so that the PBM can more accurately simulate bulk system behavior (Goldschmidt et al., 2003; Reinhold and Briesen, 2012; Barrasso, Walia and Ramachandran, 2013). This PBM-DEM method is more accurate than a PBM simulation alone but runs much faster than a complete DEM simulation. Despite the performance benefits of these coupled PBM-DEM models they still take too long to solve. In

the past parallel computing has been used to speed up computationally intensive problems such as Molecular Dynamics (MD), Computational Fluid Dynamics (CFD) and in recent years this technique has been applied to DEM and PBM simulations (Gunawan, Fusman and Braatz, 2008; A. V Prakash et al., 2013; Bettencourt, Chaturbedi and Ramachandran, 2017).

Parallel Computing is the standard procedure to solve large computational problems in many sciences. One of the methods, uses High Performance Computers (HPC) to solve over a large number of cores. A HPC consists of thousands of compute nodes connected via a high speed network. A typical compute node has multiple cores and several GBs or memory. The network that connects them, usually InfiniBand, reaches speeds of 56 Gbps. In addition, a high performance parallel filesystem accompanies the compute nodes, allowing users to read and write GBs of data efficiently. one of the challenges when implementing parallel applications is to make sure that data is written in a similar manner to serial programs.

Another form of parallel computing is to use graphics processing unit. These GPUs contain thousands of compute cores that can be used run tasks in parallel. Thus, a desktop equipped with a GPU could compute the same results as a CPU code on supercomputers in similar or lower amount of time as seen in Section 4.2.3. With the launch of Compute Unified Device Architecture (CUDA), NVIDIA made it easier to use GPUs for general parallel programming in an approach usually termed as general purpose computing on GPUs (GPGPUs). To ensure correctness of the results, an appropriate data communication implementation is required. This one of the major considerations that need to be made while parallelization of code.

Some of the benefits of using parallel computing in the pharmaceutical industry include high accuracy for parameter estimation as detailed particle-scale simulations can be performed quickly. Quick simulations can also help improve control of continuous pharmaceutical processes. This work showcases the speed improvements that can be achieved using HPCs and how they can help improve process design.

1.2 Objective

The main objective of this work is to develop a DEM-PBM coupled model that run efficiently in parallel so that it can take advantage of the computational capabilities of modern computing clusters. The future prospect of this work pertains towards the control of granulation processes but is not in scope of this study. In the course of development of the model, the DEM simulation were performed on LAMMPS Improved for General Granular and Granular Heat Transfer simulations (LIGGGHTS) (Kloss et al., 2012) to model the micro-mechanics of the Lódige CoriMix CM5 high shear granulator. A 4-Dimensional, reduced order DEM informed PBM was developed that is parallelized using hybrid techniques (Message Parsing Interface (MPI) + Open Multi-Processing (OMP)) to model the bulk processes occurring during the granulation process and is discussed in section 3.3. Section 3.4.1 describes unidirectional coupling of DEM and PBM using Radical Pilot, a framework developed in Python. Further bidirectional coupling of DEM –PBM is discussed in sections 3.4.2. An extension of the PBM onto GPUs for computation has also been discussed in sections 3.3.2 and its results in Section 4.3.2. This provides task level parallelization, to help develop an accurate model which can be run quickly on high performance computing systems. The speed improvements obtained for the parallel code and further improvements that can be implemented are discussed in Chapter 4.

Chapter 2: Background and literature review

Particulate processes are ones in which a system of discrete species exist, such as granules or catalyst pellets, that undergo changes in average composition, size, or other pertinent properties. Granulation is one such particulate process which is commonly found in the pharmaceutical industry. In this process, fine powders are converted to larger granules using a liquid binder. The three rate processes governing granulation are wetting and nucleation, consolidation and aggregation and attrition and breakage (Iveson et al., 2001; Cameron et al., 2005). As the liquid is added to the fine powder, it forms a porous nuclei that can coalesce, deform and break (Barrasso et al., 2015). As there is an alteration in the properties of these nuclei, they can take up additional liquid or breakdown to a finer powder. To understand how a granulation processes will behave with different design and operating parameter settings and formulations, experimental studies are performed which is a method referred to as Quality-by-Testing (QbT). This methodology is time consuming and expensive. Thus, newer research is focused on the QbD concept i.e., developing a mathematical model to represent the rate process.

The paradigm shift of the pharmaceutical industry for instance towards continuous manufacturing, emphasizes the need for a more accurate model. This further helps to develop better control strategies for the process. The modeling of particulate processes is more time consuming and computationally expensive when compared to fluid systems since particles are considered as individual entities rather than a continuum like fluid systems. The models discussed ahead represent the particle-particle interaction at meso and micro-scale.

2.1 Discrete Element Modeling (DEM)

Discrete Element Method is a simulation technique used to monitor the behavior of each particle as a separate entity compared to other bulk continuum models. This method tracks the movement of each particle within the space, records the collisions of each particle with the geometry as well as with each other and it is also subject to other force fields like gravity (Barrasso et al., 2015). This model is based on the Newton's laws of the motion as shown in Equations 1 & 2:

$$m_i \frac{dv_i}{dt} = F_{i,net} \quad (1)$$

$$F_{i,net} = F_{i,coll} + F_{i,ext} \quad (2)$$

Where, m_i is the mass of the particle, v_i represents the velocity of the particle, $F_{i,net}$ represents the net force on the particle, forces on the particle due to collisions and other external forces are represented in $F_{i,coll}$ and $F_{i,ext}$ respectively.

Using this method, the distance between each particle is calculated at every time step and if the distance between two particles is less than the sum of the radii (for spherical particles) a collision between the two particles is recorded. The tolerance for overlap is low in the normal as well as the tangential direction. Micro-scale DEM simulations are computationally demanding and simulations may take up to several days to replicate a few seconds of real time experiments. Many methods have been implemented to increase the speed of these simulations, such as scaling by increasing the size of the particles. These approximations are good in understanding the physics of the system but are not directly applicable to process-level simulations.

The particle-particle collisions are not always elastic, thus there is a need for models for the contact forces. The earliest elastic model was developed by Hertz and was extended by Mindlin by accounting for the tangential forces during the collisions (Adams and Nosonovsky, 2000). The Hertz-Mindlin contact model (Gantt et al., 2006; Hassanpour et al., 2013) has been utilized in this work.

2.2. Population Balance Model (PBM)

Population balance models (PBM) predict how groups of discrete entities will behave on a bulk scale due to certain effects acting on the population with respect to time (Ramkrishna and Singh, 2014). In the context of process engineering and granulation, population balance models are used to describe how the number densities, of different types of particles, in the granulator change as rate processes such as aggregation and breakage reshape particles (Barrasso, Walia and Ramachandran, 2013). A general form of population balance model is shown in Equation 3.

$$\frac{\partial}{\partial t}F(v,x,t)+\frac{\partial}{\partial v}\left[F(v,x,t)\cdot\frac{dv}{dt}(v,x,t)\right]+\frac{\partial}{\partial x}\left[F(v,x,t)\cdot\frac{dx}{dt}(v,x,t)\right]$$
$$=\Re_{formation}(v,x,t)+\Re_{depletion}(v,x,t)+\dot{F_{in}}(v,x,t)-\dot{F_{out}}(v,x,t) \quad (3)$$

In Equation (3), v is a vector of internal coordinates. For modeling a granulation process v is commonly used to describe the solid, liquid, and gas content of each type of particle. The vector x represents external coordinates, usually spatial variance. For a granulation process this account for spatial variance in the particles as they flow along the granulator.

2.3. Coupled DEM-PBMs

The use of multi-physics models has recently been adapted to understand the behavior of particle systems. These models help understand the physics of the system at various scales i.e. micro, meso and macro scale (Sen et al., 2014b). Particle process dynamics have been inferred from coupling of various physics models viz. CFD, DEM and PBM. Earlier works from Sen et al., 2014a; Barrasso et al., 2015 have successfully predicted process dynamics of the granulation process using such multi-physics models.

Initially, Ingram and Cameron (2004) coupled PBM and DEM using two different multi-scale frameworks which focused on methods of integration and information exchange required between these two methods. Later efforts on coupling of PBM and DEM were unidirectional in nature, where the collision data was obtained from the DEM and then used it in PBM. Gantt et al. (2006) used the DEM data to build a mechanistic model for the PBM. Goldschmidt et al. (2003) solved a PBM using DEM by replacing smaller particles as they successfully coalesce with larger particles. (Reinhold and Briesen, 2012)replaced a mechanistic aggregation kernel with an empirical kernel in order to prove that, in certain cases DEM simulations may not be necessary for the development of kernels. A hybrid model for one-way coupling has been reported for continuous mixing (Sen et al., 2012; Sen and Ramachandran, 2013) and is discussed in Section 3.2.

In this work, a coupled DEM and PBM model has been implemented. In general, the PBM provides meso-scale information while the DEM gives particle scale information. The combination of these two methods helps describe the process dynamics with more accuracy. However, calculations involved due to the number of particles involved in the DEM process as well as PBM become computationally very heavy, hence the motivation and need for running the simulations on HPCs and parallelization techniques.

2.4 Parallel computing and computer architecture

2.4.1 Computer Architecture

Analogous to a conventional PC a High Performance Computing cluster node has one or more CPUs and RAM. Commonly nodes are manufactured with two CPUs, each CPU is a multi-core meaning it has multiple compute cores such that each can carry out calculations separately from one another. On a node, memory is divided by CPU sockets, so each CPU has direct access to memory that is local to its own socket, however accessing memory on another socket is much slower (Jin et al., 2011). For this reason, data that is needed for computation should be stored locally to the CPU that needs it.

There are two classes of computer architecture which can be classified by memory locality features such as distributed memory systems or shared memory systems. These two classes co-exist in a cluster, thus providing the benefits of each. All the nodes exchange memory using explicit message passing while each has its own independent memory. The cores on each node can access data from the shared memory without any explicit message passing statements from the user. While designing a parallel program all these aspects need to be considered for optimal performance of the code (Adhianto and Chapman, 2007).

2.4.2. Parallel application programming interfaces

Message Passing Interface (MPI) is a common parallel computing application programming interface standard. MPI is used for distributed memory parallel computing and this is because the application programming interface will operate every MPI process as a discrete unit that does not share memory with the other processes unless explicit message passing is used. Even on shared a single node where the hardware supports shared memory computing, MPI will still operate it in a distributed memory fashion (Jin et al., 2011). Operating all cores as distinct units also means they each need their own copy of all variables used for computation which

results in a large overall memory foot print compared to a similar system if it was operated in shared memory.

Open Multi-Processing (OMP) is another application program interface stand for parallel computing. OMP is used for shared memory and can take advantage of shared memory systems which can result in much faster computation. Although, it does not work well on distributed systems. This prevents it from being used to efficiently carry out computations across multiple nodes of a cluster simultaneously (Jin et al., 2011). Since MPI is preferred for distributed computing and OMP is better for shared computing many researchers have studied the performance of MPI vs MPI+OMP methods. Many a times a tradeoff is made between optimizing a program for performance and trying to make it flexible enough to run on many different computer architectures. It was found that hybrid methods for PBMs allow the code flexibility for different architectures while still maintaining good performance (Bettencourt, Chaturbedi and Ramachandran, 2017). It was also reported that only the external (spatial) coordinates of the PBM were parallelized. In this current work external and internal (compositions) calculations are parallelized.

2.4.3 Graphics processing units (GPUs)

Graphic processing unit (GPU) were initially mainly used for vector calculations to support graphics inside a computer system. But, lately GPU manufacturers have started to promote them general computing as well. This form of computing has been gaining popularity among scientists to accelerate simulations (Kandrot and Sanders, 2011). These GPUs dominantly have a massively parallel architecture with hundreds to thousands of computational cores which can thousands of active threads simultaneously (Keckler et al., 2011). Modern GPU computing can be exploited using parallel programming languages such as OpenCL and CUDA.

CUDA is an application programming interface (API) developed by NVIDIA (NVIDIA Corporation, 2012) that enables users to program parallel code for execution on the GPU. This

is framework is an extension implemented on top of C/C++ or FORTRAN. Parallel code for the GPU is written as kernels, which theoretically are similar to functions or methods in traditional programming languages. Only few sections of the code can be written in terms of kernel while the remaining has to be executed in serial on the CPU of the system. The NVCC compiler from the CUDA toolkit prioritizes the compilation of these kernels before passing the serial section of the code to the native C/C++ compiler inside the system. There are three main parallel abstractions that exist in CUDA are grids, blocks and threads (Santos et al., 2013). Each CUDA kernel is executed serially during the execution of the program unless specified, where the kernels can be run in parallel using CUDA streams. Each kernel executes as a grid which in turn consists of various blocks which are constituted by various threads. This thread-block-grid hierarchy helps obtain fine grained data level and thread level parallelism. An illustration of this hierarchy is observed in Figure 2-1.

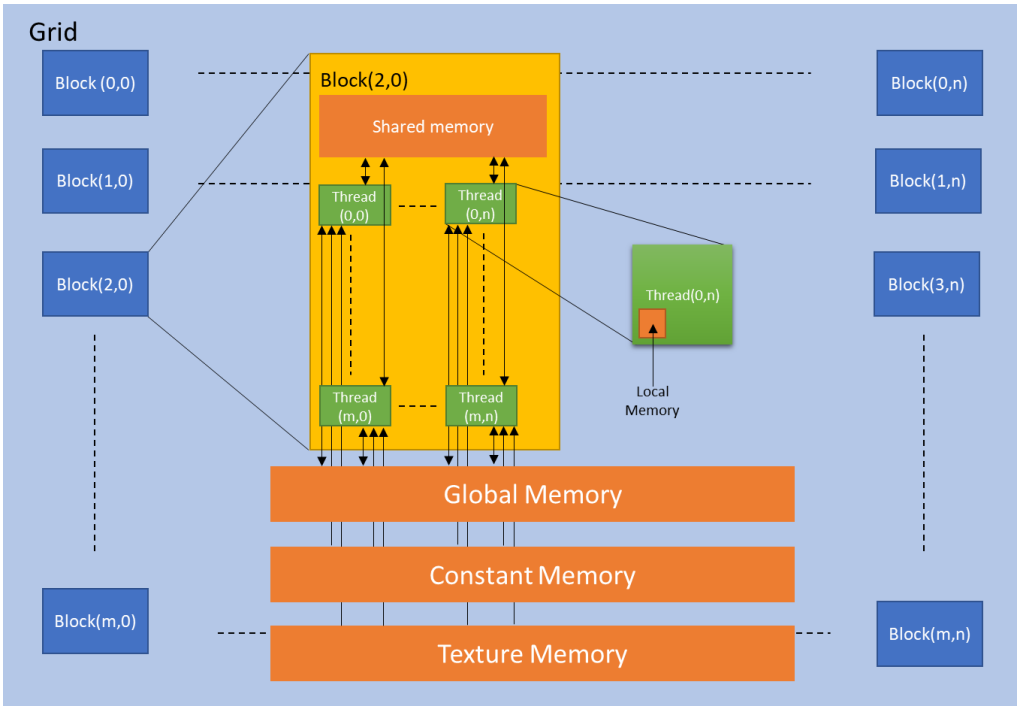


Figure 2-1: The parallel structure matrix inside the GPU and the various memories associated with each structure

Another important aspect related to GPU parallelization is the data communication between the threads. The GPU consists of various memory modules with different access limitations as

shown in Figure 2-1. The threads inside each block can communicate with each other using the shared memory. This memory is local to the block where these threads exist i.e. they are not accessible by threads from other blocks. In addition to the shared memory each thread has its own local memory where local/temporary variables for each kernel can be saved to them. The threads from different communicate with each other using the global memory which is visible to all blocks inside the GPU at the cost of higher communication times. Accessing of data from the local memory is the fastest for a thread and it slows down as we move towards shared block memory and the least for accessing data from the global GPU memory.

2.5 Previous works on parallel development and solution of PBM and DEM

The idea of parallelization to reduce the computation time of the DEM and PBM simulations have been employed by various researchers in the past. Gunawan, Fusman and Braatz (2008) used high-resolution finite volumes solution methods for the parallelization of their PBM. They performed load balancing effectively by decomposing the internal coordinates of their PBM. They achieved speed improvements up to 100 cores on one system size, but was not tested for models with more dimensions. Moreover, they mentioned that parallelization could be improved using shared memory processing. Bettencourt, Chaturbedi and Ramachandran (2017) took a hybrid approach towards the parallelization of the PBM using both MPI and OMP. The hybrid parallelization helped achieve a speed improvement of about 98% using 128 cores over the serial code. Prakash (A. V. Prakash et al., 2013; A. V. Prakash, Chaudhury and Ramachandran, 2013) used the inbuilt Parallel Computation Toolbox (PCT) in MATLAB (MathWorks™ Documentation, 2017) to parallelize their PBM on lower number of cores, but this faced the shortcomings of MATLAB's internal processing and could not achieve the speed improvements of parallelization of a program if it were written in a native programming language like C/C++ or FORTRAN.

LIGGGHTS (Kloss et al., 2012), an open-source software used to perform DEM simulations has native support for MPI for parallelizing the simulation by static decomposition which partitions space such that the area of communication between the MPI processes is minimized. Kačianauskas et al. (2010) used load balancing methods similar to a static decomposition and observed that this works well for a mono-dispersed system but the computational effort increases for simulations for poly-dispersed material. Gopalakrishnan and Tafti (2013) also reported a speed increase and a parallel efficiency of about 81% in their CFD-DEM simulation. LIGGGHTS could not take advantage of shared memory interfaces since it did not support OMP. Berger et al. (2015) implemented hybrid parallelization methods for the particle-particle interaction and the particle-wall interaction modules in LIGGGHTS and also used the Zoltan library (Boman et al., 2012) developed by Sandia National Laboratories for dynamic load balancing. They achieved a speed improvement of about 44% for simulations performed on higher number of cores, but there was no significant speed improvement for smaller core counts.

Algorithms to parallelize the PBM codes on GPU have been studied briefly by A. V Prakash, Chaudhury and Ramachandran, 2013 using the inbuilt MATLAB's parallel computing toolbox (PCT). This study was able to achieve good speed ups but could have been higher if the code had been implemented in native programming languages such as C or FORTRAN. Other works that have used GPU acceleration to improve computation times for their population balance simulations include those from various other chemical engineering processes such as crystallization (Szilágyi and Nagy, 2016), combustion (Shi et al., 2012), multiphase ow (Santos et al., 2013), coagulation dynamics (Xu et al., 2015)

2.6 Pilot abstraction and RADICAL-Pilot (RP)

A primary challenge faced is the scalable execution of multiple (often two, but possibly more) heterogeneous simulations that need to run independently but have a need to communicate

and exchange information. Traditionally each simulation is submitted as an individual job, but that invariably leads to a situation where each simulation gets through the batch-queue systems independent of the other. So although the first-through-the-queue is ready to run, it stalls fairly soon waiting for the other simulation to make it through the queue. On the other hand MPI capabilities can be used to execute both simulations as part of a single multi-node job. Thus whereas the former method suffers from unpredictable queue time for each job, the latter is suitable to execute tasks that are homogeneous and have no dependencies.

The Pilot abstraction (Luckow et al., 2012) solves these issues: The Pilot abstraction (i) uses a container-job as a placeholder to acquire resources, and (ii) decouples the initial resource acquisition from task-to-resource assignment. Once the Pilot (container-job) has acquired the resources, it can be populated with computational tasks. This functionality allows all tasks to be executed directly on the resources, without being queued individually. Thus, this approach supports the requirements of task-level parallelism and high-throughput as needed by science drivers. RADICAL-Pilot is an implementation of the Pilot abstraction in Python, engineered to support scalable and efficient launching of heterogeneous tasks across different platforms.

Chapter 3: Methods and simulation setup

3.1 Discrete Element Method (DEM)

LAMMPS Improved for General Granular and Granular Heat Transfer simulations (LIGGGHTS) v3.60 developed by DCS computing was used for all the DEM simulations performed in this study. LIGGGHTS natively supports parallelization using MPI, which helps divide the geometry into various smaller simulation boxes thus, making them run faster compared to a serial code. Parts of the source code of LIGGGHTS was also edited to enable the code to capture particle-particle and particle-wall collisions. This version of LIGGGHTS was compiled using the mvapich (mvapich2 v2.1) and Intel (Intel v15.0.2) compilers with the -O3 optimization option as well as an option to side load the process to the Xeon phi co-processors was added. The initial timing studies were performed on Stampede supercomputer located at TACC, University of Texas, Austin. The hardware configuration of each node consists of 2 8-core Intel Xeon E5-2680 processors based on the Sandy Bridge architecture, 32 GB of memory with QPI interconnects at 8.0 GT/s PCI-e lanes.

3.1.1 Geometry and Meshing

In this work, the Lódige CoriMix CM5 continuous high shear granulator has been studied. Its geometry was developed using the SolidWorks™ (Dassault Systemes). This granulator consisted of a high speed rotating element enclosed within a horizontal cylindrical casing. The casing (shown in Figure 3-1) consists of a cylinder with diameter of 120 mm at the inlet and 130 mm at the outlet and having a total length of 440 mm. A vertical inlet port is provided at one end of the casing and an angled outlet port is provided at the larger end of the case.

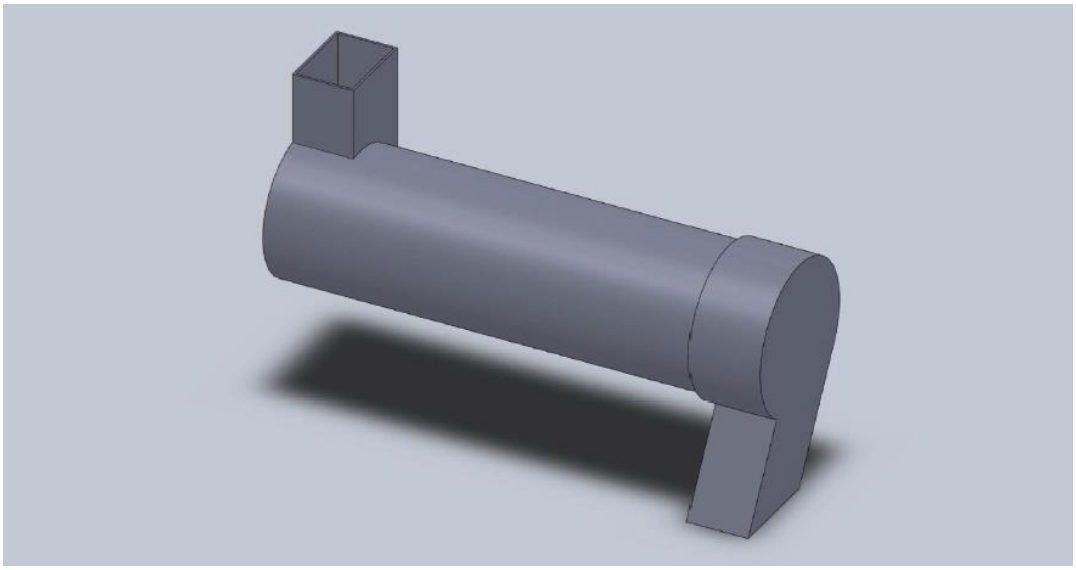


Figure 3-1: The isometric view of the Lódige CoriMix CM5 continuous high shear granulator casing

The impeller consists of a cylindrical shaft of length 370 mm and diameter 68 mm with four flattened sides 15 mm wide running along the axis. The blades are placed on these flattened sides as shown in Figure 3-2. There are three different blade elements on the shaft (Figure 3-3).

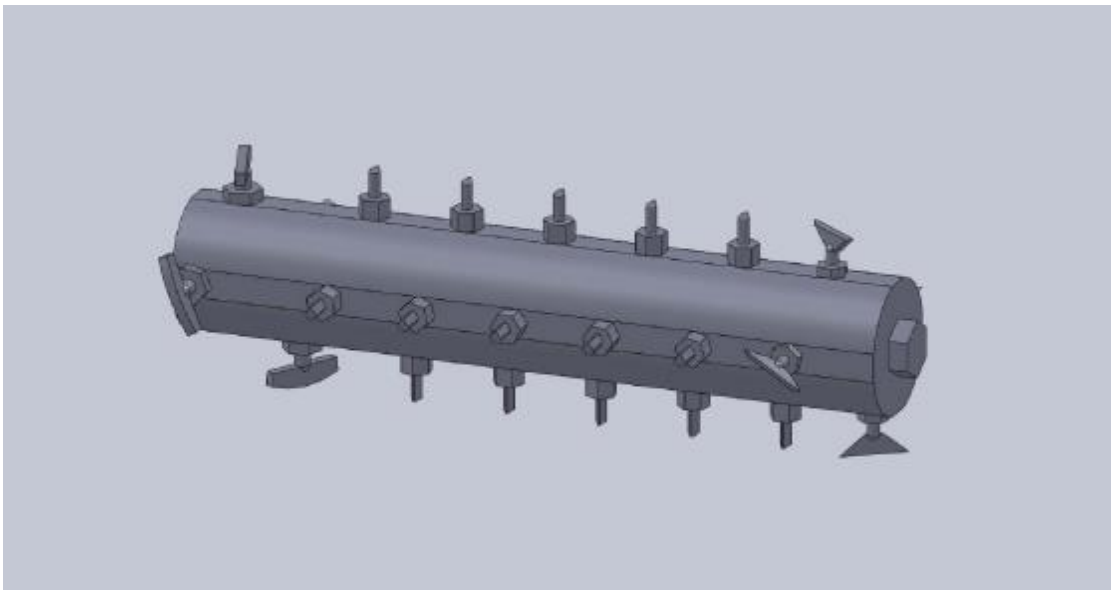
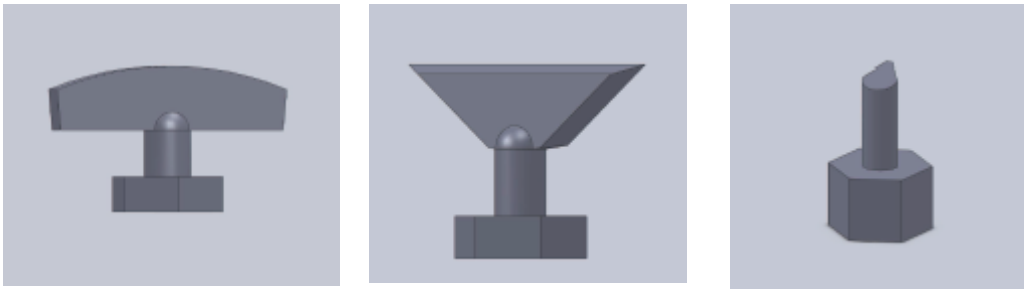


Figure 3-2: isometric view of the impeller inside the Lódige CoriMix CM5 continuous high shear granulator casing



(a) Feed Element **(b)** Exit Element **(c)** Shear Element

Figure 3-3 : Components (a) and (b) help in the forward movement of the particles while (c) aids to direct the particles to the wall inside the Lódige CoriMix CM5 continuous high shear granulator.

At the granulator inlet, there are 4 paddle shaped feed elements following which there are 20 tear drop shaped shearing elements and finally 4 trapezoidal blades near the exit. All these elements are placed in a spiral configuration. Once the geometry was built in SolidWorks™ (Dassault Systèmes) the shell and impeller were exported as stereo lithography (STL) files. The coarsest output option was used to keep the STL files small and simple for faster computation times. The origin of the geometry was not preserved while saving the STL files since it needs to be aligned in LIGGGHTS as per the process conditions. This resulted in the impeller having 2802 faces and 1281 points with approximately a file size of 775 kilobytes. The shell had 1948 faces and 720 points and size was about 544 kilobytes. MeshLab was used to align the STL files for importing into LIGGGHTS. No mesh treatments were used on the STLs. The meshes were then imported into LIGGGHTS using the write command in serial. This resulted in 50 elements of the impeller file having highly skewed elements, which have more than 5 neighbors per surface or have an angle less than 0.0181185 radians, that according to LIGGGHTS would degrade parallel performance. The write exclusion list command in LIGGGHTS was used and this exclusion list file is then used in the simulation to skip the highly skewed elements during the simulation.

3.1.2 DEM input file setting

The DEM simulation in LIGGGHTS are setup using an input script which defines the physical parameters of the particles, importing of the geometry, particle insertion commands, various physics models to be used during the simulation as well as various compute and dump commands to help print the data required for post-processing of the data. The particles were considered to be granular in nature. The Hertz-Mindlin model was used for non-adhesive elastic contact between the granular particles. The particles were inserted inside the granulator from the inlet at a constant mass flow rate of 15 kilograms per hour. The rotation speed of the impeller was kept throughout the study at 2000 rotations per minute. Such a high rotation speed was chosen since this would lead to high shear between the particles and the walls of the shell resulting in better size control of the granules. There were 2 sets of simulations that were performed, one with mono-sized particles and second consisting of a distribution of sizes. The particle radii chosen for mono-sized simulation varied 0.59mm - 2mm, consecutive particles radii had volume twice of one before them. The radii range of the distributed size simulation was 1mm - 3mm. The difference in the mechanics of these two simulations is discussed in the Chapter 4. The physical constants used for the simulations are given in Table 3-1.

Table 3-1 : Physical Properties of the particle for the LIGGGHTS input script

Parameter	**Value**	**Units**
Young's modulus of particles	8×10^6	$\mathrm{N \cdot m^{-2}}$
Young's modulus of geometry	1×10^8	$\mathrm{N \cdot m^{-2}}$
Poisson's Ratio	0.2	--
Coefficient of restitution	0.4	--
Coefficient of static friction	0.5	--
Coefficient of static friction	0.2	--
Density of the granules	500	$\mathrm{kg \cdot m^{-3}}$

The simulation data was collected after every 50,000 time steps (510−3 sec) for the visualization of the particles inside the shell, further post processing. The collisions between each of the particles and the collisions between the particle and the geometry was collected and used in the PBM.

3.1.3 DEM data post processing

The post processing of the data obtained from the DEM simulations was done using MATLAB. The first test run on the output data was to determine if the simulation had reached steady-state. The mass inside the granulator was found out by averaging it over 5 time steps and then compared to mass inside the granulator after every 10000 time steps (about 510−4 sec) with a tolerance of about 10%. If the mass was found to be constant for most of the iterations, it was considered to be at steady state. Another test to determine steady state was to monitor the number of collision inside the granulator. The visualization of the simulation data was done by running the LIGGGHTS post processing (LPP) script over the dump files to convert them into STL files. These files were then loaded in to Paraview (Ayachit, 2017) for a graphical representation of the simulation. It can be seen that the number of collision start to oscillate around a mean value. The number of collisions were then plotted and steady state time was determined. A precautionary script was also run so as to determine that no particles were lost due to overlap of the geometry with the particles as well as from particle-particle overlap.

3.2 Population Balance Modeling

The population balance equation used in this work is expressed below:

$$\frac{\delta}{\delta t}F(s_1, s_2, x, t) = \Re_{agg}(s_1, s_2, x, t) + \Re_{br}(s_1, s_2, x, t) + \dot{F}_{in}(s_1, s_2, x, t) - \dot{F}_{out}(s_1, s_2, x, t) \quad (4)$$

where, $\dot{F}(s_1, s_2, x)$ is the number of particles with an active pharmaceutical ingredients (API) volume of s_1 and an excipient volume of s_2 in the spatial compartment x. The rate of change of number of particles with time in different size classes depend on the rate of aggregation, $\Re_{agg}(s_1, s_2, x)$ and the rate of breakage, $\Re_{br}(s_1, s_2, x)$. Also, the rate of particles coming into, $\dot{F}_{in}(s_1, s_2, x)$ and going out, $\dot{F}_{out}(s_1, s_2, x)$ of the spatial compartment due to particle transfer affect the number of particles in different size classes.

The rate of change of liquid volume for a given time in each particle is calculated using the equation:

$$\begin{aligned}\frac{d}{dt}F(s_1, s_2, x)l(s_1, s_2, x, t) \\ = \Re_{liq,agg}(s_1, s_2, x) + \Re_{liq,break}(s_1, s_2, x) \\ + \dot{F}_{in}(s_1, s_2, x)l_{in}(s_1, s_2, x) - \dot{F}_{out}(s_1, s_2, x)l_{out}(s_1, s_2, x) \\ + F(s_1, s_2, x)\,\dot{l}_{add}(s_1, s_2, x)\end{aligned} \tag{5}$$

where, $l(s_1, s_2, x)$ is the amount of liquid volume in each particle with API volume of s_1 and excipient volume of s_2 in the spatial compartment x, $\Re_{liq,agg}(s_1, s_2, x)$ and $\Re_{liq,break}(s_1, s_2, x)$ are respectively the rates of liquid transferred between size classes due to aggregation and breakage. $l_{in}(s_1, s_2, x)$ and $l_{out}(s_1, s_2, x)$ are respectively the liquid volumes of the particles coming in and going out of the spatial compartment. $l_{add}(s_1, s_2, x)$ is the volume of liquid acquired by each particle in the compartment at every time step due to external liquid addition. The rate of change of gas volume for a given time is calculated in a similar manner.

The rate of aggregation $\Re_{agg}(s_1, s_2, x)$ can be calculated as (Chaturbedi et al., 2017):

$$\Re_{agg}(s_1, s_2, x) = \frac{1}{2}\int_0^{s_1}\int_0^{s_2} \beta(s_1', s_2', s_1 - s_1', s_2 - s_2', x)F(s_1', s_2', x)F(s_1 - s_1', s_2 - s_2', x)ds_1' ds_2' - F(s_1, s_2, x)\int_0^{s_{max1}-s_1}\int_0^{s_{max2}-s_2} \beta(s_1, s_2, s_1', s_2')F(s_1', s_2', x)ds_1' ds_2' \quad (7)$$

where, the aggregation kernel, $\beta(s_1, s_2, s_1', s_2')$ is expressed as a function of collision rate coefficient (C) and probability that collision results in agglomeration(ψ) (Ingram and Cameron, 2005) and is shown below:

$$\beta(s_1, s_2, s_1', s_2') = \beta_0 C(s_1, s_2, s_1', s_2', x)\psi(s_1, s_2, s_1', s_2', x) \quad (8)$$

where, β_0 is aggregation rate constant.

Collision rate coefficient (C) is a function of particle sizes and is calculated by normalizing the number of collisions between groups of particles (Gantt et al., 2006) and is obtained from LIGGGHTS DEM simulation. A recent study shows that collision frequency depends on PSD as well (Sen et al., 2014a). Collision rate coefficient for every time step can be expressed as:

$$C(s_1, s_2, s_1', s_2') = \frac{N_{coll(s_1, s_2, s_1', s_2')}}{N_{p(s_1, s_2)} Np(s_1', s_2')\Delta t} \quad (9)$$

In Equation 9, N_{coll} is the number of collision between two particles in time interval Δt & N_p is number of particle of particular size. The agglomeration (ψ) in Equation 8 can be expressed as:

$$\psi(s_1, s_2, s_1', s_2') = \begin{cases} \psi_0, LC(s_1, s_2) \geq LC_{min} \text{ and } LC(s_1', s_2') \geq LC_{min} \\ 0, LC(s_1, s_2) < LC_{min} \text{ and } LC(s_1', s_2') < LC_{min} \end{cases} \quad (10)$$

In Equation 10, LC is the liquid content of particles and LC_{min} stands for minimum liquid content required for coalescence of particles.

Particle transfer rate, $\dot{F}_{out}(s_1, s_2, x)$ in Equation 4 is calculated as:

$$\dot{F}_{out}(s_1, s_2, x) = \dot{F}(s_1, s_2, x)\left(\frac{v_{compartment}(x)dt}{d_{compartment}}\right) \quad (11)$$

where, $v_{compartment}(x)$ and $d_{compartment}$ are respectively the average velocity of particles in compartment x and the distance between the mid-points of two adjacent compartment, which is the distance particles have to travel to move to the next spatial compartment. dt is the time-step. The process parameters and physical constants used in the PBM simulation are listed in Table 3-2.

Table 3-2: Parameters used in the PBM simulation

Parameter	**Symbol**	**Value**	**Units**
Initial time step	δt	0.5	s
Mixing time	T	25	s
Granulation time	T	75	s
Velocity in axial direction	v_{axial}	1	$m \cdot s^{-1}$
Velocity in radial direction	v_{radial}	1	$m \cdot s^{-1}$
Aggregation constant	β_0	1×10^{-9}	--
Initial particle diameter	R	150	μm
Diameter of the impeller	D	0.114	m
Impeller rotation speed	RPM	2000	--
Minimum granule porosity	ϵ_{min}	0.2	--
Consolidation rate	C	0	--
Total starting particles in batch	$F_{initial}$	1×10^6	--
Liquid to solid ratio in binder	L/S	0.35	--
Number of compartments	c	16	--
Number of first solid bins	s	16	--
Number of second solid bins	ss	16	--

3.3 Discretization & parallelization of the PBM

The PBM was discretized by converting each of its coordinates in to discrete bins. For the spatial coordinates a linear bin spacing was used. For the internal coordinates, solid, liquid and gas a non-linear discretization was used (Barrasso and Ramachandran, 2012). Once the PBM had been discretized, a finite differences method was used which created a system of ordinary differential equations (ODEs) (Barrasso and Ramachandran, 2015). The numerical integration technique used to evaluate the system of ODEs was first order Euler integration as it is commonly used to solve these types of systems and author found an improvement in speed while having minimal impact on accuracy (Barrasso, Walia and Ramachandran, 2013). In order to avoid numerical instability due to the explicit nature of the Euler integration, Courant-Friedrichs-Lewis (CFL) condition must be satisfied (Courant, Friedrichs and Lewy, 1967). For our PBM model, time-step was calculated at each iteration such that, the number of particles leaving a particular bin at any time is less than the number of particles present at that time(Ramachandran and Barton, 2010).

3.3.1 Parallelizing using CPU cores

To obtain the most optimal parallel performance, when solving the PBM, workloads were distributed in a manner which took into account the shared and distributed memory aspects of the clusters, the PBM was being run on. To parallelize the model in a way which could take advantage of shared memory but still effectively run across a distributed system both MPI and OMP were implemented. One MPI process was used per CPU core and one OMP thread was used per CPU core, as Bettencourt et al. (2017) found it resulted in the best performance. MPI was used for message passing from one node to another while OMP was used for calculations on each node that could be efficiently solved using a shared memory system. An algorithm in the form of pseudo code is presented in the Appendix illustrating how the calculations are

distributed and carried out during the simulation. For each time step, the MPI processes are made responsible for a specific chunk of the spatial compartments. Then each OMP thread, inside of each MPI process, is allocated to one of the cores of multi-core CPU the MPI process is bound too. The OMP threads divide up and compute $\Re_{agg}$. After $\Re_{agg}$ is calculated the MPI processes calculate the new PSD value for their chunk at that specific time step, $F_{t,c}$. The slave processes send their F_{tc} to the master processes which collects them into the complete $F_{t,all}$. The master process then broadcasts the $F_{t,\,all}$ value to all slave processes. This decomposition of the data into different CPUs and further into various threads is illustrated in Figure 3-4. A crucial feature of the PBM is that the current PSD ($F_{t,all}$) value is used to compute a new time step size for the next iteration. This means all of the MPI processes need to have the same dynamic time step size at each iteration for the calculations to be properly carried out in parallel. Since the completely updated $F_{t,\,all}$ value is shared before calculating a new time step each process will have the same $F_{t,\,all}$ value. As a result each process calculates the same size for the new time step. Since the DEM informed PBM code required dynamic size increase for the data structure due to different number of particles being present in the DEM simulation, the PBM code developed used the Standard Template Library containers and their features available in C++ 11 and later. It requires intel v17.0.2 (GCC 5.3+) or later for compilation. Since this module was not present on the Stampede, the execution of the PBM was done on the newer Stampede2 supercomputer. Each of the compute node of the cluster consists of Intel Xeon Phi 7250 (“Knights Landing") which has 68 cores on a single socket clocked at 1.4 GHz, with 96 GB of DDR4 RAM. Each of the cores consists of 4 hardware threads. The simulations were then run by varying the number of MPI processes from 1 to 16 and the number of OMP threads from 1 to 8, thus, using a maximum of 128 cores.

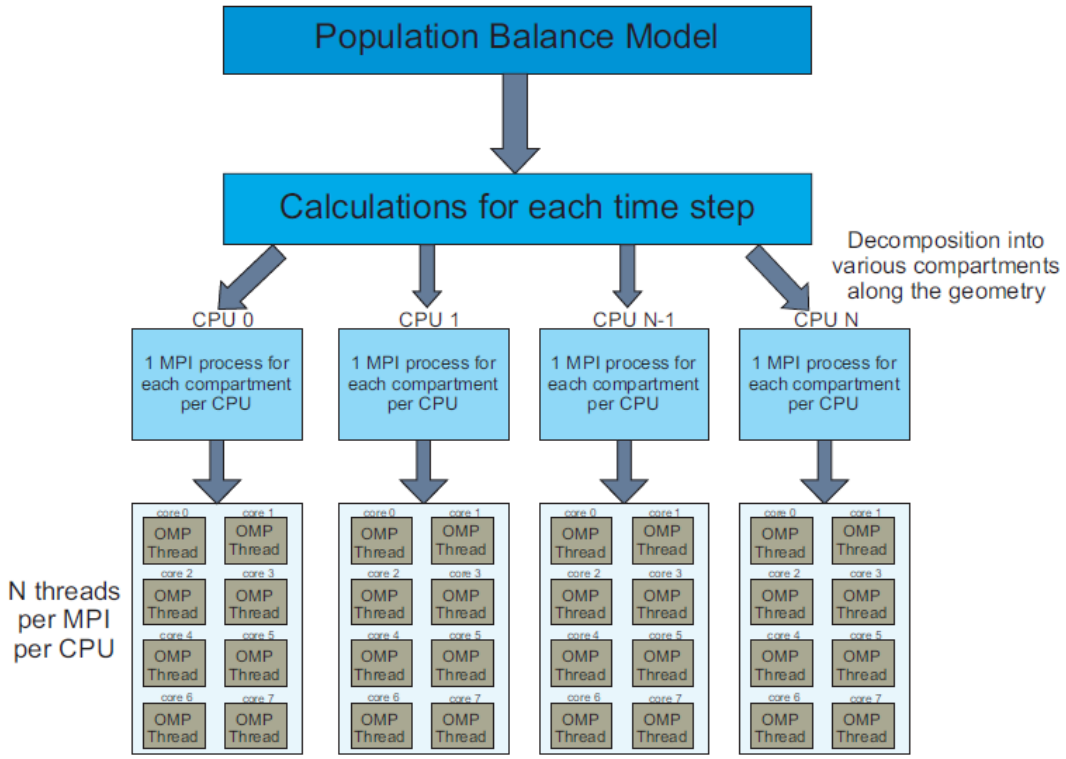


Figure 3-4: The distribution of the PBM calculations to utilize the MPI + OMP parallelization technique on the CPUs of each node

3.3.2 Parallelizing using the GPU

NVIDIA's CUDA toolkit extends the C language such that user defined functions called kernels can be created to be run on the GPU. These kernels can be executed N times in parallel using large number of threads. A thread is sequence if programmed instructions that can be managed by the computer's scheduler. A kernel depending upon the dimensions of the data can execute instructions in 1-D, 2-D or 3-D thread blocks. The kernel can also launch multiple thread blocks at once, thus increasing the number of parallel process executions known as grid. Similar to a thread block, a grid can be up to 3-D depending upon the data under study.

The code execution was split between the CPU (also called host) and GPU (also called device). Time sensitive calculations as well as mixed data calculations were handled on the CPU with a single core, while the more computationally intensive tasks were distributed on to the GPU using kernels. Like in the CPU parallelization, the geometry was split into multiple

compartments. These compartments in turn formed the number of blocks inside each GPU kernel. The number of solids used helped formed the threads in each of these blocks. The work flow of the execution can be found in Figure 3-5. The orange arrows in Figure 3-5 indicate the transfer of data from the CPU memory to the GPU memory and vice-versa whereas the blue arrows indicate the sections of the code that is sent to the GPU for parallel execution.

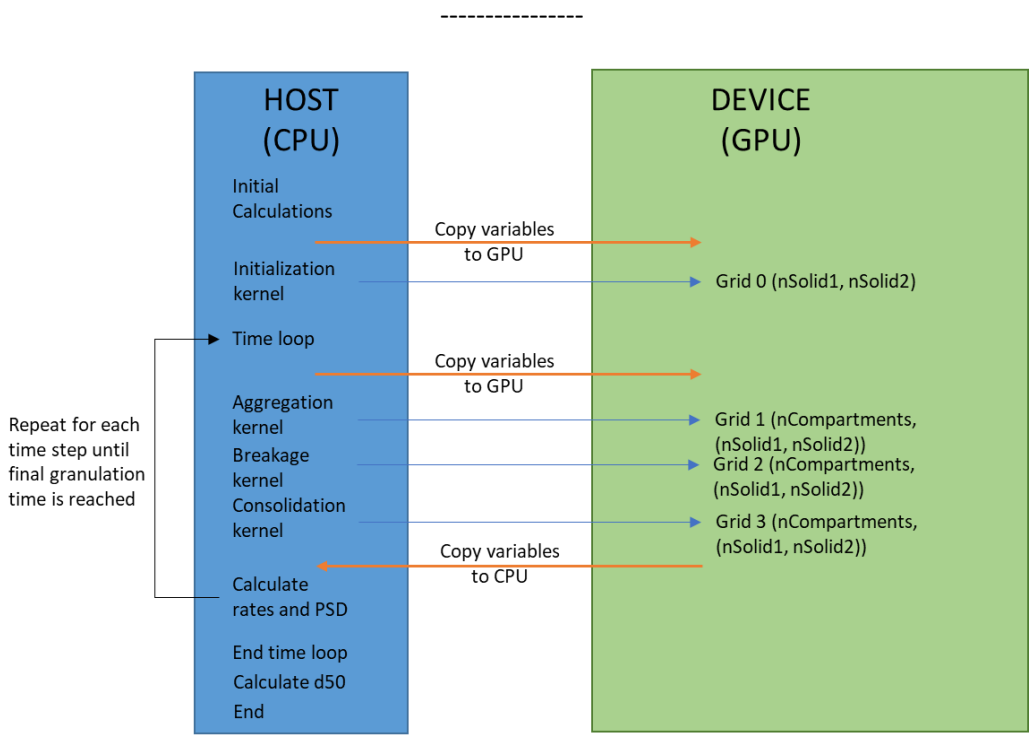


Figure 3-5: Workflow of the GPU code indicating data transfers and execution timeline of the code

These simulations were performed on a desktop computer with an Intel i7-7700k processor with 32 GB of RAM and the GPU used was a NVIDA Quadro p4000 GPU with 16 GB of GPU memory, with 1792 CUDA cores and a maximum memory bandwidth of 243 GBps. All the parameters were kept the same.

3.4 RADICAL-Pilot (RP) & coupled DEM and PBM communication

RADICAL-Pilot defines a number of abstractions to describe resources, and computational tasks. The Pilot Description is the abstraction that describes resources by defining the number of cores that will be used, the used HPC resource and the total time that the resources are

needed, and the queue that will be used. The abstraction that defines the execution of a computational task is the Compute Unit (CU). A CU defines the executable, the necessary environment that is needed during execution, execution arguments, and any file dependencies the executable may have. RP also defines two managing components, the Pilot Manager, and Unit Manager. The Pilot Manager is responsible to submit a Pilot to the selected resources. The Unit Manager is responsible to place CUs to active Pilots. As soon as a unit is placed in a pilot it starts executing on the acquired resources.

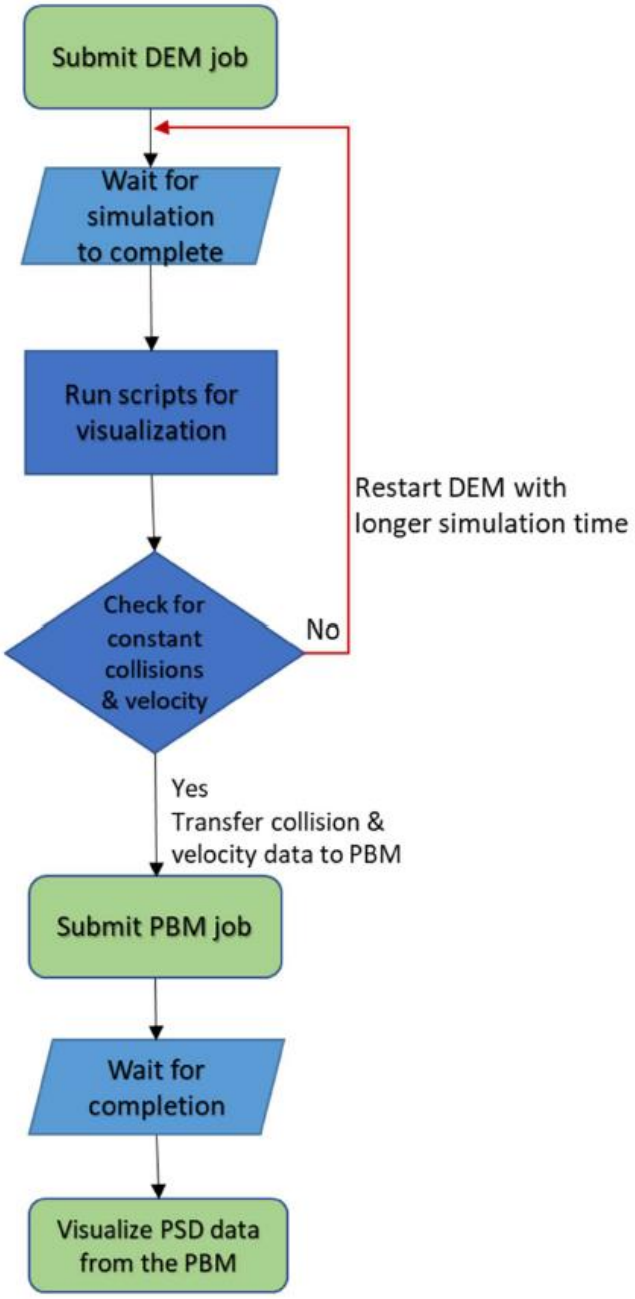
RADICAL-Pilot was utilized to execute and monitor DEM and PBM simulations. Initially, a pilot description is created to use several cores of a resource and submitted to the pilot manager. A CU description is created for the DEM simulation and submitted to the Unit Manager. As soon as the pilot is active the DEM simulation start executing. When finished a CU that describes a PBM simulation is submitted at the Unit Manager and start executing. Any data dependency is satisfied by doing the necessary file linking through the CU description.

3.4.1 One-way DEM-PBM coupling

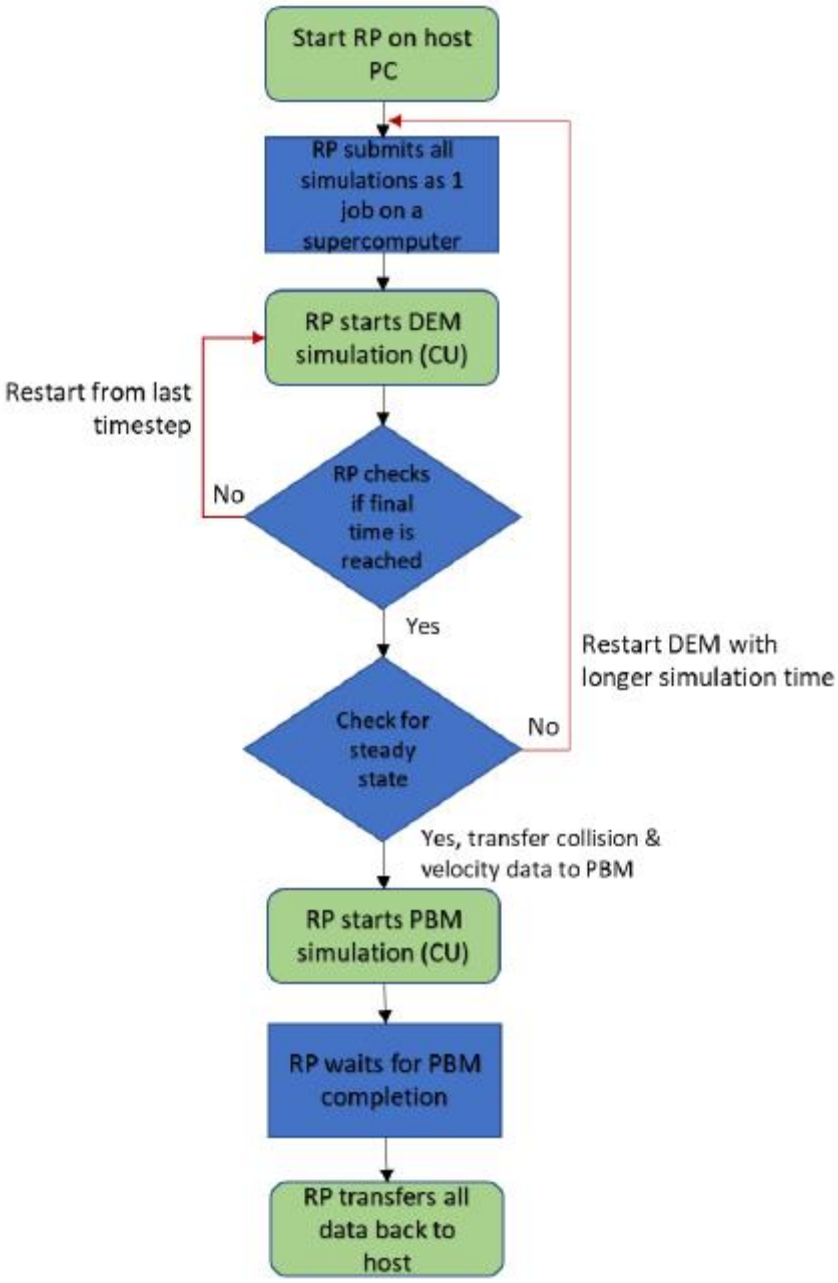
The DEM simulation was used to determine the time taken to reach steady state while the DEM data was also used to understand the steady state micro-mechanics of the powder mixture inside the high shear granulator. These simulations were also checked for particle loss as mentioned in Section 3.2.1. Physical quantities obtained from the DEM simulation like velocity, collisions were then used in the PBM simulation to give it a more mechanistic nature. Figure 3-6a represents the flow of data in the coupling process, where each of the DEM and PBM job was submitted manually to the supercomputer.

As soon as the pilot is active, the DEM simulation starts executing. When it is finished the DEM output is linked to the new CU describing the PBM simulation. The PBM is submitted by the Unit Manager and starts executing. Any data dependency is satisfied by performing the

necessary file linking through the CU description. A pictorial representation of the data flow using RP can be found in Figure 3-6b.



(a) Workflow without RP



(b) Workflow with RP

Figure 3-6: Flow charts representing differences in between the coupling of the DEM and PBM simulations manually and using RP respectively.

3.4.2 Two-way coupling and controller design

Figure 3-7 represents the workflow that was used to model the two way coupling DEM PBM for the high shear granulator. These simulations were monitored using python scripts to switch in between the two simulations. The controller scripts to monitor the DEM and PBM

simulations were written in python. Each of these scripts were executed along with its respective simulation. The DEM controller script monitored the collision and velocity data being dumped by the DEM simulation. The data was first stored and the average of the velocities for each type of particles and the total number of collisions and impacts were determined. If there was a change in either of these properties of more than 15%, an exit status was sent which led to killing the execution of the DEM and indicated to start the PBM simulation. The DEM controller script also printed out various data files required for the PBM execution as well as for the restart of the DEM simulation if needed. In the case, the properties did not show a variation of more than 15% over the span of 5s of DEM simulation time, it was considered to be in steady state and that all the simulations were halted. The PBM controller read the d50 and number of particle files which were printed by the PBM after a constant time interval. The change in each of these properties in each compartment was monitored and an exit signal was sent if they varied by more than 15%. The halting of the PBM was accompanied by the printing of files required to restart DEM with new diameters and data for restarting the PBM if needed after the DEM simulation.

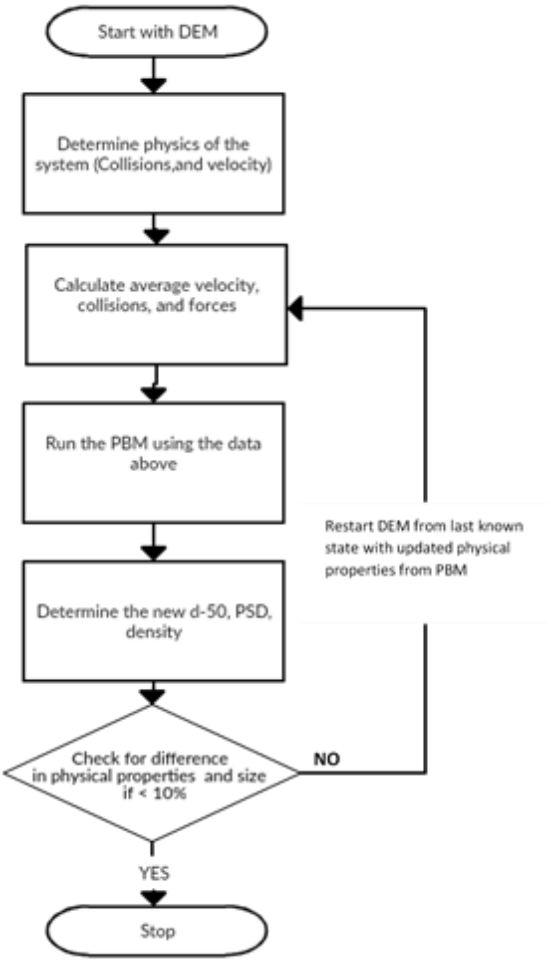


Figure 3-7: Schematic of the bi-directional approach used to model the granulator

Chapter 4: Results and Discussion

4.1 Discrete Element Method

4.1.1 Spatial decomposition studies

LIGGGHTS as discussed previously, statically decomposes the work space and each section is sent to a MPI process for calculations. Thus, the division of the space becomes an important criteria for the simulation for efficient load balancing. The initial studies were undertaken for a mono-sized particle of size 1 mm and the simulation was carried out for 0.5 second of granulation time. The initial timing studies for the decomposition were performed using 64 cores. The effect of the decomposition on the simulation time can be seen in Table 3. This indicates that dividing the x-direction in more number of compartments help increase the speed of the simulation. This is easy to comprehend since the granulator has its length parallel to the x-axis. These results also show that if the geometry is divided into more than 2 compartments in the y-axis or the z-axis the simulation time increases. This can be accounted to the increased communication required to transfer the rotating impeller mesh from one compartment in the y-axis or the z-axis to the another compartment for each time step. Since MPI is limited by communication in between the nodes, a speed decrease is observed due to increased partitioning in these directions.

When MPI is used for parallelization of a task, load balancing becomes an important parameter that needs to be considered. When the geometry is divided, the amount of computation done by one core should be in a similar to other processors. This helps in better utilization of the resources as well as make the simulations run faster. Following the results from the initial timing studies, the y and the z-axes were not divided in more than 2 compartments for 128 and 256 core simulation as well. This meant that the x-axis was divided into 32 and 64 compartments respectively. In order to avoid the expensive communication between the processes, LIGGGHTS tries to insert the particles towards the centre of the

compartment such that the number of ghost atoms are minimized. But, slicing in the x-axis reduced the space available for the insertion of the particles thus, many of the particles were inserted incorrectly. Another abnormal behaviour observed during these simulation was the particles halted at certain compartment and no particle travelled beyond this compartment in the x-direction. Thus, another set of timing studies were performed for the 128 and 256 core simulations. The comparison of simulation times have been shown in Table 4-1.

Table 4-1: Comparison of time taken for the DEM simulations using 128 and 256 core due to different spatial decomposition configurations.

Number of cores used	Slices in x-direction	Slices in y-direction	Slices in z-direction	Time taken for a 10 second simulation
128	16	2	4	264.67
128	16	4	2	247.2
256	16	2	8	271.5
256	16	8	2	252
256	16	4	4	265.32

The simulation times illustrated in Table 3 show that incorrectly slicing the geometry also affect the performance of the system. It can be noted that slicing along y-axis is more favourable than slicing along the z-direction. The insertion of particles is hindered when the geometry is cut along the z-axis as the inlet is perpendicular to it. LIGGGHTS thus provisions lesser space for the insertion of these particles, thus increasing the time of the simulation. Thus, just slicing the geometry into 2 sections along the z-axis is preferred. So, the chosen configurations for the final simulations consisted of the geometry having only 2 sections in the z-direction and more slices along the y-axis.

4.1.2 DEM performance

A test case was run for mono-sized particle of 2mm diameter for timing comparison studies. The times are plotted in Figure 4-1 indicate that using lower number of cores is not feasible for long simulations since the time taken while using 1 core is about 11x times slower. Thus, the simulations were carried out in core configurations of 64, 128 and 256 cores. The studies undertaken had 5 mono-sized population of particles of diameter 0.63, 1, 1.26, 1.59 and 2 mm simulations and one simulation consisted of particle size distribution. Figure 4-2 shows that the amount of CPU time required for a 10 second simulation of the granulator. The post processing MATLAB script was run on the dump files obtained from the simulation and it was observed that the system reached a steady state about 3-5 seconds of the simulation time. Particles with larger diameter reached steady state at a faster rate with an average hold-up of particles of about 6500. Pure timing studies are not really a good measure to represent the parallel performance of a program. Speedup of a parallel program indicates the speed increase of the program when it is run on more compute cores compared to the wall clock time when it is run in serial. It is the most common way to represent the parallel performance. Speedup is the ratio of the time taken to run the program in serial to the time taken by the program to run in parallel as shown in Equation 14. For an ideally parallelized program, the speedup is 'n' times, where n is the number of cores used.

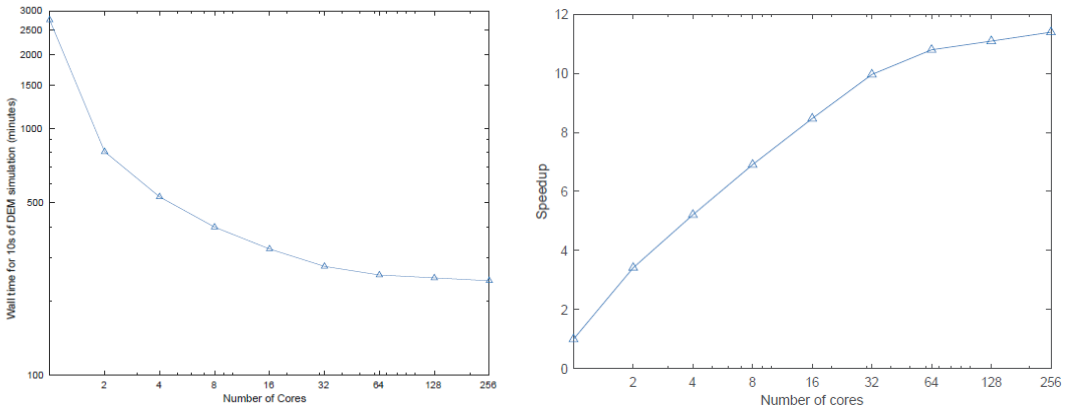
$$Speedup = \frac{Serial\ wall\ clock\ time}{Parallel\ wall\ clock\ time} \tag{14}$$

Speedup does not take into account number of processors used in the simulation, thus another metric that is used to determine the parallel performance is parallel efficiency. This metric is nothing by speedup divided by the number of cores used. Thus, parallel efficiency

normalizes speedup and gives a fractional value of the ideal speedup a program achieves with the increase in the number of cores.

$$Parallel\ Efficiency = \frac{Serial\ wall\ clock\ time}{Parallel\ wall\ clock\ time \cdot n_{cores}} \quad (15)$$

The speedup of the DEM simulations using only MPI cores is shown in Figure 4-1(b). There is a linear speed increase in the speedup up to 16 cores, after which the performance plateaus. Figure 4-1(b) indicates that there is not a significant amount of speed improvement for the simulation when 256 cores are used for the simulation over 128 cores. This speed decrease can be accounted to the communication time between the MPI processes. When the particles move from one section to another of the space, they are transferred as ghost particles from one process to another process. Thus, there is large amount of communication which is required. One of the issues of using a cluster with shared memory within the nodes but none in between nodes is that it has to rely on the networking infrastructure of the cluster which bottlenecks the communications thus leading to higher communication times. There are more sections present when 256 cores are used for the simulation, thus there is more communication in this system when compared to 128 core simulation. This excess communication makes the simulation slower though there is more processing power and it requires lesser time for other calculations. Another observation that can be made from Figure 4-2 is that the particle size distribution simulation takes more time compared to the simulations with mono-sized particles of 1.59mm and 2mm, though the mean size of particles in the distribution is 2mm. The default profile provided by LIGGGHTS indicates that the time spent in calculating the particle-particle interaction forces was higher than the mono-sized simulations. The different diameters make the interaction forces more tedious thus, making it computationally more expensive.



(a) (b)

Figure 4-1 (a) The variation in the amount of time taken for the DEM simulation of 2 mm particles as a function of number of cores. The improvements in the speed of the simulation was higher when the cores were increased from 1 to 16 when compared to speed improvements obtained with a core count greater than 16.(b) The speedup achieved in the DEM simulations of 2 mm particles. There was a linear increase up to 16 cores while the speedup seems to plateau when more number of cores are used due to larger amount of communications.

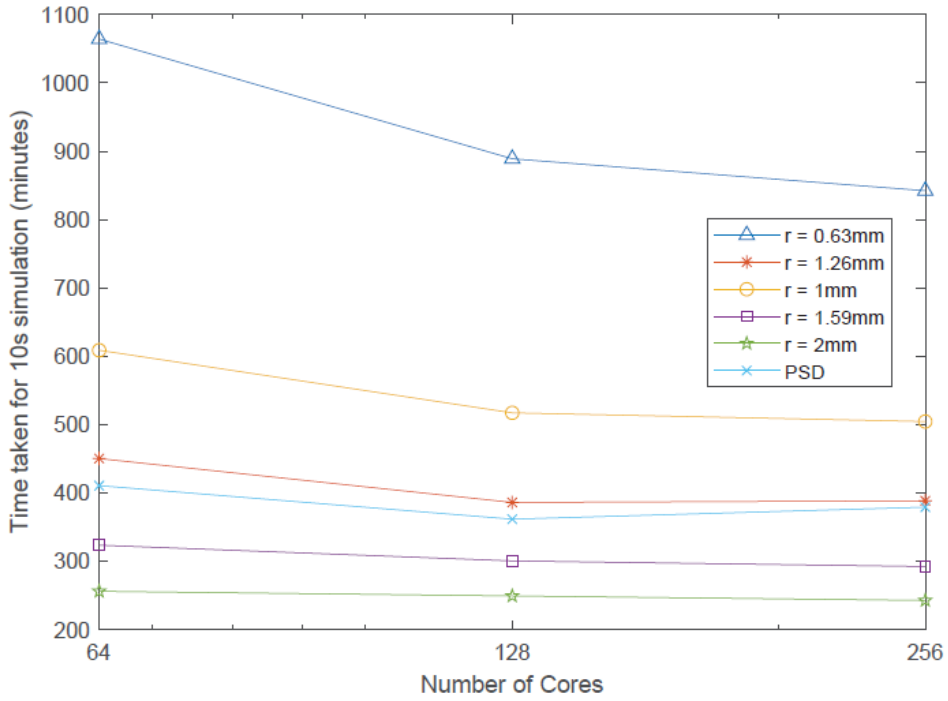


Figure 4-2: Time taken for a 10 second DEM simulation for radii ranging from 0.63 mm to 2 mm and a particle size distribution ranging from 1mm - 3mm. The speed improvements for the smaller particles was higher as the core count increased when compared to the larger particles. The larger number of smaller particles require more computational power thus benefit more as the number of cores are increased.

The communication time in LIGGGHTS is indicated by the Modify time, which is the sum of the times required to transfer the rotating mesh from one process to the other. From Figure 4-3, it can be seen that the main portion of the simulation time is taken up by the modify time. This is expected since the impeller is rotating at a very high speed of 2000 rpm. So, if the number of processes are increased the amount of time spent in transferring the mesh also increases. In the studies, the modify time as a percentage of the simulation increased from 82% to about 90%, when the core count was increased from the 64 cores to 256 cores. But, the using the higher number of cores reduces time taken to calculate particle-particle interaction as well as in neighbour calculation. Thus, a better implementation for meshing as well as decomposition of the geometry for faster simulations with higher core counts.

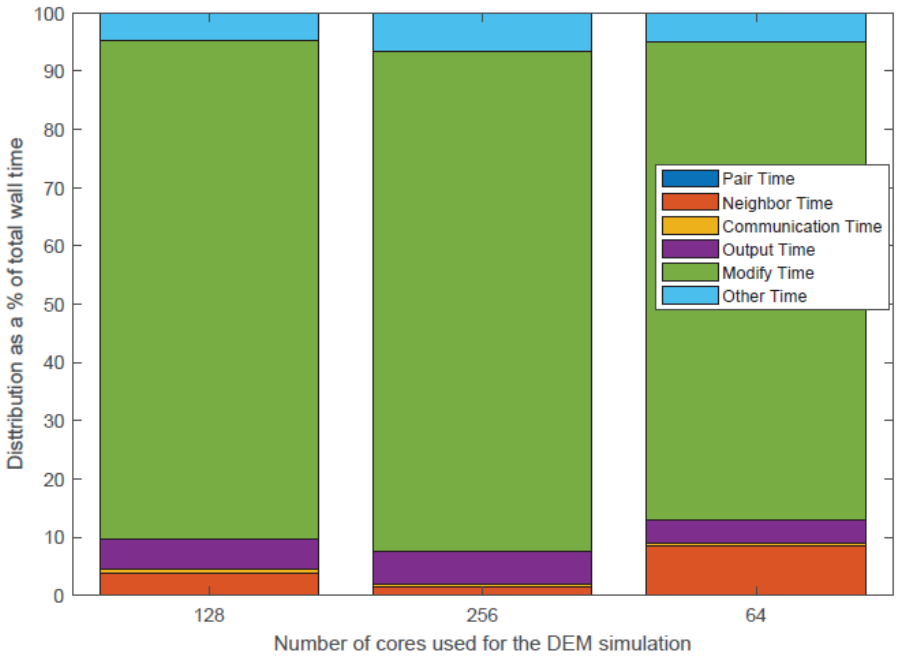


Figure 4-3: The distribution of time taken by each component of the DEM simulation with varying number of cores for the 2 mm particles. It can be observed that the percentage of time required to modify the geometry (indicated by the modify time), i.e. rotating the impeller at high speeds, was the highest.

4.2 Population Balance Model

4.2.1 PBM validation

The population balance model implemented was considered to have an inlet ow of particles in the first compartment at a constant mass ow rate of 15 kilograms per hour. The particles were assumed to have a log normal distribution with minimum diameter of the particles being about 12 _m and the maximum of 2.1 mm with a mean of 0.453 mm and a standard deviation of about 0.62 mm.

The PBM used in this study employs an aggregation kernel that takes into account the formation of a larger particle from only two smaller particles. The ratio of the rate of formation and the rate of depletion due to aggregation helps us monitor whether the PBM satisfies the conservation of mass. Since this PBM takes into account the aggregation of only 2 particles at once, the ratio of the particles is expected to have a value of 0.5 during the aggregation process. This ratio was reported by the PBM to be 0.5 throughout the simulation, validating the accuracy of the PBM used.

Figure 4-4 shows four different particle size distribution plots at four different time instants. Figure 4-4 (a) shows the distribution of the particles at 30 seconds, where we expect to have the highest number of the smaller particles. Since the degree of aggregation that has occurred is very low, there is a jump in the number of particles of the smaller size. The particle size distributions at 2 intermediate times of 50 seconds and 75 seconds are plotted in Figures 4-4 (b) & 124-4c) respectively. These illustrate that there is an increase in the number of larger particles and a subsequent decrease in the number of smaller particles. Figure 4-4 (d) shows the distribution of the particle size at the end of the granulation process. It can be seen that the number of particles in the higher diameter bins have increased.

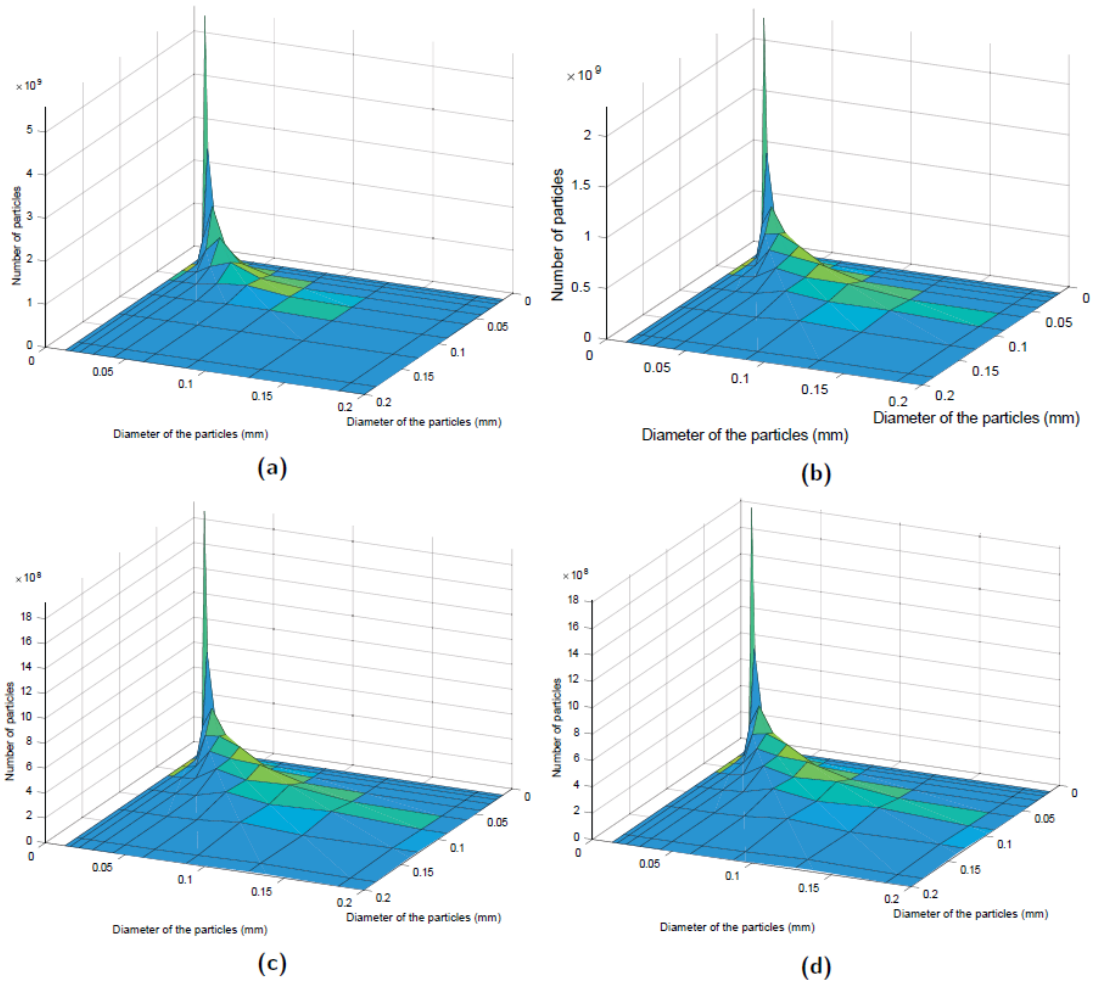


Figure 4-4: Representation of the total number of particles inside all compartments with diameters less than 0.2 mm in mm after (a) 30s (b) 50s (c) 75s and (d) 100s of PBM simulation (Mixing takes place for the first 25 seconds).

4.2.2 PBM performance

The PBM was run for the results obtained from each of the aforementioned DEM simulations. Since the PBM has been parallelized using hybrid techniques, a combination of MPI and OMP cores were used to perform the simulations. Figure 4-5 shows the average time taken by a PBM simulation for a total of 100 seconds of the granulation process, which includes 25 seconds of the mixing time and 75 seconds of granulation time. The time taken for simulating all DEM scenarios by a single set of core configuration of in less than 10% of each other, thus, an average time for a single core configuration was used to illustrate the performance. These simulations were run in a varied configuration of cores ranging from 1 to 128. The cores were

initially increased to 16 by increasing only the number of MPI processes. To increase the number of cores used, 8 OMP threads were employed for a configuration of 32(4 MPI and 8 OMP), 64(8 MPI and 8 OMP) and 128 cores (16 MPI and 8 OMP). Figure 4-5 shows that the program scales well to about 32 cores but then, the improvement in the performance is negligible. The scaling with the only MPI cores shows substantial increase in performance. Figure 4-6(a) depicts the speed up achieved by the hybrid parallel PBM code. It can be seen when the MPI cores used are increased from 1 to 16 cores the speed up achieved is almost linear. This speed can be accounted to the way MPI has been implemented inside the code. Each compartment of the granulator is offloaded on one MPI process, thus making 16 MPI processes the ideal since the granulator has been divided into 16 compartments. When less than 16 cores are used, more than 1 compartment is sent to a single MPI process, which leads to a decrease in performance. The implementation of OMP on top of the MPI parallelization helps improve the performance by the about 10%. The calculations inside the OMP parallel section of the code consists of large number of nested loop which have been known to be difficult to parallelize (He, Chen and Tang, 2016) using the native C++'s OMP libraries. The amount of communication time spent in between these threads is much higher than the speed increase achieved by using higher number of cores. One thread waits for another thread to complete processing the outer loops thus lower performance increase is achieved by using the OMP implementation.

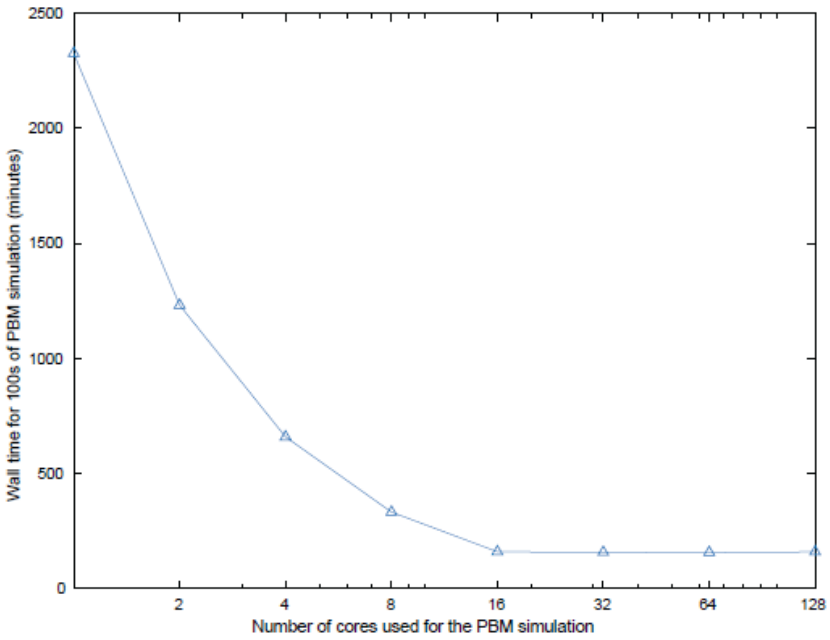
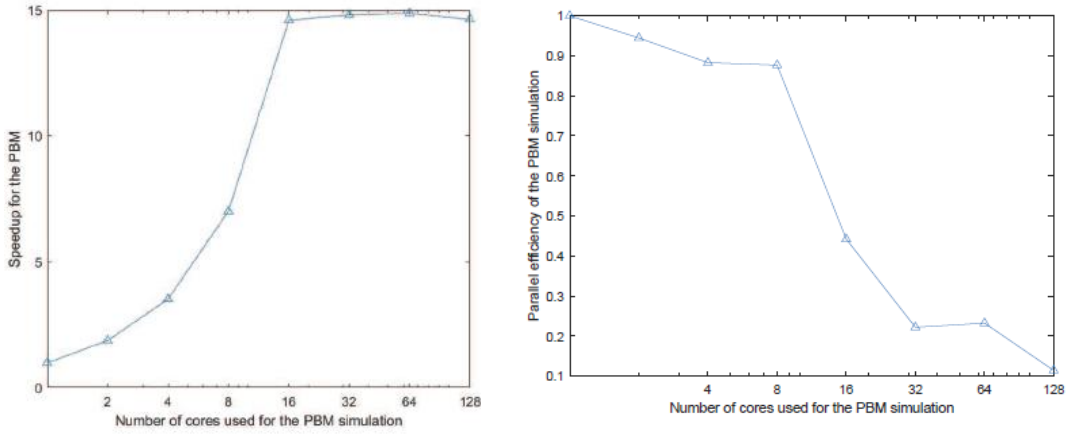


Figure 4-5: Average time taken to run the PBM simulation which consisted of 25 seconds of mixing time and 75 seconds of granulation time at different core configurations. There was a steady decrease as the number of MPI processes were increased, but the improvements on increasing the OMP threads were not that significant.

A similar behaviour as in Figure 4-6(b) where the parallel efficiency of the program has been plotted against the number of the cores used. This efficiency decreases as the number of cores used increase. The decrease in efficiency initially up till 8 cores is steep as the communication time in between the MPI process decreases the efficiency, deviating its value from the ideal of 1 to about 0.4. With the increase in the MPI processes to 16 and then the implementation of OMP the efficiency decreases as there is no major increase in the performance when compared to the increase in the number of cores used. The efficiency falls to as low as 11% when 128 cores are used. Thus, there is scope for improvement in the parallel implementation of the program using OMP, especially in section of the code where nested loops are present.



(a) (b)

Figure 4-6: (a) The speedup achieved by the hybrid parallel implementation of the PBM program. It can be seen the initial speedup up to 16 cores as expected as from an ideal parallel program where as it becomes almost constant from 32 cores to 128 cores (b) The parallel efficiency for the hybrid parallelized PBM code. The efficiency of the code decreased as the number of cores are increased. The higher number of cores have very low efficiency of about 11% which depicted that there is large time being spent in communication in between the cores.

4.2.3 PBM GPU performance

The GPU parallel code timings were not compared directly to the CPU version since it would have been unfair since the hardware configurations used were completely different. There are several non-code related latencies and delays that could affect the performance of the code. Thus, no comparison was performed between the 2 code versions.

A soft scaling was performed for the GPU code. This refers to change of the size of the problem is in direct correlation with the computational power required for its execution. In this study, the sections of geometry were varied from 2 to 16 compartments, while the number of solids from 1 to 16 for both the solids. Using this configuration, the maximum number of threads required would be 4096 while the minimum would be 2 threads. Figure 4-7 shows timing comparison between the studies. It can be seen that as the size of the problem increases there is a small increase in the time of the simulation which can directly be linked to increase in the number of calculations as well problem initiation times would be higher. A larger time is required for problems which require threads greater than the number of CUDA cores present

on the GPU (here 1792), since the extra blocks would need to wait for the earlier blocks to complete their simulations before they can be executed.

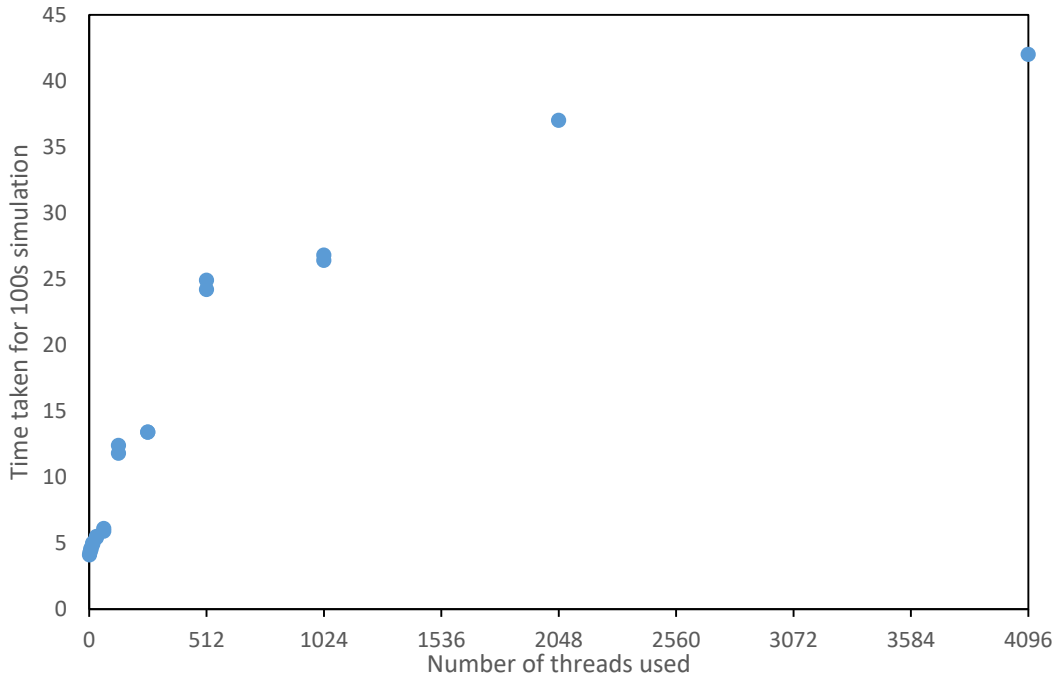


Figure 4-7: The benefit of using a GPU over a CPU is that chagning the size / complexity of the problem does not affect the performance significantly. Only when the threads need to wait in a GPU, a increase in computational time is observed

4.3 Coupling of DEM and PBM

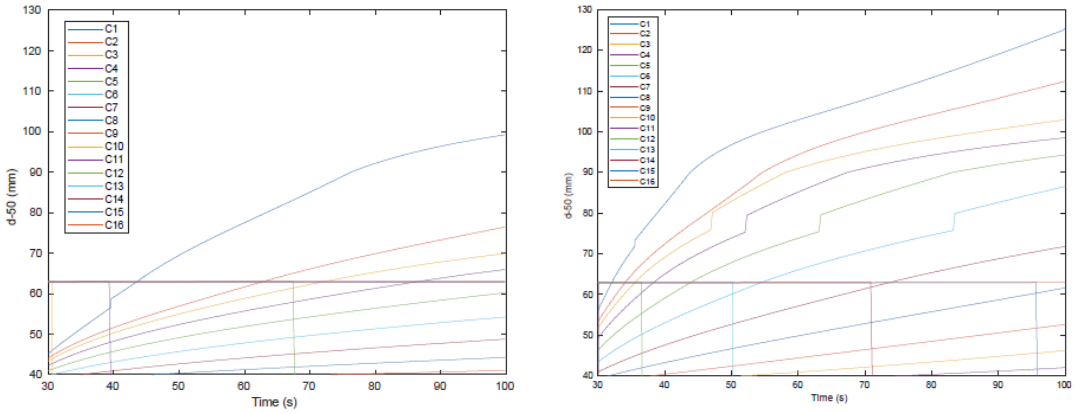
4.3.1 One way coupling

The micro-scale simulations provide an insight about the physics of the system usually by tracking each particle. This micro-scale simulation data is useful for the development of macroscale mechanistic models which take into account the dynamics of bulk of the particles and not individual particles. A similar approach has been implemented in this work, where a mechanistic aggregation kernel was developed from the DEM particle-particle collisions. Thus, the aim of this section is to illustrate that the physics of the system does not change to a great extent with the change in the size of the particles or the distribution of the particles.

Two simulations from the current scenario will be compared, the DEM + PBM simulation of the 2mm mono-sized particle and the second being the simulation where the inserted

particles were in a size distribution. The ratio of rate of formation to the rate of depletion, both due to aggregation observed during these simulations was 0.5 which indicated that the PBM is stable. This meant that the mono-sized and the PSD simulation were stable.

One of the metrics to determine the physics of the system after a PBM simulation is to check the median diameter (D50) plots of the system after the granulation process. D50 indicates the maximum diameter of particles that constitute 50% of the total mass. These diameters vary along the length of the granulator since the granulated particles take time to pass through the granulator and that there is not enough liquid content in the later sections of the granulator to encourage the formation of granules. Thus, the D50 for compartments in the latter section of the granulator is low. Figures 4-8(a) & 4-8(b) show the D50 plots of the mono-sized and PSD respectively. It can be seen that both these plots have a similar behaviour when it comes to the nature of the increase of the D50 during the granulation process. There is a difference of about 20% in the final diameter of the particles predicted, which at micrometre scale does not affect the final product quality. This slight deviation is observed due to the sudden jump in the rate of the aggregation which increases when particles from one compartment with higher D50 get transferred to the next compartment with a lower D50. The advantage of running the mono-sized simulation compared to the PSD simulation is the time taken to simulate the DEM. It can be seen from Figure 4-4 that the 2mm mono-sized particle took about 1.6 times less than the PSD simulation time for the DEM. Since the physics of both the systems are not different, using the mono-sized simulations for the DEM help save time on the overall simulations.



(a) (b)

Figure 4-8 (a) D50 of the particles obtained after 100 s of granulation time (25 s of mixing and 75 s of liquid addition) for the 2 mm mono-sized particle DEM simulation. (b) D50 of the particles obtained after 100 s of granulation time (25 s of mixing and 75 s of liquid addition) for the distributed particle size DEM simulation. The trend observed was similar to the trend found in Figure 4-8a.

4.3.2 Two way coupling

This coupled DEM-PBM simulation were tested on the Stampede2 supercomputer with the Knights Landing configuration. Each of the compute node of the cluster consists of Intel Xeon Phi 7250 "Knights Landing" which has 68 cores on a single socket clocked at 1.4 GHz, with 96 GB of DDR4 RAM. Each of the cores consists of 4 hardware threads. Simulations were run on various configuration of the cores used for each of the component of the system. The number of cores used for the DEM simulation were 64, while the PBM used 4 MPI processes. The DEM and PBM controller scripts used 1 process each for their respective executions.

The coupled system took 10 DEM and 9 PBM simulations to reach steady state. Total simulation time of DEM execution was 3.575 seconds and PBM was 60 seconds.

4.3.2.1 Improved system dynamics

The advantage of using a multiscale model in parallel is that the physical properties of the particles can be updated to the new values while moving among the simulations. This not only helps achieve better results but also reduces the time required for simulating the system. Granulation is used to control the size of the granules of a powder mixture. The median diameter of this powder mixture is usually used to characterise the behaviour of the system.

For better control, median diameter of this system should increase gradually such that the time of granulation can be capped.

Figure 4-9 shows the evolution of median diameter of the simulated wet granulation process inside a high shear granulator with time. A slow but steady increase initially in the diameter is observed which is followed by breakage of granules into smaller ones due to the high amounts of shear present as the impeller rotates at high RPMs. This method of prediction utilized dynamic data from the DEM simulations making the PBM simulations more real-time. The observed trend is better than previous works which only took into account only a serial coupling i.e. using the DEM collision information into the PBM for a single iteration.

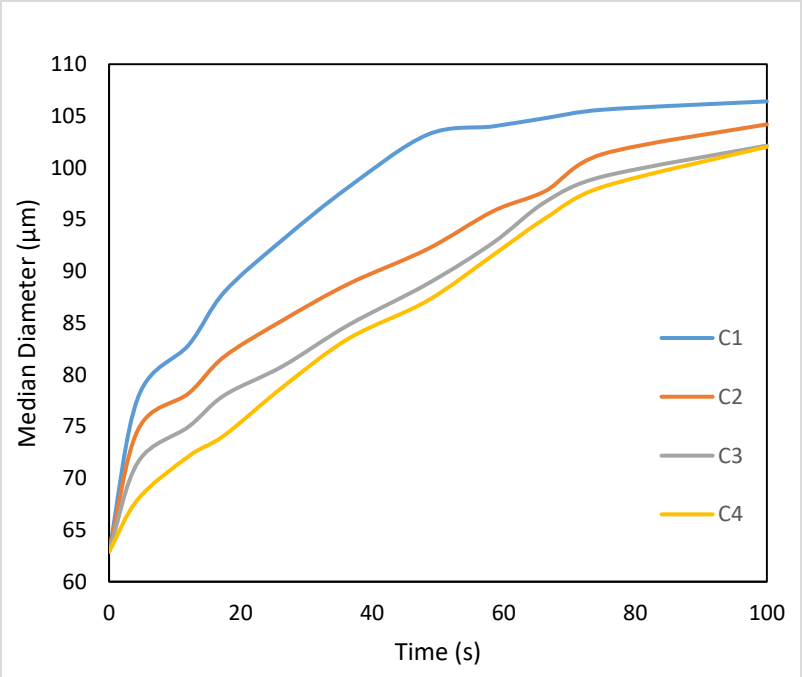


Figure 4-9: Median diameter of the system after a bi-directional DEM-PBM coupling as a function of time.

Chapter 5: Conclusions and Future Directions

This study presents various scenarios which can aid the modelling of a high shear granulator. In order to control the high shear granulation process, a model needs to be build that is not only accurate but also fast and efficient at predicting the outcomes. Various approaches undertaken in this study have been aimed towards achieving a fast, accurate and efficient model. The parallelisation of DEM using LIGGGHTS provided a quick, accurate solution to a rather computationally heavy problem within a manageable timespan. The mechanistic nature of the PBM also makes it more precise in prediction of process results and help detect anomalies in a better manner. Using the CPU or the GPU version of the code is dependent on the user as well as the hardware available. It can also been from chapter 4 that the median results for the bi-directional and the unidirectional only differ by about 5-10% depending upon the compartment which is fairly acceptable. Thus, the more efficient way would be use one way forward simulation results to increase the speed of the simulations.

One of the major drawbacks of this study is that the obtained values were not cross-checked with experimental values. A high importance should also be given to management of historical data since it would also aid decision making. A potential area of study could be statistical analysis of experimental data which can be fed into the model to obtain feasible ranges of operation. Further using this data regime maps could be developed to help understand the granulation dynamics in a better manner. This would all eventually increase the product quality and lead to the better manufacturing practices. This would also help reduce the number as well as the cost of expensive experimental studies that need to be undertaken. Thus, this study serves as a precursor to various developments that would eventually lead to an accurate model that could predict the granulation process in a faster and more accurate manner.

Appendix

The PBM CPU parallelization described in section 3.3.1 is depicted in an algorithmic manner below:

Algorithm 1 Parallel Population Balance Model

```
1: procedure POPULATION BALANCE MODEL(N_Comp,N_MPI,N_OMP)        ▷ N_Comp is the number of compartments
2:     Divide N_Comp in N_MPI
3:     while t < t_final do
4:         for ∀n_Comp in 1 MPI process do
5:             for ∀n in N_OMP do
6:                 Calculate ℜ_agg for solid bins s_1,s_2
7:             end for
8:             Calculate n_particles using Euler's method
9:         end for
10:        Collect n_particles from N_MPI                       ▷ Master process collects all data
11:        Calculate timestep using CFL condition
12:        t_new = t + timestep
13:    end while
14: end procedure
```

References

MathWorks™ Documentation (2017) 'Parallel Computing Toolbox - MATLAB®

Adams, G. G. and Nosonovsky, M. (2000) 'Contact modeling—forces', Tribology International. Elsevier, 33(5), pp. 431–442.

Adhianto, L. and Chapman, B. (2007) 'Performance modeling of communication and computation in hybrid {MPI} and {OpenMP} applications', Simulation Modelling Practice and Theory. Elsevier, 15(4), pp. 481–491.

Barrasso, D., Eppinger, T., Pereira, F. E., Aglave, R., Debus, K., Bermingham, S. K. and Ramachandran, R. (2015) 'A multi-scale, mechanistic model of a wet granulation process using a novel bi-directional PBM–DEM coupling algorithm', Chemical Engineering Science, 123, pp. 500–513. doi: 10.1016/j.ces.2014.11.011.

Barrasso, D. and Ramachandran, R. (2012) 'A comparison of model order reduction techniques for a four-dimensional population balance model describing multi-component wet granulation processes', Chemical engineering science. Elsevier, 80, pp. 380–392.

Barrasso, D. and Ramachandran, R. (2015) 'Multi-scale modeling of granulation processes: bi-directional coupling of {PBM} with {DEM} via collision frequencies', Chemical Engineering Research and Design. Elsevier, 93, pp. 304–317.

Barrasso, D., Walia, S. and Ramachandran, R. (2013) 'Multi-component population balance modeling of continuous granulation processes: a parametric study and comparison with experimental trends', Powder technology. Elsevier, 241, pp. 85–97.

Berger, R., Kloss, C., Kohlmeyer, A. and Pirker, S. (2015) 'Hybrid parallelization of the {LIGGGHTS} open-source {DEM} code', Powder Technology. Elsevier, 278, pp. 234–247.

Bettencourt, F. E., Chaturbedi, A. and Ramachandran, R. (2017) 'Parallelization methods for efficient simulation of high dimensional population balance models of granulation', Computers & Chemical Engineering. Elsevier, 107(Supplement C), pp. 158–170.

Boman, E. G., Çatalyürek, Ü. V, Chevalier, C. and Devine, K. D. (2012) 'The {Zoltan} and {Isorropia} parallel toolkits for combinatorial scientific computing: Partitioning, ordering and coloring', Scientific Programming. IOS Press, 20(2), pp. 129–150.

Cameron, I. T., Wang, F. Y., Immanuel, C. D. and Stepanek, F. (2005) 'Process systems modelling and applications in granulation: A review', Chemical Engineering Science. Elsevier, 60(14), pp. 3723–3750.

Chaturbedi, A., Bandi, C. K., Reddy, D., Pandey, P., Narang, A., Bindra, D., Tao, L., Zhao, J., Li, J., Hussain, M. and Ramachandran, R. (2017) 'Compartment based population balance model development of a high shear wet granulation process via dry and wet binder addition', Chemical Engineering Research and Design. Elsevier, 123, pp. 187–200.

Courant, R., Friedrichs, K. and Lewy, H. (1967) 'On the partial difference equations of mathematical physics', {IBM} journal of Research and Development. IBM, 11(2), pp. 215–234.

Cundall, P. A. and Strack, O. D. L. (1979) 'A discrete numerical model for granular assemblies', geotechnique, 29(1), pp. 47–65.

Gantt, J. A., Cameron, I. T., Litster, J. D. and Gatzke, E. P. (2006) 'Determination of

coalescence kernels for high-shear granulation using {DEM} simulations', Powder Technology. Elsevier, 170(2), pp. 53–63.

Goldschmidt, M. J. V, Weijers, G. G. C., Boerefijn, R. and Kuipers, J. A. M. (2003) 'Discrete element modelling of fluidised bed spray granulation', Powder Technology. Elsevier, 138(1), pp. 39–45.

Gopalakrishnan, P. and Tafti, D. (2013) 'Development of parallel {DEM} for the open source code {MFIX}', Powder technology. Elsevier, 235, pp. 33–41.

Gunawan, R., Fusman, I. and Braatz, R. D. (2008) 'Parallel high-resolution finite volume simulation of particulate processes', AIChE Journal. Wiley Subscription Services, Inc., A Wiley Company, 54(6), pp. 1449–1458. doi: 10.1002/aic.11484.

Hancock, B. C. and Ketterhagen, W. R. (2011) 'Discrete element method ({DEM}) simulations of stratified sampling during solid dosage form manufacturing', International journal of pharmaceutics. Elsevier, 418(2), pp. 265–272.

Hassanpour, A., Pasha, M., Susana, L., Rahmanian, N., Santomaso, A. C. and Ghadiri, M. (2013) 'Analysis of seeded granulation in high shear granulators by discrete element method', Powder technology. Elsevier, 238, pp. 50–55.

He, J., Chen, W. and Tang, Z. (2016) '{NestedMP}: Enabling cache-aware thread mapping for nested parallel shared memory applications', Parallel Computing. Elsevier, 51, pp. 56–66.

Ingram, G. D. and Cameron, I. T. (2004) 'Challenges in multiscale modelling and its application to granulation systems', Asia-Pacific Journal of Chemical Engineering. Wiley Online Library, 12(3–4), pp. 293–308.

Ingram, G. D. and Cameron, I. T. (2005) 'Formulation and comparison of alternative multiscale models for drum granulation', Computer Aided Chemical Engineering. Elsevier, 20, pp. 481–486.

Iveson, S. M., Litster, J. D., Hapgood, K. and Ennis, B. J. (2001) 'Nucleation, growth and breakage phenomena in agitated wet granulation processes: a review', Powder technology. Elsevier, 117(1), pp. 3–39.

Jin, H., Jespersen, D., Mehrotra, P., Biswas, R., Huang, L. and Chapman, B. (2011) 'High performance computing using {MPI} and {OpenMP} on multi-core parallel systems', Parallel Computing. Elsevier, 37(9), pp. 562–575.

Kačianauskas, R., Maknickas, A., Kačeniauskas, A., Markauskas, D. and Balevičius, R. (2010) 'Parallel discrete element simulation of poly-dispersed granular material', Advances in Engineering Software. Elsevier, 41(1), pp. 52–63.

Kandrot, E. and Sanders, J. (2011) Cuda By Example: An Introduction To General-Purpose Gpu Programming. Addison-Wesley Professional. Available at: https://books.google.com/books?id=6mwanQAACAAJ.

Keckler, S. W., Dally, W. J., Khailany, B., Garland, M. and Glasco, D. (2011) 'GPUs and the future of parallel computing', IEEE Micro. IEEE, 31(5), pp. 7–17.

Kloss, C., Goniva, C., Hager, A., Amberger, S. and Pirker, S. (2012) 'Models, algorithms and validation for opensource {DEM} and {CFD--DEM}', Progress in Computational Fluid Dynamics, an International Journal. Inderscience Publishers, 12(2–3), pp. 140–152.

Litster, J. (2016) Design and Processing of Particulate Products. Cambridge University Press.

Luckow, A., Santcroos, M., Merzky, A., Weidner, O., Mantha, P. and Jha, S. (2012) 'P*: A model of pilot-abstractions', in E-science, 2012 IEEE 8th International Conference on, pp. 1–10.

NVIDIA~Corporation (2012) 'NVIDIA CUDA C Programming Guide'. NVIDIA Corporation, 2701 San Tomas Expressway, Santa Clara, CA 95050. Available at: https://developer.download.nvidia.com/compute/DevZone/docs/html/C/doc/CUDA_C_Programming_Guide.pdf.

Prakash, A. V., Chaudhury, A., Barrasso, D. and Ramachandran, R. (2013) 'Simulation of population balance model-based particulate processes via parallel and distributed computing', Chemical Engineering Research and Design. Institution of Chemical Engineers, 91(7), pp. 1259–1271. doi: 10.1016/j.cherd.2013.01.017.

Prakash, A. V., Chaudhury, A. and Ramachandran, R. (2013) 'Parallel simulation of population balance model-based particulate processes using multicore CPUs and GPUs', Modelling and Simulation in Engineering, 2013. doi: 10.1155/2013/475478.

Prakash, A. V, Chaudhury, A., Barrasso, D. and Ramachandran, R. (2013) 'Simulation of population balance model-based particulate processes via parallel and distributed computing', Chemical Engineering Research and Design. Elsevier, 91(7), pp. 1259–1271.

Prakash, A. V, Chaudhury, A. and Ramachandran, R. (2013) 'Parallel simulation of population balance model-based particulate processes using multicore {CPUs} and {GPUs}', Modelling and Simulation in Engineering. Hindawi Publishing Corp., 2013, p. 2.

Ramachandran, R. and Barton, P. I. (2010) 'Effective parameter estimation within a multi-dimensional population balance model framework', Chemical Engineering Science. Elsevier, 65(16), pp. 4884–4893.

Ramachandran, R., Immanuel, C. D., Stepanek, F., Litster, J. D. and Doyle, F. J. (2009) 'A mechanistic model for breakage in population balances of granulation: Theoretical kernel development and experimental validation', Chemical Engineering Research and Design. Elsevier, 87(4), pp. 598–614.

Ramkrishna, D. and Singh, M. R. (2014) 'Population balance modeling: current status and future prospects', Annual review of chemical and biomolecular engineering. Annual Reviews, 5, pp. 123–146.

Reinhold, A. and Briesen, H. (2012) 'Numerical behavior of a multiscale aggregation model--coupling population balances and discrete element models', Chemical engineering science. Elsevier, 70, pp. 165–175.

Rogers, A. J., Hashemi, A. and Ierapetritou, M. G. (2013) 'Modeling of particulate processes for the continuous manufacture of solid-based pharmaceutical dosage forms', Processes. Multidisciplinary Digital Publishing Institute, 1(2), pp. 67–127.

Sen, M., Barrasso, D., Singh, R. and Ramachandran, R. (2014a) 'A multi-scale hybrid CFD-DEM-PBM description of a fluid-bed granulation process', Processes. Multidisciplinary Digital Publishing Institute, 2(1), pp. 89–111.

Sen, M., Barrasso, D., Singh, R. and Ramachandran, R. (2014b) 'A Multi-Scale Hybrid CFD-DEM-PBM Description of a Fluid-Bed Granulation Process', Processes. Multidisciplinary Digital Publishing Institute, 2(1), pp. 89–111. doi: 10.3390/pr2010089.

Sen, M., Dubey, A., Singh, R. and Ramachandran, R. (2012) 'Mathematical development and comparison of a hybrid {PBM-DEM} description of a continuous powder mixing process',

Journal of Powder Technology. Hindawi Publishing Corporation, 2013.

Sen, M. and Ramachandran, R. (2013) 'A multi-dimensional population balance model approach to continuous powder mixing processes', Advanced Powder Technology. Elsevier, 24(1), pp. 51–59.

Seville, J., Tüzün, U. and Clift, R. (2012) Processing of particulate solids. Springer Science & Business Media.

Shi, Y., Green, W. H., Wong, H. W. and Oluwole, O. O. (2012) 'Accelerating multi-dimensional combustion simulations using GPU and hybrid explicit/implicit ODE integration', Combustion and Flame. The Combustion Institute., 159(7), pp. 2388–2397. doi: 10.1016/j.combustflame.2012.02.016.

Szilágyi, B. and Nagy, Z. K. (2016) 'Graphical processing unit (GPU) acceleration for numerical solution of population balance models using high resolution finite volume algorithm', Computers & Chemical Engineering. Elsevier Ltd, 91, pp. 167–181. doi: 10.1016/j.compchemeng.2016.03.023.