Parallelization of pseudo-particle modeling and its application in simulating gas–solid fluidization

Jianxin Lu\textsuperscript{a,b}, Jiayuan Zhang\textsuperscript{b,*}, Xiaowei Wang\textsuperscript{b}, Limin Wang\textsuperscript{a,b}, Wei Ge\textsuperscript{b,*}

\textsuperscript{a} Graduate School of Chinese Academy of Sciences, Beijing 100049, China
\textsuperscript{b} State Key Laboratory of Multiphase Complex Systems, Institute of Process Engineering, Chinese Academy of Sciences, Beijing 100190, China

1. Introduction

With the limitation of experimental technology in exploring micro- and meso-scale mechanisms in chemical reactors and the fast development of computer capacity, numerical simulation has attracted more and more attention in the research of process engineering (Li \textit{et al.}, 2005). In recent years, discrete simulation has become a hot point in the research of particle–fluid systems. Pseudo-Particle Modeling (PPM) is a particle method proposed by Ge and Li in 1996 \cite{Ge, Li} that discretizes the fluid into a large number of smooth and rigid pseudo-particles, which are much smaller than the real particles but larger than molecules or atoms, as shown in Fig. 1. With this approach, the computation of fluid dynamics is transformed into a series of collisions among pseudo-particles (PPs), and thereby facilitating and unifying the interaction algorithms of three kinds of collisions among solid particles (SP) and pseudo-particles, i.e., SP–SP, PP–SP and PP–PP.

In PPM, each pseudo-particle has four properties: mass ($m$), radius ($r$), position ($\mathbf{P}$) and velocity ($\mathbf{v}$), among which $m$ and $r$ are constant in the whole process. In the simulation process, all the particles move synchronously in the same time step ($\Delta t$). In each time step, all particles move independently, possibly under some external forces. At the end of each step, if the distance between the centers of two particles $|\mathbf{P}_1 - \mathbf{P}_2|$ is less than the sum of their radii ($r_1 + r_2$), and they are moving towards each other, they will collide as two rigid and smooth particles. The new velocities after collision are as follows:

\begin{align*}
\mathbf{v}_1' &= \mathbf{v}_1 - \frac{2m_2}{m_1 + m_2} \frac{(\mathbf{v}_1 - \mathbf{v}_2)(\mathbf{P}_2 - \mathbf{P}_1)}{|\mathbf{P}_2 - \mathbf{P}_1|^2}(\mathbf{P}_2 - \mathbf{P}_1), \\
\mathbf{v}_2' &= \mathbf{v}_2 + \frac{2m_1}{m_1 + m_2} \frac{(\mathbf{v}_1 - \mathbf{v}_2)(\mathbf{P}_2 - \mathbf{P}_1)}{|\mathbf{P}_2 - \mathbf{P}_1|^2}(\mathbf{P}_2 - \mathbf{P}_1),
\end{align*}

where the indexes 1 and 2 mean the two different particles and $\mathbf{v}^*$ means velocity after collisions.

Many works have been done since PPM was proposed. For instance, PPM has successfully reproduced the “long time tail” phenomena at high gas concentrations \cite{Ge & Li}; it has also been used to simulate some classical flow such as plane Poiseuille flow and flow around static solid particles, with reasonable results \cite{Ge & Li}, indicating its feasibility and accuracy in discrete simulation of single-phase flows.

As for the application in particle–fluid systems, Ge and Li (2003) also simulated typical phenomena in fluidization, including clustering, slugging and bubbling, where periodic boundary condition was used due to limitations on computation capacity and hence on simulation scale in earlier times \cite{Ge & Li}.
Zhang et al. upgraded the algorithms of PPM and used it to simulate more realistic gas–solid systems (Ge, Zhang, Li, & Li, 2003; Zhang, Ge, & Li, 2004). Considering that a real system was connected to the outside, the periodic boundary condition was replaced by an open boundary condition, where the pressure and temperature of inlet and outlet can be controlled separately. Simulation of a fluidized bed with 2500 solid particles and $4.5 \times 10^5$ pseudo-particles successfully demonstrated the evolution process of heterogeneous structures (Zhang, 2004). Another more significant application of PPM is the verification of the stability criterion, $\text{N}_{st} = \min$, i.e., the stability condition of the energy-minimization multi-scale (EMMS) model (Zhang, Ge, & Li, 2005). Before 2004, all simulations with PPM were executed by serial computing.

Since the simulation capacity is much limited in serial computing with single CPU and the scale is still much smaller than a real system, it is but natural to develop a parallel algorithm of this method. The first attempt was made in 2003 by Zhang and Wang (Wang et al., 2005; Zhang, 2004) with 1-D spatial decomposition and dynamic load balancing. Due to the sequential and non-additive nature of the hard-sphere model, parallelization is more complicated as compared to traditional soft-sphere models. In this paper, a 2-D spatial decomposition algorithm of PPM for parallel simulation of gas–solid fluidization was carried out based on Zhang and Wang’s work, and then computation efficiency was optimized by analysis of the time consumption of the main operation functions before a final demonstration on a gas–solid system with 102,400 solid particles and $1.8 \times 10^7$ pseudo-particles.

### 2. Parallelization of hard-sphere PPM

#### 2.1. Spatial decomposition

In parallel computing, the key problem is to divide a large computation task into a certain number of parts which can be executed on independent CPUs simultaneously with some necessary communications at some intervals. In this way, the simulation scale can be enlarged while the increment of time consumption is endurable. Generally, there are three decomposition algorithms: Atom Decomposition, Force Decomposition and Spatial Decomposition (Plimpton, 1995). As the forces in PPM are short-range and our application objective is mainly fluidization system with a rather regular geometry, spatial decomposition was adopted in this work.

Since the fluidized bed is simplified as a rectangle in our 2-D simulation, and the particles only have collisions with adjacent particles due to the hard-sphere algorithm, the flow field is partitioned into sub-spaces along width and height (as shown in Fig. 2) and each sub-region is assigned to a unique processor, which deals with the calculation task of the particles belonging to its respective domain. Certainly, the processors must communicate but only with the four adjacent processors, i.e., the upper and the lower processors, and the left and right processors.

Compared to serial computation, a task assigning process is needed before parallel computation. That is, all particles (including SPs and PPs) are designated to a processor beforehand according to their initial positions. When a particle moves out of its sub-domain into a neighboring sub-domain, deleting and adding operation would be necessarily implemented in the corresponding processors.

#### 2.2. Parallelization algorithm

Fig. 3 shows the basic flowchart of parallel algorithm for gas–solid fluidized bed together with the major functions (or steps) of the program. Different from serial computing, special treatments (e.g., those highlighted by red italic in Fig. 3) need to be done for parallel computing, and even more, for parallel algorithm in hard-sphere model.

In the initialization progress, the uniform distributions and random velocities are given to both PPs and SPs. For parallel computing, spatial decomposition also needs to be done here and both SPs and PPs are stored in their linked lists in each processor.

During the computing period, in each step the PPs and SPs first move under the external forces and their displacements are calcu-
Fig. 3. Flowchart for parallel computing.

lated. Here if any particle moves out of its sub-domain, it will be exchanged to a new destination processor; that is, deleted from the original processor's particle list and inserted to the new processor's list.

Then, for the new distribution, interactions between particles are calculated. There are three types of regions in each sub-domain: inner region, boundary region type A that is shared by two adjacent processors (i.e., the black shadowed in Fig. 4) and boundary region type B that is shared by four adjacent processors (i.e., the blue region in Fig. 4) and only exists in 2-D spatial decomposition. For particles in the inner region, the calculation is the same as that in serial computing. For particles in boundary regions, exchange of particles’ information in neighboring boundaries should be executed first. As in 1-D spatial decomposition, there are boundary regions A between two adjacent processors, moreover, for 2-D spatial decomposition, there are also boundary regions B for boundary calculation. Details of computation of particles in boundaries are discussed in Section 2.3.

After computing all potential collisions, a step for adjustment of temperature and pressure of the whole system is followed to keep the system in constant operation condition (Zhang et al., 2004). This step needs a global communication between processors. All the processors will send their local information (velocities, etc.) to the root processor. Then the root processor will calculate the average velocity, etc. of the flow and send the scaling parameters to all other processors.

Afterwards, a step of dynamic load balancing is performed to keep equivalent numbers of particles and thus equivalent computing times in each processor as detailed in Section 2.4.1. This step is very important to assure the high efficiency of parallel computing.

2.3. Exchange information and collisions in boundaries

In order to reduce the times spent on exchanging information between adjacent processors and then reduce the communication cost, a bidirectional exchange mode is used as detailed in Fig. 4.

Step I: Collisions are executed but only on particles located between the upper and lower boundary regions whose thickness is one solid particle diameter $d$.

Step II: Collecting the particles in both upper and lower boundary regions and then sending the boundary particles to its neighbor processors and receiving boundary particles from its neighbor processors at the same time.

Step III: Computing the interactions of the particles in the boundary regions with a thickness of $2d$.

Step IV: For 2-D spatial decomposition, exchanging the boundary information for the second time and computing the particles interaction in Region B.

Note that the same collisions are executed in adjacent processors in Step III & IV and the same particles in adjacent processors have the same properties, and therefore, only one exchange operation (i.e., Step II) among boundary regions is needed in our simulation. Since the solid particle is much larger than pseudo-particles, the thickness of the boundary region is defined by the dimension of solid particles. As particles are binned to cells having an edge length of solid particle diameter $d$, this thickness is also chosen to be $d$. Furthermore, because the particles in the nearby cells need to collide with the particles in the boundary regions, a border with thickness of $2d$ is to be exchange between adjacent processors as shown in Fig. 4. That is, the thickness of boundary region in each processor is $1d$, but particles in a border region of $2d$ are to be exchanged. After exchanging, binning particles to cells and searching particles in each cell are operated with the same sequence in adjacent processors, then the same particles in different processors will encounter exactly the same collisions and have the same properties.

In 2-D spatial decomposition, there is an extra step for boundary particles calculations. This boundary is shown as Region B (blue shadowed) in Fig. 4. The particles’ velocities in this region will be
updated and the particles in red rectangle are necessarily needed for this computation. In Fig. 4 particles in the black shadowed region (Region A) are first updated. But for each processor, such as P(m,n), it only holds the updated boundary information of boundaries 1 and 2. Hence for the blue region calculation, the relative updated boundary information are needed from boundaries 3 and 4. Thus a second time exchange is introduced and the interactions on the corner particles are computed.

2.4. Optimization of computation efficiency

Large-scale simulation by parallel computing may cost considerable amount of computing time and electric power; therefore, in order to avoid unnecessary consumption, computation efficiency of the parallelized algorithm need to be optimized prior to practical application.

2.4.1. Dynamic load balancing

One key feature of fluidized system is the multi-scale heterogeneous structures, especially the heterogeneous structure in axial direction that makes each processors hold different particle amounts. Since the computing load in each processor is roughly proportion to the number of particles therein, the processors with fewer particles have to wait for those with more particles in each step for synchronization, which will undoubtedly lead to a low efficiency of parallel computation.

In our simulation of a gas–solid fluidized bed, as mentioned in Section 2.1, a uniform spatial decomposition was initially designated due to our homogeneously initial distribution of the particles in the flow field. Due to the spontaneous evolution of heterogeneous structure, different computation load must occur and get more serious as the computation proceeds, which will results in an obvious decrease of computation efficiency.

To maintain a relative uniform load in each processor, it is necessary to add a dynamic load balancing mechanism in the parallel algorithm, which is to adjust the borders between processors to make each processor have almost the same number of particles. Wang et al. (2005) used DLB for the parallel algorithm of a soft-sphere model and improved the parallel efficiency up to 8%. This technology was also used in our algorithm to generate a non-uniform spatial decomposition as shown in Fig. 2c and an efficiency improvement about 30% was achieved as shown in Fig. 7.

2.4.2. Resetting particle array

In order to figure out the main factors affecting the program’s efficiency, we recorded and analyzed the time cost of each function in the program in detail. The time distribution is shown in Fig. 5, where a gas–solid fluidization system containing $10^5$ solid particles and $2.6 \times 10^6$ pseudo-particles was simulated. It was decomposed

Fig. 4. Boundary exchange and computing.

Fig. 5. Time cost ratio of the major functions (time step is 71,000).
It is obvious that the most time-consuming functions are sending and receiving boundary information, pseudo-particle collisions, temperature and velocities control, binning particles to cells, pseudo-particle movements, etc. After a further analysis of the functions above, one common feature was found that they all visited the data list which held pseudo-particle information. Then we realized that as the data array of pseudo-particles became more and more disordered due to particle migration between adjacent processors. In the program, a particle is always added at the tail of the data array, which becomes consequently longer and longer but discontinuous in the memory after a time period with interspaces left by particles moved to another processor.

As the array becomes longer, it will cost more and more time to search the empty position on the data chain as shown in Fig. 6a and b, and it will also exhaust memory resource at certain time. To solve this problem, we reset the particle array every certain time steps as shown in Fig. 6c by gathering all the disordered particles from the data list, and then arrange them in a new array, so the particles become continuous again in the memory. This treatment together with DLB has got significant effect in saving the computing time more than 50%, as shown in Fig. 7.

3. Simulation of a gas–solid fluidization system

Gas–solid circulating fluidized bed is a typical particle fluid system, which is very common in chemical engineering processes. To investigate the microscopic mechanism of the gas–solid interactions, and also the multi-scale properties of the system, a gas–solid fluidized bed of typical size was simulated. It also served as an exemplification of the parallelization.

Fig. 8 shows some snapshots of the simulated system containing 102,400 solid particles and $1.8 \times 10^7$ pseudo-particles. Spatial decomposition topology is 6 by 10 divisions in width and height respectively. The simulation was taken on ten nodes equipped with dual Intel® Xeon® E5430 quad-core CPUs and 16GB DDRII-800
Table 1
Major parameters of the simulated system.

<table>
<thead>
<tr>
<th></th>
<th>Dimensionless</th>
<th>Dimensional</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>18,000</td>
<td>6.8 m</td>
</tr>
<tr>
<td>H</td>
<td>3960</td>
<td>1.56 m</td>
</tr>
<tr>
<td>W</td>
<td>102,400</td>
<td>102,400</td>
</tr>
<tr>
<td>n_s</td>
<td>102,400</td>
<td>102,400</td>
</tr>
<tr>
<td>n_p</td>
<td>1.8e7</td>
<td>0.9024</td>
</tr>
<tr>
<td>ε</td>
<td>0.9024</td>
<td>0.9024</td>
</tr>
<tr>
<td>v_0</td>
<td>0.08</td>
<td>4.8 mm</td>
</tr>
<tr>
<td>d_s</td>
<td>12.0</td>
<td>0.4 mm</td>
</tr>
<tr>
<td>d_p</td>
<td>1.0</td>
<td>9.81 m/s^2</td>
</tr>
<tr>
<td>g</td>
<td>6e-8</td>
<td></td>
</tr>
<tr>
<td>Re</td>
<td>0.0044</td>
<td>0.0026</td>
</tr>
<tr>
<td>Re_p</td>
<td>0.95</td>
<td>1</td>
</tr>
<tr>
<td>Re_g</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>u_g</td>
<td>1.125 m/s</td>
<td>1.293 kg/m^3</td>
</tr>
<tr>
<td>u_s</td>
<td>0.6649 m/s</td>
<td>824.5 kg/m^3</td>
</tr>
<tr>
<td>α_s</td>
<td>0.95</td>
<td>1</td>
</tr>
<tr>
<td>α_w</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>α_g</td>
<td>1</td>
<td>214.4</td>
</tr>
<tr>
<td>Re</td>
<td>0.1812</td>
<td></td>
</tr>
</tbody>
</table>

Re_p = ε(U_g - U_s)d_s/v_0.

Fig. 9. Distributions of solids concentration and velocities. (a) Axial distribution of solid particles; (b) radial distribution of solid particles; (c) radial distribution of solids velocity.

memory, which are connected by Gigabit Ethernet. The simulation parameters are listed in Table 1.

The high resolution of discrete simulation on micro scale makes it possible to investigate the meso- and macro-structure of particle–fluid systems. The gas–solid system is dominated by the floatage of the gas flow and the gravity of the solids. The underlining mechanism is that the gas flow tends to blow up the solid clusters and make channels for continuous flow, thus reducing the flow resistance, while the gravity forces the solids to drop down. As a result, the radial and axial heterogeneous structures form spontaneously. The solids are motivated by both the floatage and gravity, forming the axial distribution of dilute flow in upper space and dense flow in the bottom.

Fig. 9 shows the axial (a) and radial (b) distribution of solids concentration and radial distribution of solids velocities (c) in steady state respectively, which were reasonable in real system and also consistent with our simulation results by serial computing (Zhang et al., 2004), indicating the success of 2-D parallelization of hard-sphere PPM.

4. Conclusions

In this paper, a parallel algorithm for 2-D hard-sphere pseudo-particle modeling was implemented successfully with spatial decomposition. Dynamic load balancing was used based on the number of particles in each processor, and the data array of particles was reset at certain time interval to keep its continuity. By means of these two improvements, the computation efficiency was increased up to about 50% on the tested system. As a demonstration of our paralleled program, a gas–solid fluidized system with 102,400 solid particles and 1.8 × 10^7 pseudo-particles was simulated and reasonable results have been obtained. This work established a basis for
the scalability of PPM, and our next work is to upgrade the algorithm to 3-dimensional and achieve more realistic and large-scale simulation.

Acknowledgements

We thank the Designated Funding for Winners of President’s Awards of Chinese Academy of Sciences (CAS, 2006) and financial supports from the National Natural Science Foundation of China (NSFC) under the Grant No. 20221603 and 20706057.

References


