Chemical Engineering Science 104 (2013) 451–459

Contents lists available at ScienceDirect

Chemical Engineering Science

journal homepage: www.elsevier.com/locate/ces

A GPU-based parallelized Monte-Carlo method for particle coagulation using an acceptance–rejection strategy

Jianming Wei [a,b], Frank Einar Kruis [a,b,*]

[a] Institute for Nanostructures and Technology (NST), University of Duisburg-Essen, Campus Duisburg, 47048 Duisburg, Germany
[b] Center for Nanointegration Duisburg-Essen (CENIDE), University of Duisburg-Essen, Campus Duisburg, 47048 Duisburg, Germany

HIGHLIGHTS

- An acceleration method for a Monte-Carlo method for coagulation is developed.
- It estimates the maximum coagulation rate based on the mean coagulation rate.
- The scheme is parallelized on a low-cost Graphics card (GPU).
- Large speedups (> 100) are obtained for systems having a large number of cells.
- Hence, it is especially suited for combining stochastic particle dynamics and CFD.

ARTICLE INFO

Article history:
Received 5 December 2012
Received in revised form
17 June 2013
Accepted 8 August 2013
Available online 27 September 2013

Keywords:
Coagulation
Simulation
Particulate processes
Population balance
Monte-Carlo
GPU

ABSTRACT

A graphics processing unit (GPU)-based Monte Carlo (MC) algorithm for particle coagulation using an acceptance–rejection (AR) strategy leading to improved computing efficiency has been developed and validated. The use of GPUs in high-performance computing is attractive due to the low cost per core, currently some 1–2 EUR. The GPU-implementation developed takes full advantage of the intrinsic parallel property featured by the AR strategy, namely, multiple AR attempts are carried out independently on many threads simultaneously. It uses an efficient way to obtain an estimation for the maximum coagulation kernel from the mean kernel. The method has been benchmarked by a sectional method validating its computing accuracy. Especially when a large number of cells is being handled at the same time, remarkable speed-up factors are achieved. This makes the method, a choice when population balances have to be solved in a CFD environment, which is demonstrated by means of a case study describing simultaneous coagulation, nucleation and diffusion in 1D. In summary, the simulations show that a MC method for particle coagulation based on the AR strategy can be efficiently parallelized on a GPU.

1. Introduction

The size evolution of nanoparticles is relevant for many fields of science and engineering (Friedlander, 1997; Ramkrishna, 2000) such as gas-to-particle conversion processes, combustion processes, material synthesis, flue gas cleaning technology, etc. Among the various mechanisms involved, coagulation is one of the most common particle growth mechanisms and is often decisive for the evolution of particle size distribution (PSD). It influences the particle size and number concentration as two particles having collided form one new and larger particle. Mathematically, coagulation is governed by the following population balance equation (PBE):

$$\frac{\partial n_p(v,t)}{\partial t} = \frac{1}{2}\int \beta(v-u,u,t)n_p(v-u,t)n_p(u,t)du - n_p(v,t)\int \beta(v,u,t)n_p(u,t)du \quad (1)$$

where $n_p(v,t)$ is the particle size distribution function at time t and β is the collision frequency function (i.e., coagulation kernel) for two particles with volume v and u. The first term on the right-hand side of Eq. (1) indicates the increase in the numbers of particles with volume v due to collisions between two smaller ones; the second term stands for decrease in number due to coagulation with other particles. The factor 1/2 is introduced since collisions are counted twice in the integral.

* Corresponding author at: Institute for Nanostructures and Technology (NST), University of Duisburg-Essen, Campus Duisburg, Bismarckstr. 81, 47048 Duisburg, Germany.
Tel.: +49 203 379 2899; fax: +49 203 379 3268.
E-mail address: einar.kruis@uni-due.de (F.E. Kruis).

0009-2509/$ - see front matter © 2013 Published by Elsevier Ltd.
http://dx.doi.org/10.1016/j.ces.2013.08.008

Broadly speaking, there are two classes of methods, namely deterministic and nondeterministic methods, that are now applicable to solve the above PBEs. The first class includes sectional methods (Jenong and Choi, 2001; Landgrebe and Pratsinis, 1990; Mitrakos et al., 2007; Wu and Biswas, 1998), moment methods (Brown et al., 2006; Park et al., 2001; Terry et al., 2001; Yamamoto, 2004; Yu et al., 2008), the latter features Monte Carlo (MC) methods (Efendiev and Zachariah, 2002; Garcia, 1987; Kruis et al., 2000; Lee and Matsoukas, 2000; Liffman, 1992; Maisels et al., 2004; Shah et al., 1977; Smith and Matsoukas, 1998; Zhao and Kruis, 2011; Zhao et al., 2010, 2005a, 2005b; Zhao and Zheng, 2009a, 2009b, 2011, 2013; Zhao and Kruis, 2008). Among these methods, MC methods are receiving more and more attention, as in general they offer the following features:

- their stochastic and discrete nature makes them ideally suited to handle discrete events;
- the methods do not require any priori information of the PSD, such as shape of the distribution;
- each individual particle is treated as a single object which consists of several properties such as diameter, charge, composition, morphology, etc.;
- the accuracy is dependent only on the number of samples, regardless of the number of independent particle properties, hence, new properties of particles can be incorporated with minimum efforts, making the method easily extendible and flexible; and
- it is generic simple, robust and easy to code.

The major limitation of the MC methods lies in their relatively high computing costs, in particular for the cases where large numbers of simulation particles are involved. This is because MC methods are essential statistical simulation methods with their computational precision being inversely proportional to the square root of the total number of samples (here, simulation particles) (Liffman, 1992). In other words, in order to increase the accuracy of the simulation by a factor of 10, the number of samples should be increased by a factor of 100. Therefore, the number of simulation particles should be sufficiently large in order to obtain sufficient accuracy. On the other hand, CPU speed and memory capacity pose considerable limitations to systems with large number of particles. For example, current MC software on conventional PCs can deal with some 10^5 simulation particles without problems, but will reach the limits of computer speed and memory in systems where a large number of cells each having 10^5 simulation particles are simulated. In such a case one may resort to large parallelized mainframe computers so that results can be produced within a reasonable time. As a matter of fact, one of the most effective uses of parallel architectures is to simply perform independent MC simulations on different processors (Landau and Binder, 2009). A multi-cell system as is often used in chemical engineering modeling tools such as computational fluid dynamics (CFD) is a typical example of a system in which the MC simulations can be made independently on multiple processors. Currently, typical parallel architectures include CPU-based high performance clusters (HPC), graphic processing units (GPU) (NVIDIA, 2008), GPU-based clusters, etc. Compared with CPU computing, the NVIDIA® GeForce® series GPUs proved to be a more promising tool for large computing-intensive problems, as their computational capacities improve relentlessly and far outpaces that of the general-purpose microprocessors. For example, as far as the hardware in this work is concerned, the peak performance of the Intel CPU with four cores running on 2.66 GHz can attain 42.56 GigaFLOPS (10^9 float point operation per second) peak performance, whereas NVIDIA GTX 285 GPU with 240 cores can easily deliver 1062 GigaFLOPS peak performance. The use of GPUs in high-performance computing is also attractive due to the low cost per core, currently some 1–2 EUR. Moreover, the GPU solutions are also quite economical due to the low electricity consumption, minimal requirements for workspace and no need of high-power air conditioning when compared with HPC clusters (Molnar et al., 2010).

Two important characteristics of the GPU are the following. First, gaining access to the different cores of the GPU by means of the common unified device architecture (CUDA) (NVIDIA, 2007) software released by NVIDIA when imbedded in standard C software running on the CPU, a large number of copies of a GPU function called “kernel” is distributed to the available cores and executed there. This distribution uses a kernel grid, which is subdivided into blocks and each block is further subdivided into various threads. Second, the GPU contains both on-chip shared memory and off-chip global memory. For obtaining maximum performance, the GPU allows that the different types of memory can be selected by the programmer. The global memory offers a large memory size but has slow access whereas the shared memory has fast access but a much smaller memory size and it is visible only by other threads in the same block.

In this study, we focus on designing a MC method using an acceptance–rejection rule (AR) for coagulation simulation especially suited for parallelization on a GPU. Comparing to its counterpart, the Inverse scheme (Kruis et al., 2000), the rule adopted by the AR scheme for choosing the particle pair to be coagulated is more simple and straightforward. More significantly, the method used by the AR scheme when dealing with particle coagulation features an inherent parallelism, which allows it to take full advantage of the GPU architecture characterized by a great large number of light-weighted threads. As can be seen later in this paper, the computing performance using the AR strategy on a CPU–GPU mixed architecture can lead to improvements compared to a single CPU platform, depending upon the different size scales of the problem to be investigated. Beside developing an efficient parallel AR-based MC algorithm, efforts have also been made in this work to design a new scheme to quickly evaluate the maximum value of the coagulation rate, as it plays a crucial role in the choice of coagulation pair. The new scheme also fully uses the parallel features of GPU by drawing the intermediate results of the coagulation rate from different threads (in a block) to estimate the maximum value of the coagulation rate.

The rest of the paper is outlined as follows. Section 2 gives a detailed introduction to the MC method used in this work, followed by a description of the new strategy chosen. Section 3 addresses in detail the implementation of the parallel MC algorithm on the GPU. The simulation results and analysis are presented in Section 4. In Section 5 the paper is concluded.

2. Parallel MC and its implementation on GPU

The present work adopts a new MC method developed recently, the differentially weighted (DW) MC method (Zhao and Kruis, 2008). This DW MC introduces a weight property for each simulation particle, which allows to keep track of the particle size distribution over the full particle size spectrum and to preserve the simulation history of each particle. It has been selected here as (1) it automatically keeps the number of simulation particles constant, and (2) the particle weights make it very well suited for implementation in a multi-cell transport model, where simulation particles coming from different cells having different concentrations collide with each other. The DW MC method describes the coagulation rate between any two fictitious particles as

$$\beta'_{i,j} = \beta_{i,j} \frac{2w_j \max(w_i, w_j)}{w_i + w_j}, \tag{2}$$

where $\beta_{i,j}$ is the collision kernel between particle i and j, which is equal to that of $\beta(u,v)$ in Eq. (1); w_i and w_j are particle weights. Note that coagulation kernel $\beta_{i,j}$ is symmetric, i.e., $\beta_{i,j}=\beta_{j,i}$. Note that $\beta'_{i,j}$ is in general not any more symmetric because the introduced w_i is not necessarily equal to w_j.

2.1. Acceptance–rejection scheme used by DW MC

The DW MC method handles particle coagulation in a stochastic manner. Presently, two strategies are widely employed by a variety of MC methods, i.e., the inverse method (Nanbu, 1980) and the acceptance–rejection (AR) method (Garcia, 1987), with the major difference in the way of selecting the particle pair to be coagulated. This work focuses on the AR scheme, which uses an acceptance–rejection rule to decide whether a coagulation event between a selected particle pair, i and j, is occurring or not (i.e., is accepted or rejected). This rule takes the form

$$r \le p_{i,j} = \frac{\beta'_{i,j}}{\beta'_{\max}}, \tag{3}$$

where $p_{i,j}$ is the coagulation probability between particle i and j, $\beta'_{\max}$ is the maximum coagulation rate over all particle pairs, and r is a random number generated uniformly in the interval (0,1), i.e., $r \sim U(0,1)$. The particle pair will be accepted if Eq. (3) is true, or rejected otherwise. If the latter is the case, a new particle pair will be chosen at random and checked again with Eq. (3). This randomly select-and-check procedure (called AR attempt in the remaining of this paper) proceeds until a particle pair has been found.

Following the determination of the particle pair, the average time step, $\langle\tau\rangle$, for the coagulation event between can be estimated via

$$\langle\tau\rangle = \frac{1}{\sum_{k=1}^{n_s}\sum_{l=1,l\neq k}^{n_s}\beta'_{k,l}}, \tag{4}$$

where n_s is the number of particles used in simulation.

The time needed for an exact evaluation of $\langle\tau\rangle$ scales as n_s^2. For large n_s, however, one may take the average of the denominator in Eq. (4) using only one particle pair, namely the selected one to be coagulated (Garcia, 1987). This method is very efficient but imprecise. A more precise method for estimating $\langle\tau\rangle$ is to use the values of $\beta'_{i,j}$ of particle pairs earlier rejected (Smith and Matsoukas, 1998), namely,

$$\langle\tau\rangle = \frac{2}{n_s(n_s-1)\overline{\beta'}}, \tag{5}$$

where $\overline{\beta'} = \sum_{m=1}^{n_a}(\beta'_{k,l})/n_a$, with n_a being the attempts made before accepting a particle pair.

2.2. Accelerated AR scheme including a novel method for estimating $\beta'_{\max}$

Like estimating time step, the cost for the calculation of $\beta'_{\max}$ scales with n_s^2 if all particle pairs are considered. A more efficient and simple method is to use a pre-computed and over-estimated $\beta'_{\max}$. However, due to the large and constant denominator, the acceptance rate is deteriorating as the coagulation is advancing.

An alternative way to obtain $\beta'_{\max}$ is by computing the collision kernel directly with knowledge of the lower and upper bound of the particle size spectrum. This is because for realistic coagulation kernels, such as the Brownian kernel, the maximum value occurs always between the smallest and largest particle size. The computing cost for determining the extreme sizes is in general trivial, hence this scheme is computational efficient. Indeed, this is a good choice on the CPU, but some problems may arise when applying it to simulations on GPU: (1) one has to pre-fetch information of the particle sizes saved on the global memory very frequently before comparing them; however, access to global memory is general fairly slow; (2) the comparison itself is a time-consuming process, as it is a sequential-prone process and may not be effectively parallelized (Wei and Kruis, 2013). In summary, it is wise to avoid this scheme for the GPU.

We propose a new GPU-friendly method to estimate $\beta'_{\max}$ such that the previous drawbacks can be circumvented. The key to our new scheme is to establish a relation between the maximum coagulation rate, $\beta'_{\max}$, and total coagulation rates, $\overline{\beta'}$, from attempts being made for finding a particle pair. In particular, $\beta'_{\max}$ is expressed by

$$\beta'_{\max} = \gamma\overline{\beta'} = \gamma\sum_{1}^{n_a}\beta'_{i,j}/n_a, \tag{6}$$

where γ is the coefficient linking $\overline{\beta'}$ to $\beta'_{\max}$. Clearly, if an estimation of γ can be made on the basis of the relevant coagulation kernel and the form of the expected size distribution, then one can efficiently estimate $\beta'_{\max}$ as $\overline{\beta'}$ is known because it is used to determine the time step by Eq. (5). Unfortunately, it is difficult to get a description of γ analytically, but numerical means can efficiently be used to exploit the characteristic range of γ. In this paper, we do this for the coagulation kernel which is the most relevant for nanoparticles in gases, the free-molecular regime where

$$\beta_{i,j} = K_f(d_i+d_j)^2\left(\frac{1}{d_i^3}+\frac{1}{d_j^3}\right)^{1/2}, \tag{7}$$

where d_i and d_j stand for diameters of particle i and j, respectively, $K_f=(3\kappa_B T/\rho_p)^{1/2}$ with T being the temperature, κ_B the Boltzmann constant and ρ_p the particle density.

The value of γ will also depend on the form of the particle size distribution. For the free-molecular regime, the resulting size distributions are self-preserving and close to log-normal (Lai et al., 1972). Therefore, unimodal log-normal distributions characteristic for a coagulating aerosol are generated

$$f_{uni}(\ln d) = \frac{1}{\sqrt{2\pi}\ln\sigma_g}\exp\left(-\frac{(\ln d-\ln d_g)^2}{2(\ln\sigma_g)^2}\right), \tag{8}$$

where σ_g is the geometric standard deviation and d_g is the geometric mean diameter.

However, other mechanisms such as nucleation or transport can introduce new particles in the simulation volume. This is simulated by considering a bimodal particle size distribution

$$f_{bi}(\ln d) = \frac{1}{\sqrt{2\pi}\ln\sigma_{g1}}\exp\left(-\frac{(\ln d-\ln d_{g1})^2}{2(\ln\sigma_{g1})^2}\right) + \frac{N_2}{N_1}\frac{1}{\sqrt{2\pi}\ln\sigma_{g2}}\exp\left(-\frac{(\ln d-\ln d_{g2})^2}{2(\ln\sigma_{g2})^2}\right) \tag{9}$$

with the indices 1 and 2 now referring to the two modes and N_2/N_1 as the ratio of the number concentrations. The two modes represent the nucleation mode and the coagulation mode, or two modes stemming from different locations and having different coagulation history.

By varying σ_g, various unimodal particle size distributions are simulated. Furthermore, by discretizing the specific particle size distribution, i.e., dividing the size distribution into limited size bins, $\beta'_{\max}$ is obtained by considering all possible combinations of different size bins. $\overline{\beta'}$ is calculated by picking randomly a large number of different particle pairs. With both $\beta'_{\max}$ and $\overline{\beta'}$ being known, γ can be calculated from Eq. (6).

Figs. 1 and 2 show the behavior of γ for different values of σ_g for a unimodal and a bimodal particle size distribution, respectively. Basically, γ increases along with σ_g for the case where particle size

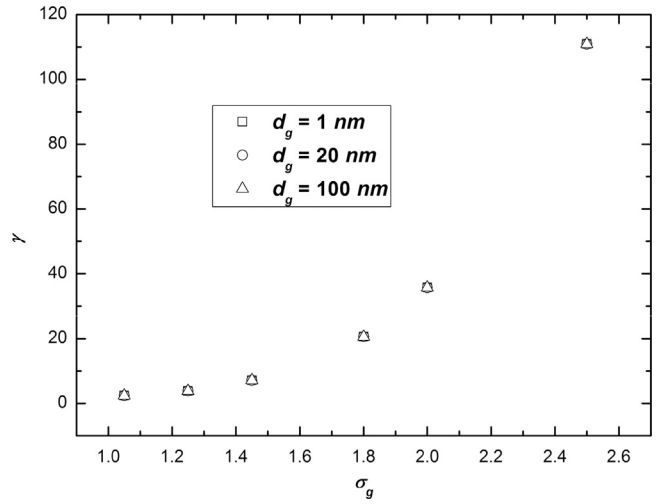

Fig. 1. γ as a function of the geometric standard deviation for a unimodal particle size distribution.

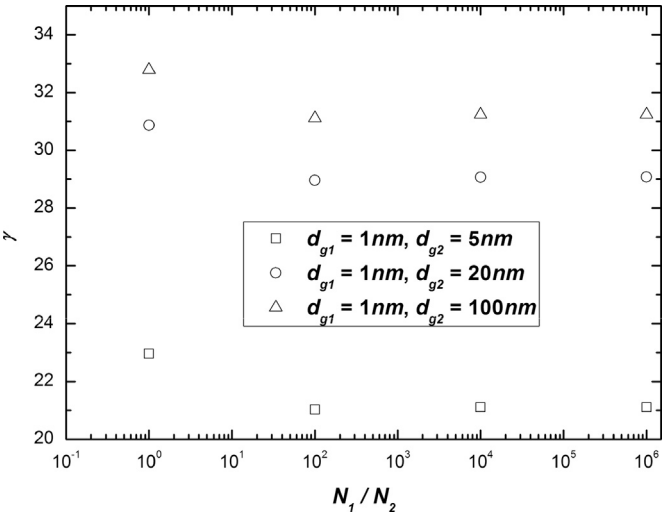

Fig. 2. γ for a bimodal particle size distribution.

distribution is unimodal. It seems to be independent of d_g. More precisely, γ increases from around 1 when σ_g=1.0, to 40 for σ_g=2.0. Larger values for σ_g do not occur in free-molecular coagulation, only when a bimodal distribution develops larger values for the integral value of σ_g are found. The bimodal case is however investigated separately.

The bimodal case is more complicated, as γ now is depending on more parameters, like the ratio N_2/N_1. Fig. 2 shows that N_2/N_1 is however not strongly influencing the value of γ. When d_g of the two modes becomes more disparate, γ increases. However, $\gamma \leq 33$ for the chosen case $\sigma_{g1}=\sigma_{g2}=1.5$. It is likely that γ becomes larger when taking for both σ_{g1} and σ_{g2} values above 1.5. γ is increased to about 100 when $\sigma_{g1}=\sigma_{g2}=2$.

Several options are in principle available for the choice of γ. The first option is to determine first whether the size distribution can be described by a unimodal or bimodal lognormal size distribution, and determine the characteristic parameters of the size distribution. These are then used to choose an optimal value of γ on the basis of Figs. 1 and 2. The second option is to take a worst-case estimation of γ, and keep it constant throughout the simulation. In this work, the second option has been chosen for the following reasons. As will be seen later, the GPU implementation of the AR scheme is especially efficient for the multi-cell system. For a single cell the speed gain is too small to justify the extra programming effort. For the multi-cell case, each cell will have its own size distribution and thereby an optimal value of γ. For the algorithm as will be presented in Section 3 it is not advantageous to have for each cell a different value of γ. The algorithm would become much complicated, anyway for each cell the same number of AR judgments are performed as it is not possible to have different number of threads for each single block (here, a single cell). Furthermore, it is computationally costly to determine the size distribution characteristics after each step. Finally, for obtaining a fairly accurate value of $\overline{\beta'}$ anyway a not too small number of particle pairs has to be selected and its coagulation kernel determined, using a small value of γ does not decrease this computational effort. All these considerations make the choice for option 2 inevitable.

In this work, we have chosen γ=40. This value is adequate to deal in the free-molecular regime with unimodal distributions with σ_g up to 2.0, as well as bimodal distributions with σ_g up to 1.5. Its value will be in most cases too large, so that more pairs than necessary will be needed. However, the numerical penalty of this is relatively small, as will become clear in Section 3. In cases homogenous nucleation plays an important role, its optimal value might have to reconsidered, but in case coagulation and transport being the main mechanism this value is considered to be adequate.

3. Implementation of the accelerated AR scheme for particle coagulation on the GPU

A characteristic feature of the AR scheme for particle coagulation is that two arbitrary AR attempts for choosing a coagulated pair are uncorrelated, in the sense that no information exchange is necessary between them. Thus, multiple attempts for finding an accepted particle pair can be carried out simultaneously and independently. This type of data parallelism possible in case of the AR scheme makes it very suitable to be implemented in a massive parallel manner on the GPU. It has to be realized that in a CPU version of the AR scheme, the AR attempts are halted as soon as a particle pair is accepted, whereas in a GPU version simultaneously a fixed number of AR attempts are carried out. In the last case, several possibilities exist as follows:

(1) *No accepted particle pair is found*: in that case the procedure has to be repeated.
(2) *One accepted particle pair is found*: this one is selected for coagulation. The procedure of the differentially weighted MC method (Zhao and Kruis, 2008) is applied to generate two new particles having new properties and weighting factors. This procedure conserves the number of simulation particles, which is perfectly compatible with the algorithm described in this section.
(3) *Two or more accepted particle pairs*: accepting more than one pair, although theoretically possible when also adjusting $\langle\tau\rangle$, would lead to too complicated CUDA kernels which usually lead to a loss of the speed gained by the GPU. Therefore, only one of the accepted pairs is randomly selected as definitive choice.

As explained in Section 2.2, the accelerated AR scheme requires multiple attempts for finding an accepted particle pair. In most cases, the conservative choice of γ=40 which for most cases will be leading to an overestimation of $\beta'_{\max}$. It has to be realized, however, that another task is to determine $\overline{\beta'}$ so that the time step can be determined with sufficient precision. This is typically done by randomly selecting a sufficient number of particle pairs, determining their individual β' and calculating the mean value $\overline{\beta'}$. Even with an optimal (low) value of γ which in principle would allow to use a small number of particle pairs for the AR attempts, more particle pairs are

necessary to determine $\overline{\beta'}$ adequately. In view of computational efficiency and algorithmically simplicity, it is optimal to take the same number of particle pairs for determining $\overline{\beta'}$ and for performing AR attempts.

GPU programming is strongly dependent on the choice of blocks and thread numbers. It is important to notice that within a single block, the various threads can make use of the shared memory, which has a very fast access but is visible only by other threads in the block. In this work, the goal is to develop an algorithm suitable not only for simulating a single compartment or cell but also for many cells simultaneously, as most numerical methods for simulating particle dynamics are ultimately applied in a multi-cell or CFD environment. Therefore, in the following we consider a system having n_c cells containing each n_s simulation particles, whereas the number of (simultaneous) AR attempts for a single cell as well as the number of particle pairs use to determine $\overline{\beta'}$ is fixed as n_a. An efficient way of distributing the computational tasks is to choose the number of blocks equal to n_c and the number of threads equal to n_a. The algorithm can handle values of n_s much larger than n_a. The only limitation is that the global memory of the GPU has to be large enough to accommodate $n_c n_s$ simulation particles in total. With a global memory of 1.0 GB as is the case for the NVIDIA GTX 285, it is possible to use $n_s = 120{,}000$ when $n_c = 1000$. The maximum number of blocks is currently 65,535, but is relatively easy to consider a larger number of cells by means of introducing simple loops in the blocks. The maximum thread number, currently 512 for CUDA V1.3, is however the absolute limit to the number of parallel AR attempts due to the data exchange and synchronization between threads. The influence of the number of blocks and threads on the computational efficiency as well as accuracy will be studied in Section 4.

The complete accelerated AR scheme, as shown in Fig. 3, basically consists of the following four steps:

(1) In each thread, two random number are generated for randomly selecting particles i and j and the corresponding coagulation rate is computed.
(2) When all threads are ready (the threads allow synchronization), the coagulation rate from all threads in a block is added up to obtain $\overline{\beta'}$ and calculate $\beta'_{\max}$, with the aid of Eq. (6). This is done with the help of the fast shared memory available to each thread.
(3) A third random number is generated in each thread to make the judgment according to Eq. (3) on the acceptance or rejection of the pair under consideration.
(4) The outcome of an AR attempt made on an individual thread is either acceptance (marked as A) or rejection (marked as R). One of the A-particle pairs is randomly selected as definitive choice. Under CUDA, this is automatically obtained when all threads with A write the indices of their selected particle pair to the same memory address. These indices will be automatically overwritten when more than one A is found, the memory address will contain the results of the thread which was the last one to send its results. As this is a random process, there is no need for a selection procedure to come to a single selected particle pair.

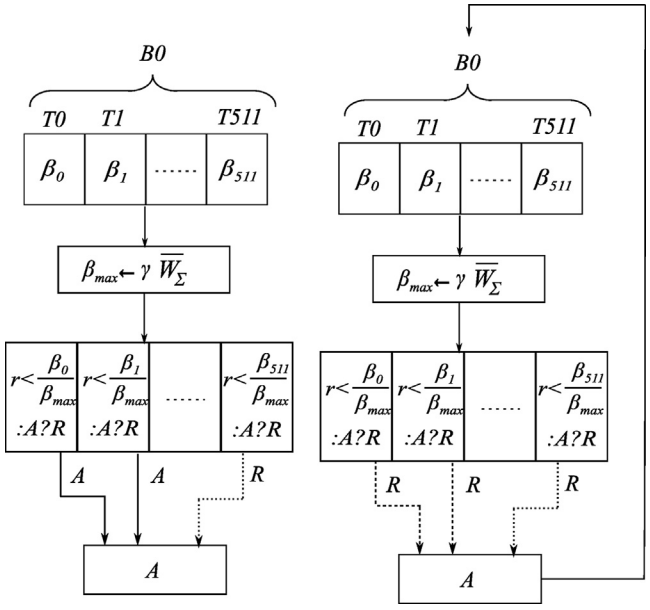

Fig. 3. Implementation of the accelerated AR scheme on the GPU.

Obviously, this procedure has to be repeated for any block (cell) until a particle pair has been accepted. When a pair has been selected, the new particle properties (both diameters and weighting factors) are calculated and the time step is calculated for each cell (block) on the basis of the estimated $\overline{\beta'}$. As the decision to continue the simulation has to be taken on the CPU level (as the CPU launches the GPU kernels), the time step information is sent to the CPU. Individual cells can be finished (meaning, the block execution is stopped) as soon as their total simulation time reaches the required time. All particle properties are kept in the GPU global memory. Specific properties such as d_g and σ_g are calculated with the help of special CUDA kernels, accessing the GPU global memory.

4. Simulation results and discussion

In this section, the accelerated AR scheme for the GPU is validated by means of a comparison to a benchmark solution and a study is made of the computing efficiency of the GPU AR scheme. Also a case study of simultaneous coagulation, nucleation and transport due to diffusion is presented.

4.1. Validation procedure

The GPU-based algorithm is validated by comparison with a benchmark numerical solution, which is here an implementation of a discrete sectional model (Lu, 1994) using 200 sections and a sectional spacing of 1.12. The system used for validation is the classical case in which an initially monodisperse distribution ($\sigma_{g,0} = 1.0$) with $d_{g,0} = 3$ nm evolves into the self-preserving size distribution. The results will be expressed as function of t/τ_{char} in which τ_{char} is the time needed to decrease the initial particle concentration by a factor of 2, given by (Kodas, 1999)

$$\tau_{char} \approx \frac{1}{(3V_t/4\pi)^{1/6} K_f N_0^{5/6}}, \tag{10}$$

where V_t is the total aerosol volume (m^3/m^3) and N_0 is the initial particle number concentration.

For evaluation of statistical errors, a number of runs (n_r) with different random numbers and different number of simulation particles (n_s) are carried out. The mean error in d_g is evaluated over $n_t = 16$ different time points logarithmically distributed between $10^{-3}\tau_{char}$ and $10^{3}\tau_{char}$

$$\varepsilon_{mean} = \frac{\sum_{i=1}^{n_t} \left(\left| \sum_{j=1}^{n_r} d_g(i,j)/n_r - d_g^{sec}(i) \right| / d_g^{sec}(i) \right)}{n_t}. \tag{11}$$

The simulation calculations were performed on a desktop system equipped with an Intel Core™ 2 Quad 2.66 GHz Processor, 8 GB RAM and NVIDIA Geforce GTX 285 GPU. The maximum number of threads on this GPU is 512.

4.2. Validation of accelerated AR scheme on the GPU

The evolution of particle size and geometric standard deviation calculated by means of the benchmark discrete-sectional model as well as the GPU-version of the accelerated-AR scheme using $\gamma=40$ are shown in Figs. 4 and 5, respectively. The calculation results obtained from GPU were based on 100 simulation runs. As can be seen, particle size increases due to coagulation from $10^{-3}\tau_{char}$ to $10^{3}\tau_{char}$. Likewise, the geometric standard deviation of particle size distribution, σ_g, increases from a mono-disperse size distribution at the early phase of the coagulation, $\sigma_g=1$, to a self-preserving size distribution, $\sigma_g=1.46$, as shown in Fig. 5. A very good agreement can be seen between the accelerated AR scheme and the benchmark solution.

The statistical fluctuations of the results are inherent to Monte Carlo simulations. The statistical scatter can be reduced by increasing the number of simulation runs n_r or increasing the number of simulations particles n_s. Furthermore, the optimal number of threads on the GPU n_{th} has to be determined. By increasing n_{th} a more precise estimation of $\overline{\beta'}$ is made, leading to a more accurate the mean time step $\langle\tau\rangle$. However, the drawback is that the computing time will be larger. Fig. 6 shows the mean statistical error ε_{mean} in d_g and σ_g as function of n_s and n_{th},

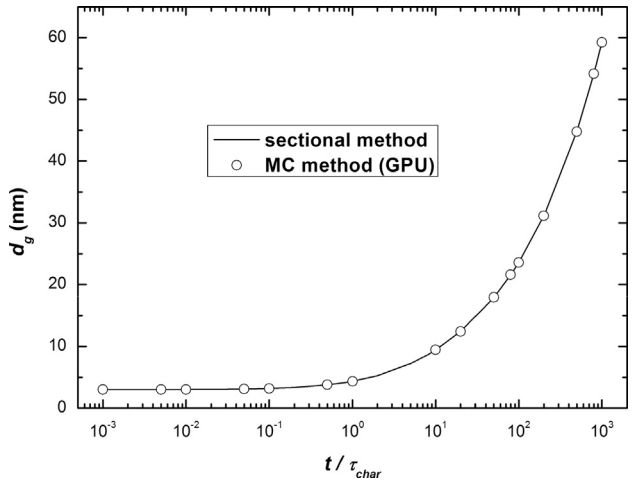

Fig. 4. Particle size comparison between the sectional method and DW MC using the accelerated AR scheme.

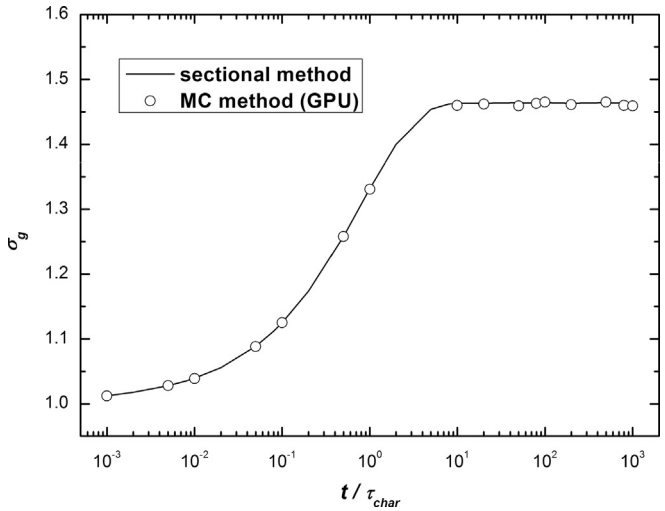

Fig. 5. Comparison of geometric standard deviations between the sectional method and DW MC using the accelerated AR scheme.

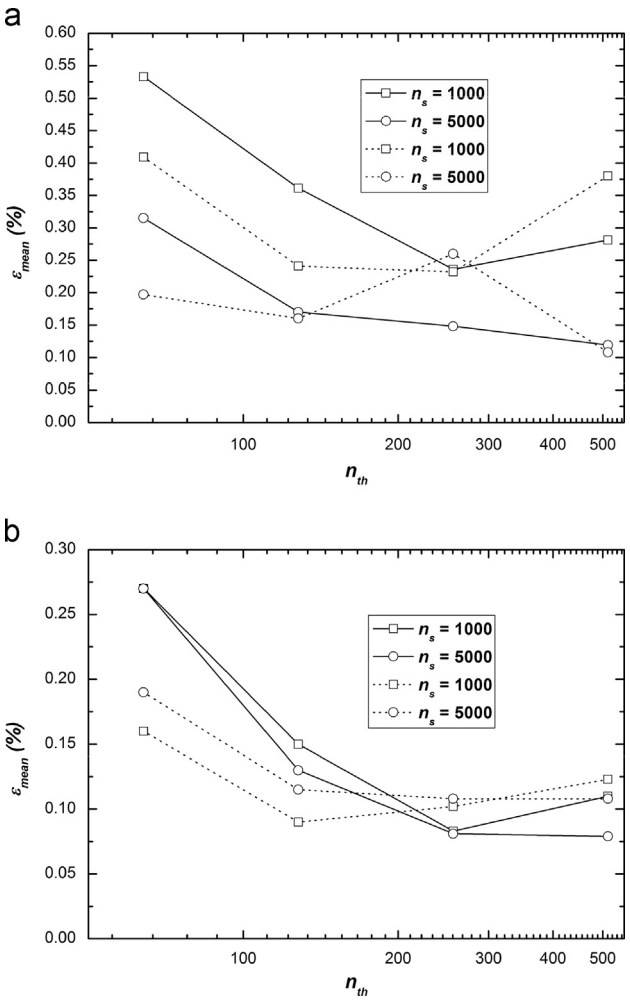

Fig. 6. Mean error ε_{mean} in the particle diameter d_g (solid line) and standard deviation σ_g (dashed line) calculated with the GPU-based MC with AR scheme as a function of the number of threads per block n_{th} under the number of MC runs: (a) $n_r=5$ and (b) $n_r=100$.

in Fig. 6(a) for $n_r=5$ and in Fig. 6(b) for $n_r=100$. As can be seen, the particle size and geometric standard deviation obtained from the accelerated AR scheme are fairly accurate, as ε_{mean} is always less than 0.55%. Further careful examination of the figures shows that indeed, ε_{mean} decreases when n_r or n_s increases, although some statistical scattering occurs. Interestingly, the effect of increasing n_s is relatively small in case n_r is large. It seems to be sufficient to choose either n_r or n_s large. Increasing n_{th} leads to an improvement in accuracy, however the effect is less pronounced for larger n_{th} and beyond $n_{th}=256$ no improvement can be detected. Choosing a too large n_{th} is therefore inefficient, as the computing time rises with increasing n_{th}.

4.3. Computational efficiency of accelerated AR scheme on the GPU

As the purpose of the present paper is to develop an accelerated AR scheme based on the GPU, the performance improvement of the parallel AR scheme on the GPU has been investigated. Clearly, the most straightforward way for evaluating the improvement by using the GPU is to directly measure the computer time used on the CPU and GPU respectively, and calculate the associated speedup, α, which can be defined as

$$\alpha=\frac{t_C}{t_G}, \tag{12}$$

with t_C being the execution time on the CPU, and t_G the computer time on the GPU. It should be pointed out here that in order to perform an objective comparison, the times of AR attempt being made on the CPU is constrained to a prescribed number that is equal to the thread number per block on the GPU, i.e., $n_a = n_{th}$. Evidently, this has changed slightly the sequential way in which the AR code on the CPU works, namely the code is instead of being terminated when a desired pair is found, it now will continue to complete the remaining trials even if a desired pair has been found. Although for CPU code above strategy may need more computing overhead, but it leads to a better computing accuracy in estimating $\langle\tau\rangle$ using Eq. (7) and guarantees an impartial comparison between the CPU and GPU as in both cases the same precision in estimating $\langle\tau\rangle$ is used.

Many factors, including total cell number (n_c), thread number per block (n_{th}), particle number confined in a cell (n_s), influence the computing performance on the GPU. Hence, how the speedup behaves under these factors will particularly be studied. Fig. 7 shows how α behaves for different n_c, keeping $n_{th}=512$ and $n_s=2000$. Clearly, α increases from smaller than 3 to more than 100 as n_c is increased from 1 to 1000. Apparently the speedup is relatively small for the single-cell case, and does not justify the effort of GPU programming. This is caused by the fact that only a single block is used in that case, and therefore does not make full use of the GPU power. However, for the case when dealing with a large number of cells the number of blocks increases and therefore speedup also increases. Speedups of approximately 100 have been observed by many GPU developers, especially when many blocks are used to deal with multiple cells (Tolke and Krafczyk, 2008).

The speedup is also investigated as a function of the number of threads, now concentrating on the multiple cell case with $n_c=1000$ and applying $n_s=2000$. In Fig. 8 it can be seen that α increases with increasing n_{th}, reaching values as high as 180 for the largest number of threads possible on this type of GPU. However, it should be stressed here that one should not judge whether the highest efficiency on the GPU is acquired just from the speedup. For a better judgment, a direct comparison between the computer time used on the CPU and GPU is required, as shown in Fig. 9.

From Fig. 9 it can be seen that the computing time increases by almost a factor of 10 when moving from $t=\tau_{char}$ to $t=10^3\tau_{char}$. From coagulation theory it is expected that in this time the particle number concentration decreases by $\sim$1000 (2^{10}) and that approximately 10 times the concentration has been halved, which would indeed take 10 times the time needed for simulating up to $t=\tau_{char}$.

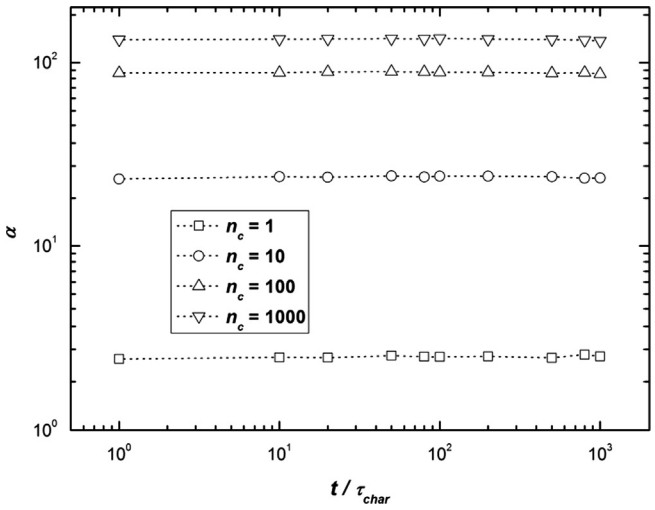

Fig. 7. Speedup achieved on the GPU for different numbers of cells.

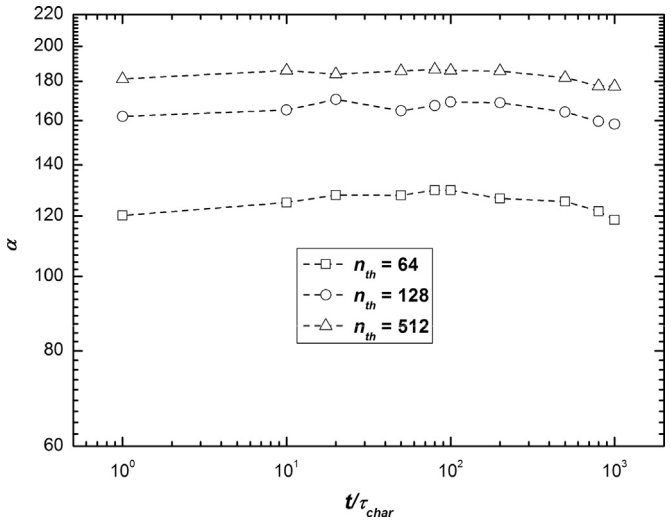

Fig. 8. Speedup achieved on the GPU for different numbers of threads.

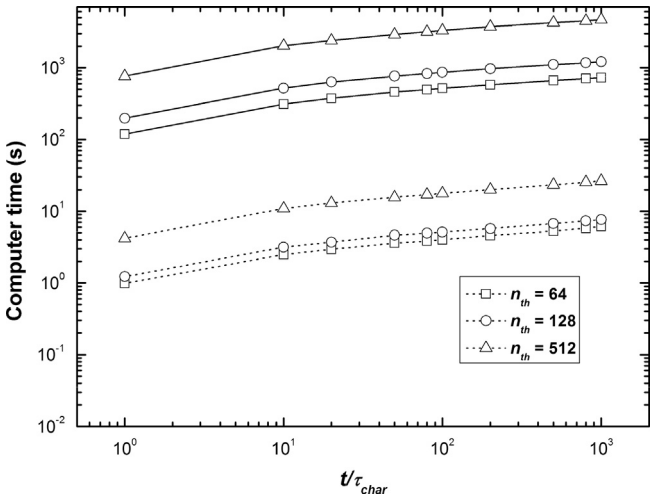

Fig. 9. Computer time used by the CPU and GPU for different numbers of threads per block.

It takes just 1 s with $n_{th}=64$ to perform the simulation for 1000 cells up to $t=\tau_{char}$, however it needs about 4 s with $n_{th}=512$. As we have seen in Figs. 5 and 6 that no real improvement in accuracy occurs when moving from $n_{th}=128$ to $n_{th}=512$, we can conclude that in this case applying $n_{th}=128$ is the optimal choice with respect to computing time and accuracy. This can be explained by the fact that performing simultaneously 512 AR attempts is not necessary, as in virtually all situations one pair is already found with 128 AR attempts. Probably the optimal situation is when the mean number of AR attempts before finding a pair is slightly lower than n_{th}. Overall, when using the accelerated AR scheme on the GPU for the multi-cell system with $n_s=1000$ the computing time for coagulation up to $t=10^3\tau_{char}$ is below 10 s when $n_{th}=128$. This means for a single cell a full coagulation simulation takes less than 10 ms, which makes the accelerated AR scheme on GPU the method of choice for coupled simulation of particle dynamics and transport which without GPU acceleration is very time-consuming (Kruis et al., 2012).

Another significant factor that affects the computational efficiency is n_s. Fig. 10 reports the actual time used on the CPU and GPU with different n_s. It is found that the computing time is approximately linearly dependent on n_s. For example, the

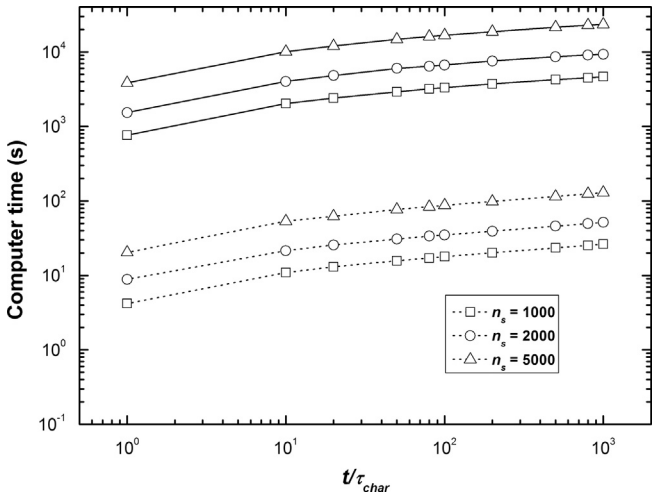

Fig. 10. The computer time used by the GPU and CPU for different numbers of simulation particles.

computer time used on the GPU up to $t=\tau_{char}$ is increased from around 4 s to 8 s, as the number of n_s is doubled from 1000 to 2000.

4.4. Case study: particle diffusion, coagulation and nucleation in 1D space

In the following case study we demonstrate that the developed method can be readily combined with a transport model, for the sake of simplicity diffusional transport in a 1-dimensional (1D) space is chosen. This procedure is governed by

$$\frac{\partial n_p}{\partial t}+\nabla(\vec{U}\,n_p)-\nabla(D_f\nabla n_p)=S \tag{13}$$

where particle number concentration $n_p=n_p(v,x,t)$ is now a function of not only particle size and time, but also position, $\vec{U}$ is the external flow velocity and is set to 0 in this case study (pure diffusion), D_f is the particle diffusion coefficient, $D_f=\kappa_B TC_c/3\pi\mu d$ with C_c being the Cunningham correction factor and μ being the dynamic viscosity, S is the continuous nucleation term. The first term of Eq. (13), $\partial n_p/\partial t$, is given by Eq. (1).

A 1D domain of length 0.25 m has been evenly divided into $n_c=1001$ cells. The number of particles in each cell is set to $n_s=1000$. Hence it has up to 10^6 particles to be handled in the simulation. Particles with diameter of 1 nm and number concentration of $N_0=10^{17}\ \mathrm{m}^{-1}$ are continuously produced with a nucleation rate of $2\times10^{22}\ \mathrm{m}^{-1}\ \mathrm{s}^{-1}$ in the central cell (starting cell) of the domain. The temperature is 300 K and the time step is 0.02 s.

Particle diffusion in the computational domain has been simulated as a random walk. Both diffusion and coagulation are put in the same framework and implemented on the GPU after parallelization. Figs. 11 and 12 show the evolution of particle number concentration and size as a function of position, at different times, i.e., $t=5$ s and 30 s. Shown are the mean results from 10 simulation runs with different random numbers.

Fig. 11 shows the increase in particle number concentration due to diffusion. In Fig. 12 it is shown that the particle diameter has two symmetrical peaks in its distribution and the largest particle size does not occur at the center of the domain where the highest number concentration is found. The latter is due to the presence of the nucleation term S in Eq. (13), which keeps adding fresh particles to the central cell. As a result, particles in the vicinity of the center cell have less chance to grow larger; at the same time, particles that are a bit far away from the central cell become the largest, thus leading to the two size peaks. It can also be noted that

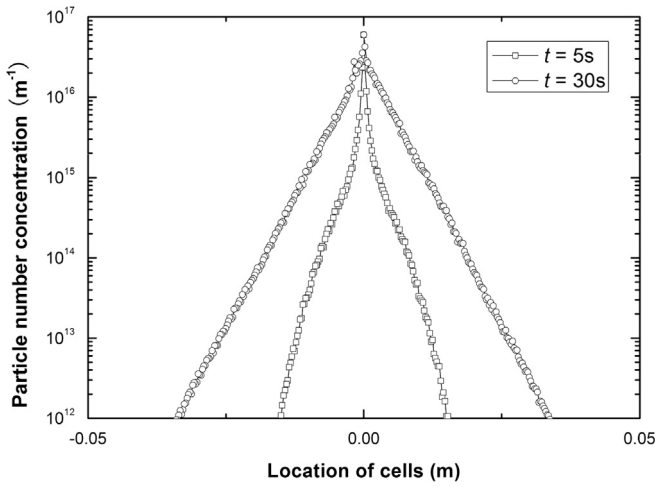

Fig. 11. Case study of simultaneous diffusion, nucleation and coagulation in 1D: particle number concentration as function of position at different times.

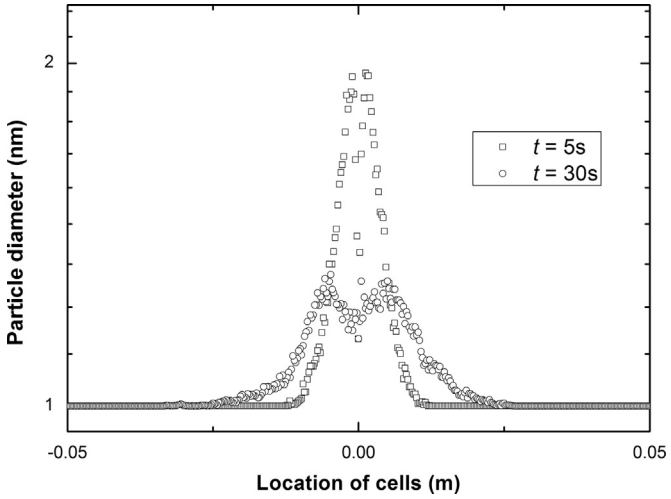

Fig. 12. Case study of simultaneous diffusion, nucleation and coagulation in 1D: particle diameter as function of position at different times.

the presence of diffusion and nucleation has retarded particle coagulation. To see this point, with the foregoing parameters particle size will increase from initial diameter of 1 nm to around 12.5 nm within 30 s in a zero-dimensional system without diffusion and nucleation; however, it reaches only 2 nm in the presence of these two effects.

In this example we clearly demonstrate that not only the mechanism of particle coagulation, but particle transport as well (as in above example) can effectively be parallelized. The GPU-based simulation takes only 950 s on a PC equipped with a model GTX 285 NVIDIA GPU. Also, procedures for describing intra-particle mechanisms sintering, surface growth, etc., can be very efficiently parallelized on the GPU because they do not require particle interactions. Mechanisms such as nucleation and breakage which do increase the number of simulation particles will be more challenging and require an active control of the number of simulation particles.

5. Conclusions

A GPU-based parallel MC algorithm for particle coagulation based on the AR scheme has been developed, aiming at decreasing the computing time. The proposed parallel MC algorithm takes full

advantage of the intrinsic parallel property of the AR scheme, that is, multiple AR attempts can be done independently and simultaneously. At the same time, the coagulation coefficients calculated to that purpose are used to obtain an estimation of the mean coagulation rate. In addition, a new method for quickly estimating the maximum coagulation rate on the basis of the mean coagulation rate has been proposed. For the free-molecular regime an efficient choice is $\beta'_{\max} = 40\overline{\beta'}$. The parallel AR algorithm was first benchmarked with a sectional method to check its computational accuracy. The best results in view of accuracy and computational efficiency was used for a moderate number of parallel threads, as for the maximum number of threads the accuracy was not significantly better. Further, the speedup of the accelerated parallel AR method on the GPU was investigated. It was found that the GPU-accelerated AR scheme is especially efficient for systems containing a large number of cells, for 1000 cells a speedup over 100 was reached whereas for a single cell the speedup is only 3. It makes the method therefore especially suited for cases where MC-based particle dynamics is to be coupled to CFD. It is shown in a case study that the method can indeed be very effectively coupled to a transport model, when both coagulation, nucleation and transport are handled on the GPU level.

Acknowledgments

The Research Project no. 330 Z of the Research Association Institut für Energie- und Umwelttechnik e. V. (IUTA), has been funded by the AiF within the agenda for the promotion of industrial cooperative research and development (IGF) by the German Federal Ministry of Economics and Technology based on a decision of the German Bundestag and the European Union's Seventh Framework Program (EU FP7) under Grant Agreement no. 280765 (BUONAPART-E). We acknowledge Prof. H. Zhao for helpful discussions and his effort in coding the sectional method.

References

Brown, D.P., Kauppinen, E.I., Jokiniemi, J.K., Rubin, S.G., Biswas, P., 2006. A method of moments based CFD model for polydisperse aerosol flows with strong interphase mass and heat transfer. Computer & Fluids 35, 762–780.
Efendiev, Y., Zachariah, M.R., 2002. Hybrid Monte-Carlo method for simulation of two-component aerosol coagulation and phase segregation. Journal of Colloid and Interface Science 249, 30–43.
Friedlander, S.K., 1997. Smoke, Dust and Haze: Fundamentals of Aerosol Behavior. Wiley, New York.
Garcia, A.L., 1987. A Monte Carlo simulation of coagulation. Physica 143A, 535–546.
Jenong, J., Choi, M., 2001. A sectional method for the analysis of growth of polydisperse non-spherical particles undergoing coagulation and coalescence. Journal of Aerosol Science 32, 565–582.
Kodas, T.T., 1999. Aerosol Processing of Materials. Wiley-VCH, New York.
Kruis, F.E., Maisels, A., Fissan, H., 2000. Direct simulation Monte Carlo method for particle coagulation and aggregation. AIChE Journal 46, 1735–1742.
Kruis, F.E., Wei, J.M., Zwaag, T.V.D., Haep, S., 2012. 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. Chemical Engineering Science 70, 109–120.
Lai, S.K., Friedlander, S.K., Pich, J., Hidy, G.M., 1972. The self-preserving size distribution for Brownian coagulation in the free-molecular regime. Journal of Colloid and Interface Science 39, 395–405.
Landau, D., Binder, P.K., 2009. A Guide to Monte-Carlo Simulation in Statistical Physics. Cambridge University Press, Cambridge.
Landgrebe, J.D., Pratsinis, S.E., 1990. A discrete-sectional model for powder production by gas phase chemical reaction and aerosol coagulation in the free-molecular regime. Journal of Colloid and Interface Science 139, 63–86.
Lee, K., Matsoukas, T., 2000. Simultaneous coagulation and break-up using constant-N Monte Carlo. Powder Technology 110, 82–89.
Liffman, K., 1992. A direct simulation Monte-Carlo method for cluster coagulation. Journal of Computational Physics 100, 116–127.
Lu, S., 1994. Collision integrals of discrete-sectional model in simulating powder production. AIChE Journal 40, 1761–1764.
Maisels, A., Kruis, F.E., Fissan, H., 2004. Direct Monte Carlo simulation for simultaneous nucleation, coagulation, and surface growth in dispersed systems. Chemical Engineering Science 59, 2231–2239.
Mitrakos, D., Hinis, E., Housiadas, C., 2007. Sectional modeling of aerosol dynamics in multi-dimensional flows. Aerosol Science and Technology 41, 1076–1088.
Molnar, F., Szakaly, T., Meszaros, R., Lagzi, I., 2010. Air pollution modelling using a Graphics Processing Unit with CUDA. Computer Physics Communications 181, 105–112.
Nanbu, K., 1980. Direct simulation scheme derived from the Boltzmann equation. Journal of the Physical Society 49, 2042–2049.
NVIDIA, 2007. Compute Unified Device Architecture Programming Guide, version 2.0.
NVIDIA, 2008. NVIDIA GeForce GTX 285 Specifications.
Park, S.H., Lee, K.W., Shimada, M., Okuyama, K., 2001. Alternative analytical solution to condensational growth of polydisperse aerosols in the continuum regime. Journal of Aerosol Science 32, 187–197.
Ramkrishna, D., 2000. Population Balance: Theory and Applications to Particulate Systems in Engineering. Academic Press, San Diego.
Shah, B.H., Ramkrishna, D., Borwanker, J.D., 1977. Simulation of particulate systems using the concept of the interval of quiescence. AIChE Journal 23, 897–904.
Smith, M., Matsoukas, T., 1998. Constant-number Monte Carlo simulation of population balances. Chemical Engineering Science 53, 1777–1786.
Terry, D.A., McGraw, R., Rangel, R.H., 2001. Method of moments for a laminar flow aerosol reactor model. Aerosol Science and Technology 34, 353–362.
Tolke, J., Krafczyk, M., 2008. TeraFLOP computing on a desktop pc with GPUs for 3D CFD. International Journal of Computational Fluid Dynamics 22, 443–456.
Wei, J.M., Kruis, F.E., 2013. GPU-accelerated Monte Carlo simulation of particle coagulation based on the inverse method. Journal of Computational Physics 249, 67–79.
Wu, C.Y., Biswas, P., 1998. Study of numerical diffusion in a discrete-sectional model and its application to aerosol dynamics simulation. Aerosol Science and Technology 29, 359–378.
Yamamoto, M., 2004. A moment method of an extended log-normal size distribution application to Brownian aerosol coagulation. Journal of Aerosol Science 19, 41–49.
Yu, M.Z., Lin, J.Z., Chan, T., 2008. A new moment method for solving the coagulation equation for particles in Brownian motion. Aerosol Science and Technology 43, 781–793.
Zhao, H., Kruis, F.E., 2011. Monte Carlo simulation for aggregative mixing of nanoparticles in two-component systems. Industrial & Engineering Chemistry Research 50, 10652–10664.
Zhao, H., Kruis, F.E., Zheng, C., 2010. A differentially weighted Monte Carlo method for two-component coagulation. Journal of Computational Physics 229, 6931.
Zhao, H., Zheng, C., 2009a. Correcting the multi-Monte Carlo method for particle coagulation. Powder Technology 193, 120–123.
Zhao, H., Zheng, C., 2009b. A new event-driven constant volume method for solution of the time evolution of particle size distribution. Journal of Computational Physics 228, 1412–1428.
Zhao, H., Zheng, C., 2011. Two-component Brownian coagulation: Monte Carlo simulation and process characterization. Particuology 9, 414–423.
Zhao, H., Zheng, C., 2013. A population balance-Monte Carlo method for particle coagulation in spatially inhomogenous systems. Computer & Fluids 71, 196–207.
Zhao, H., Zheng, C., Xu, M.H., 2005a. Multi-Monte Carlo approach for general dynamic equation considering simultaneous particle coagulation and breakage. Powder Technology 154, 164–178.
Zhao, H., Zheng, C., Xu, M.H., 2005b. Multi-Monte Carlo method for coagulation and condensation/evaporation in dispersed systems. Journal of Colloid and Interface Science 286, 195–208.
Zhao, H.B., Kruis, F.E., 2008. Reducing statistical noise and extending the size spectrum by applying weighted simulation particles in Monte Carlo simulation of coagulation. Aerosol Science and Technology 43, 781–793.