Parallel High-Resolution Finite Volume Simulation of Particulate Processes

Rudiyanto Gunawan
Dept. of Chemical and Biomolecular Engineering, National University of Singapore, Singapore 117576

Irene Fusman and Richard D. Braatz
Dept. of Chemical and Biomolecular Engineering, University of Illinois, Urbana, IL 61801

DOI 10.1002/aic.11484
Published online April 18, 2008 in Wiley InterScience (www.interscience.wiley.com).

Population balance models (PBMs) are widely used to describe the dynamics of particulate systems. The simulation of PBMs becomes challenging with increasing numbers of dimensions and integral terms associated with phenomena, such as coalescence and breakage, which has motivated efforts to develop more computationally efficient numerical algorithms. This article presents high-resolution finite-volume methods for the parallel simulation of PBMs that scale well to ~100 processors on a linux cluster. The high-resolution methods provide second-order accuracy with an accurate tracking of sharp fronts. The ParticleSolver software implementing these methods is verified by application to PBMs for (1) aerosol coagulation and condensation, (2) the formation of gold nanoparticles by nucleation, disproportionation, and coagulation, and (3) the nucleation and growth of inorganic crystals with a time-varying shape distribution.
 AIChE J, 54: 1449–1458, 2008
Keywords: population balance equations, aggregation, coagulation, crystallization, distributed parameter systems, numerical analysis

Introduction

Particulate processes are industrially prevalent and include melt and solution crystallization,[1,2] bioreactors of microbial cell culture,[3] and emulsion and suspension polymerization.[4,5] The properties of the end products of these processes typically depend on the distribution of the particulate characteristics.[6] Population balance models (PBMs) describe the time trajectory of the particulate distribution by way of a hyperbolic* (integro-)partial differential equation whose analytical solution is intractable except for simple cases. For these reasons, considerable efforts have been put forth to develop numerical methods for simulating PBMs.[7,8]

A population balance equation (PBE) has the form of

$$\frac{\partial f(x,t)}{\partial t} + \frac{\partial (G(x,t) f(x,t))}{\partial x} = h(x,t,f(x,t)) \qquad (1)$$

where $f(x,t)$ is the distribution function (also called the population density), t is time, x is an internal coordinate (e.g., size, mass, or age), $G(x,t)$ is the (state) velocity at time t, and $h(x,t,f)$ is the generation-disappearance rate of particles, i.e., source and sink terms. For example, in crystallization, x is the size of crystals measured by length or volume, $f(x,t)$ is the crystal size distribution (CSD), $G(x,t)$ is the growth rate, and $h(x,t,f)$ consolidates all processes involving crystal birth and death. More generally, birth and death of particles can arise from a wide range of processes, including nucleation, aggregation/coagulation, and breakage/attrition. The rates of such processes usually depend on some integral functions of the crystal size dis-

Correspondence concerning this article should be addressed to R. Gunawan at chegr@nus.edu.sg.

*A PBE can also include terms such as growth dispersion which are modeled by parabolic term(s), but such PBEs are much easier to solve numerically than the hyperbolic case. So the focus here as in the vast majority of past articles in the literature is on the hyperbolic case. Nearly all methods for solving the hyperbolic PBE, including those in this article, can be used to solve a PBE with parabolic terms by just lumping the parabolic terms in the right-hand side of the PBE (1).

tribution, and thus the PBE in general is an integro-partial differential equation. In addition, if the distribution function depends on multiple internal coordinates or is spatially varying, then the distribution of particles is described by a multidimensional PBE in which the second term in (1) is replaced by a summation over all of the elements of the system coordinates.

Since several reviews of the various methods for simulating PBEs are available including their pros and cons,[7–10] the focus here will only be on the most relevant methods. Finite-difference methods (FDMs) were popular in the 1990s because of their simplicity and ease of extension to multidimensional simulations. However, these methods perform poorly in tracking particle distributions with sharp fronts (discontinuities), where first-order methods give numerical diffusion and second-order methods introduce numerical dispersion that results in nonphysical oscillations. More recently, high-resolution finite-volume (HRFV) methods developed primarily to solve the hyperbolic equations that arise in astrophysics and gas dynamics[11] were adapted for the solution of multidimensional PBEs with size-independent growth rates.[12] These methods avoid numerical dispersion while suppressing numerical diffusion by adding a weighted anti-diffusion term, where the weight is a function of the local smoothness of the distribution. Subsequent articles developed HRFV algorithms for particulate processes with size-dependent growth rates,[13,14] combinations of growth, nucleation and aggregation,[15] and spatially varying distributions,[16,17] as well as incorporating adaptive meshing for further improvements in computational efficiency.[18]

There has been much recent interest in coupling the solution of PBEs with computational fluid dynamics to assess the effects of nonideal mixing on the particle size distribution.[17,19–22] This involves solving a spatially varying PBE, which typically has four or more coordinates (three spatial coordinates and one or more internal coordinates). Typically such simulations require on the order of a day. Further, there is interest in incorporating PBEs for complex particulate processes within optimization algorithms to enable the design and control of particle size,[23–25] in which case it would be desired to simulate a PBE within seconds to minutes. This motivates the development of numerical algorithms for solving PBEs with orders-of-magnitude speedup.

This article addresses this need through the parallel implementation of HRFV algorithms. The numerical algorithms are implemented as Fortran subroutines which provide various HRFV methods for solving general one- and two-dimensional PBEs. The subroutines also provide the basis for simulating higher dimensional systems through application of the dimensional (Godunov's) splitting method. The numerical performance of the parallel implementations is demonstrated for aerosol coagulation and condensation, gold nanoparticle formation, and the crystallization of potassium dihydrogen phosphate (KDP) crystals with a time-varying shape distribution. The results were compared to that obtained using the standard algorithms for solving PBEs-the upwind and the Lax–Wendroff methods. The serial and parallel Fortran subroutines are available in the ParticleSolver software.[26]

High-Resolution Algorithms

HRFV methods were originally developed for simulating hyperbolic systems of conservation equations, which typically arise in modeling the motion of shock waves in fluids. There are two equivalent forms of the conservation law[11]: the differential form

$$\frac{\partial u(z,t)}{\partial t} + \frac{\partial}{\partial z} F(z,t,u) = 0, \tag{2}$$

and the integral form

$$\frac{d}{dt} \int_{z_1}^{z_2} u(z,t)\, dz = F(z_1,t,u(z_1,t)) - F(z_2,t,u(z_2,t)) \tag{3}$$

where z and u denote the spatial and state variables, respectively, and $F(z,t,u)$ is called the flux function. The integral form describes a mass or quantity balance, where the change in the total quantity equals to the sum of the fluxes at the boundaries z_1 and z_2. Integrating both sides of the PBE (1) with respect to x results in the corresponding integral form

$$\frac{d}{dt} \int_{x_1}^{x_1} f(x,t)\, dx = G(x_1,t) f(x_1,t) - G(x_2,t) f(x_2,t) + \int_{x_1}^{x_2} h(x,t,f)\, dx \tag{4}$$

which is similar to (3) except for the last term. The integral form of the PBE (4) describes the particle count/number balance, where the number of particles of states between x_1 and x_2 changes due to the velocity in the state space (such as crystal growth) and the particle generation-disappearance processes. Although HRFV methods for (3) do not directly apply to the PBE (4) due to the nonhomogenous integral term associated with $h(x,t,f)$, the extra term can be handled by employing fractional-step methods such as Godunov's or Strang's splitting (as summarized later in this article).

Finite-volume methods (FVMs) are based on the integral form of the conservation law, with various FVMs resulting from using different approximations to the flux function $F(z,t,u)$. In contrast, FDMs are designed to solve the differential form by approximating the derivatives with finite-difference approximations. The interpretations of the simulation results differ between the two classes of methods. If the state space is divided into a finite number of intervals or grid cells centered on x_n, then the value f_n^m in FVMs represents the averaged $f(x,t)$ over the nth grid cell:

$$f_n^m \approx \frac{1}{\Delta x} \int_{x_N - \frac{1}{2}\Delta x}^{x_N + \frac{1}{2}\Delta x} f(x,t_m)\, dx, \tag{5}$$

where Δx is the size of the cell and t_m is the m-th time point $(t_m) = m\Delta t$. In contrast, the variable f_n^m in FDMs typically describes the value of $f(x,t)$ at the state x_n and time t_m, i.e., $f_n^m = f(x_n,t_m)$. That is, FVMs return the time evolution of the quantity averaged over subregions in the state space, while FDMs give the time evolution of the quantity at specific points in the state space. While this manuscript uses a uniform cell size, a nonuniform mesh can be used to further improve computational efficiency for PBEs with orders of magnitude differences in the size range.[18]

The (explicit) finite-volume time-marching algorithm for a homogeneous PBE ($h(x,t,f) = 0$) can be derived from (3) to give

$$f_n^{m+1} = f_n^m - \frac{\Delta t}{\Delta x}(F_{n+1/2}^m - F_{n-1/2}^m), \tag{6}$$

where the term $F_{n-1/2}^m(F_{n+1/2}^m)$ describes the time-average flux of particles through the left (right) boundary of the cell. Since the exact value for the flux function is generally difficult to obtain, its approximation is used in the finite-volume time-marching algorithm. High-resolution methods are hybrids between a first-order flux approximation that behaves well around discontinuities and a higher order flux function that gives good accuracy in the smooth region of the solution. In general, high-resolution methods use a corrective (antidiffusive) term to improve the accuracy of a first-order method. For example, such a correction term can be based on a modification of the anti-diffusive flux function used in the Lax–Wendroff method. The modification is in the use of a flux limiter function to control the degree of correction as needed, depending on the smoothness of the solution. That is, through weighting functions, the corrective actions should be minimal around discontinuities as to preserve the benefits of the first-order flux function in tracking sharp fronts, and should be maximal in the smooth region of the solutions to attain the most accurate solutions and to attenuate numerical diffusion. Numerous high-resolution algorithms can arise not only from selecting different combinations of first-order flux function and anti-diffusive term, but also from choosing various flux limiter functions. The formulation of various HRFV algorithms for PBEs have previously been reported[13,14] and a selection of flux limiter functions is available.[11]

The standard high-resolution algorithms are applicable to homogeneous one-dimensional PBEs. Since PBMs are generally multidimensional and nonhomogenous, a dimensional splitting technique can be used to decompose the multidimensional PBE into multiple one-dimensional problems, in which any high-resolution (or 1D-time-marching) algorithm can be sequentially applied.[12,13] This decomposition, which is also known as Godunov's splitting, belongs to the class of fractional step methods. The nonhomogenous term can also be treated in a similar manner using Godunov's splitting. At each time step, the time-marching algorithm consists of a high-resolution update along each dimension, followed by the addition of the nonhomogenous contribution.

Particulate Aggregation

Aggregation models are usually written in terms of particle volume, since the total volume of particles is conserved with two particles collide to from a larger particle. The rates of appearance of larger particles and disappearance of smaller particles are given by

$$B(v,t) = \frac{1}{2}\int_0^v \beta(v-\varepsilon,\varepsilon)f(v-\varepsilon,t)f(\varepsilon,t)\,d\varepsilon \tag{7}$$

$$D(v,t) = f(v,t)\int_0^\infty \beta(v,\varepsilon)f(\varepsilon,t)\,d\varepsilon, \tag{8}$$

where $B(v,t)$ and $D(v,t)$ denote the birth and death rates of particles of volume v, respectively, and $\beta(v,\varepsilon)$ is the coalescence kernel that describes the collision frequency between particles of volumes v and ε creating a particle of volume $v + \varepsilon$.[27–30] In crystallization processes, the crystal size is often represented by its characteristic length. Rewriting the aggregation functions in (7) and (8) in terms of the crystal length gives[29,31,32]

$$B(L,t) = \frac{L^2}{2}\int_0^L \frac{\beta((L^3-\lambda^3)^{1/3},\lambda)f((L^3-\lambda^3)^{1/3},t)f(\lambda,t)}{(L^3-\lambda^3)^{2/3}}\,d\lambda \tag{9}$$

$$D(L,t) = f(L,t)\int_0^\infty \beta(L,\lambda)f(\lambda,t)\,d\lambda, \tag{10}$$

where $B(L,t)$ and $D(L,t)$ are now the birth and death rates of crystals of size L due to aggregation, respectively, and $\beta(L,\lambda)$ is the coalescence kernel for crystals of length L and λ.

These birth and death rates appear as source-sink terms within $h(L,t,f)$. The coalescence kernel can assume several forms. A size-independent aggregation kernel is simply a constant β_0, while size-dependent kernels are given by.[27,33,34]

$$\beta(v,\varepsilon) = \beta_1(v+\varepsilon) \tag{11}$$

when volume is the internal coordinate, or

$$\beta(L,\lambda) = \beta_1(L^3+\lambda^3) \tag{12}$$

when length is the internal coordinate. The total cost of computing the aggregation birth and death rates as in (7) and (8) at each time step using the standard second-order Simpson's rule[35] is $O(N^2)$, where N is the total number of grid cells. The evaluation of the birth term also scales linearly with the crystal size v or L, due to the integration upper limit.

Since the cost of a high-resolution update is $O(N)$ at each time step, the integral evaluations dominate the simulation time in aggregating systems. To reduce the computational cost, the ParticleSolver software computes the birth rate according to the equivalent formulation:

$$B(v,t) = \int_0^{v/2} \beta(v-\varepsilon,\varepsilon)f(v-\varepsilon,t)f(\varepsilon,t)\,d\varepsilon. \tag{13}$$

In addition, if the coalescence kernel is a polynomial function of the internal coordinate, then the death rate can be written as functions of the moments of the distribution. By computing the moments only once at each time step, the total cost of evaluating the aggregation death rate reduces to $O(N)$. However, these reductions still do not lead to a change in the order of the computational cost, which motivates the parallel implementation of the high-resolution method.

Parallel High-Resolution Algorithms

As with any parallel programming implementation, the simulation speedup depends on the computational load balancing across the processors. For homogeneous PBEs, the

computational load for each grid cell is the same, and thus the computation distributes evenly among the processors. Nonhomogenous source-sink terms in PBE are typically integral functions of the distribution (e.g., nucleation, aggregation), and the evaluation of these integral functions can lead to varying computation load across the grid cells. In the case of aggregation terms as described in the previous section, the load increases linearly with the grid cell index. In our implementation, the computation is distributed such that the load is incrementally higher with increasing processor rank in the first half of the size range, and is incrementally lower with increasing rank in the other half, as explained in the pseudo-code later.

The second consideration is the management of communication among the processors. A master-slave parallel paradigm is taken such that the master processor is responsible in collecting the result of one time-marching simulation from each slave processor and broadcasting the complete set of f_n^m back to the slave processors for the next time update. If there exist other ordinary differential equations, such as those associated with mass balances, the time-marching calculation will be done by the master processor and then broadcasted to the slaves. The communication from the slaves to the master is split into two as shown in the pseudo-code below, to avoid overloading the network bandwidth.

The pseudo-code of the HRFV time-marching algorithm in the parallel implementation for N_S slave processors and N total grid cells run for each time step is

```
if (in the Master processor)
    receive from Slaves
    update f_n^m
    perform time-marching algorithm for other ODEs

(e.g., mass balances)
    update other variables (e.g., concentrations)
else if (in a Slave processor of rank r)
    for grid cell index = r to N/2 by N_s
      {perform HRFV time − marching of f_n^m in the range}
    send updated f_n^m to Master
    for grid cell Index = N − r to N/2 by N_s
      {perform HRFV time-marching of f_n^m in the range}
    send updated f_n^m to Master
end if
    broadcast and receive f_n^m and other variables from
    Master to Slaves
```

For aggregating systems, the first for-loop in the pseudo-code allocates linearly increasing computation load to higher ranked slave processors. Conversely, the second for-loop assigns increasing load to lower ranked slaves. Such balancing provides nearly equal computation load among the processors for constant and linearly increasing or decreasing computational load. Since the higher ranked slave will take increasingly longer simulation time in the first for-loop, the communications from the slaves to the master will happen in the rank-ascending sequence. Conversely, the communications in the second for-loop will happen in the rank-descending sequence due to the reversal in the load assignment. All these were necessary to avoid overloading the network bandwidth, where all slaves simultaneously send the updated f_n^m to the master. Compared to the standard parallel implementation [e.g., using simple OpenMP on the time-marching algorithm in (6)], this combination of load balancing and communication bandwidth management leads to nearly perfect speedup with the number of processors.

Numerical Examples

Numerical examples demonstrate the accuracy and speed of the HRFV methods in simulating various PBEs arising in the modeling of aggregating systems and batch crystallization. The first example is used to verify that the numerical algorithms accurately and efficiently simulate a simple model for aerosol coagulation and condensation, which has an analytical solution. This is followed by a more complicated PBE for gold nanoparticle formation which includes nucleation and Brownian aggregation, and a crystallization with time-varying shape distribution used to compare HRFV methods.

Example 1: A simple model for aerosol coagulation and condensation

For particles that simultaneously grow and aggregate at constant rates G_0 and β_0 with the initial distribution:

$$f_0(v) = \frac{N_0}{v_0} e^{v/v_0}, \tag{14}$$

the analytical solution to the PBE is[34]:

$$f(v,t) = \frac{M_0^2/M_1}{1 - 2\Lambda v_0\left(\frac{N_0 - M_0}{M_1}\right)} \exp\left[-\frac{\frac{M_0}{M_1}\left(v - 2\Lambda v_0\left(\frac{N_0}{M_0} - 1\right)\right)}{1 - 2\Lambda v_0\left(\frac{N_0 - M_0}{M_1}\right)}\right], \tag{15}$$

where

$$M_0 = \frac{2N_0}{2 + \beta_0 N_0 t},$$
$$M_1 = N_0 v_0\left[1 - \frac{2G_0}{\beta_0 N_0 v_0}\ln\left(\frac{2}{2 + \beta_0 N_0 t}\right)\right], \text{and}$$
$$\Lambda = \frac{G_0}{\beta_0 N_0 v_0}. \tag{16}$$

Figure 1 compares the distributions computed by the high-resolution, upwind, and Lax–Wendroff methods using the simulation parameters in Table 1. The upwind method produced a smeared distribution, whereas the Lax–Wendroff method produced nonphysical oscillations for volumes below the discontinuity at 0.01 μm^3. As expected from theory,[11] the high-resolution method has lower numerical diffusion than the upwind method and lower numerical dispersion than the Lax–Wendroff method. This is consistent with a past study for this PBE.[15]

Using an IBM BladeCenter computing cluster with 112 processors, the parallel HRFV algorithm initially achieved a linear speed-up (Figure 2a). This is consistent with the nearly

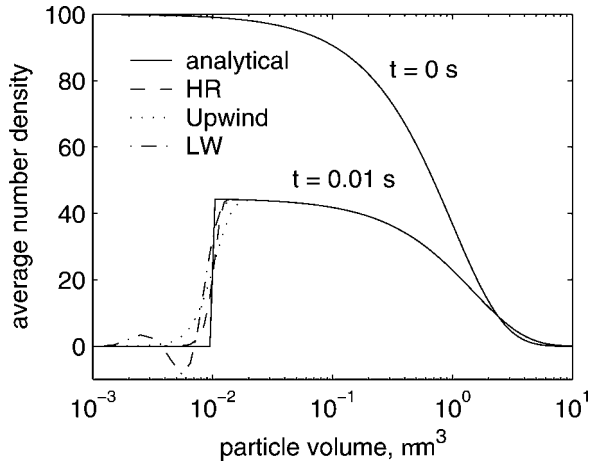

Figure 1. Comparison of the simulated particle distributions for constant growth and aggregation using the high-resolution (HR), upwind, and Lax–Wendroff (LW) methods (Example 1).

The high-resolution profile (HR) is both the serial and the parallel high-resolution numerical distributions with the simulation parameters in Table 1). The HR method used the van Leer limiter.

perfect allocation of integral calculations among the slave processors. The speed-up becomes sub-linear with increasing number of processors (>20 processors) due to the increased communication load, as expected for the distributed memory architecture of the linux cluster. Nevertheless, the parallel code achieves ~90% speed-up efficiency up to 60 processors, and ~80% efficiency to 100 processors. The run-time reduced from 38 min on a single processor to 30 s using 100 processors on the linux cluster, which is certainly fast enough to use in optimization algorithms for parameter estimation or process design. Even better speedup improvements would be obtained using a supercomputer with shared memory, which would reduce the communication overhead.

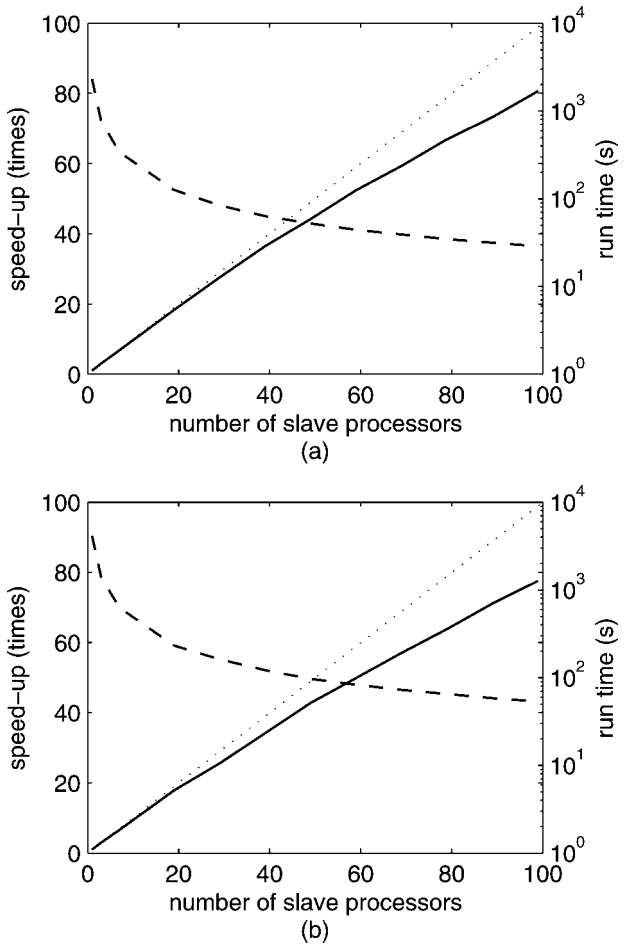

Figure 2. The speed-up of the parallel high-resolution algorithm and the corresponding run time for (a) Example 1 and (b) Example 2, using an IBM BladeCenter high performance computing (HPC) linux cluster with Intel® Dual Xeon Quad-Core 1.6 GHz processors (112 total).

The dotted line represents perfect speed-up.

Example 2: Gold nanoparticles formation by citrate method

Gold nanoparticles can be synthesized through the reduction of tetrachloroauric acid ($HAuCl_4$) by trisodium citrate.[36] The gold particles grow in size mainly due to disproportionation of aurous chloride and coagulation of nanoparticles. A model of this process was previously presented in which the PBE describes the distribution of particle volumes.[37] Here, an alternative PBE is derived to give the distribution of radii. The model describes the gold particle formation and growth processes according to:

$$
\begin{aligned}
&T + C \xrightarrow{k_c} M + S \ \text{(reduction)}\\
&3M \xrightarrow[\text{particle}]{k_h} T + \text{particle mass (disproportionation)}\\
&3M \xrightarrow{k_N, s} \text{nucleus} + T \ \text{(nucleation)}\\
&S \xrightarrow{k_s} D\\
&D + 4T \xrightarrow{k_d} 4M + \text{other products}
\end{aligned}
\tag{17}
$$

where the notation follows the original publication: T, C, M, S, and D denote the auric, citrate, aurous, dicarboxy acetone, and acetone species, respectively. The corresponding PBE with respect to the distribution of particulate radii is given by

Table 1. Simulation Parameters in Example 1

Variable	Description	Parameter Values in Example	Units
Δt	Time step size	10^{-5}	s
Δx	Grid cell size	10^{-3}	μm^3
N_0	Number of crystals	100	dimensionless
v_0	Mean volume of charge	1	μm^3
β_0	Size-independent coalescence kernel	1	1/s
G_0	Side-independent growth constant	1	$\mu m^3/s$

Table 2. Simulation Parameters in Example 2

Variable	Description	Parameter Values in Example	Units
Δt	Time step size	2×10^{-4}	s
Δx	Grid cell size	2×10^{-9}	cm
k_c	Reduction by citrate	1.25×10^{3}	1/(M s)
k_h	Disproportionation	6.5×10^{-5}	L/(cm^2 s)
k_n	Nucleation	1.0×10^{33}	1/(M^5 L s)
k_s	Acetone formation	1	1/s
k_d	Reduction by acetone	4×10^{2}	1/(M s)
ρ	Gold molar density	0.1	mole/cm^3
R_0	Nucleate size	2×10^{-7}	cm
Θ	Temperature	373.15	K
μ	Viscosity at Θ = 374.15 K	2.79×10^{-4}	Pa s

$$\frac{\partial f(r,t)}{\partial t} + \frac{\partial}{\partial r}\left\{\frac{1}{2\pi\rho}k_h[M]f(r,t)\right\} = 2k_n[M]^3[S]^2\delta(r-r_0) - f(r,t)\int_{r_0}^{\infty}\frac{q(r,r')}{W}f(r',t)\,dr' + \frac{1}{2}\int_{r_0}^{r}\frac{q(r-r',r')}{W}f(r-r',t)f(r',t)\,dr' \quad (18)$$

where r and r_0 denote the nanoparticle and nucleus radius, respectively, and ρ is the molar density of gold. The coagulation kernels are given by the Brownian collision frequency

$$q(r,r') = \frac{2k\Theta}{3\mu}\left(\frac{1}{r}+\frac{1}{r'}\right)(r+r') \quad (19)$$

and the coagulation efficiency

$$\log W = \frac{0.0056}{-9[\zeta + 1.5(1-\zeta)]}\log(3[C]_0 + [T]_0) + 27.5, \quad (20)$$

where k is the Boltzmann constant, Θ is the temperature, μ is the viscosity, and the factor ζ is the fraction of area on the nanoparticle that is occupied by gold ions:

$$\zeta = \frac{1}{1 + 0.1\frac{[C]}{[M]+[T]}} \quad (21)$$

In addition to the PBE, the mass balance equations for T, C, M, S, and D are given by

$$\frac{d[T]}{dt} = -k_c[C][T] - k_d[D][T] + k_h[M]\int_{r_0}^{\infty} r^2 f(r,t)\,dr + k_n\rho\left(\frac{4}{3}\pi r_0^3\right)[M]^3[S]^2$$

$$\frac{d[C]}{dt} = -k_c[C][T]$$

$$\frac{d[M]}{dt} = k_c[C][T] + k_d[D][T] - 3k_h[M]\int_{r_0}^{\infty} r^2 f(r,t)\,dr - 3k_n\rho\left(\frac{4}{3}\pi r_0^3\right)[M]^3[S]^2$$

$$\frac{d[S]}{dt} = k_c[C][T] - k_s[S]$$

$$\frac{d[D]}{dt} = k_s[S] - \frac{1}{4}k_d[D][T] \quad (22)$$

with the initial conditions $[T] = [T]_0$, $[C] = [C]_0$, and $[M] = [S] = [D] = 0$. The model parameters are listed in Table 2 according to the previously published values.[37]

Figure 3 compares the predicted average diameter of the gold nanoparticles with experimental data, as a function of the ratio of citric acid to gold concentrations. The simulations were performed until 99% of the gold ions in solution were reduced to nanoparticles. The result is in good agreement with the original publication based on particle volumetric size (see Figure 3 in the article by Kumar et al.[37]). Again, the parallel HRFV implementation provided a good scale-up with ~80% efficiency at 100 slave processors, reducing the simulation time from 70 min on a single processor to less than 1 min, as seen in Figure 2b. This speedup would permit the optimization of process conditions (e.g., $[T]_0$, $[C]_0$, and temperature) to accomplish the desired nanoparticle size distribution, which is not practically feasible on a single PC workstation.

Further, seeded batch simulations were performed with 10-nm seed particles at a concentration of 2×10^{-5} M of gold until 99% of the gold ions in the solution were reduced to form nanoparticles. Two combinations of Δx and Δt were used to compare simulation accuracy. This problem is challenging in its simulation due to significant nucleation coupled with Brownian aggregation, and severe discontinuity caused by the seed distribution, which is a Dirac delta function at 10 nm. As shown in Figure 4, the predicted distribution from the lower resolution simulation misses the sharp peaks of the higher resolution simulation. The predicted average diameters from the two simulations differ only by 0.22% (17.973 nm vs. 18.013 nm).

Example 3: Crystallization with time-varying shape distribution

The purpose of this example is to compare the performance of different HRFV algorithms for simulating a PBE

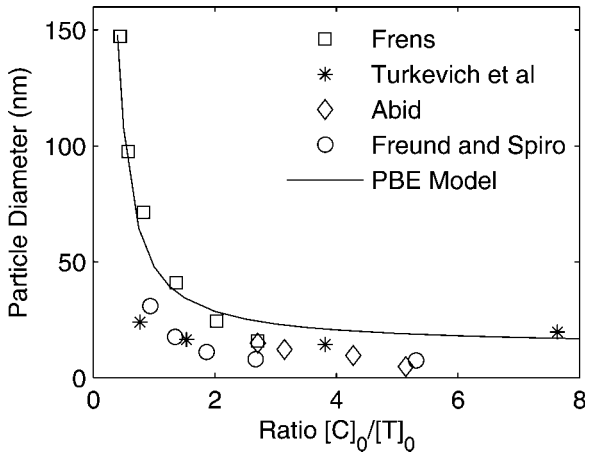

Figure 3. Average diameter of gold nanoparticles formed by the citrate reduction method as predicted by the PBE model, in comparison to values reported in the literature.[36,38–40]

The simulations were performed with $[T]_0 = 3 \times 10^{-4}$ M using the van Leer limiter.

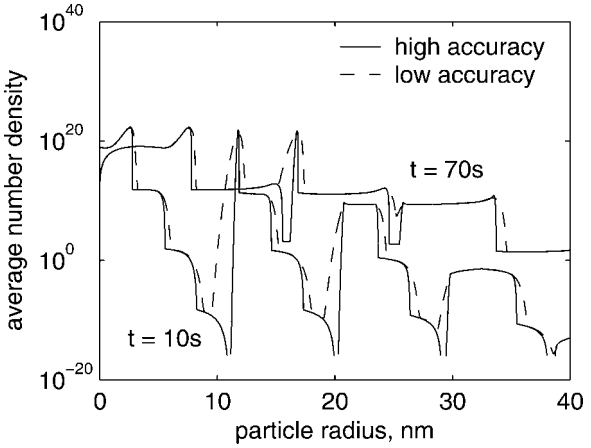

Figure 4. Size distribution of gold nanoparticles by the citrate reduction method with seed particles of 10 nm in size.

The high accuracy simulation used $\Delta x = 0.005$ nm and $\Delta t = 10^{-3}$ s, while the low accuracy simulation used $\Delta x = 0.05$ nm and $\Delta t = 10^{-2}$ s with both simulations using the van Leer limiter. The simulations were performed with $[T]_0 = 3 \times 10^{-4}$ M and $[C]_0 = 1.5 \times 10^{-5}$ M until 99% of the gold in solution has been incorporated into the nanoparticles.

with multiple characteristic dimensions, which characterizes a time-varying shape distribution. In this example, the distribution of KDP crystals in Figure 5 in a batch crystallizer is described by the PBE:

$$\frac{\partial f(L_1,L_2,t)}{\partial t}+\frac{\partial\{G_1(L_1,L_2,c,T)f\}}{\partial L_1}+\frac{\partial\{G_2(L_1,L_2,c,T)f\}}{\partial L_2} = B_0(c,T)\delta(L_1,L_2) \quad (23)$$

where L_1 and L_2 are the characteristic length scales, c is the solute concentration, and T is the solution temperature. The growth rates depend linearly on crystal size and follow a power law functionality with respect to the relative supersaturation:

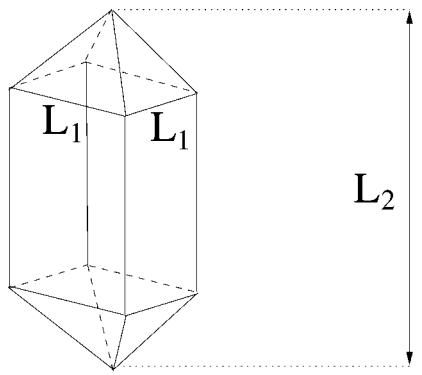

Figure 5. Characteristic lengths of KDP crystals.

Table 3. Kinetic Parameters of KDP Batch Crystallization

Variable	Value	Units
b	2.04	dimensionless
k_b	7.49×10^{-8}	particles/(μm^3 s)
g_1	1.48	dimensionless
k_{g1}	12.1	μm/s
g_2	1.74	dimensionless
k_{g2}	100.75	μm/s

$$G_1(L_1,L_2,t)=k_{g1}\left(\frac{c-c_{\text{sat}}(T)}{c_{\text{sat}}(T)}\right)^{g_1}(0.1+0.06L_1) \quad (24)$$

$$G_2(L_1,L_2,t)=k_{g2}\left(\frac{c-c_{\text{sat}}(T)}{c_{\text{sat}}(T)}\right)^{g_2}(0.1+0.06L_2), \quad (25)$$

where the kinetic parameters g_1, g_2, k_{g1}, and k_{g2} listed in Table 3 are based on experimental data.[41] The nucleation kinetics $B_0(c,T)$ follows a similar power law form

$$B_0(c,T)=k_bV\left(\frac{c-c_{\text{sat}}(T)}{c_{\text{sat}}(T)}\right)^{b}, \quad (26)$$

where V is the total volume of crystals in the system and the kinetics k_b and b are in Table 3. The solution temperature is gradually cooled according to:

$$T(t)=32-4(1-e^{-t/310})[^{\circ}\text{C}] \quad (27)$$

which lowers the saturated solute concentration[42]

$$c_{\text{sat}}(T)=9.3027\times10^{-5}T^2-9.7629\times10^{-5}\,T+0.2087 \quad [(\text{g solute})/(\text{g water})] \quad (28)$$

As solute molecules move from solution into crystals, the solute concentration decreases in time according to the mass balance:

$$\frac{dc}{dt}=-\rho_c\int_0^{\infty}\int_0^{\infty}f(L_1,L_2,t)(2G_1(L_1L_2-L_1^2)+G_2L_1^2)\,dL_1\,dL_2. \quad (29)$$

Table 4. Numerical Accuracy of Numerical Algorithms in Example 3

Methods	Average Absolute Error
Upwind	5.2×10^{-4}
Lax–Wendroff	5.7×10^{-4}
High-Resolution Minmod	3.0×10^{-4}
High-Resolution Superbee	6.9×10^{-5}
High-Resolution MC	1.7×10^{-4}
High-Resolution van Leer	2.1×10^{-4}

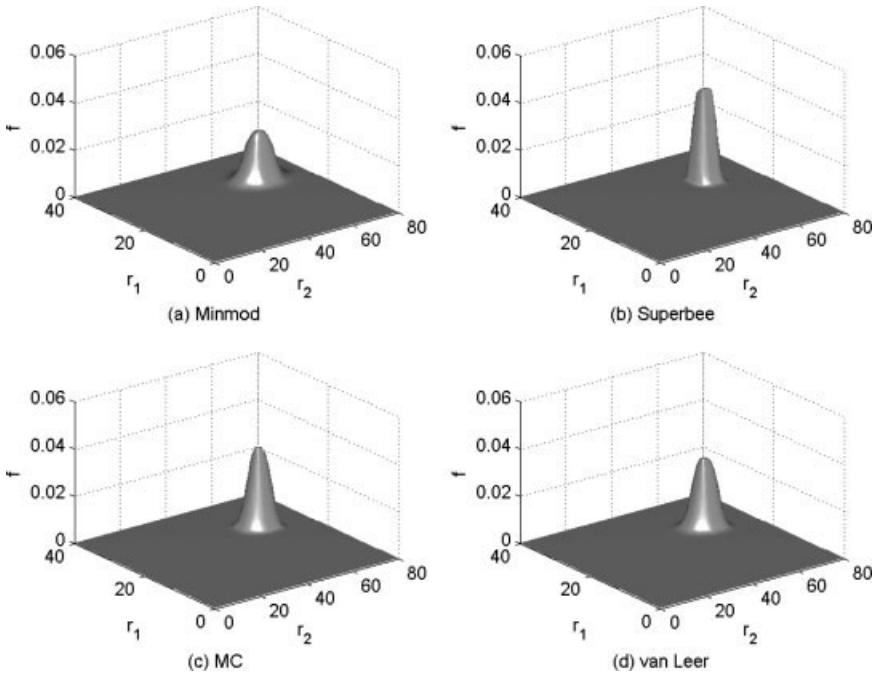

Figure 6. Comparison of the crystal size distributions from the high-resolution (a) Minmod, (b) Superbee, (c) MC, and (d) van Leer methods using moderate grid sizes $\Delta L_1 = \Delta L_2 = 0.5$ μm at $t = 80$ s (Example 3).

The seed (initial) CSD is

$$f(L_1, L_2, 0) = \begin{cases} -3.48 \times 10^{-4}(L_1^2 + L_2^2) + 0.136(L_1 + L_2) - 26.6 & \text{for } 18.05\ \mu\text{m} \le L_1, L_2 \le 21.05\ \mu\text{m} \\ 0 & \text{elsewhere} \end{cases} \tag{30}$$

The PBE has no known analytical solution, so the numerical accuracy was measured against a simulated CSD with much smaller grid cells ($\Delta L_1 = \Delta L_2 = 0.05$ μm). Table 4 reports the average absolute errors in the simulated CSDs for moderate grid sizes ($\Delta L_1 = \Delta L_2 = 0.5$ μm) for the upwind, Lax–Wendroff and four HRFV methods, and Figure 6 shows the CSDs for the HRFV methods. All of the HRFV methods gave higher accuracy than the upwind and Lax–Wendroff methods with the Superbee flux limiter providing the best accuracy (Table 4). The upwind method gave much higher numerical diffusion than the other methods, whereas the Lax–Wendroff method gave good accuracy near the peak of the CSD, but had numerical dispersion with nonphysical two-dimensional oscillations for smaller length scales (plots not shown). The Superbee HRFV method captured the shape the best on average, whereas the MC and van Leer HRFV methods captured the roundness of the peak more effectively than the Superbee method (Figure 6). While the van Leer limiter has performed very well in many past applications of HRFV methods,[11,12] this application indicates that other HRFV methods can give higher accuracy for some PBMs, which motivates the inclusion of these alternative HRFV methods into the ParticleSolver software.

Conclusion

Serial and parallel implementations of HRFV methods for simulating general PBEs were presented. These methods provide second-order accuracy in the smooth region of the solution and accurate sharp-front tracking. For all examples in this article as well as additional examples included with the ParticleSolver software, all of the HRFV methods gave higher accuracy than the upwind and Lax–Wendroff methods. While the van Leer limiter has performed well in many past applications of HRFV methods,[10,11] application to a PBM for a crystallization with time-varying shape distribution indicated that other HRFV methods can give higher accuracy for some PBMs. For PBMs of aerosol coagulation and condensation and the formation of gold nanoparticles by nucleation, disproportionation, and coagulation, the parallel high-resolution subroutine provided nearly linear speedup which enables the direct incorporation of PBEs into numerical algorithms that require iterative model calculation such as in most algorithms for parameter estimation and process design. The ParticleSolver subroutines are available for use as a stand-alone code or for incorporation into more complicated process simulations and optimizations.

Literature Cited

1. Hulburt HM, Katz S. Some problems in particle technology. *Chem Eng Sci*. 1964;19:555–574.
2. Randolph A, Larson MA. *Theory of Particulate Processes*, 2nd ed. San Diego: Academic Press, 1988.
3. Daoutidis P, Henson MA. Dynamics and control of cell populations in continuous bioreactors. *AIChE Symp Ser*. 2002;326:274–289.
4. Wang Y, Doyle FJ. Reachability of particle size distribution in semibatch emulsion polymerization. *AIChE J*. 2004;50:3049–3059.
5. Yuan HG, Kalfas G, Ray WH. Suspension polymerization. *Polymer Rev*. 1991;31:215–299.
6. Teipel U. Particle technology: design of particulate products and dispersed systems. *Chem Eng Tech*. 2004;27:751–756.
7. Ramkrishna D. *Population Balances: Theory and Applications to Particulate Systems in Engineering*. San Diego: Academic Press, 2000.
8. Rawlings JB, Miller SM, Witkowski WR. Model identification and control of solution crystallization processes: a review. *Ind Eng Chem Res*. 1993;32:1275–1296.
9. Costa CBB, Maciel MRW, Filho RM. Considerations on the crystallization modeling: population balance solution. *Comp Chem Eng*. 2007;31:206–218.
10. Braatz RD. Advanced control of crystallization processes. *Annu Rev Contr*. 2002;26:87–99.
11. LeVeque RJ. *Finite Volume Methods for Hyperbolic Problems*. New York: Cambridge University Press, 2002.
12. Ma DL, Tafti DK, Braatz RD. High resolution simulation of multidimensional crystal growth. *Ind Eng Chem Res*. 2002;41:6217–6223.
13. Gunawan R, Fusman I, Braatz R. High resolution finite volume algorithms for simulating multidimensional population balance equations with nucleation and size-dependent growth. *AIChE J*. 2004;50:2738–2749.
14. Qamar S, Elsner MP, Angelov IA, Warnecke G, Seidel-Morgenstern AA. Comparative Study of high resolution schemes for solving population balances in crystallization. *Comp Chem Eng*. 2006;30:1119–1131.
15. Qamar S, Warnecke G. Numerical solution of population balance equations for nucleation, growth and aggregation processes. *Comp Chem Eng*. 2007;31:1576–1589.
16. Ma DL, Tafti DK, Braatz RD. Optimal control and simulation of multidimensional crystallization processes. *Comp Chem Eng*. 2002;26:1103–1116.
17. Woo XY, Tan RBH, Chow PS, Braatz RD. Simulation of mixing effects in antisolvent crystallization using a coupled CFD-PDF-PBE approach. *Cryst Growth Des*. 2006;6:1291–1303.
18. Qamar S, Ashfaq A, Warnecke G, Angelov IA, Elsner MP, Seidel-Morgenstern A. Adaptive high-resolution schemes for multidimensional population balances in crystallization processes. *Comp Chem Eng*. 2007;31:1296–1311.
19. Chen P, Sanyal J, Dudukovic MP. CFD modeling of bubble columns flows: implementation of population balance. *Chem Eng Sci*. 2004;59:5201–5207.
20. Bermingham SK, Verheijen PJT, Kramer HJM. Optimal design of solution crystallization processes with rigorous models. *Chem Eng Res Des*. 2003;81:893–903.
21. Fox RO. *Computational Models for Turbulent Reacting Flows*. Cambdrige: Cambridge University Press, 2003.
22. Lian GP, Moore S, Heeney L. Population balance and computational fluid dynamics modelling of ice crystallisation in a scraped surface freezer. *Chem Eng Sci*. 2006;61:7819–7826.
23. Costa CBB, Maciel MRW, Filho RM. Factorial design technique applied to genetic algorithm parameters in a batch cooling crystallization optimization. *Comp Chem Eng*. 2005;29:2229–2241.
24. Irizarry R. A generalized framework for solving dynamic optimization problems using the artificial chemical process paradigm: applications to particulate processes and discrete dynamic systems. *Chem Eng Sci*. 2005;60:5663–5681.
25. Shi D, Mhaskar P, El-Farra NH, Christofides PG. Predictive control of crystal size distribution in protein crystallization. *Nanotechnology* 2005;16:S562–S574.
26. Gunawan R, Fusman I, Braatz RD. ParticleSolver. Available at http://cheed.nus.edu.sg/~chegr/ParticleSolver.htm or http://brahms.scs.uiuc.edu/lssrl/software/ParticleSolver. 2007.
27. Friedlander SK. *Smoke, Dust and Haze*. New York: Wiley, 1977.
28. Gelbard F, Seinfeld JH. Numerical solution of the dynamic equation for particulate systems. *J Comput Phys*. 1978;28:357–375.
29. Hounslow MJ, Ryall RL, Marshall VR. A discretized population balance for nucleation, growth, and aggregation. *AIChE J*. 1988;34:1821–1832.
30. Nicmanis M, Hounslow MJ. Finite-element methods for steady-state population balance equations. *AIChE J*. 1998;44:2258–2272.
31. Hounslow MJ. A discretized population balance for continuous systems at steady-state. *AIChE J*. 1990;36:106–116.
32. Litster JD, Smit DJ, Hounslow MJ. Adjustable discretized population balance for growth and aggregation. *AIChE J*. 1995;41:591–603.
33. Golovin AM. The solution of the coagulation equation for raindrops, taking condensation into account. *Sov Phys Dokl*. 1963;8:191–193.
34. Ramabhadran TE, Peterson TW, Seinfeld JH. Dynamics of aerosol coagulation and condensation. *AIChE J*. 1976;22:840–851.
35. Press WH, Flannery BP, Teukolsky SA, Vetterling WT. *Numerical Recipes in C: The Art of Scientific Computing*, 2nd ed. New York: Cambridge University Press, 1993.
36. Turkevich J, Stevenson PC, Hillier J. A study of the nucleation and growth in the synthesis of colloidal gold. *Discuss Faraday Soc*. 1951;11:55–75.
37. Kumar S, Gandhi KS, Kumar R. Modeling of formation of gold nanoparticles by citrate method. *Ind Eng Chem Res*. 2007;46:3128–3136.
38. Frens G. Controlled nucleation for the regulation of the particle size in monodisperse gold suspensions. *Nature* 1973;241:20–22.
39. Abid JP. Laser induced synthesis and nonlinear optical properties of metal nanoparticles, PhD Thesis, Ecole Polytechnique Federale de Lausanne, France, 2003.
40. Freund P, Spiro M. Colloidal catalysis: the effect of sol size and concentration. *J Phys Chem*. 1985;89:1074–1077.
41. Gunawan R, Ma DL, Fujiwara M, Braatz RD. Identification of kinetic parameters in a multidimensional crystallization process. *Int J Mod Phys B*. 2002;16:367–374.
42. Togkalidou T, Fujiwara M, Patel S, Braatz RD. Solute concentration prediction using chemometrics and ATR-FTIR spectroscopy. *J Cryst Growth*. 2001;231:534–543.

Appendix

The simulations in this article were performed using the ParticleSolver software, which consists of Fortran 77 serial and parallel implementations of the HRFV algorithm with various flux limiters, as well as the upwind, Lax–Wendroff, and other non-HR methods (see Table A1). The solver provides one- and two-dimensional implementations of the HRFV methods with Godunov's splitting, and a one-dimensional parallel programming implementation based on the message-passing interface (MPI). The parallel implementation makes use of the master-slave programming model as described above, in which the master processor performs data gathering and concentration updates, while the slave processors carry out the high-resolution time-marching computations. The subroutines apply to general PBEs by replacing the various kinetics terms in the subroutines. These subroutines were developed and tested using the free distribution of Microsoft Fortran Developer and GNU Fortran g77 with OpenMPI.

The serial subroutines FV1D and FV2D provide one and two-dimensional HRFV implementations, respectively. These subroutines are declared as

Table A1. Variable Definition in ParticleSolver

Variable	Description	Comment
delx, delx1, delx2	Grid cell size	The numbers 1 and 2 in here and in other variables refer to each independent internal coordinates.
delt	Time step size	
tstep	Number of time steps	
ncell, ncell1, ncell2	Number of grid cells	
$\boldsymbol{f}$	Particle distribution vector	The vector $\boldsymbol{f}$ contains f_n^m defined in (5)
C	Solute concentration	Optional variable for crystallization process. Setting C to –1 will deselect C tracking.
Method	Flux limiter selector	Method = 1: Upwind Method = 2: Lax-Wendroff Method = 3: Beam-Warming Method = 4: Fromm Method = 5: Minmod (HR) Method = 6: Superbee (HR) Method = 7: MC (HR) Method = 8: van Leer (HR) (HR: high resolution)
xinit, xinit1, xinit2	Smallest internal coordinate	
tinit	Initial time	
initsz, initsz1, initsz2	Size of nucleating crystals	Optional variable for crystallization process. Setting these to –1 will deselect nucleation tracking.
ierr	MPI error code	
myrank	Processor rank	
numprocs	Number of processors	

Table A2. User-Defined Functions

Description	Format	Comment
Velocity (growth)	1D: GROWTH (x, t, C) 2D: GROWTH1 (x1, x2, t, C) GROWTH2 (x2, x2, t, C)	x, x1, x2: coordinate values t: time C: solute concentration
Source	1D: SOURCE (n, t, f, delx, ncell, C) 2D: SOURCE (I, J, t, f, delx1, delx2, ncell1, ncell2, C)	n: 1D grid cell index I, J: 2D grid cell index
Nucleation	1D: NUCLT(t, f, delx, ncell, C) 2D: NUCLT(t, f, delx1, delx2, ncell1, ncell2, C)	optional, the nucleus size is set in the variables initsz, initsz1, and initsz2 accordingly.
Mass balance	1D: DCDT (t, C, f, delx, ncell, xinit, initsz) 2D: DCDT (t, C, f, delx1, delx2, ncell1, ncell2, xinit1, xinit2, initsz1, initsz2)	optional, for use when the velocity and source functions depend on other concentration(s).

SUBROUTINE FV1D (delx, delt, tstep, ncell, f, C, method, xinit, tinit, initsz)

SUBROUTINE FV2D (delx1, delx2, delt, tstep, ncell1, ncell2, f, C, method, xinit1, xinit2, tinit, initsz1, initsz2)

The parallel subroutine PFV1D derives from FV1D with additional arguments associated with the message-passing interface:

SUBROUTINE PFV1D (delx, delt, tstep, ncell, f, C, method, xinit, tinit, initsz, ierr, myrank, numprocs)

Table A1 defines each input variable in the subroutines. At each call, the subroutine performs the high-resolution time marching algorithm in (6) using the starting particle distribution f from time *tinit* to time $tinit + tstep \times delt$ with the selected flux limiter function according to the variable *method*. User input functions define the velocity function G, the source function h, the nucleation function (optional), and the solute concentration derivative function (optional). The last two functions are typically used in the simulation of crystallization processes (see Example 3). Other user-defined functions should adhere to the format presented in Table A2.

Manuscript received Jun. 26, 2007, and revision received Feb. 18, 2008.