Journal of Computational Physics 281 (2015) 844–863

Contents lists available at ScienceDirect

Journal of Computational Physics

www.elsevier.com/locate/jcp

Accelerating population balance-Monte Carlo simulation for coagulation dynamics from the Markov jump model, stochastic algorithm and GPU parallel computing

Zuwei Xu, Haibo Zhao *, Chuguang Zheng

State Key Laboratory of Coal Combustion, Huazhong University of Science and Technology, 1037 Luoyu Road, Wuhan 430074, PR China

ARTICLE INFO

Article history:
Received 1 July 2014
Received in revised form 16 October 2014
Accepted 28 October 2014
Available online 31 October 2014

Keywords:
Population balance
Monte Carlo
Coagulation
Markov jump
Majorant kernel
GPU parallel computing

ABSTRACT

This paper proposes a comprehensive framework for accelerating population balance-Monte Carlo (PBMC) simulation of particle coagulation dynamics. By combining Markov jump model, weighted majorant kernel and GPU (graphics processing unit) parallel computing, a significant gain in computational efficiency is achieved. The Markov jump model constructs a coagulation-rule matrix of differentially-weighted simulation particles, so as to capture the time evolution of particle size distribution with low statistical noise over the full size range and as far as possible to reduce the number of time loopings. Here three coagulation rules are highlighted and it is found that constructing appropriate coagulation rule provides a route to attain the compromise between accuracy and cost of PBMC methods. Further, in order to avoid double looping over all simulation particles when considering the two-particle events (typically, particle coagulation), the weighted majorant kernel is introduced to estimate the maximum coagulation rates being used for acceptance–rejection processes by single-looping over all particles, and meanwhile the mean time-step of coagulation event is estimated by summing the coagulation kernels of rejected and accepted particle pairs. The computational load of these fast differentially-weighted PBMC simulations (based on the Markov jump model) is reduced greatly to be proportional to the number of simulation particles in a zero-dimensional system (single cell). Finally, for a spatially inhomogeneous multi-dimensional (multi-cell) simulation, the proposed fast PBMC is performed in each cell, and multiple cells are parallel processed by multi-cores on a GPU that can implement the massively threaded data-parallel tasks to obtain remarkable speedup ratio (comparing with CPU computation, the speedup ratio of GPU parallel computing is as high as 200 in a case of 100 cells with 10 000 simulation particles per cell). These accelerating approaches of PBMC are demonstrated in a physically realistic Brownian coagulation case. The computational accuracy is validated with benchmark solution of discrete-sectional method. The simulation results show that the comprehensive approach can attain very favorable improvement in cost without sacrificing computational accuracy.

* Corresponding author. Tel.: +86 27 87542417 8208; fax: +86 27 87545526.
E-mail address: klinsmannzhb@163.com (H. Zhao).

http://dx.doi.org/10.1016/j.jcp.2014.10.055
0021-9991/

1. Introduction

Coagulation between particles is ubiquitous in many different fields of nature and engineering [1], including atmospheric physics (aerosol dynamics), combustion (the growth of particulate matter, soot and PAH), chemical engineering (e.g., polymerization, granulation, and crystallization), nanoparticles synthesis, and so on. A particle coagulation event refers to the process that two particles coalesce to form a bigger particle, leading to the increase of particle size and the decrease of particle number, i.e., the dynamic evolution of particle size distribution (PSD). Among the various particle dynamic events, coagulation is the most demanding event for modeling, as it always involves two particles. The population balance equitation (PBE) for particle coagulation, which characterizes coagulation dynamics in term of the time evolution of PSD, is represented by the following mathematical equation (Smoluchowski coagulation equation):

$$\frac{\partial n(v,t)}{\partial t} = \frac{1}{2}\int_{v_{\min}}^{v} \beta(v-u,u,t)n(v-u,t)n(u,t)\mathrm{d}u - n(v,t)\int_{v_{\min}}^{v_{\max}} \beta(v,u,t)n(u,t)\mathrm{d}u, \tag{1}$$

where $n(v,t)$ with dimension $\mathrm{m}^{-3}\,\mathrm{m}^{-3}$ is the particle size distribution function (PSDF) at time t, so that $n(v,t)\mathrm{d}v$ is the number concentration of particles with size range between v and $v+\mathrm{d}v$ at time t; $\beta(v,u,t)$ is coagulation kernel for two particles of volume v and u at time t, $\mathrm{m}^3\,\mathrm{s}^{-1}$. The first term on the right-hand of Eq. (1) is the birth term, accounting for the formation of a particle of volume v due to the coagulation event between a particle of volume u (smaller than v) and a particle of volume $(v-u)$; and the coefficient $1/2$ is raised from the fact that one coagulation event is related to two particles. The second term is the death term, representing the disappearance of a particle of volume v due to coagulation event with any other particle.

Because of the partial integro-differential nature of the PBE, it is difficult to solve it directly. Only for a few ideal cases can we get analytical solutions, otherwise we can only get approximate solutions by numerical methods. The deterministic scheme such as sectional method and method of moments [2,3] is capable of solving Eq. (1) either through an appropriate discretization scheme or by quadrature. However, there are some difficulties and challenges such as complicated mathematical models (especially for multivariate population balance) and discrete errors for these deterministic methods [4,5]. A practical approach to model these systems is to use Monte Carlo (MC) simulation, which is based on random sampling statistical theory and describes the dynamic evolution of discrete system in a natural way. The population balance-Monte Carlo (PBMC) method is able to deal with high-dimensional problem in a simple and straightforward manner. Similar to the Lagrangian particle tracking method, PBMC can gain the information about particle history and trajectory crossing and can obtain the details of the dynamic evolution of particles and describe the multi-dimensional, multi-component, and polydispersed particle population [6–9]. Owing to the above-described advantages, PBMC constitutes an important class of methods for the numerical solution of the PBE. There are many efficient Monte Carlo methods for the solution of PBEs [10–14]. These methods consist in an artificial realization of the population dynamics of a finite system (finite number of particles in a very small volume) to estimate the properties of the whole system. The estimate becomes exact as the number of particles approaches infinity.

However, there are two drawbacks associated with the MC methods: uncertain statistical noise and large computational cost. It is very important for the MC methods to utilize limited number of simulation particles to reach an acceptable accuracy. In recent years several PBMC methods have been developed to keep the total number of simulation particles within prescribed bounds, even though the number concentration of real particles changes dramatically. Prominent among these are the constant-number method by Matsoukas et al. [13,15] and the differentially-weighted MC (DWMC) method by us [16,17] and other scientists [18,19]. The constant-number method can constrain the statistical noise and computational cost within an acceptable range. The differentially-weighted method captures the coagulation dynamics in dispersed systems with low noise and is simultaneously able to track the size distribution over the full size range [16]. The stochastic weighted particle methods [18] and weighted flow algorithms [19] are based on systems of weighted simulation particles and the weight transfer functions are constructed such that the number of computational particles does not change during coagulation events. With regard to the differentially-weighted MC method, the simulation particles have different private weights. The event probability of one simulation particle relates to not only its rate of corresponding dynamic process but also its private weight. The total number of simulation particles is kept constant by adjusting their private weights to the corresponding dynamic events [8].

For the two-particle events such as coagulation, the PBMC has to calculate/update the interaction probability of any particle pair real-timely (to obtain probability distribution of random events and the waiting time between two successive events). The double looping over all simulation particles is thus required in normal PBMC methods, and the computational cost reaches $O(N_s^2)$, where N_s is the simulation particle number. Broadly speaking, there are two classes of ways to accelerate MC simulation: one is to design a set of tricks to reduce the computational complexity; another is parallel processing of cumbersome problems. In fact, the parallel computing uses more computing devices simultaneously to reduce computational time, depending too much on hardware resources. Therefore, it is more widely developed to design the lightweight algorithm of PBMC in the past. Kruis et al. [11] proposed the smart bookkeeping technology to avoid a large number of recalculations of the coagulation rates of particles not participating in coagulation, in such a way that the CPU time is greatly saved without loss in accuracy. Wagner et al. [20,21] and Kraft et al. [14] developed an efficient MC which utilized the

majorant kernel to calculate the coagulation rate without double looping over all particles. The CPU time increases linearly with N_s, rather than as N_s^2 (as with the conventional MC). Wei et al. [22] present a fast acceptance–rejection scheme that can boost the performance of Monte Carlo methods for particle coagulation by establishing a connection between the information of particle pairs and the maximum coagulation rate. Lécot et al. [23,24] used the quasi-Monte Carlo to accelerate the convergence rate, in which pseudo-random numbers were replaced by quasi-random numbers or low-discrepancy point sets (which are "evenly distributed"). Similar idea in terms of good lattice point set was also used by Kruis et al. [25] to estimate the maximum of coagulation kernel with a remarkable gain in efficiency. We [26] introduce a weighted majorant kernel to estimate the maximum coagulation rate being used for acceptance–rejection process by a single looping over all particles, and meanwhile the mean time-step of coagulation event is estimated by summing the coagulation kernels of rejected and accepted particle pairs. Hence the double looping of particles is avoided and there is a linear dependence of computational time with the number of simulation particles. Another measure to accelerate PBMC simulation is to simulate coagulation between particle species rather than between discrete simulation particles, which exhibits an excellent improvement in efficiency and memory demand [19,27–33]. However, their computational accuracy is generally lower than the conventional particle-based MC.

To speed up computations, a single problem or "task" can be split into multiple subtasks, which are executed simultaneously on multiple processors. As a matter of fact, one of the most effective uses of parallel architectures (typically, including CPU-based high performance clusters (HPC), GPU (graphitic processing unit), GPU-based clusters, etc.) is to simply perform independent MC simulations of different processors [34]. A recent trend in computer science and related fields is the use of general purpose computing on GPU [35–37]. NVIDIA® has developed a general purpose parallel computing architecture, CUDA™ (Compute Unified Device Architecture), which carries out the parallel computing in NVIDIA GPU to solve many complex computational problems in a fraction of the time required on a CPU. Parallel processing with CUDA has exhibited enormous potential for high-performance and cheap computing in terms of the low electricity consumption and minimal requirements for work space [35]. For some traditional computation–intensive PBMC methods, e.g., the acceptance–rejection MC [38], the time-driven DSMC [10], the smart bookkeeping DSMC [11], the event-driven constant-volume method [17] etc., the abundant data parallelism embedded within the MC method is realized as it will allow an efficient parallelization on multi-core GPU [35,37]. Presently, most of accelerating PBMCs by GPU parallel computing only deal with zero-dimensional system in a single grid (or cell). However, an accurate modeling the particle dynamics need to obtain the spatiotemporal evolution of particle population in spatially inhomogeneous systems. In this sense, it is even more necessary to accelerate the PBMC in spatially inhomogeneous systems by GPU parallel computing. Actually, the greatest significant advantage of GPU parallel computing is the fact that it can simultaneously process multi-grid (or cell) problem (e.g., spatially inhomogeneous system [8]). A multi-cell system that is often used in chemical engineering modeling tools such as computational fluid dynamics (CFD) is a typical example in which the MC simulations can be made independently of multiple processors due to the rich data parallelism embedded [36]. Kruis et al. [25] present a parallel computing for CFD-based stochastic aerosol modeling, utilizing HPC system based on a Linux cluster, whereas the parallel efficiency is not especially high and can be made more efficient on GPU.

In this paper, we present a comprehensive framework to accelerate PBMC simulation of particle coagulation dynamics from three aspects, namely, Markov jump model, stochastic algorithm and GPU parallel computing. Firstly, the Markov jump model constructs a coagulation-rule matrix of differentially-weighted simulation particles, and this investigation focuses on the three typical coagulation rules (i.e., the lower-bound jump, the upper-bound jump, the median jump). The number of time loopings is reduced by improving the efficiency of adjustable Markov jump. The constant-number scheme is adopted to maintain sample space and guarantee stochastic accuracy of PBMC simulation. Secondly, the weighted majorant kernel is introduced into the Markov jump model to build fast stochastic algorithms, which increases greatly the efficiency of selecting particle pairs and performing coagulation event in acceptance–rejection process. Thirdly, GPU acceleration implements the massively threaded data-parallel processing tasks for multi-cell system (as an example, 100 cells), and Markov jump model and weighted majorant kernel are performed in each cell. A physically realistic case with Brownian coagulation kernel in the free molecular regime is simulated. The computational accuracy is validated using the discrete-sectional method as benchmark solution. The computational efficiency is measured by comparing with Monte Carlo methods using other schemes.

2. Methodology

2.1. Markov jump model: the coagulation-rule matrix of differentially-weighted simulation particles

The concept of weighting simulation particles is widely utilized by the PBMC to overcome the conflict between large numbers of real particles and limited CPU speed and memory capacity. The key of PBMC simulation is how to construct an appropriate Markov jump model. Based on the model, a rule for imagining a coagulation event between two simulation particles is designed and the coagulation rate between simulation particles having different weights is derived. In order to use differentially-weighted simulation particles, we have proposed two coagulation rules successively: the full coagulation rule and the probabilistic coagulation rule. With respect to the full rule, it is specified that each real particle from two coagulated simulation particles must coagulate and coagulates only once. As far as the probabilistic coagulation rule is concerned, each real particle from simulation particle i undergoes a real coagulation event with a probability of $\min(w_i, w_j)/w_i$, and

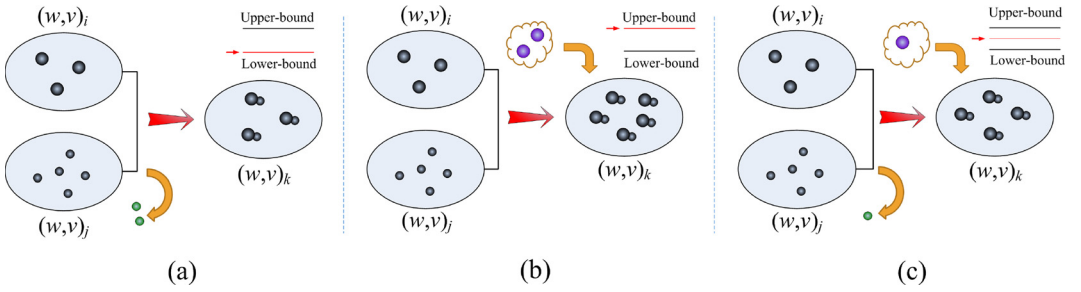

Fig. 1. Schematic representation of coagulation rules of Markov jump model: (a) the lower-bound jump; (b) the upper-bound jump; (c) the median jump.

each real particle from j does so with a probability of $\min(w_i, w_j)/w_j$, where w_i and w_j are the private number weights of simulation particle i and j, respectively. Thus, on average, only $\min(w_i, w_j)$ real particles from i or j participate in real coagulation. Based on the two rules, we have developed the multi-Monte Carlo method [39–42] and differentially-weighted MC method [16,17,43] respectively. These differentially-weighted methods were validated to exhibit low statistical noise in capturing the temporally-dependent coagulation dynamics and be able to track the size distribution over the full size range [16].

Here we consider a general Markov jump model. Based on general asymmetric coagulation algorithms [19], an appropriate Markov jump model could be constructed, so as to derive the coagulation rate between simulation particles with different weights. Considering a coagulation event between simulation particle i (with volume v_i and private weight w_i) and j (with v_j and w_j). The event results in a new simulation particle k, whose size is $v_k = v_i + v_j$ while whose weight (w_k) is determined by the coagulation rule which actually determines the coagulation probability of two simulation particles (that is, the coagulation probability of i is specified as w_k/w_i, and that of j as w_k/w_j). For the Markov jump model, we introduce the heuristic Smoluchowski coagulation equation [19]:

$$\frac{\partial c(u,t)}{\partial t} = \frac{1}{2}\int_0^u B(u,v)p_{XX1}(u,v)c(v)c(u-v)\mathrm{d}v - c(u)\int_0^\infty B(u,v)p_{0XX}(u,v)c(v)\mathrm{d}v, \tag{2}$$

where $c(u,t) = n(u,t)/w(u)$ is the size distribution function of simulation particles, the rate for a coagulation event between particle i and j which generates particle k has to be determined:

$$B_{ij}p_{XX1}(i,j) = \beta_{ij}\frac{w_i w_j}{w_k}, \tag{3}$$

where p_{XX1} is birth probability of particle k due to the coagulation (as for certain event, $p_{XX1} = 1$, namely $B_{ij} = \beta_{ij}w_i w_j/w_k$), and the death probability of particle i and j is

$$p_{0XX}(i,j) = \frac{\beta_{ij}w_j}{B_{ij}}, \tag{4}$$

$$p_{X0X}(i,j) = \frac{\beta_{ij}w_i}{B_{ij}}. \tag{5}$$

The total coagulation rate (R_i) of simulation particle i with any other simulation particles and the total coagulation rate (R) of simulation system are calculated as

$$R_i = \frac{1}{V^2}\sum_{j=1,j\neq i}^{N_s} B_{ij}, \tag{6}$$

$$R = \frac{1}{2}\sum_{i=1}^{N_s} R_i = \frac{1}{2V^2}\sum_{i=1}^{N_s}\sum_{j=1,j\neq i}^{N_s} B_{ij}, \tag{7}$$

where V is the volume of the simulation system. In the event-driven mode, the mean time-step of performing a coagulation event can be calculated as:

$$\langle\tau\rangle = \frac{1}{VR} = \frac{2V}{\sum_{i=1}^{N_s}\sum_{j=1,j\neq i}^{N_s} B_{ij}}, \tag{8}$$

where the mean time-step is affected by the w_k of different Markov jump, factually, increasing with the decreasing of the w_k.

Table 1
Definition of three typical coagulation rules.

Particle pair	$(w_i, v_i; w_j, v_j)$ $(w_i \le w_j)$		
	The lower-bound jump	The upper-bound jump	The median jump
w_k	$\min(w_i, w_j)$	$\max(w_i, w_j)$	$(w_i + w_j)/2$
B_{ij}	$\beta_{ij}\max(w_i, w_j)$	$\beta_{ij}\min(w_i, w_j)$	$\beta_{ij}[2w_i w_j/(w_i + w_j)]$
p_{XX1}	1	1	1
p_{0XX}	$\min(w_i, w_j)/w_i$	$\max(w_i, w_j)/w_i$	$(w_i + w_j)/2w_i$
p_{X0X}	$\min(w_i, w_j)/w_j$	$\max(w_i, w_j)/w_j$	$(w_i + w_j)/2w_j$
δn	-$\min(w_i, w_j)$	$-\max(w_i, w_j)$	$-(w_i + w_j)/2$
δm	$-(w_j - w_i)v_j$	$(w_j - w_i)v_i$	$(w_j - w_i)(v_i - v_j)/2$

Where, δn and δm are the change of the number and mass during coagulation event, respectively.

Table 2
Parameters in the constant-number management for Markov jump.

	Total mass	Total number	Mass concentration	The system volume
Before coag. event	$m = \sum_{i=1}^{N_s} w_i v_i$	$n = \sum_{i=1}^{N_s} w_i$	$M = m/V$	V
After recovery	$m' = m + \delta m + w_r v_r$	$n' = n + \delta n + w_r$	$M = m'/V'$	$V' = Vm'/m$

Generally, we can regulate w_k within the interval $[\min(w_i, w_j), \max(w_i, w_j)]$, assigning w_k with a great degree of freedom to construct a coagulation-rule matrix. Here, we focus on three typical values of w_k: $\min(w_i, w_j)$, $\max(w_i, w_j)$ and $(w_i + w_j)/2$. The corresponding coagulation rules are three elements of coagulation-rule matrix, and three kinds of Markov jump are named as lower-bound jump, upper-bound jump and median jump, respectively. For the three kind of typical Markov jump, the different definitions of coagulation rules are summarized in Table 1, and corresponding schematic representation of these coagulation rules is illustrated in Fig. 1. Take the median jump model shown in Fig. 1(c) as an example. The coagulation probability of i is 4/3 (which is allowed to be greater than 1 in our models), and that of j is 4/5. As a result, all real particles in i take part in coagulation event, and another one foreign real particle having same size with i is also added to realize the coagulation; only 4 real particles in j participate in the coagulation, and the remaining one holds after the coagulation. The coagulation event necessarily (that is, $p_{XX1} = 1$) generates a new particle k with volume of $(v_i + v_j)$ and weight of $(w_i + w_j)/2$ (which is allowed to be not integer in our models). One coagulation event means subtracting one from the total number of simulation particles. In order to continuously maintain a constant number (N_s) of simulation particles, one simulation particle is randomly selected and copied from the other particles, and then the constant-number scheme [13,15] which adjusts the volume of the simulation system (V) to keep constant mass concentration throughout number restoration is adopted here. The parameters in the constant-number management for Markov jump are shown in Table 2.

2.2. Fast PBMC: a highly efficient algorithm based on acceptance–rejection processes with weighted majorant kernel

Acceptance–rejection (AR) scheme is a quite straightforward but effective way to randomly and reasonably choose coagulation particle pairs. If the following condition is met:

$$r < p_{ij} = \frac{B_{ij}}{B_{\max}}, \tag{9}$$

the two randomly selected particles i and j is accepted as a coagulation pair, where r is a random number uniformly distributed between 0 and 1, p_{ij} is the coagulation probability and $B_{\max}$ is the maximum coagulation rate.

Calculating the mean time-step $\langle\tau\rangle$ by Eq. (8) and obtaining the maximum coagulation rate $B_{\max}$ are time-consuming. Obviously, the complexity of computing scales as $O(N_s^2)$ in Eq. (8), which is relatively high for large N_s. In this situation sampling estimation may reduce Eq. (8) to

$$\langle\tau\rangle = \frac{2V}{N_s(N_s - 1)\langle B_{ij}\rangle} \approx \frac{2V N_{\text{sam}}}{N_s(N_s - 1)\sum_{q=1}^{N_{\text{sam}}} B_{ij,q}}, \tag{10}$$

where N_{sam} is sample size of coagulation rate set $\{B_{ij}\}_{N_s \times N_s}$, being an adjustable value much less than N_s^2. Better yet, a random sampling process is included in the AR attempts to choose a desirable coagulation pair, so N_{sam} can be replaced with the number of AR attempts (N_{AR}). Another cumbersome problem that is faced by using AR scheme is the computation of $B_{\max}$ due to searching in coagulation rate set $\{B_{ij}\}_{N_s \times N_s}$. We got some inspiration to estimate $B_{\max}$ from the majorant kernel [14,21]. As known, it is possible to transfer a traditional kernel β_{ij} to a majorant kernel $\hat{\beta}_{ij}$ through amplifying and factoring reasonably. The form of majorant kernel is $\hat{\beta}_{ij} = \sum_k [h_k(i) \times g_k(j)]$ that is used to study approximation properties regarding the mass flow equation proposed by Eibeck et al. [20,21], where $h_k(i)$ and $g_k(j)$ are pending factors. Similarly, we could construct weighted majorant kernel $\hat{B}_{ij}$ from regular coagulation rate B_{ij}, in such a way that the maximum value

Table 3
Weighted majorant kernels for different Markov jump models.

	Weighted majorant kernel
The lower-bound jump	$\hat{B}_{ij} = \sqrt{2}K_{\mathrm{fm}} v_j^{1/6} w_{\max}[(v_{\max}/v_j)^{2/3} + (v_{\max}/v_j)^{1/6} + 1 + (v_{\min}/v_j)^{-1/2}]$
The upper-bound jump[a]	$\hat{B}_{ij} = \sqrt{2}K_{\mathrm{fm}} v_j^{1/6} w_j[(v_{\max}/v_j)^{2/3} + (v_{\max}/v_j)^{1/6} + 1 + (v_{\min}/v_j)^{-1/2}]$
The median jump[b]	$\hat{B}_{ij} = \sqrt{2}K_{\mathrm{fm}} v_j^{1/6} w_{\max}[(v_{\max}/v_j)^{2/3} + (v_{\max}/v_j)^{1/6} + 1 + (v_{\min}/v_j)^{-1/2}]$

[a] In the case, we have $B_{ij} = \beta_{ij}\min(w_i, w_j)$, where $\beta_{ij} \le \hat{\beta}_{ij} \le \sqrt{2}K_{\mathrm{fm}} v_j^{1/6}[(\frac{v_{\max}}{v_j})^{2/3} + (\frac{v_{\max}}{v_j})^{1/6} + 1 + (\frac{v_{\min}}{v_j})^{-1/2}]$, $\min(w_i, w_j) \le w_j$ as before. Therefore $B_{ij} \le \sqrt{2}K_{\mathrm{fm}} v_j^{1/6} w_j[(\frac{v_{\max}}{v_j})^{2/3} + (\frac{v_{\max}}{v_j})^{1/6} + 1 + (\frac{v_{\min}}{v_j})^{-1/2}]$ as required, we can arrange $\hat{B}_{ij} = \sqrt{2}K_{\mathrm{fm}} v_j^{1/6} w_j[(\frac{v_{\max}}{v_j})^{2/3} + (\frac{v_{\max}}{v_j})^{1/6} + 1 + (\frac{v_{\min}}{v_j})^{-1/2}]$ as a weighted majorant kernel.

[b] In the case, we have $B_{ij} = \beta_{ij}[2w_i w_j/(w_i + w_j)]$, where $\hat{\beta}_{ij} \le \sqrt{2}K_{\mathrm{fm}} v_j^{1/6}[(\frac{v_{\max}}{v_j})^{2/3} + (\frac{v_{\max}}{v_j})^{1/6} + 1 + (\frac{v_{\min}}{v_j})^{-1/2}]$, $2w_i w_j/(w_i + w_j) \le w_{\max}$ as before. Therefore $B_{ij} \le \sqrt{2}K_{\mathrm{fm}} v_j^{1/6} w_{\max}[(\frac{v_{\max}}{v_j})^{2/3} + (\frac{v_{\max}}{v_j})^{1/6} + 1 + (\frac{v_{\min}}{v_j})^{-1/2}]$ as required, we can arrange $\hat{B}_{ij} = \sqrt{2}K_{\mathrm{fm}} v_j^{1/6} w_{\max}[(\frac{v_{\max}}{v_j})^{2/3} + (\frac{v_{\max}}{v_j})^{1/6} + 1 + (\frac{v_{\min}}{v_j})^{-1/2}]$ as a weighted majorant kernel.

$\hat{B}_{\max}$ of weighted majorant kernel is obtained by only single looping over all simulation particles. The maximum majorant kernel $\hat{B}_{\max}$ is used to approximate the maximum coagulation rate $B_{\max}$ in Eq. (9), and is further used to search coagulation particle pairs randomly using the AR scheme. Aiming to the weighted coagulation kernel $B_{ij} = 2\beta_{ij} w_j \max(w_i, w_j)/(w_i + w_j)$ used in the DWMC method [16,17], we have constructed the corresponding weighted majorant kernel and proposed a fast DWMC method [26,44,45]. In this work, we further show how to construct the weighted majorant rates which correspond to three typical Markov jump models described above.

We take Brownian coagulation kernel in the free molecular regime as an example. The regular kernel reads

$$\beta_{ij} = K_{\mathrm{fm}}\left(v_i^{1/3} + v_j^{1/3}\right)^2\left(v_i^{-1} + v_j^{-1}\right)^{1/2}, \tag{11}$$

where coagulation coefficient $K_{\mathrm{fm}} = (\frac{3}{4\pi})^{1/6}(\frac{6k_{\mathrm{B}}T}{\rho_{\mathrm{p}}})^{1/2}$ with k_{B} being the Boltzmann constant, T is the temperature and ρ_{p} is the particle density. Then, the majorant kernel [14,20]

$$\hat{\beta}_{ij} = \sqrt{2}K_{\mathrm{fm}}\left(v_i^{2/3} + v_j^{2/3}\right)\left(v_i^{-1/2} + v_j^{-1/2}\right), \quad \beta_{ij} \le \hat{\beta}_{ij}$$

can be transformed to

$$\hat{\beta}_{ij} = \sqrt{2}K_{\mathrm{fm}} v_j^{1/6}\left[\left(\frac{v_i}{v_j}\right)^{2/3} + \left(\frac{v_i}{v_j}\right)^{1/6} + 1 + \left(\frac{v_i}{v_j}\right)^{-1/2}\right]. \tag{12}$$

With the correlation $\frac{v_{\min}}{v_j} \le \frac{v_i}{v_j} \le \frac{v_{\max}}{v_j}$ ($v_{\min}$ and $v_{\max}$ are the minimum and maximum volumes of particles respectively), we get

$$\hat{\beta}_{ij} \le \sqrt{2}K_{\mathrm{fm}} v_j^{1/6}\left[\left(\frac{v_{\max}}{v_j}\right)^{2/3} + \left(\frac{v_{\max}}{v_j}\right)^{1/6} + 1 + \left(\frac{v_{\min}}{v_j}\right)^{-1/2}\right]. \tag{13}$$

With regard to the lower-bound jump model,

$$B_{ij} = \beta_{ij}\max(w_i, w_j) \le \hat{\beta}_{ij} w_{\max}. \tag{14}$$

So we can define the weighted majorant kernel as following:

$$\hat{B}_{ij}(i) = \sqrt{2}K_{\mathrm{fm}} v_i^{1/6} w_{\max}\left[\left(\frac{v_{\max}}{v_i}\right)^{2/3} + \left(\frac{v_{\max}}{v_i}\right)^{1/6} + 1 + \left(\frac{v_{\min}}{v_i}\right)^{-1/2}\right]. \tag{15}$$

First of all, $w_{\max}$, $v_{\min}$ and $v_{\max}$ are found through a single-looping over all simulation particles; subsequently, another single-looping to find $\hat{B}_{\max}$ in Eq. (15). It is noted that the weighted majorant kernel $\hat{B}_{ij}$ also has the following characteristics (like the normal majorant kernel): (1) $\hat{B}_{ij} \ge B_{ij}$ for all i, j; (2) $\hat{B}_{ij}$ is only related to internal variables of one simulation particle (rather than particle pair), i.e., $\hat{B}_{ij} = \sum_k [f_k(j)]$, so that only single looping over all simulation particles is enough to estimate the maximum value over all B_{ij}; (3) $B_{\max}/\hat{B}_{\max}$ is close to 1 as possible so that the AR process choosing coagulation pair(s) is highly efficient. Table 3 summarizes weighted majorant kernels in the free molecular regime for three typical Markov jumps.

2.3. GPU acceleration: on CUDA platform

The above-described acceleration schemes are building on serial computation on CPU. The proposed PBMC schemes are further implemented on a GPU after a full parallelization to further improve the efficiency. For the GPU acceleration, three

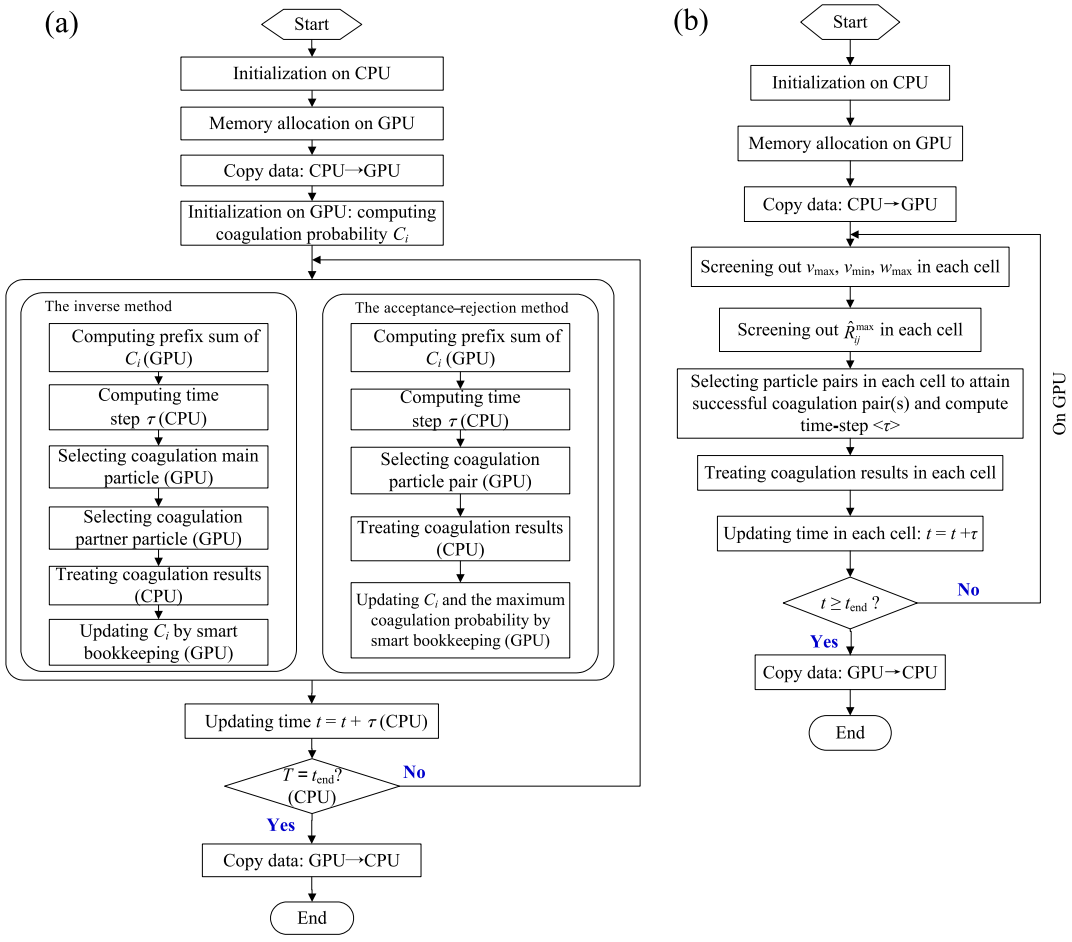

Fig. 2. Flow chart of the parallel algorithm on GPU: (a) PBMC in a single cell; (b) the fast PBMC in a multi-cell system.

parallel versions of this code for simulating particle coagulation were investigated: GPU parallel computing of PBMC in a single cell, GPU parallel computing of fast PBMC in a single cell, and a multi-cell GPU parallel computing. Simulations in this work were achieved on a workstation equipped with an Intel® Xeon™ E5-2680 2.7 GHz Processor, 64 GB RAM, and an NVIDIA® Tesla™ C2075 GPU. The Tesla™ C2075 has 448 cores with 16 KB of shared memory per block and 4 GB of global memory. It was operated in single precision under Red Hat enterprise 6.0 Linux, with the CUDA Toolkit version 5.0. In addition, all particle properties are saved in the GPU global memory, and some variables of particle population such as the geometric mean diameter (d_g) and the geometric standard deviation of PSD (σ_g) are calculated by means of some CUDA statistics kernels.

2.3.1. GPU parallel computing of PBMC in a single cell

We implement the GPU parallel computing of the differentially weighted PBMC for particle coagulation. The main idea is to implement the highly threaded data-parallel processing tasks by using GPU. The selection of coagulated particle pair by the inverse scheme and the AR scheme are both implemented using CUDA on GPU. The parallel algorithm for PBMC in a single cell on GPU is designed and shown in Fig. 2(a), which includes the following CUDA kernel functions (Here, it is assumed that the number of simulation particle N_s is constant equal to 10 000, and one dimensional grids and blocks are defined.):

(a) Initialization process

The initialization involves three sub-kernels. The first sub-kernel computes R_i for each particle based on Eq. (6) as it involves $N_s - 1$ terms, which executes double looping over all particle pairs, thus totaling $N_s(N_s - 1)$ threads in the CUDA kernel where each thread is used to compute one coagulation kernel (B_{ij}). With thread number $N_{th} = 1024$ per block as an example, the number of blocks required to deal with a specified particle is $\lceil N_s/N_{th} \rceil = 10$, and total number of blocks is $\lceil N_s/N_{th} \rceil \cdot N_s = 10^5$. Due to the limitation in the maximum number (here, 65535) of blocks allowed by CUDA, the maximum block number is chosen as $\lfloor 65535/\lceil N_s/N_{th} \rceil \rfloor \cdot \lceil N_s/N_{th} \rceil = 65530$, when the block index surpasses this value,

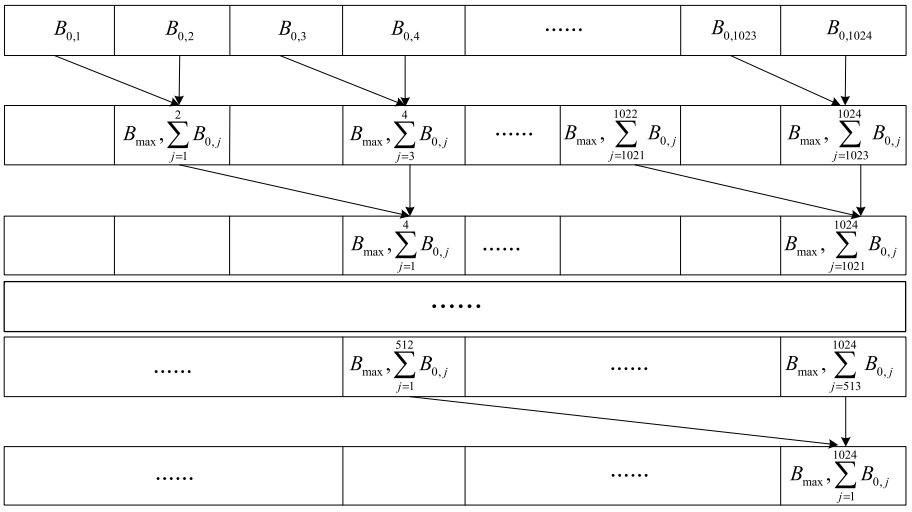

Fig. 3. Schematic illustration of searching for $B_{\max}$ and summating B_{ij} in a block.

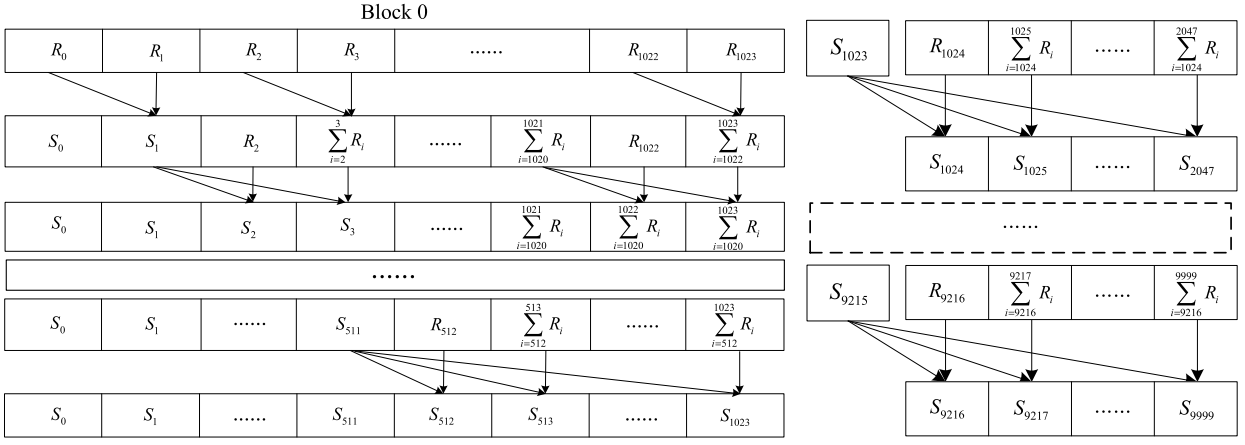

Fig. 4. Schematics illustrating for the summation procedure: (a) left, in Block 0 as an example; (b) right, in all ten blocks.

and the calculations are repeated. After the thread calculated the coagulation kernel, it is stored in the shared memory (N_{th} floating-point memory allocated for each block). In the second sub-kernel, all of coagulation kernel in each block are summated and compared by a rapid parallel reduction method (In this way, each block reduces a portion of particle array by tree-based approach to summate B_{ij} and to search the maximum of coagulation kernel $B_{\max}$ iteratively, as shown in Fig. 3.), and then the partial sums and the maximum of coagulation kernel for each block are stored in global memory. The third sub-kernel calculates R_i and $B_{\max}$ for all simulation particles and stores them in global memory.

(b) Computing the prefix-sum of coagulation kernel

The CUDA kernel consists of two sub-kernels to compute the prefix-sum $S_i = \sum_{k=0}^{i} R_k$ of each particle by a parallel reduction method. The first sub-kernel calculates the sum over the R_i in each block, and stores them in global memory, as shown in Fig. 4(a). In the second sub-kernel, the individual sum in each thread is added with the previous block sums, as shown in Fig. 4(b). In one thread the mean time-step is calculated based on Eq. (8), $\langle\tau\rangle = 2/(V S_{9999})$.

(c) Selecting coagulation pair in the inverse scheme

In the inverse scheme, a particle pair (i, j) can be accepted as a coagulation pair when the following conditions are satisfied in turn

$$\langle\tau\rangle V S_{i-1}/2 < r_1 \leq \langle\tau\rangle V S_i/2, \tag{16}$$

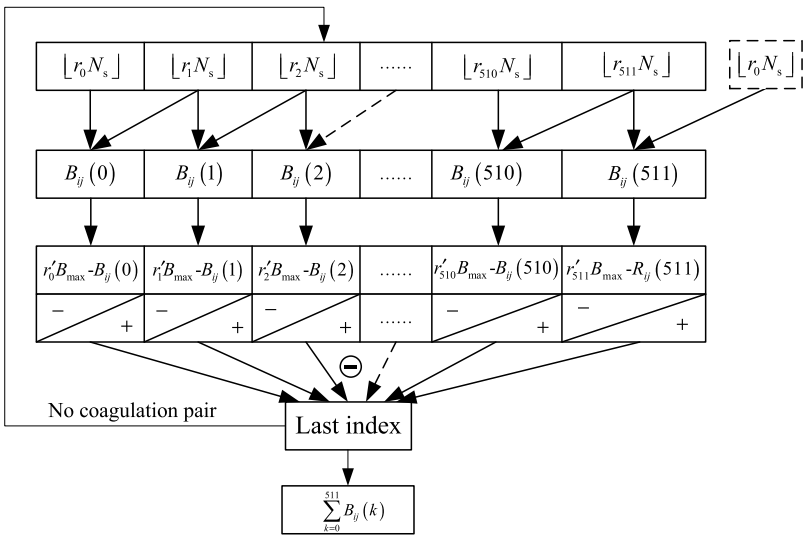

Fig. 5. The parallel implementation of AR attempt on GPU.

and

$$\frac{\langle\tau\rangle}{2V}\sum_{m=1,m\neq i}^{j-1} B_{im} < r_2 - \frac{\langle\tau\rangle V}{2} S_{i-1} \leq \frac{\langle\tau\rangle}{2V}\sum_{m=1,m\neq i}^{j} B_{im}, \tag{17}$$

where the random number r_1, r_2 are uniformly distributed in the interval $[0, 1]$. A random number r is generated based on a seed value stored in global memory in one thread. Then the values of $(\langle\tau\rangle V S_i - r)$ transformed from Eq. (16) for total N_s threads are calculated and placed in global memory. The particle i is the first selected particle of coagulation pair when either there is $\langle\tau\rangle V S_i - r = 0$ or the adjacent two values satisfy the following condition

$$\left(\langle\tau\rangle V S_{i-1} - r\right) \cdot \left(\langle\tau\rangle V S_i - r\right) < 0. \tag{18}$$

Subsequently, the second particle is selected in a similar way with Eq. (17) based on the first selected particle.

(d) Selecting coagulation pair in the AR scheme

Here, N_{th} is equal to N_{AR} but less than N_s, and a random number is generated by the MTGP32 Mersenne Twister algorithm from CURAND library (NVIDIA® Corporation, CUDA CURAND Library, 2010) for randomly selecting a particle in a thread. Then coagulation kernel of each two particles in adjacent threads is computed. The second random number is generated in each thread to judge the acceptance or rejection of the particle pair according to the value of $(rB_{max} - B_{ij})$ with the help of Eq. (9). Afterwards, the status of an AR attempt occupying an individual thread is either acceptance (status value as negative, namely $rB_{max} - B_{ij} < 0$) or rejection (status value as non-negative, namely $rB_{max} - B_{ij} \geq 0$). An overwriting action [36], which can automatically obtain a coagulation pair when all threads with negative status value write the indexes of their selected particle pair to a same memory address, is performed. This process has to be repeated for any block until a coagulation pair has been found. A schematic diagram of the data over the threads in a block is shown in Fig. 5.

(e) Treating coagulation results

According to the Markov jump model and coagulation rule, as previously mentioned in Section 2.1, the new particle internal variables (both size and weight) are calculated and the elapse time is updated. Coagulation dynamics can be considered "finished" as soon as their total evolution time reaches the prespecified time length.

(f) Smart bookkeeping for updating

The values of R_i for all simulation particles need to be updated after a coagulation event, because the size and weight of the two selected simulation particles have changed as a result of that event. For the two selected particles m and n, R_m and R_n are recalculated based on Eq. (6) whereas for all other particles the R_i can be modified according to

$$R_i^{new} = R_i - B_{im} - B_{in} + B_{im}^{new} + B_{in}^{new}. \tag{19}$$

It is worth emphasizing that all of coagulation event processing is fully conducted on GPU, and the only communication between CPU and GPU occurs during the initialization and result output.

2.3.2. GPU parallel computing of the fast PBMC in a single cell

The substantial accomplishment of reducing the computational load has shown in a single homogeneous system by a series of fast algorithms, e.g., the majorant kernel fictitious jumps method [14], the fast PBMC [26] etc. As mentioned in Section 2.2, the fast PBMC does not need to compute R_i, R and $B_{\max}$, whereas it just depends on the maximum of weighted majorant kernel $\hat{B}_{\max}$ and estimates time-step based on the acceptance–rejection processes. Therefore, the accelerated fast PBMC in a single-cell system mainly includes the following steps different from Section 2.3.1.

(a) Searching for $v_{\max}$, $v_{\min}$, $w_{\max}$ as well as $\hat{B}_{\max}$

Firstly, a parallel reduction method is adopted within each block to find $v_{\max}$, $v_{\min}$, $w_{\max}$ gradually, similar to searching $\hat{B}_{\max}$ in Section 2.3.1(a). Then $\hat{B}_{\max}$ is attained with the aid of Eq. (15).

(b) Selecting coagulation pair and calculating mean time-step $\langle\tau\rangle$

Selecting coagulation pair has been mentioned in Section 2.3.1(d). Moreover, while all threads are ready (allowing synchronization) after a coagulation pair has been found, the coagulation kernel B_{ij} from all threads ($N_{\text{th}} = N_{\text{AR}}$) in a block is added up to compute $\langle\tau\rangle$ by Eq. (10).

2.3.3. A multi-cell GPU parallel computing

It is important to note that the simulation of the coagulation dynamics can in principle be performed for different cells in parallel without any interference. The goal of this work is to develop a method which allows to implement multi-cell parallel computation running on GPU with fast PBMC in each cell. Therefore, we consider a system having N_{c} cells each containing N_{s} simulation particles, whereas the number of AR attempts for a single cell as well as the number of particle pairs use to determine $\langle\tau\rangle$ is fixed as N_{AR}. In such a way, an efficient way of allocating the computational tasks is to choose the number of blocks N_{B} equal to N_{c} and the number of threads N_{th} equal to N_{s} or N_{AR}. The maximum thread number, currently 1024 for CUDA V5.0, is however the absolute limit to the number of parallel AR attempts due to the data exchange and synchronization between threads. The complete acceleration algorithm, as shown in Fig. 2(b), mainly consists of steps similar to Section 2.3.2 in each cell, however, this procedure has to be repeated for any block (cell) until a coagulation event has occurred.

The basic framework of multi-cell parallel computing is suitable for CFD-PBM simulation. In this sense, particle dynamics in all of cells can be simulated on GPU within each time-step of CFD simulation, so as to save calculation time.

2.3.4. GPU parallel computing of fast PBMC based on time-driven approach

It should be pointed out that all above PBMC are based on event-driven strategy. However, in practice time-driven PBMC and CFD have the internal consistency, which is the same time-step in favor of coupling solution. We have proposed a unified scheme for event-driven and time-driven [46]. The time-driven PBMC allows multiple events to be handled at the same time, and thereby the total number of events within a time-step is generally far greater than 1. So as to there are still high parallelism. While in the time-driven PBMC, many coagulation events may occur within a time-step. We define the factor (p) of the number of coagulated simulation particles to the whole simulation particle number within time-step Δt, $2/N_{\text{s}} \le p \le 1$, and p is an adjustable parameter according to Δt. The time-step in the time-driven PBMC is thus calculated as:

$$\Delta t = \frac{pN_{\text{s}}V}{\sum_{i=1}^{N_{\text{s}}}\sum_{j=1,j\neq i}^{N_{\text{s}}} B_{ij}}. \tag{20}$$

It means that there are pN_{s} coagulation events within Δt, allocating the computational tasks is to set the number of threads N_{th} equal to pN_{s} for selecting coagulation pairs by GPU parallel computing.

3. Simulation results and discussions

In this work, a physically realistic case with initially monodispersed particle population (i.e. $\sigma_{\text{g0}} = 1$) and with the Brownian coagulation kernel in the free molecular regime is simulated to explore the performance of PBMC methods with different acceleration schemes in precision and efficiency. At the onset of simulation, the initial diameter is $d_0 = 3$ nm with density $\rho_{\text{p}} = 1000\ \text{kg m}^{-3}$, and initial particle number concentration is $n_0 = 10^{17}\ \text{m}^{-3}$ as well as initial weight for each simulation particle is calculated as $w = n_0/N_{\text{s}}$. The simulations are carried out at standard atmospheric pressure (1.01×10^5 Pa) and at ambient temperature 300 K, experiencing over a long time scale, $[0, 1000\tau_{\text{char}}]$, where τ_{char} is the characteristic coagulation time, defined as the time required equal to the half-value period of particle concentration. τ_{char} can be calculated as [35]

$$\tau_{\text{char}} = \left(\frac{3k_{\text{B}}Td_0n_0^2}{\rho_{\text{p}}}\right)^{-1/2}. \tag{21}$$

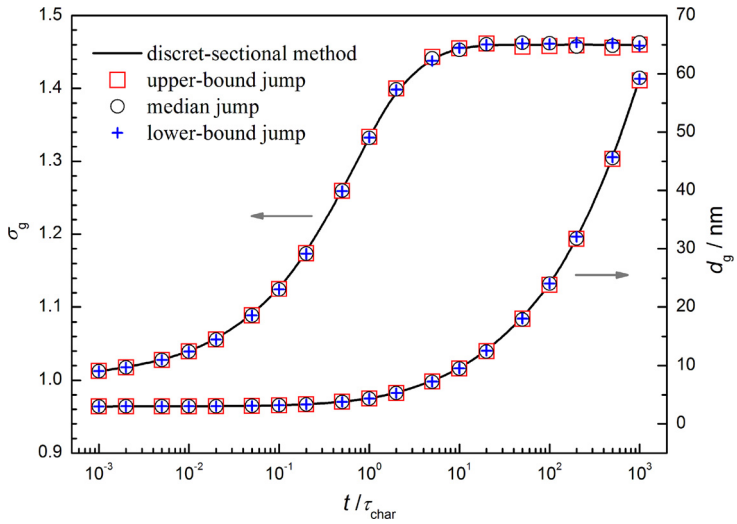

Fig. 6. Validation of the evolution of particle geometric mean diameter d_g and geometric standard deviation σ_g with dimensionless simulation time.

3.1. Validation procedure

Firstly, three typical Markov jump models are tested on CPU, respectively, all based on traditional AR scheme while choosing coagulation pairs. Here a single-cell or spatially homogeneously system is simulated. Subsequently, the fast AR scheme with weighted majorant kernel is used to simulate the Brownian coagulation case in the single-cell system.

With respect to GPU parallel computing, the acceleration effect of PBMC was demonstrated at first, and then the fast PBMC was processed in GPU parallel, both of them being in a single cell system. Factually, a whole computing domain, which may be spatially inhomogeneous, should be divided into some compartments or cells, which is also a requirement when population balances have to be solved in a CFD environment. In this premise we consider a computing domain consisting of 100 cells where each of them contains 2000 simulation particles, and there is no transport and interaction between cells. Finally, the comprehensive accelerating approach, which PBMC with the most efficient Markov jump (the upper-bound jump) model and fast AR scheme is used to simulate coagulation dynamics in each cell simultaneously based on the framework of GPU parallel computing, is for the first time demonstrated in efficiency and precision.

The simulation results are compared with these of a benchmark numerical solution, which is here an implementation of a discrete sectional model [47] using 200 sections and a sectional spacing of 1.12 [35].

3.2. Test of Markov jump model

The computing accuracy of three typical Markov jumps based on traditional AR method on CPU was first verified by comparison with the benchmark solution. The mean values were obtained from 10 simulation runs and using 10 000 simulation particles, as seen in Fig. 6. The curves show the increase in particle geometric mean diameter d_g and the evolution of the geometric standard deviation σ_g from 1.0 initially to a self-preserving size distribution (SPSD) benchmark value ($\sigma_g = 1.46$). The simulation results show a very good agreement with the benchmark solution. In addition, the relative errors of d_g and σ_g are evaluated according to:

$$\delta_X(t) = \frac{|X_{\mathrm{PBMC}}(t) - X_{\mathrm{Benchmark}}(t)|}{X_{\mathrm{Benchmark}}(t)} \times 100\%, \tag{22}$$

$$\bar{\delta}_X = \frac{\sum_{i=1}^{N_t} \delta_X(i)}{N_t}, \tag{23}$$

where X as the symbol for these parameters d_g and σ_g, the mean error $\bar{\delta}_X$ is calculated over $N_t = 19$ different time points logarithmically distributed between $0.001\tau_{\mathrm{char}}$ and $1000\tau_{\mathrm{char}}$. Fig. 7 shows the relative error of three typical Markov jumps with time evolution. Obviously, the scattering of simulation results is basically within 0.5% and 1.5% for σ_g and d_g, respectively, and mean error is all less than 0.7%. Although the relative errors of upper-bound jump almost are the biggest, it is also acceptable. Furthermore, Fig. 8 shows that the time-step number and the mean time-step increase with time evolution. It can be clearly seen that the mean time-step of the upper-bound jump is the biggest and the time-step number of which is the lowest. Consequently, the CPU time consumed (953.3 s) achieves the minimum, being close to one quarter of that of the lower-bound jump (4072.8 s) due to almost same CPU time for execution at each time-step looping. The median jump model attain a compromise between precision and cost, where its relative errors in d_g and σ_g are among these of the upper-bound and lower-bound jump models, and its CPU time consumed is 2610.9 s.

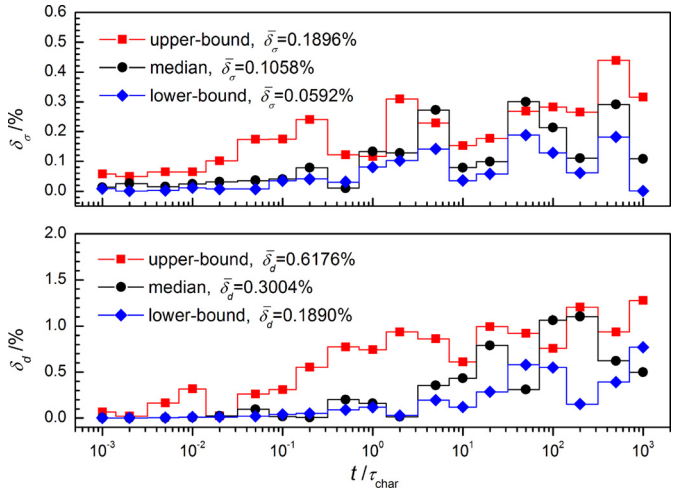

Fig. 7. The relative error of d_g and σ_g calculated on CPU by comparison to the benchmark solution.

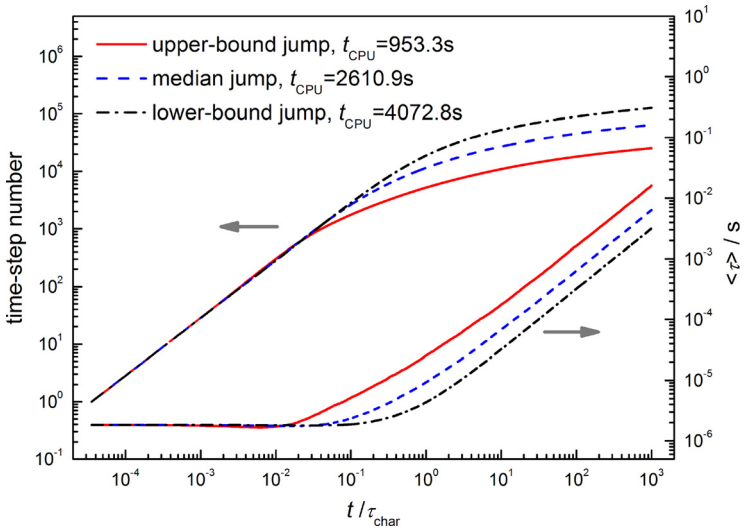

Fig. 8. The comparison of time-step number and mean time-step for three typical Markov jumps.

In order to demonstrate the performance advantage of DWMC method with appropriate Markov jump to equally weighted MC (e.g., the constant-number method), a theoretical case [32] with constant coagulation kernel and the initially polydispersed distribution (here exponential distribution) is simulated:

$$n(v,0)=\frac{N_0}{v_{g0}}\exp\left(-\frac{v}{v_{g0}}\right), \tag{24}$$

where the initial total number concentration of real particles is $N_0 = 1.0 \times 10^6$ cm^{-3}, the initial geometric mean size is $v_{g0} = 0.029$ μm^3, the initial geometric standard deviation $\sigma_{g0} = 1.5$. The coagulation kernel is $\beta_{ij} = K = 6.405 \times 10^{-10}$ cm^3 s^{-1}, and the characteristic coagulation time $\tau_{char} = 2/(KN_0) \approx 1561.3$ s. The continuous size distribution is divided into 200 bins by the logarithmically spaced law in the range from 10^{-5} to 1 μm^3, and the simulation particle number of every bin is greater than 10.

Here, the simulation results from the three DWMC methods and the Constant-N method are compared with analytical solutions. As shown in Fig. 9, three DWMC methods exhibit higher precision of the particle size distribution function in the less populated regions. However, among these DWMC methods, the higher computational accuracy is, the larger computational cost is (as Fig. 9(c)), which is actually ascribed to the more detailed division of random jumps in, e.g., the lower-bound jump model. Therefore, constructing appropriate coagulation rules and Markov jump models provides a route to coordinate the relation between cost and accuracy of PBMC methods to a certain extent.

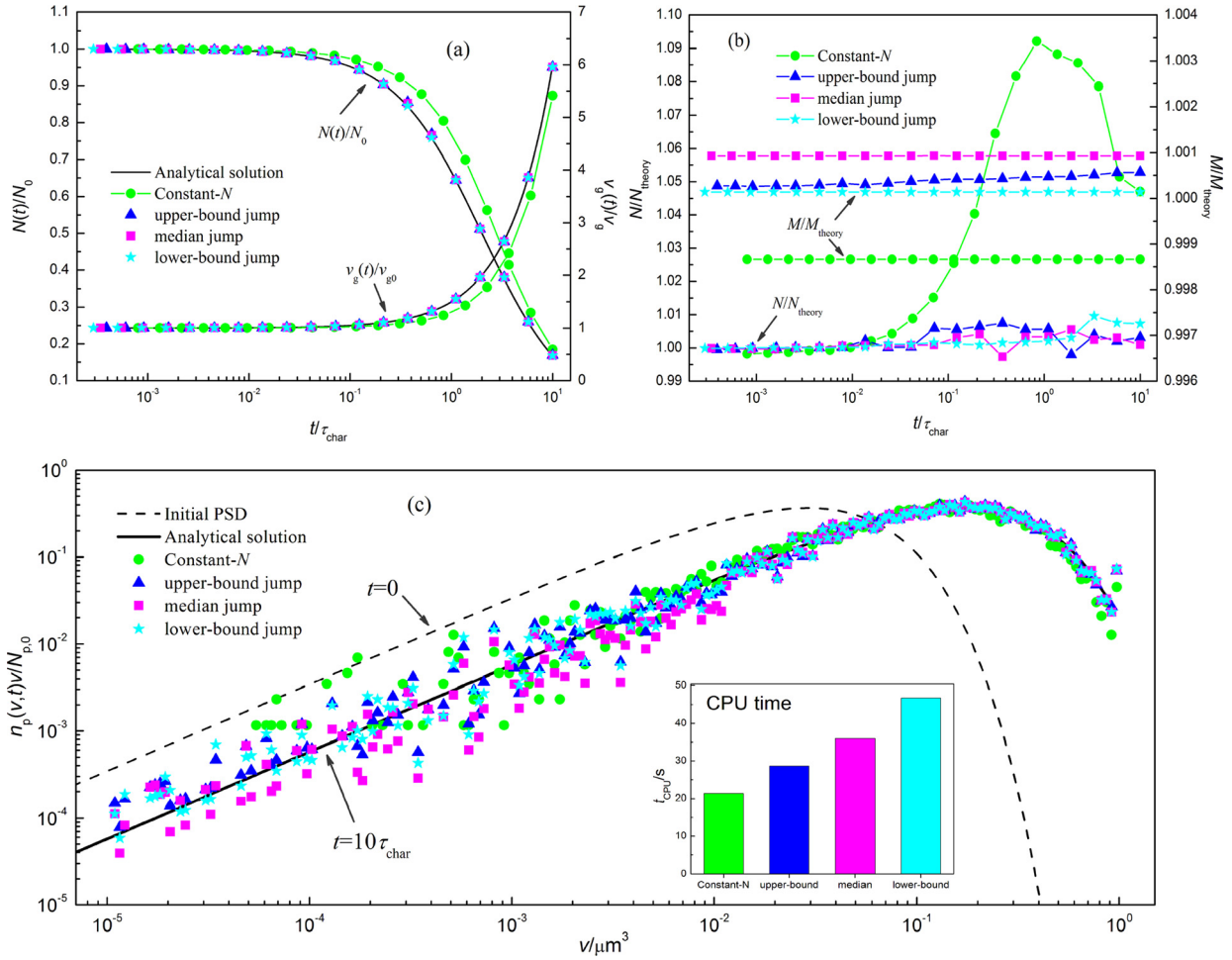

Fig. 9. Validation of the Markov jump of PBMC: (a) The particle number and standard deviation; (b) The error of moments; (c) the time evolution of PSD and the CPU time.

3.3. Test of the fast PBMC method

All of three typical Markov jumps based on the fast AR scheme with weighted majorant kernel were carried out on CPU. Likewise, the computing accuracy is compared with the benchmark solution as shown in Figs. 10 and 11. The same evolution trends as well as similar errors of d_g and σ_g are displayed. Compared with the results of the above, errors increase slightly and remain favorably low. Moreover, the discrete-sectional model [48] provided classical benchmark solutions of SPSD. In the self-preserving formulation, the dimensionless particle size is defined as $\eta = v/\bar{v} = Nv/M$, and the dimensionless number distribution function as $\psi = Mn(v,t)/N^2$, where $\bar{v}$ is the average volume, N and M are number and mass concentrations respectively. Fig. 12 shows an SPSD of the fast PBMC with upper-bound jump in the terminal point of simulation ($1000\tau_{\text{char}}$), agreeing well with the benchmark solution. The efficiency improvement of the fast PBMC has been investigated, as shown in Fig. 13. It is obvious that the fast PBMC is more efficient than Monte Carlo methods using other schemes [14,22] at present. What is more, the fast PBMC with upper-bound jump is the most efficient method.

3.4. Test of PBMC method accelerated on GPU

For GPU parallel computing of PBMC with upper-bound jump in a single cell, the selection of coagulated particle pair by the inverse method and the AR method are both implemented on GPU. The computing time on the GPU (t_{GPU}) is compared to the one on the CPU (t_{CPU}) with the serial implementation. Fig. 14(a) shows the speedup ratio ($\alpha = t_{\text{CPU}}/t_{\text{GPU}}$) of the inverse method and the AR method for different numbers of simulation particle N_s from 1000 to 16 000, and speedup ratio between several and about 190 are obtained. The speedup ratio increases nearly linearly as simulation particle number increases, being fairly independent of the simulation time, whereas it is lower during initialization of simulation because the computing time on GPU includes the time for data transfer between CPU and GPU. Fig. 14(b) shows that the computing

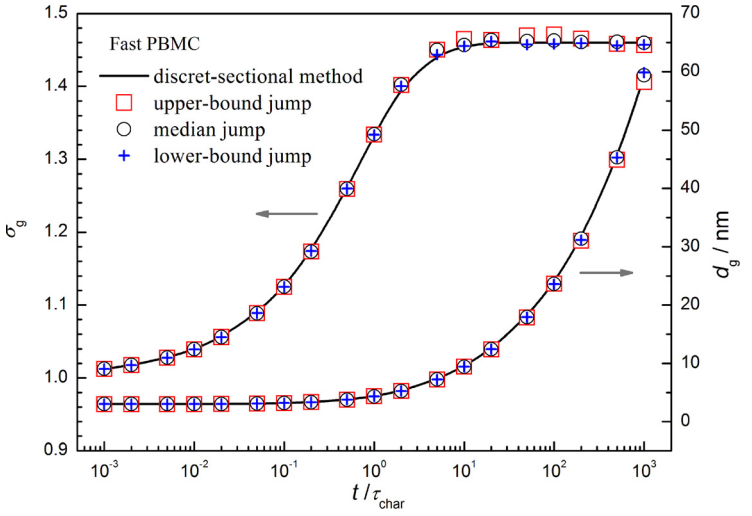

Fig. 10. Validation of the evolution of d_g and σ_g with dimensionless time by the fast PBMC.

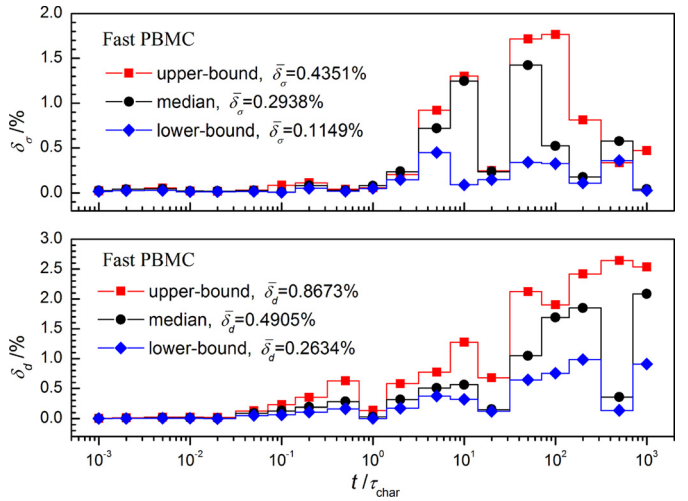

Fig. 11. The relative error of d_g and σ_g calculated on CPU with the fast PBMC by comparison to the benchmark solution.

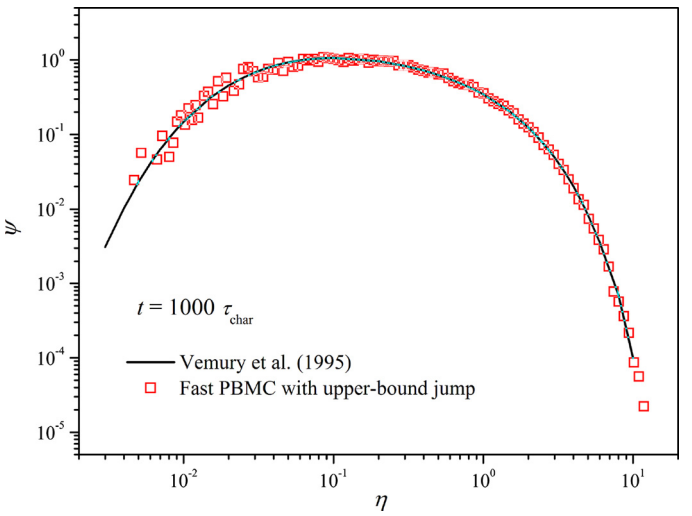

Fig. 12. The self-preserving size distributions of Brownian coagulation in the free molecular regime by the fast PBMC with upper-bound jump.

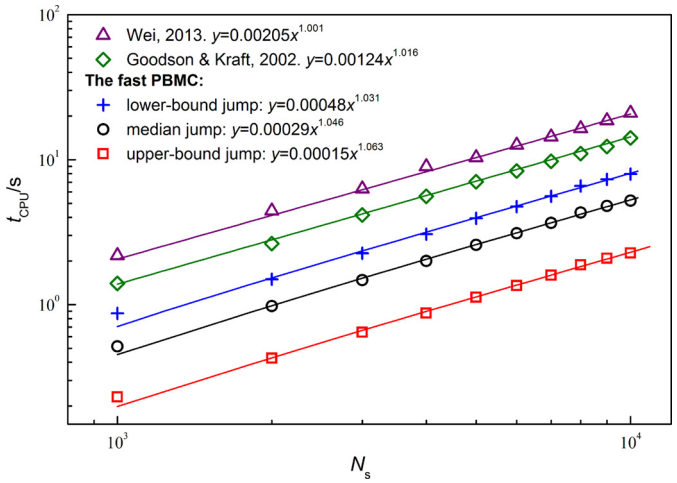

Fig. 13. CPU time vs. simulation particle number for five acceleration schemes.

accuracy is compared with the benchmark solution. As can be seen, the particle size and geometric standard deviation obtained from the four kinds of strategy are all accurate, with the mean statistical errors being less than 0.62%.

After having validated PBMC on GPU, we now turn towards implementing the fast PBMC on GPU for a single cell simulation, as the purpose of the GPU implementation is to better accelerate the simulation. For the fast PBMC with upper-bound jump (N_s fom 1000 to 18 000), as shown in Fig. 15(a), the numerical experiments exhibit that GPU computing-time consumption was higher than that of CPU in some cases (N_s from 1000 to 12 000). The fast PBMC have significantly reduced the amount of calculation, whose GPU acceleration effect is not well. The statistical accuracy of the results is shown in Fig. 15(b). Interestingly, the mean errors in d_g and σ_g of the AR method on GPU are less than that of the AR method on CPU. The reason is that N_{th} for the AR attempts on GPU is greater than N_{AR} on CPU in practice, leading to a more accurate the mean time-step $\langle\tau\rangle$ as explained by Eq. (10).

As the purpose of the GPU implementation is to accelerate multi-cell simulations, a multi-cell system consisting of 100 cells has been demonstrated at first, in which each cell is computed by the fast PBMC with upper-bound jump. Fig. 16 shows the relative errors of d_g and σ_g in each cell at the terminal point of simulation ($t = 1000\tau_{char}$). The relative errors can be reduced by increasing the number of GPU threads N_{th}. There is a significant decrease found while N_{th} from 64 to 128, and less obvious reductions from 128 to 512. This is because the number of threads, which is also the number of AR attempts, can affect accuracy of the mean time-step $\langle\tau\rangle$ to a certain extent according to Eq. (10) (here, $N_{sam} = N_{th}$). Meanwhile, the number of GPU threads also determines the computational efficiency; the optimal number of threads has to be found. As shown in Fig. 17, the smallest time consumption appears while N_{th} is equal to 128, otherwise there are multiple cycles to choose coagulation pair while N_{th} is equal to 64 and more thread synchronization occurs while N_{th} is equal to 256 or 512. This means that the optimal number of threads per block on GPU has to be determined. Obviously, the most straightforward way for evaluating the performance improvement by using GPU is to directly measure the computing time, and then compare it with that of other methods. Fig. 18 shows that the time consumption of three ways to speed up changes along with the number of simulation particles. It is found that the computing time is approximately linearly dependent on N_s, and the fast PBMC with upper-bound jump on GPU obtains the acceleration of the most significant. Fig. 19 shows how α behaves for different N_c and N_s, keeping $N_{th} = 128$. Clearly, increasing N_c leads to an improvement in speedup ratio that increases from less than 1 to more than 30 as N_c is from 1 to 1000. Only one block is used in the single-cell case, which does not make full use of the GPU power, so it is more efficient that many blocks are used to deal with multiple cells. However, the effect is less pronounced for increasing N_s because the computational complexity of the fast PBMC and the GPU parallel computing is all as $O(N_s)$.

Furthermore, we implement the GPU parallel computing of the fast PBMC based on time-driven approach in a single-cell system. The parameter p in this paper is equal to 0.0128, and N_s is equal to 10 000. There are 128 coagulation events within a time-step Δt on average, namely $N_{th} = 128$. From Fig. 20(a) it can be seen that the speedup ratio increases by almost a linear function when N_s is increased from 1000 to 18 000. The relative errors of d_g and σ_g increase as time evolution, however they are typically less than 3%, as shown in Fig. 20(b), and it is also obvious that the fast PBMC on GPU has higher accuracy. Similar phenomenon is also found for the fast PBMC on GPU based on event-driven in the previous results. Obviously, the time-driven fast PBMC is more efficient than the event-driven fast PBMC on GPU.

4. Conclusions

The population balance-Monte Carlo (PBMC) has become increasingly popular because the discrete and stochastic nature of the MC method is especially suited for particle dynamics. However, for the two-particle events (typically, particle

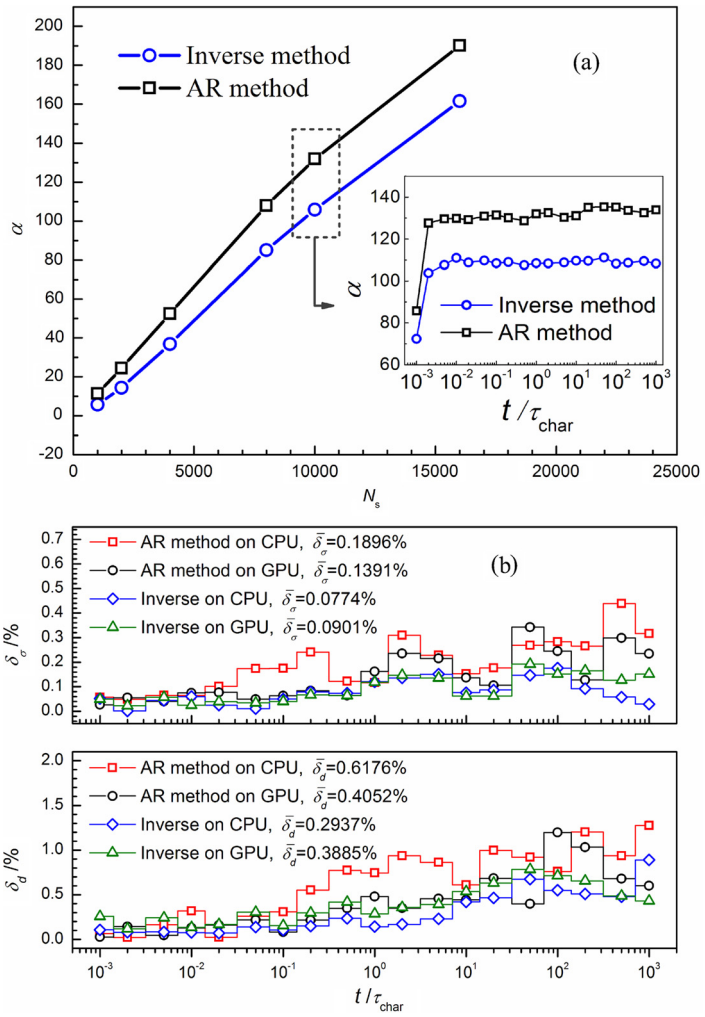

Fig. 14. GPU parallel computing of PBMC in a single cell: (a) the speedup ratio; (b) the relative error of d_g and σ_g.

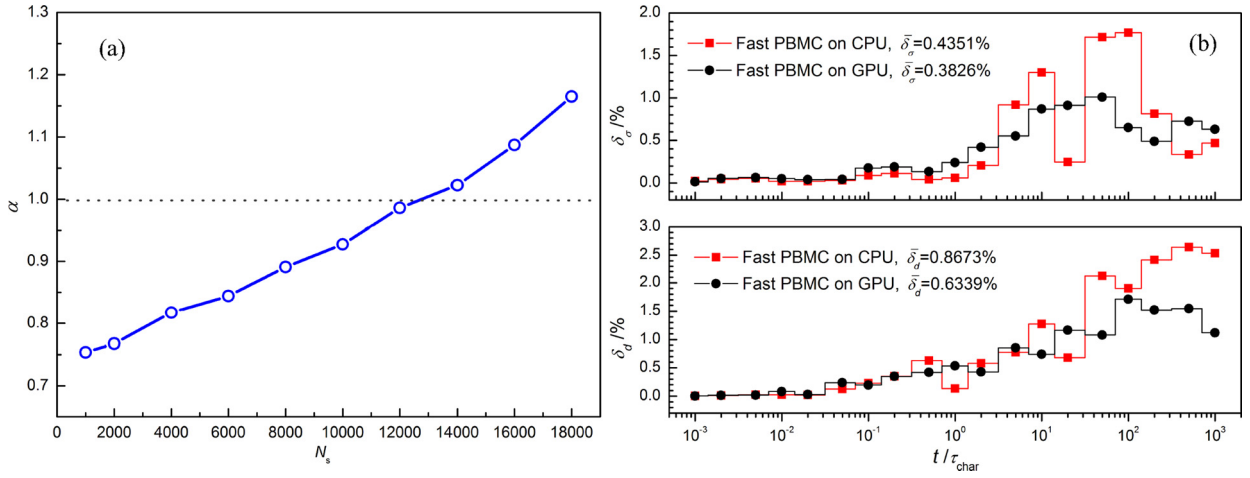

Fig. 15. GPU parallel computing of the fast PBMC in a single cell: (a) the speedup ratio; (b) the relative error of d_g and σ_g.

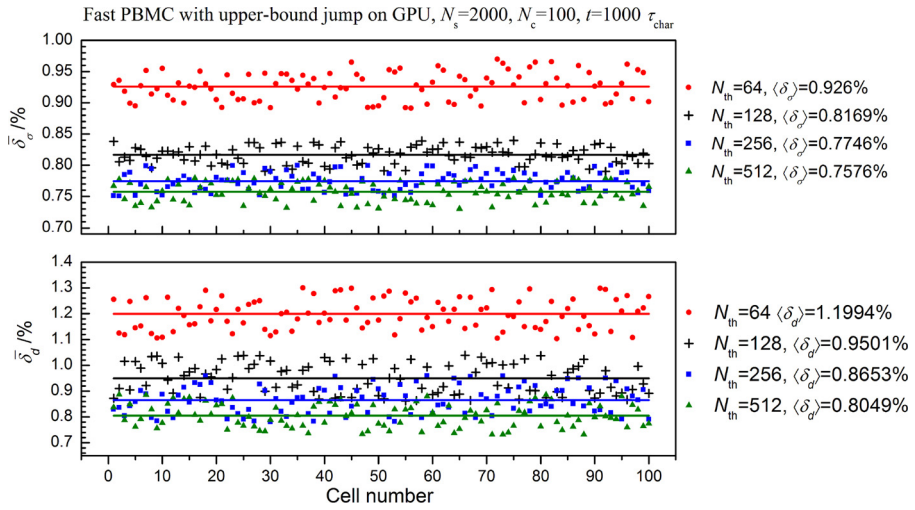

Fig. 16. The relative error of d_g and σ_g of each cell calculated on GPU with the fast PBMC by comparison to the benchmark solution.

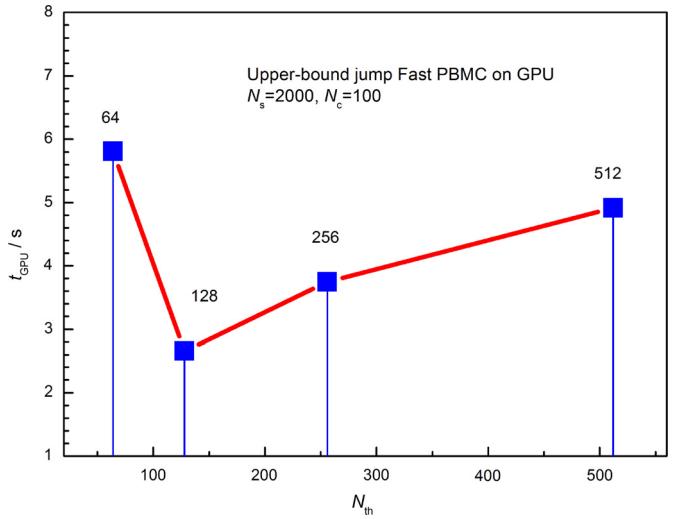

Fig. 17. Time consumption on GPU for different numbers of threads.

coagulation), the double looping over all simulation particles is required in normal PBMC methods, and the computational cost is $O(N_s^2)$. In order to improve computational efficiency of PBMC for coagulation while maintain its computational accuracy at the same time, the PBMC is accelerated on three levels. On the level of Markov jump model, a new coagulation rule model that describes coagulation between two differentially weighted simulation particles was constructed and validated. In this way, the upper-bound jump is chosen because its computing time is greatly reduced whereas errors are limited within the acceptable range in engineering application. On the stochastic algorithm level, the fast-sampling method based on AR scheme is proposed via introducing the weighted majorant kernel, to avoid the direct and time-consuming computation of the maximum coagulation rate and mean time-step. We have determined the fastest Markov jump (the upper-bound jump), the most efficient procedure for selecting coagulation pair (the fast acceptance–rejection scheme), and the optimal number of threads on GPU ($N_{th} = 128$ for AR attempts). Simulation results shows the computing time was only $O(N_s)$, and very remarkable speed-up ratio is achieved. The GPU parallel computing provides a large number of parallel threads at low costs, which is especially efficient for conventional PBMC in a single-cell system. However, the fast PBMC has significantly reduced the computational load, being not appropriate for using GPU computing in a single-cell system. The simulation of a multi-cell system takes full advantage of the intrinsic parallel property of real-world evolution, that is, multiple cells can be tracked independently and simultaneously by the fast PBMC. The GPU-based parallel algorithm for a multi-cell system consisting of 100 cells is validated at first, in which each cell is computed by utilizing the fast PBMC with upper-bound jump. The best results in view of accuracy and computational efficiency is to utilize a moderate number of parallel threads, as for the maximum number of threads the efficiency is not significantly better. An optimal number of threads has been found in this case, namely the smallest time consumption while N_{th} is equal to 128. The speedup of the accelerated parallel

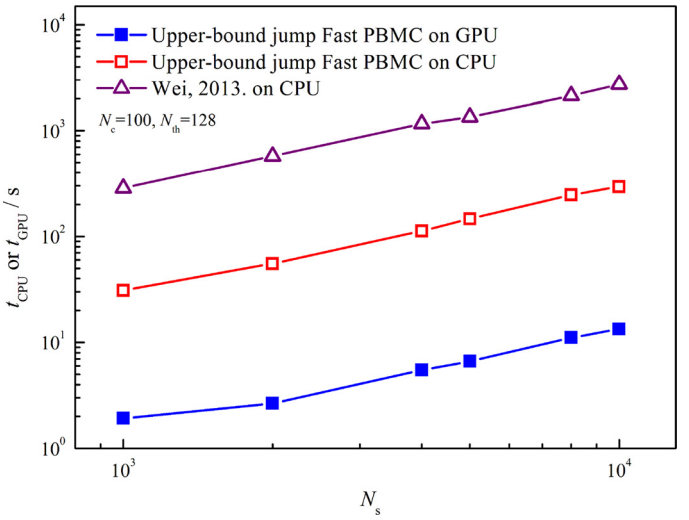

Fig. 18. Computing time required by the CPU or GPU for different numbers of simulation particles per cell.

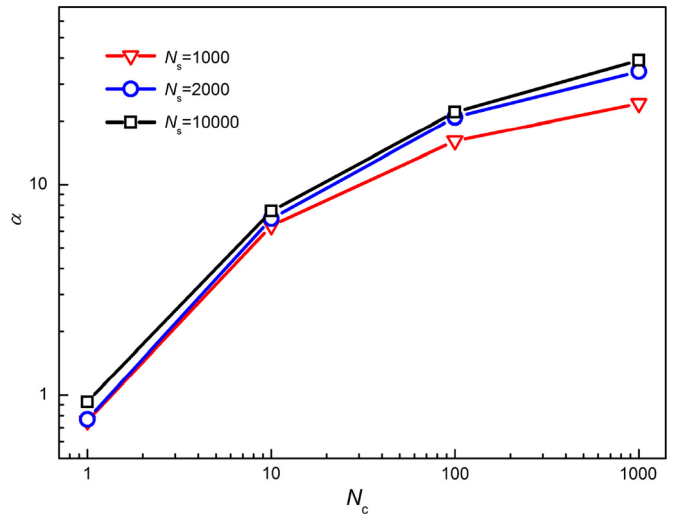

Fig. 19. Speedup achieved on GPU for different numbers of cells N_c and particle numbers N_s.

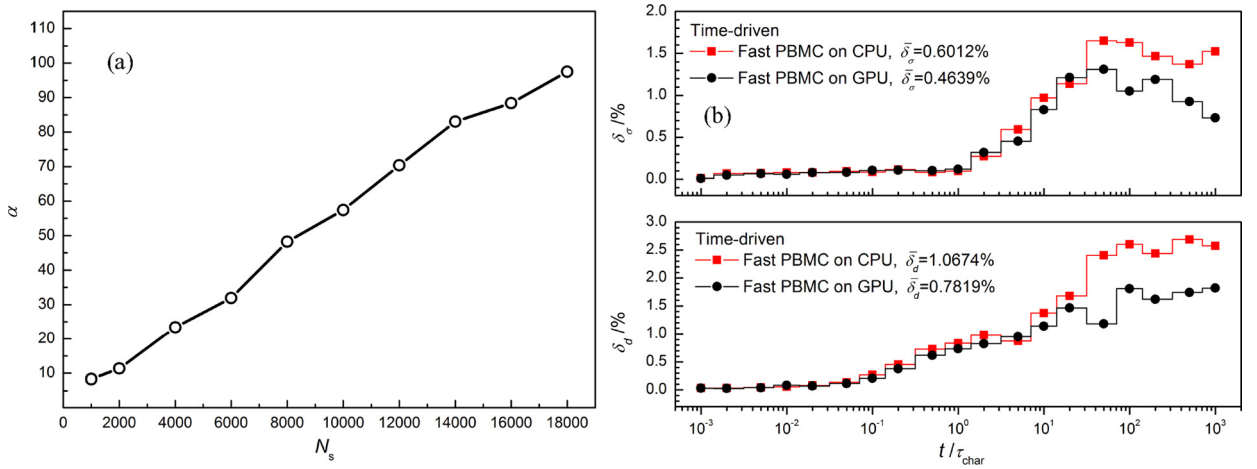

Fig. 20. GPU parallel computing of fast PBMC based on time-driven approach in a single cell: (a) the speedup ratio; (b) the relative error of d_g and σ_g.

on GPU was investigated. It was found that the GPU-accelerated is especially efficient for systems containing a large number of cells. Moreover, the time-driven fast PBMC have higher parallel efficiency and the same time-step in favor of coupling with CFD. On the whole, the comprehensive approach therefore especially suited for cases where PBMC is to be coupled to CFD when the particle properties change due to other mechanisms such as transport from other cells in a CFD surrounding.

Acknowledgements

The work was partly presented and benefited from discussions at "PBM2013" conference (5th International Conference on Population Balance Modeling in Bangalore, India). The research was supported by the National Natural Science Foundation of China (51276077, 51390494), and Open Project of State Key Laboratory of Multiphase Complex Systems (MPCS-2011-D-02). Meanwhile, Prof. F.E. Kruis and Dr. J. Wei (Institute for Technology of Nanostructures, University of Duisburg–Essen), are also appreciated for the related communication and discussion.

References

[1] S.K. Friedlander, Smoke, Dust and Haze: Fundamentals of Aerosol Dynamics, 2nd ed., Oxford University Press, New York, 2000.
[2] M. Frenklach, S.J. Harris, Aerosol dynamics modeling using the method of moments, J. Colloid Interface Sci. 118 (1987) 252–261.
[3] F. Gelbard, Y. Tambour, J.H. Seinfeld, Sectional representations for simulating aerosol dynamics, J. Colloid Interface Sci. 76 (1980) 541–556.
[4] M. Yu, J. Lin, Nanoparticle-laden flows via moment method: a review, Int. J. Multiph. Flow 36 (2010) 144–151.
[5] S. Rigopoulos, Population balance modelling of polydispersed particles in reactive flows, Prog. Energy Combust. Sci. 36 (2010) 412–443.
[6] H. Zhao, F.E. Kruis, C. Zheng, Monte Carlo simulation for aggregative mixing of nanoparticles in two-component systems, Ind. Eng. Chem. Res. 50 (2011) 10652–10664.
[7] H. Zhao, C. Zheng, Two-component Brownian coagulation: Monte Carlo simulation and process characterization, Particuology 9 (2011) 414–423.
[8] H. Zhao, C. Zheng, A population balance-Monte Carlo method for particle coagulation in spatially inhomogeneous systems, Comput. Fluids 71 (2013) 196–207.
[9] P. Tandon, D.E. Rosner, Monte Carlo simulation of particle aggregation and simultaneous restructuring, J. Colloid Interface Sci. 213 (1999) 273–286.
[10] K. Liffman, A direct simulation Monte-Carlo method for cluster coagulation, J. Comput. Phys. 100 (1992) 116–127.
[11] F.E. Kruis, A. Maisels, H. Fissan, Direct simulation Monte Carlo method for particle coagulation and aggregation, AIChE J. 46 (2000) 1735–1742.
[12] A. Maisels, F. Einar Kruis, H. Fissan, Direct simulation Monte Carlo for simultaneous nucleation, coagulation, and surface growth in dispersed systems, Chem. Eng. Sci. 59 (2004) 2231–2239.
[13] M. Smith, T. Matsoukas, Constant-number Monte Carlo simulation of population balances, Chem. Eng. Sci. 53 (1998) 1777–1786.
[14] M. Goodson, M. Kraft, An efficient stochastic algorithm for simulating nano-particle dynamics, J. Comput. Phys. 183 (2002) 210–232.
[15] Y. Lin, K. Lee, T. Matsoukas, Solution of the population balance equation using constant-number Monte Carlo, Chem. Eng. Sci. 57 (2002) 2241–2252.
[16] H. Zhao, F.E. Kruis, C. Zheng, Reducing statistical noise and extending the size spectrum by applying weighted simulation particles in Monte Carlo simulation of coagulation, Aerosol Sci. Technol. 43 (2009) 781–793.
[17] H. Zhao, C. Zheng, A new event-driven constant-volume method for solution of the time evolution of particle size distribution, J. Comput. Phys. 228 (2009) 1412–1428.
[18] R.I.A. Patterson, W. Wagner, M. Kraft, Stochastic weighted particle methods for population balance equations, J. Comput. Phys. 230 (2011) 7456–7472.
[19] R.E.L. DeVille, N. Riemer, M. West, Weighted Flow Algorithms (WFA) for stochastic particle coagulation, J. Comput. Phys. 230 (2011) 8427–8451.
[20] A. Eibeck, W. Wagner, An efficient stochastic algorithm for studying coagulation dynamics and gelation phenomena, SIAM J. Sci. Comput. 22 (2000) 802–821.
[21] A. Eibeck, W. Wagner, Stochastic particle approximations for Smoluchowski's coagulation equation, Ann. Appl. Probab. 11 (2001) 1137–1165.
[22] J. Wei, A fast Monte Carlo method based on an acceptance–rejection scheme for particle coagulation, Aerosol Air Qual. Res. 13 (2013) 1273–1281.
[23] C. Lécot, W. Wagner, A quasi-Monte Carlo scheme for Smoluchowski's coagulation equation, Math. Comput. 73 (2004) 1953–1966.
[24] C. Lécot, A. Tarhini, A quasi-stochastic simulation of the general dynamics equation for aerosols, Monte Carlo Methods Appl. 13 (2008) 369–388.
[25] F.E. Kruis, J. Wei, T. van der Zwaag, S. Haep, Computational fluid dynamics based stochastic aerosol modeling: combination of a cell-based weighted random walk method and a constant-number Monte-Carlo method for aerosol dynamics, Chem. Eng. Sci. 70 (2012) 109–120.
[26] Z. Xu, H. Zhao, C. Zheng, Fast Monte Carlo simulation for particle coagulation in population balance, J. Aerosol Sci. 74 (2014) 11–25.
[27] I.J. Laurenzi, S.L. Diamond, Monte Carlo simulation of the heterotypic aggregation kinetics of platelets and neutrophils, Biophys. J. 77 (1999) 1733–1746.
[28] R. Irizarry, Fast Monte Carlo methodology for multivariate particulate systems—I: Point ensemble Monte Carlo, Chem. Eng. Sci. 63 (2008) 95–110.
[29] R. Irizarry, Fast Monte Carlo methodology for multivariate particulate systems—II: PEMC, Chem. Eng. Sci. 63 (2008) 111–121.
[30] S. Shekar, W.J. Menz, A.J. Smith, M. Kraft, W. Wagner, On a multivariate population balance model to describe the structure and composition of silica nanoparticles, Comput. Chem. Eng. 43 (2012) 130–147.
[31] S. Shekar, M. Sander, R.C. Riehl, A.J. Smith, A. Braumann, M. Kraft, Modelling the flame synthesis of silica nanoparticles from tetraethoxysilane, Chem. Eng. Sci. 70 (2012) 54–66.
[32] E. Debry, B. Sportisse, B. Jourdain, A stochastic approach for the numerical simulation of the general dynamics equation for aerosols, J. Comput. Phys. 184 (2003) 649–669.
[33] N. Riemer, M. West, R.A. Zaveri, R.C. Easter, Simulating the evolution of soot mixing state with a particle-resolved aerosol model, J. Geophys. Res., Atmos. 114 (2009) D09202.
[34] D.P. Landau, K. Binder, A Guide to Monte Carlo Simulations in Statistical Physics, Cambridge University Press, 2009.
[35] J. Wei, F.E. Kruis, GPU-accelerated Monte Carlo simulation of particle coagulation based on the inverse method, J. Comput. Phys. 249 (2013) 67–79.
[36] J. Wei, F.E. Kruis, A GPU-based parallelized Monte-Carlo method for particle coagulation using an acceptance–rejection strategy, Chem. Eng. Sci. 104 (2013) 451–459.
[37] J. Wei, A parallel Monte Carlo method for population balance modeling of particulate processes using bookkeeping strategy, Physica A 402 (2014) 186–197.
[38] A.L. Garcia, C. van den Broeck, M. Aertsens, R. Serneels, A Monte Carlo simulation of coagulation, Physica A 143 (1987) 535–546.
[39] H. Zhao, C. Zheng, M. Xu, Multi-Monte Carlo method for coagulation and condensation/evaporation in dispersed systems, J. Colloid Interface Sci. 286 (2005) 195–208.
[40] H. Zhao, C. Zheng, M. Xu, Multi-Monte Carlo approach for general dynamic equation considering simultaneous particle coagulation and breakage, Powder Technol. 154 (2005) 164–178.
[41] H. Zhao, C. Zheng, M. Xu, Multi-Monte Carlo method for particle coagulation: description and validation, Appl. Math. Comput. 167 (2005) 1383–1399.

[42] H. Zhao, C. Zheng, Correcting the multi-Monte Carlo method for particle coagulation, Powder Technol. 193 (2009) 120–123.
[43] H. Zhao, C. Zheng, The event-driven constant volume method for particle coagulation dynamics, Sci. China Ser. E, Technol. Sci. 51 (2008) 1255–1271.
[44] X. Hao, H. Zhao, Z. Xu, C. Zheng, Population balance-Monte Carlo simulation for gas-to-particle synthesis of nanoparticles, Aerosol Sci. Technol. 47 (2013) 1125–1133.
[45] H. Zhao, F. Einar Kruis, Dependence of steady-state compositional mixing degree on feeding conditions in two-component aggregation, Ind. Eng. Chem. Res. 53 (2014) 6047–6055.
[46] H. Zhao, F.E. Kruis, C. Zheng, A differentially weighted Monte Carlo method for two-component coagulation, J. Comput. Phys. 229 (2010) 6931–6945.
[47] S.-Y. Lu, Collision integrals of discrete-sectional model in simulating powder production, AIChE J. 40 (1994) 1761–1764.
[48] S. Vemury, S.E. Pratsinis, Self-preserving size distributions of agglomerates, J. Aerosol Sci. 26 (1995) 175–185.