
Alastair J. Smith¹, Clive G. Wells², Markus Kraft¹,³

released: 23 June 2017

¹ Department of Chemical Engineering and Biotechnology
University of Cambridge
New Museums Site
Pembroke Street
Cambridge, CB2 3RA
United Kingdom
E-mail: mk306@cam.ac.uk

² Hughes Hall
University of Cambridge
Wollaston Road
Cambridge, CB1 2EW
United Kingdom

³ School of Chemical and Biomedical Engineering
Nanyang Technological University
62 Nanyang Drive
6357459
Singapore

Preprint No. 186

Keywords: Monte Carlo, stochastic particle method, population balance, mass flow algorithm, single particle method, CFD
Abstract

This paper develops for the first time the mathematical foundations of an efficient stochastic algorithm for solving steady-state population balance equations, which is suitable for coupling to computational fluid dynamics codes. The properties and performance of the algorithm are charted using a systematic comparison with an existing stochastic algorithm. The algorithm is extended to 1D geometries with non-uniform flows. The method is enhanced by the inclusion of additional stochastic jump processes to improve its convergence characteristics and studies are performed in order to establish values of key parameters rendering the method suitable for CFD applications.

Highlights

- The mathematical foundations of the single particle method are developed.
- The convergence and numerical properties of the method are investigated in detail.
- Additional jump processes are introduced to improve the convergence character.
- The method is extended to spatially dependent problems with non-uniform velocities.
- Parameter studies are performed in order to justify the use of the method in CFD applications.
## Contents

1 Introduction .................................................. 3

2 Mass Flow Formulation of The Smoluchowski Equation ............. 5
   2.1 The Smoluchowski Equation .................................. 5
   2.2 Weak Form & Stochastic Generators ........................ 6

3 The Single Particle Method ..................................... 8
   3.1 Essential Idea of The Single Particle Process ................ 8
   3.2 Derivation ................................................ 9
      3.2.1 Field Particle Mass Density ............................ 9
      3.2.2 Replacement Process .................................. 10
      3.2.3 Test Particle Coagulation & Outflow ................. 10
   3.3 The Algorithm ............................................ 11
   3.4 Choice of Free Parameters .................................. 12
   3.5 Numerical Results ......................................... 12
   3.6 Convergence Study ........................................ 15
   3.7 Computational Efficiency .................................. 18

4 Spatially Inhomogeneous Geometries ............................... 18
   4.1 Implementation ............................................ 19
   4.2 Time-Dependent Case ...................................... 22

5 An Improved Algorithm ........................................ 23
   5.1 Improving the Convergence Behaviour ........................ 23
   5.2 Updated Numerical Results .................................. 24

6 Conclusions .................................................................. 27

7 Appendix .................................................................... 27
   7.1 Mathematical Analysis of The SPM ............................ 27
   7.2 Jumps & Rates ................................................. 31

References ................................................................... 35
1 Introduction

Stochastic methods are a very highly developed class of solution scheme for population balance equations (PBEs). The use of this approach to simulate population balances and rate processes has enjoyed widespread discussion in the literature [17, 19, 34]. A direct simulation Monte Carlo (DSMC) method, a probabilistic approach to solving the Boltzmann equation, in which fluid flows are modelled using a simulation of stochastic particles which represent a large number of real particles was proposed by Bird [7, 8, 9]. Particles are transported through a simulation of physical space in a realistic manner that is directly coupled to physical time such that unsteady flow characteristics can be accounted for, inter-particle interactions being calculated using probabilistic, phenomenological models.

The classic reference for the phenomenological approach to modelling coalescence and breakage in a turbulent dispersion is the work of Coulaloglou and Tavlarides [11, 12]. This work has been updated with various enhancements in order to solve complex problems that involved particle collisions [6, 25, 27, 28, 38, 42]. The DSMC method has been extended by adding a stochastic model of coalescence in order to describe droplet formation in clouds by Gillespie [18]. Eibeck and Wagner [13] used stochastic methods in order to study coagulation and fragmentation problems, derived the direct simulation algorithm (DSA) and introduced majorant kernels in order to reduce the complexity class of the algorithm. This algorithm was extended to include a source term for gas-phase reactions by Goodson and Kraft [20] and Balthasar and Kraft [5]. Wells and Kraft [47] compare DSA with a mass flow version for solving sintering-coagulation equations. Patterson and Kraft [32] increased the computational efficiency of this method by noticing that nonlinear processes dominate the computational cost, so these could be prioritised and linear processes deferred (the Linear Process Deferment Algorithm). Lapeyre et al. [26] give a survey of the application of stochastic methods to more general transport equations.

The key advantage of this class of algorithm is that, whilst they may be outperformed by the alternatives when the number of dimensions is low, they scale computationally very favourably with increasing number of internal coordinates. As the number of dimensions increase, the true efficacy of these methods is realised, and the advantage of the dimension-independent $1/\sqrt{N}$ scaling comes into its own; deterministic methods typically being an exponential function of dimension. However, there exists a number of strategies for increasing the computational efficiency of stochastic methods, even in a low number of dimensions, as mentioned above.

One problem is that particle birth processes (such as inception or breakage) can cause the number of particles being simulated to become unfeasibly large (which can be both CPU and memory intensive). Conversely, death events can serve to deplete the particle distribution to the point of becoming statistically unrepresentative (a particular problem for kernels with homogeneity $> 1$ which are known to be gelling [43]), and so increasing the variance. This can be resolved by introducing a particle doubling strategy (in which the ensemble is duplicated) and a particle contraction strategy (in which random particles are deleted from the ensemble), respectively, an approach employed by Celnik [10], Sabelfeld et al. [37]. Smith and Matsoukas
present a simple alternative solution in which the number of particles is kept constant, regardless of whether the processes result in an actual net loss or gain in the number of particles.

Another source of inefficiency is in choosing pairs of collision partners to coalesce, which is quadratic in the number of particles. This can be partially resolved by using an efficient binary tree data structure to store the ensemble \([24, 47]\), which can improve the efficiency of the acceptance-rejection algorithm used to select indices from \(O(N^2)\) to \(O(N \log N)\). Further improvements were suggested by Eibeck and Wagner \([13, 14]\), who introduced majorant kernels, allowing the particles to be selected independently. The use of these kernels, however, reduces the waiting time between events, necessitating the introduction of fictitious jumps (null events) to preserve the integrity of the simulation.

Babovsky \([3]\) introduced a stochastic algorithm in which the particles are used to represent mass density rather than number density, achieving an effective reduction in the variance of the algorithm. Eibeck and Wagner \([14, 15]\) used this mass flow representation to develop a stochastic algorithm, the mass flow algorithm (MFA) which could accurately resolve the particle size distribution with a reduced number of stochastic particles. This algorithm was used to simulate the dynamics of nanoparticles by Goodson and Kraft \([20]\) and later, to study coalescence/breakage in a liquid-liquid extraction process by Goodson and Kraft \([21]\). Morgan et al. \([30]\) used the algorithm to study nanoparticle formation in a TiCl\(_4\) flame reactor.

For steady flows, a further simplification is possible, by using a variant of the MFA, known as the single particle method (SPM), often referred to in the literature as a test particle method \([44]\). The essential idea behind this method is that we can effectively linearise the collision kernel by introducing a second mass density, the test particle mass density, which converges to the ordinary (field) particle mass in the steady-state limit. The test particle can then coalesce with a fictitious collision partner which is efficiently selected from the field particle ensemble, the particle always having a collision partner, even in a low-density region. To ensure the local distribution of field particles remains representative, it is necessary to periodically update the field particles mass density with that of the test particle at a rate proportional to the test particle’s residence time. This approach was originally applied to the solution of the Boltzmann equation by Haviland and Lavin \([22]\), but has been used more recently to study coalescence/breakage problems in zero dimensional reactors by Ramkrishna \([35]\), Ramkrishna et al. \([36]\).

The mathematical literature on stochastic methods is dominated by proofs of the existence and uniqueness of solution algorithms to the underlying PBE \([2, 16, 31]\), rather than the efficacy and numerical behaviour of the various approaches for real systems. The purpose of this paper is to develop the mathematical foundations of the SPM, before focussing on these more pragmatic aspects, conducting a detailed analysis of its convergence properties and performing the parameter studies necessary to justify its use in a computational fluid dynamics (CFD) context.

The structure of the paper is as follows. In §2, we outline an existing stochastic particle algorithm (MFA), which will be used in order to establish a reference frame
from which to conduct a systematic comparison. The SPM is introduced in §3, and extended to spatially inhomogeneous systems, with non-uniform flows in §4. In §5, an improved algorithm is presented which relies on the introduction of new jump processes to improve its convergence characteristics, which were motivated by a detailed mathematical analysis of the existing method. However, we begin by introducing the fundamental equation describing the evolution of coagulating particles in the next section.

2 Mass Flow Formulation of The Smoluchowski Equation

2.1 The Smoluchowski Equation

Consider some domain $\Omega$ of volume $V$. Let $n^{in}(x)$ denote the number of particles of size $x \in \mathbb{R}^+$ entering $\Omega$ every $\alpha > 0$ units of time, and $\beta(x) > 0$ denote their size-dependent residence time. The number density of particles of size $x$ at time $t > 0$, $n(x,t)$, obeys a more general form of the coagulation equation first introduced by Smoluchowski [41] (which we shall hereafter refer to as the Smoluchowski equation)

$$\frac{\partial n(x,t)}{\partial t} = \frac{1}{2} \int_0^x K(x - y, y)n(y,t)n(x - y, t) \, dy + \frac{n^{in}(x)}{\alpha V} - \int_0^\infty K(x, y)n(y,t)n(x, t) \, dy - \frac{n(x,t)}{\beta(x)},$$

(1)

where $K : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$ is the symmetric coagulation kernel, and the probability that two particles of sizes $x$ and $y$ coalesce during a small time interval $dt$ and volume $dV$ is proportional to $K(x, y) \, dt/ \, dV$ (note that $K$ is not a pure rate because it has dimensions of volume/time rather than $1$/time). In this same time interval, a particle can leave $\Omega$ with probability proportional to $dt/\beta(x)$, or a new particle can enter $\Omega$ with probability proportional to $dt/V/\alpha$. This equation is a fundamental mean-field model for cluster growth and arises in a diverse range of fields including physical chemistry, astrophysics, meteorology and the dynamics of biological systems. Aldous [2] gives a comprehensive general survey of existing literature in stochastic coalescence theory and discusses many of the applications of this equation, in addition to a number of interesting open problems. Pego [33] gives a review of some of the more recent work in the field. Melzak [29] discusses the local existence and uniqueness of solutions in general terms.

According to (1), the particle concentration $n(x,t)$ can increase either by the coagulation of particles of sizes $y < x$ and $x - y$ (first term) or simply by a particle of size $x$ being incepted into $\Omega$ (second term), and can decrease by the coagulation of a particle of size $x$ with any other particle of size $y$ (third term) or simply by a particle of size $x$ leaving the system after a size dependent time $\beta(x)$ (last term).
2.2 Weak Form & Stochastic Generators

We now express (1) in terms of mass density \( m(x,t) \), which occurs more frequently in applications and offers numerous advantages \([35]\). The mass density of particles which have mass \( x \) at time \( t \) is

\[
m(x,t) = xn(x,t),
\]

and the total mass within \( \Omega \) is

\[
M = \int_\Omega m(x,t) \, dx.
\]

The number of particles entering \( \Omega \) of size \( x \) is given by \( m^{in}(x) = xn^{in}(x) \) and the total flow rate of particles through \( \Omega \) is \( Q_{in}/\alpha \), where

\[
Q_{in} = \int_\Omega m^{in}(x) \, dx.
\]

From now on, we choose units in which \( V = 1 \) (or otherwise absorb \( V \) into the definition of \( \alpha \)). Substituting \( n(x,t) = m(x,t)/x \) into (1) and multiplying by \( x \) furnishes the mass flow equation:

\[
\frac{\partial m(x,t)}{\partial t} = \int_\mathbb{R} \frac{K(x-y,y)}{y} m(y,t) m(x-y,t) \, dy + \frac{m^{in}(x)}{\alpha} - \int_\mathbb{R} \frac{K(x,y)}{y} m(y,t) m(x,t) \, dy - \frac{m(x,t)}{\beta(x)},
\]

where we have noted that \( K(x,y) = 0 \) for \( x \leq 0 \) or \( y \leq 0 \), so that the limits of integration can be extended over the whole of \( \mathbb{R} \). The factor of \( \frac{1}{2} \) disappears from the first integral because coagulation reduces the number particles but leaves their mass unaltered, which is mathematically manifested through the partial fractions decomposition \( x/(y(x-y)) = 1/y + 1/(x-y) \) and the symmetry of the coagulation kernel.

The derivation of (2) from (1) can be made mathematically rigorous, and the form of an appropriate algorithm to solve the equation more transparent by introducing the weak form of the equation. In order to do this, we multiply (2) by an arbitrary test function \( \varphi(x) \) (i.e., a smooth function with compact support) and integrate over \( x \). After some algebra and upon the introduction of a mass density measure \( Q(t, dx) \), we find

\[
\frac{\partial}{\partial t} \langle \psi, Q(t) \rangle = \langle \psi, Q_0 \rangle + \int_\mathbb{Z} \frac{\psi(x)}{\alpha} Q_{in}(t, dx) - \int_\mathbb{Z} \frac{\psi(x)}{\beta(x)} Q(t, dx) + \int_\mathbb{Z} \left[ \psi(x+y) - \psi(x) \right] \frac{K(x,y)}{y} Q(t, dx) Q(t, dy),
\]

where the factor of \( \frac{1}{2} \) disappears for the same reason as before. Taking \( \mathbb{Z} = \mathbb{R}^+ \), \( Q_{in}(dx) = m^{in}(x) \, dx \) and \( Q(t, dx) = m(x,t) \, dx \) reproduces (2) under appropriate
regularity assumptions [15]. Once again, the integral can be extended over the whole of $\mathbb{R}$ by appealing to the properties of $K$.

In order to solve this equation, we introduce a sequence of measure-valued jump processes

$$Q^{(N)} = \left\{ \lambda \sum_{i=1}^{N} \delta_{x_i} \mid \lambda, x_i \in \mathbb{R}, \right\}$$

such that the sequence converges in distribution to $Q$ as $N \to \infty$. We now let

$$Q^{(N)} = \lambda^{(N)} R^{(N)},$$

and thus accordingly we set

$$R^{(N)} = \sum_{i=1}^{N} \delta_{x_i}, \quad \text{and} \quad \lambda^{(N)} = \lambda_0 \left( \frac{N}{N-1} \right)^{\pi}.$$

The measure $R$ depends on a set of stochastic particles and $\lambda$, an overall scaling factor. The parameter $\lambda_0$ is calculated from the original number density of particles in the system $n_0(x)$ by $\lambda_0 = n_0(x)/N$.

In order to find the generators of the processes described by this equation, we define the quantity $\Psi(\pi, R)$ by

$$\Psi(\pi, R) = \int_{\mathbb{Z}} \psi(x)Q(dx) = \lambda_0 \left( \frac{N}{N-1} \right)^{\pi} \int_{\mathbb{Z}} \psi(x)R(dx). \quad (4)$$

Consequently, we will generate solutions to

$$\frac{\partial}{\partial t} \Psi(\pi, R) = \mathcal{A}_i^N \Psi(\pi, R) + \mathcal{A}_c^N \Psi(\pi, R) + \mathcal{A}_o^N \Psi(\pi, R), \quad \forall N, \quad (5)$$

where $\mathcal{A}_i^N$, $\mathcal{A}_c^N$ and $\mathcal{A}_o^N$ are the stochastic generators for inflow, coagulation and outflow respectively. Comparing terms from (5) with those in (3) reveals the form of these generators, and enables us to cast the equation into the form of a Markov jump process. We define jumps

$$J_c(\pi, R, x, y) = (\pi + 1, R + \delta_1 - \delta_y),$$

$$J_i(\pi, R, y) = (\pi + 1, R + \delta_1 - \delta_y),$$

$$J_o(\pi, R, x, y) = (\pi - 1, R - \delta_x + \delta_y),$$

for coagulation, inflow and outflow, respectively, with corresponding rates given by

$$\rho = \lambda N \sum_{i=1}^{N} \sum_{j=1}^{N} \frac{K(x_i, x_j)}{x_j}, \quad \mu = \frac{1}{\lambda \alpha} \frac{N-1}{N}, \quad \text{and} \quad \vartheta = \sum_{i=1}^{N} \frac{1}{\beta(x_i)}. \quad (6)$$

7
Algorithm 1: The Mass Flow Algorithm (MFA)

\[\text{input} : N, L, K, \text{error}, m\text{in}, \alpha, \beta(x),\]
\[\text{initial particle distribution } (x_1, \ldots, x_N);\]
\[\text{output} : \text{Final particle size distribution } \{x_i(t), i \in \mathbb{N}\},\]
\[\text{first K moments } m_k(t);\]
\[\begin{align*}
\pi & \leftarrow 0, \\
L & \leftarrow 0, \\
\text{while } L \lt \text{error} \text{ do} \\
\text{calculate} & \text{: total rate } \rho(x) \leftarrow \rho(x) + \theta(x) + \mu(x) \text{ according to (6)}; \\
\text{generate} & \text{: an exponentially distributed time step } \tau \text{ with parameter } \rho(x), \text{ i.e., the waiting time } \tau \text{ is chosen so that it is distributed}\] \\
\text{update} & \text{: } \lambda \leftarrow \lambda_0 \left(\frac{N}{N-1}\right) \pi, M \leftarrow \lambda N, L \leftarrow L + \tau; \\
\text{with probability } \mu(x)/\rho(x) \text{ goto Inflow, with probability } \theta(x)/\rho(x) \text{ goto Outflow else goto Coagulation}; \\
\text{Inflow} & \text{A particle enters } \Omega. \\
& \bullet \text{ Add a particle of size } y \sim m\text{in}(x) \text{ to the computational domain, } \Omega. \\
& \bullet \pi \leftarrow \pi + 1 \\
& \text{goto Update}; \\
\text{Outflow} & \text{A particle leaves } \Omega. \\
& \bullet \text{ Remove a uniformly chosen particle } x_i \text{ from } \Omega \text{ and replace it by a duplicate of a particle } x_j, \text{ chosen uniformly from the particle ensemble.} \\
& \bullet \pi \leftarrow \pi - 1 \\
& \text{goto Update}; \\
\text{Coagulation} & \text{Two particles collide and coagulate.} \\
& \bullet \text{ Choose } i \text{ and } j \text{ with } i \neq j \text{ according to the distribution } \frac{K(x_i, x_j)}{\rho(x_i) \rho(x_j)}; \\
& \bullet \text{ Add a particle of size } x_i + x_j \text{ to the particle ensemble and remove one of size } x_i \\
& \text{goto Update};
\end{align*}\]

Taken together, these rates and jumps lead to Algorithm 1, which will form the basis of our numerical comparison.

3 The Single Particle Method

3.1 Essential Idea of The Single Particle Process

Following Vikhansky and Kraft [44], we start from the Smoluchowski equation in mass flow form (2). We then introduce the mass density of field particles, \(\phi(x, t)\), which converges to the mass density of test particles \(m(x, t)\) as \(t \to \infty\). (2) is then reduced to a pair of linear equations, one to describe the test particle coagulation (with a field particle) and outflow:

\[
\frac{\partial m(x, t)}{\partial t} = \int_{\mathbb{R}} \frac{K(x - y, y)}{y} m(x - y, t) \phi(y, t) \, dy - \int_{\mathbb{R}} \frac{K(x, y)}{y} m(x, t) \phi(y, t) \, dy - \frac{m(x, t)}{\beta(x)},
\]

and another to describe the replacement of a field particle by a test particle:

\[
\frac{\partial \phi(x, t)}{\partial t} = m\text{in}(x) - \frac{\phi(x, t)}{\alpha}.
\]
(7) is a linear equation in $m$ which describes the evolution of the test particles, whilst (8) is a linear equation in $\phi$ which describes the evolution of the field particles. As a consequence of linearising (2), it is now necessary to solve these two coupled linear equations simultaneously.

### 3.2 Derivation

#### 3.2.1 Field Particle Mass Density

The distribution of field particles $\phi$ is represented as a sum of discrete measures

$$\phi(y, t) = \frac{\Phi}{N} \sum_{i \in \mathbb{Z}} \delta_{y_i}(dy),$$

where $\Phi$ is the total mass density of field particles, so

$$\Phi = \int_{\mathbb{R}} \phi(y, t).$$

Integrating (8) over all $x \in \mathbb{R}$, gives

$$\frac{d\Phi(t)}{dt} = \frac{Q_{in}}{\alpha} - \Phi(t). \tag{9}$$

Using a finite difference approximation, we have

$$\Phi(t_{n+1}) = (1 - \frac{\tau}{\alpha}) \Phi(t_n) + \frac{Q_{in} \tau}{\alpha},$$

where $\tau = t_{n+1} - t_n$. This prescribes how we should update the total mass density of field particles, $\Phi$, at each time step within the algorithm:

$$\Phi^n \leftarrow \left(1 - \frac{\tau}{\alpha}\right) \Phi^n. \tag{10}$$

followed by

$$\Phi^n \leftarrow \Phi^n + \frac{Q_{in} \tau}{\alpha}, \tag{11}$$

At the end of each iteration, we need to update the mass density of field particles,

$$\Phi^{n+1} \leftarrow \Phi^n,$$

but in order to facilitate the stability of the scheme, we update the mass density using under-relaxation as

$$\Phi^{n+1} \leftarrow \varepsilon_1 \Phi^{n+1} + (1 - \varepsilon_1) \Phi^n, \tag{12}$$

where $\varepsilon_1$ is the under-relaxation factor.
3.2.2 Replacement Process

Starting from (8) we can deduce an equation for the evolution of the normalised mass density of field particles, i.e., a PDF transport equation for

\[ f(x, t) = \frac{\phi(x, t)}{\Phi(t)}. \]  \hfill (13)

To this end we substitute (13) into (8), which gives

\[ \frac{d\Phi}{dt} f(x, t) + \Phi(t) \frac{\partial f}{\partial t} = \frac{m^{in}(x)}{\alpha} - \frac{\Phi(t)f(x, t)}{\alpha}, \]

into which we substitute (9) to get

\[ \frac{\partial f(x, t)}{\partial t} = \frac{m^{in}(x)}{\alpha \Phi} - \frac{Q^{in}f(x, t)}{\alpha \Phi} \]

\[ = \frac{Q^{in}}{\alpha \Phi} \left( \frac{m^{in}(x)}{Q^{in}} - f(x, t) \right). \] \hfill (14)

This implies that the particles should be replaced with rate \( \frac{Q^{in}}{\alpha \Phi} \)

per unit time, or with probability proportional to \( \frac{Q^{in}r}{\alpha \Phi} \),

at each time step.

3.2.3 Test Particle Coagulation & Outflow

The test particle coagulation and outflow processes are treated independently, and can be derived in a similar manner to the processes in §2.2.

Let \( Z = (x, P^n, M, \Phi^n) \) and consider the generator

\[ G(\Psi, t)(Z) = \frac{\Phi^n}{N} \sum_{i=1}^{N} \frac{K(x, y^n_i)}{y^n_i} (\Psi(J^n_i(Z)) - \Psi(Z)) \]

\[ + \frac{1}{\beta(x)} (\Psi(J_{out}(Z)) - \Psi(Z)) \]

with jumps \( J \) defined by

\[ J_i^c(Z) = (x + z_i, P^n, M, \Phi^n), \]  \hfill (coagulation)

\[ J_{out}(Z, y_{i+1}^{n+1}, \Phi_{i+1}^{n+1}) = (\cdot, P + \frac{1}{N} \sum_{i=1}^{N} (\delta_{y_{i+1}^{n+1}} - \delta_{y_{i}^{n}}), \]
\[ M, \varepsilon_1 \Phi^{n+1} + (1 - \varepsilon_1)\Phi^n. \]

The total rate for these processes is given by

\[ \varrho(z, t) = \rho(x) + \frac{1}{\beta(x)} = \frac{\Phi^n}{N} \sum_{i=1}^{N} K(x, y^n_i) \frac{1}{y^n_i} + \frac{1}{\beta(x)}, \]

where \( \rho(x) \) is the coagulation rate and \( 1/\beta(x) \) is the test particle outflow rate (now there is just a single particle).

### 3.3 The Algorithm

**Algorithm 2: The Single Particle Method (SPM)**

<table>
<thead>
<tr>
<th>Input</th>
<th>( N, L, K, ) iterMax, ( m^0, \alpha, \beta(x) ), initial particle distribution ( (x_1, \ldots, x_N) )</th>
<th>Output</th>
<th>Steady-state PSD ( {x_i, i \in \mathbb{N}} ), first ( K ) moments ( m^k )</th>
</tr>
</thead>
<tbody>
<tr>
<td>begin</td>
<td>( \Phi_p \leftarrow 1 ); ( y_{p,i}, y_{f,i} \leftarrow 1, \forall i \in \mathbb{N} ); iter ( \leftarrow 0 )</td>
<td>while iter &lt; iterMax do</td>
<td>( \Phi_f \leftarrow 0 ); A test particle enters the system.</td>
</tr>
<tr>
<td>Inflow</td>
<td>* TestParticleActive ( \leftarrow ) True;</td>
<td>calculate</td>
<td>( \rho(x) \leftarrow \frac{\Phi_p}{N} \sum_{i=1}^{N} K(x, y^n_i) \frac{1}{y^n_i}, \rho(x) \leftarrow \rho(x) + \frac{1}{\beta(x)} )</td>
</tr>
<tr>
<td>Replacement</td>
<td>generate</td>
<td>( r ) with parameter ( \rho(x) );</td>
<td>update</td>
</tr>
<tr>
<td>Coagulation</td>
<td>( \Phi_f \leftarrow \Phi_f ) ( \frac{\varepsilon_2 \alpha Q_{in} r}{\alpha \Phi_p} ), (15)</td>
<td>replace a uniformly chosen field particle by the test particle: ( y_{f,i} \leftarrow x );</td>
<td>With probability ( \rho(x)/\varrho ) goto Coagulation else goto Outflow; Two particles collide and coagulate.</td>
</tr>
<tr>
<td>Outflow</td>
<td>Choose a field particle index ( i ) (collision partner) with probability</td>
<td>and coagulate with the test particle: ( x \leftarrow x + y_{f,i} );</td>
<td>The test particle leaves the system.</td>
</tr>
<tr>
<td></td>
<td>( K(x, y^n_i) \Phi_p ) ( \frac{N \rho(x) y^n_i}{\Phi^n} )</td>
<td>* Under-relax the field particles' mass density: ( \Phi_p \leftarrow \varepsilon_1 \Phi_f + (1 - \varepsilon_1)\Phi_p );</td>
<td></td>
</tr>
<tr>
<td></td>
<td>and update the field particles array: ( y_{f,i} \leftarrow y_{f,i}, \forall i );</td>
<td>* Update the field particles array: ( y_{f,i} \leftarrow y_{f,i}, \forall i );</td>
<td></td>
</tr>
<tr>
<td></td>
<td>testParticleActive ( \leftarrow ) False;</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>update</td>
<td>iter ( \leftarrow ) iter + 1;</td>
<td></td>
</tr>
</tbody>
</table>

This suggests an algorithm which is similar to Algorithm 1, but in which particle inflow is replaced by the field particle replacement step. The test particle coagulation and outflow events are treated independently from this. With probability \( \rho(x)/\varrho \), the test particle coagulates with a field particle, chosen according to the distribution

\[ \frac{K(x, y^n_i) \Phi^n}{N \rho(x) y^n_i}. \]
otherwise the test particle exits the system. From this, we construct Algorithm 2 which solves (2) in the steady-state limit. It is equivalent to the simultaneous equations (7), (9) and implicitly (14), the PDF transport equation.

3.4 Choice of Free Parameters

Algorithm 2 contains three free simulation parameters: the number of field particles, \( N \), an underrelaxation factor, \( \varepsilon_1 \), and an additional factor to tune the frequency with which the field particle distribution is updated (test particle replacement events), \( \varepsilon_2 \). In order to complete the description of the algorithm, we must specify suitable values for these parameters. To this end, we consider the probability that a test particle will leave \( \Omega \) without replacing a field particle, which is given by the product of (15) over all time steps:

\[
\prod_j \left( 1 - \frac{\varepsilon_2 Q_{\text{in}} \tau_j}{\alpha \Phi_p} \right) \approx 1 - \varepsilon_2 \frac{Q_{\text{in}}}{\alpha \Phi_p} \sum_j \tau_j = 1 - \varepsilon_2 \frac{Q_{\text{in}}}{\alpha \Phi_p} t_{\text{res}},
\]

where we have used the assumption that \( \tau_j \ll t_{\text{res}} \), and the fact that \( \exp(-x) \approx 1 - x \) for sufficiently small \( x \). Now \( \Phi \sim t_{\text{res}} Q_{\text{in}} / \alpha \), so in order to keep (16) positive, we must have \( \varepsilon_2 < 1 \). In all of the simulations performed in this work, we took \( \varepsilon_2 = 0.5 \).

Numerical experiments corroborated the findings of Vikhansky and Kraft [44], and revealed that in order to represent \( \phi \) adequately and avoid correlation between consecutive collisions, the number of field particles, \( N \), needs to be approximately equal to the number of collisions that a test particle undergoes before leaving the computational domain, \( \Omega \). The parameter \( \varepsilon_1 \) gives the interval over which \( \Phi \) is calculated. According to (12), the mass density at the \( n \)th step of the algorithm is averaged over approximately \( 1/\varepsilon_1 \) previous iterations. In these experiments, it was found that choosing \( 0.01 \leq \varepsilon_1 \leq 0.1 \) prevented oscillations of the particle mass density in \( \Omega \) from becoming too large.

3.5 Numerical Results

We implemented both algorithms in C++ using a number of computational enhancements, including the generation of collision events using fictitious jumps [13], and efficient particle index selection using a type of inversion [45, 46], which we implemented using a binary tree data structure [24, 47].

It is known that at least three particular classes of kernel are analytically soluble: the constant, additive and multiplicative. Calculations were conducted for all of these kernels, and the behaviour of the solutions and the properties of the algorithms studied in detail. However, of these, we present only results for the multiplicative kernel. The behaviour of the algorithms for this class is entirely indicative of the
others, in addition to offering a slightly more challenging numerical experiment than the others.

The multiplicative kernel is given by

\[ K(x, y) = Axy, \ A \in \mathbb{R}. \]

Simulations were performed with particle in and outflow rates which were constant and equal, \( \alpha = \frac{1}{10} = \beta \) and with a multiplicative coagulation constant \( A = \frac{1}{2} \). We assumed mono-dispersed initial conditions \( m(x, 0) = \delta(1 - x)dx \), and uniformly distributed inflowing particles, \( m^{\text{in}}(x) \sim U[0, 1] \). The same parameters were used for both algorithms. All calculations were performed on a single core of an Intel\textsuperscript{®} Xeon\textsuperscript{R} X5472 (Harpertown) processor with a clockspeed of 3.00 GHz and 12 Mb of L2 cache.

A comparison between the steady-state particle size distributions is given in Figure 1, with \( N = 2^{16} \) stochastic particles\textsuperscript{2} and \( L = 100 \) runs in each case. Figure 2a shows a comparison between the steady-state solution for the first 3 non-trivial moments calculated using a low resolution SPM simulation (\( N = 2^9 \) field particles, \( L = 10 \)) along with the analytical solution (which can be established by taking moments of (2) \[39\]). Figure 2b shows the same curves for a much higher resolution (\( N = 2^{16} \) field particles, \( L = 100 \)) run.

Figure 3 illustrates the convergence of the test particle mass density to the field

\[ \text{Particle Mass Density} \]

\[ \text{Particle size} \]

\[ \text{SPM (65536 particles, 100 runs)} \]

\[ \text{MFA (65536 particles, 100 runs)} \]

---

\textbf{Figure 1:} Steady-state particle size distribution (PSD) for a system with particles undergoing coagulation described by \( K(x, y) = \frac{1}{2}xy \), with \( \alpha = \frac{1}{10} = \beta \), \( n(x, 0) = \delta(x-1)dx \) and \( m^{\text{in}}(x) \sim U[0, 1] \). Left (blue) bars in the stacked histogram show the results of calculations performed using the single particle method (\( N = 65536 \) field particles, \( L = 100 \)), whilst the right (green) bars show the solution obtained using the mass flow algorithm (\( N = 65536, L = 100 \)).

\textsuperscript{2}N.B., In the SPM this refers to the number of field particles.
Figure 2: Convergence to steady-state of the first three non-trivial moments in a system with particles undergoing coagulation described by \( K(x, y) = \frac{1}{2}xy \), with \( \alpha = \frac{1}{10} = \beta \), \( n(x, 0) = \delta(x - 1)dx \) and \( m_i^{\text{in}} \sim U[0, 1] \).

Figure 3: Convergence of test particle mass density to field particle mass density as the number of iterations increases.
particle mass density, \( M \rightarrow \Phi \), as the number of replacement processes increases and the simulation tends towards steady-state.

In Figure 4 we show the results from a number of simulations in which the inflow rate \( Q_{in}/\alpha \), was varied, in order to see how this affects the \( L_2 \) error in the first three non-trivial moments. We observe that the error increases as \( \alpha \) decreases. This is because for large values of \( \alpha \), the reactor is flow dominated, so mostly in and outflow events are performed with little coagulation. As \( \alpha \) decreases, coagulation starts to dominate, increasing the variance in the PSD, and hence the variance in the moments.

![Figure 4: \( L_2 \) error in moments with varying \( Q_{in}/\alpha \). All simulations performed with \( N = 2^{10} \) and \( L = 100 \).](image)

### 3.6 Convergence Study

In this section we shall study the convergence properties of the SPM, by comparing its performance with that of the MFA. In particular, we will examine how quickly the simulation converges to the exact solution as \( N \) increases. We will appeal to two of the most important theorems in the calculus of probabilities: the strong law of large numbers, which will allow us to justify the convergence of the method; and the central limit theorem, which will enable us to estimate the rate of convergence.

Applications typically require only global properties of the particle size distribution, for example, moments, volume fraction, etc., which are functionals of the form

\[
F(t) = \int_{0}^{\infty} \varphi(x)m(x,t) \, dx.
\]  

(17)

These functionals can be approximated by the random variable

\[
\xi^{(N)}(t) = \frac{1}{N} \sum_{i=1}^{N} \varphi(x_i(t))
\]  

(18)
in the limit as \(N \to \infty\). We take \(\varphi(x) = x^k, k \in \mathbb{Z}\), so our functionals approximate the integer moments of mass density. It can be shown that these functionals of (2) with a multiplicative kernel admit analytical solutions, making these choices of functional and kernel particularly useful for establishing the numerical properties of the algorithm [39].

In order to calculate the expectation and variance of (18), we perform a number, \(L\), of independent runs, with corresponding random variables \(\xi^{(N,1)}(t), \ldots, \xi^{(N,L)}(t)\). The empirical mean is the arithmetic mean of these random variables

\[
\eta_1^{(N,L)}(t) = \frac{1}{L} \sum_{i=1}^{L} \xi^{(N,i)}(t).
\]

The variance is given by \(\text{var} \xi^{(N)}(t) = \mathbb{E}[\xi^{(N)}(t)^2] - \mathbb{E}^2 \xi^{(N)}(t)\), which is approximated using the (asymptotically unbiased) empirical variance

\[
\eta_2^{(N,L)}(t) = \frac{1}{L-1} \sum_{i=1}^{L} \xi^{(N,i)}(t)^2 - \eta_1^{(N,L)}(t)^2.
\]

The error in the approximation is defined to be

\[
e^{(N,L)} = \left| \eta_1^{(N,L)}(t) - F(t) \right|
\]

and is comprised of two contributions: a \textit{systematic error} and a \textit{statistical error}. The systematic error is the difference between the true value of the functional and the expectation of the random variable approximating it

\[
e^{(N,L)}_{\text{sys}} = \mathbb{E}[\xi^{(N)}(t)] - F(t),
\]

whilst the statistical error is the difference between the empirical mean and the expectation of the random variable

\[
e^{(N,L)}_{\text{stat}} = \eta_1^{(N,L)}(t) - \mathbb{E}[\xi^{(N)}(t)].
\]

The sample paths \(\xi^{(N,i)}(t), i \in \mathbb{N}\) constitute a collection of independent and identically distributed random variables, with mean \(\mu\) and variance \(\sigma^2\), so we can appeal to the central limit theorem [23] in order to furnish us with confidence bounds for the sample mean. We find that

\[
e^{(N,L)}_{p} = z \sqrt{\frac{\eta_2^{(N,L)}(t)}{L}}
\]

is a probabilistic upper bound for the statistical error, were \(z\) is found from the inverse cumulative normal distribution function of the required confidence level. In this work, we use a 99.9% confidence level, so by symmetry, \(z = \Phi^{-1}(0.9995) = 3.2905\).
In order to analyse the statistical error on the interval \([0, T]\), we divide the interval into \(M\) equispaced subintervals \([t_i, t_{i+1}]\), with \(t_0 = 0, t_M = T\) and \(\delta t = t_{i+1} - t_i = T/M\). The statistical error is then characterised by the statistic

\[
  c_{\text{stat}} = \max_i \{ c_p^{(N,L)}(t_i) \}.
\]

In the case of a multiplicative kernel, we have analytical expressions for our functionals \(F(t)\), so we can analyse the systematic error on the interval \([0, T]\) directly by calculating the mean of (19) over all timesteps:

\[
  c_{\text{tot}} = \frac{1}{1 + M} \sum_{i=0}^{M} c^{(N,L)}(t_i).
\]

The errors \(c_{\text{tot}}\) and \(c_{\text{stat}}\) are calculated for \(F(t) = m_1(t)\), i.e., the average particle size, along with the CPU time for a single run and the CPU time per million particles in Table 1. It should be noted, in particular, that it includes values for a single stochastic particle \((N = 1)\), and the errors, whilst significant are not unacceptably large, even in this pathological case.

**Table 1: Numerical convergence study for the single particle method (SPM). CPU times are in seconds \((N \times L = 2^{13} \times 10^2 \sim 8.1 \times 10^5)\).**

<table>
<thead>
<tr>
<th>(N)</th>
<th>(c_{\text{stoch}})</th>
<th>(c_{\text{tot}})</th>
<th>(t_{\text{CPU}}/L)</th>
<th>(t_{\text{CPU}}/N \times 10^6)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.01818</td>
<td>0.2814</td>
<td>0.0005181</td>
<td>5.181</td>
</tr>
<tr>
<td>2</td>
<td>0.01328</td>
<td>0.1707</td>
<td>0.0006455</td>
<td>3.227</td>
</tr>
<tr>
<td>4</td>
<td>0.008832</td>
<td>0.1103</td>
<td>0.000943</td>
<td>2.358</td>
</tr>
<tr>
<td>8</td>
<td>0.004348</td>
<td>0.05449</td>
<td>0.002245</td>
<td>2.806</td>
</tr>
<tr>
<td>16</td>
<td>0.004336</td>
<td>0.05032</td>
<td>0.003756</td>
<td>2.347</td>
</tr>
<tr>
<td>32</td>
<td>0.001982</td>
<td>0.029</td>
<td>0.00627</td>
<td>1.959</td>
</tr>
<tr>
<td>64</td>
<td>0.001089</td>
<td>0.01708</td>
<td>0.01025</td>
<td>1.601</td>
</tr>
<tr>
<td>128</td>
<td>0.0004688</td>
<td>0.004622</td>
<td>0.02656</td>
<td>2.075</td>
</tr>
<tr>
<td>256</td>
<td>0.0002266</td>
<td>0.005472</td>
<td>0.0996</td>
<td>3.891</td>
</tr>
<tr>
<td>512</td>
<td>0.0002761</td>
<td>0.003713</td>
<td>0.3208</td>
<td>6.266</td>
</tr>
<tr>
<td>1024</td>
<td>0.0001847</td>
<td>0.004186</td>
<td>1.309</td>
<td>12.78</td>
</tr>
</tbody>
</table>

For sufficiently large \(L\), the systematic error exceeds the statistical error, so we can estimate the order of convergence. The values of \(c_{\text{tot}} \pm c_{\text{stat}}\) are plotted against \(N\) for both algorithms in Figure 5a. The solid line has a gradient of \(1/N\), showing the classic Monte Carlo convergence order.

Notice that both algorithms exhibit the standard \(1/N\) convergence order, but the single particle method has a slighter lower constant, meaning the convergence to a given error tolerance is slightly faster.
3.7 Computational Efficiency

In order to analyse the computational efficiency of the method, we plot the total error as a function of CPU time in Figure 5b. As we might expect, CPU time increases linearly with decreasing $c_{\text{tot}}$, hence we achieve the typical power law relationship characteristic of a Monte Carlo algorithm. The lines are the those of best fit through the data. Figure 5b shows that whilst the single particle method is slower to reach higher tolerances, it becomes faster than mass flow as the tolerances decrease. We conclude that the single particle method takes a longer time to reach steady-state, but has better resolution for low $N$.

In fact, as Figures 6a & 6b (along with Table 1) show, the single particle method has adequate performance even for a low number of stochastic field particles, $N$. This is important for CFD applications, because when tracking particles through a large number of cells, in implies that typically very few particles per cells will suffice. The steady-state distribution of particles being represented by the total collection of all particles in all cells in the computational domain, making the method highly efficient in such applications.

4 Spatially Inhomogeneous Geometries

As we have seen, the single particle method gives us the steady-state distribution of particles in a zero-dimensional (or completely homogeneous) system. However, using an approach outlined in Smith et al. [39], we consider these zero-dimensional systems to be cells, $\Omega_j$, which we can network together to form a quasi-one dimensional
geometry, the inflowing distribution of particles being given by the steady-state distribution of particles in the previous cell. This will afford us full spatial resolution in (at least) one dimension. The construction of this one-dimensional geometry from a string of zero-dimensional cells is illustrated in Figure 7.

**Figure 7:** One-dimensional geometry. The geometry is constructed by linking $J$ cells $\Omega_j$, with the inflow rate of each cell determined by the local velocity in that cell along with its grid spacing. The first cell, $\Omega_1$, gives the boundary condition, and is equivalent to the distribution of inflowing particles, $m^{in}$, in the zero-dimensional case.

### 4.1 Implementation

The code was modified to read in a grid of cells, along with their coordinates and the velocity of the background fluid at that point. The distribution of inflowing particles, $m^{in}(x)$, becomes a boundary condition and the in and outflow rates of cell $1 \leq j \leq J$ are prescribed by the fluid velocity and grid spacing at that cell (suitably nondimensionalised) $1/\alpha_j = u_j/\delta r_j = 1/\beta_j$.

We have two viable strategies for the implementation of this algorithm: (i) we can
iterate to steady-state in each cell before transporting to the next, the distribution of inflowing particles in the next cell being given by the steady-state distribution in the previous cell; or (ii) we can transport the particles from each cell and iterate until we have global convergence across the entire domain.

The results of a simulation adopting strategy (i) (cell-wise convergence), with the obvious cell ordering, with a domain discretised into $J = 100$ cells are illustrated in Figure 8a. We see that the variance seems to increase as we pass through a larger number of cells. This is for two reasons. As we have shown in Figure 4, for a single cell simulation, the variance of the solution increases with increasing inflow rate.

An additional source of error is that $m_k$ for a given cell is the steady-state distribution of $m_k$ in the previous cell, so any error in this will be propagated to the next cell. To see why this is, we can express the steady-state solution in the first cell calculated using the first strategy as the true steady-state, plus some small error. Performing a Taylor expansion, we can see how this error propagates to the second cell, and so on, throughout the computational domain. We found that the error grows at least linearly with number cells, which is a most unsatisfactory situation. To alleviate this problem, we adopt the second strategy: particle tracking.

### Algorithm 3: Spatially Dependent Single Particle Method (SPM)

```
input : $N, L, C, K, \text{iterMax}, m^m$, initial particle distribution
\begin{align*}
\{x_1, \ldots, x_N\};
\end{align*}
output : Steady-state PSD $(x(j), i \in N)$, first $K$ moments $m_k(j)$ in cell $j$, $1 \leq j \leq J$;
```

```
begin
\begin{align*}
\Phi_p & \leftarrow 1; \\
y_p,i, y_f,i & \leftarrow 1, \forall i \in N; \\
\text{iter} & \leftarrow 0; \\
\text{while iter} < \text{iterMax} \text{ do}
\end{align*}
```

#### Inflow
```
A test particle enters the system.
\begin{align*}
\text{Generate a test particle } x \sim m^m;
\end{align*}
```

```
\begin{align*}
\text{while } j \leq J \text{ do}
\end{align*}
```

#### Replacement
```
\begin{align*}
\Phi_f & \leftarrow 0; \\
\text{testParticleActive} & \leftarrow \text{True}; \\
\text{while testParticleActive} \text{ do}
\end{align*}
```

#### Coagulation
```
\begin{align*}
\text{Choose a field particle index } i \text{ (collision partner) with probability } K(x, y_p,i) \Phi_p N \rho(x) y_p,i, \\
\text{and coagulate with the test particle: } x \leftarrow x + y_p,i;
\end{align*}
```

#### Outflow
```
\begin{align*}
\text{The test particle leaves the system.}
\end{align*}
```

```
\begin{align*}
\text{Under-relax the field particles’ mass density: } & \Phi_f \leftarrow \epsilon_1 \Phi_f + (1 - \epsilon_1) \Phi_p; \\
\text{Update the field particles array: } & y_p,i \leftarrow y_f,i, \forall i; \\
\text{testParticleActive} & \leftarrow \text{False};
\end{align*}
```

```
\begin{align*}
\text{update} & : j \leftarrow j + 1; \\
\text{update} & : \text{iter} \leftarrow \text{iter} + 1;
\end{align*}
```

```
\text{For } n \in \mathbb{N}; x = \frac{\epsilon_2 \rho(x)}{\Phi_p} \\
\text{with probability } \epsilon_2 \rho(x)/\Phi_p \text{ goto Coagulation else goto Outflow;}
```

```
\text{Two particles collide and coagulate.}
```

```
\text{Choose a field particle index } i \text{ (collision partner) with probability } K(x, y_p,i) \Phi_p N \rho(x) y_p,i, \\
\text{and coagulate with the test particle: } x \leftarrow x + y_p,i;
```

```
\text{Under-relax the field particles’ mass density: } \Phi_f \leftarrow \epsilon_1 \Phi_f + (1 - \epsilon_1) \Phi_p; \\
\text{Update the field particles array: } y_p,i \leftarrow y_f,i, \forall i; \\
\text{testParticleActive} \leftarrow \text{False};
```

```
\begin{align*}
\text{update} & : j \leftarrow j + 1; \\
\text{update} & : \text{iter} \leftarrow \text{iter} + 1;
\end{align*}
```
```
Figure 8: Comparison between distribution of steady-state values for moments of mass density calculated using SPM solver with analytical solution in a 1D geometry of length 1 unit (discretised into 100 cells) with \( K(x,y) = \frac{1}{2}xy \) in a uniform background velocity field of \( u = 1 \), using two implementation strategies.
In this strategy, rather than considering the processes affecting the test particle in each cell in isolation, we instead track the test particle through the entire domain. This approach leads to Algorithm 3.

We use this implementation to reproduce the simulations of Figure 8a. The results are illustrated in Figure 8b. Notice that as the number of cells passed through increases, the variance is now broadly constant for the lower moments, and increases much more modestly for the higher moments.

We perform a convergence study, this time varying the number of cells in addition to the number of stochastic particles $N$, whilst keeping the number of runs constant at $L = 100$. This gives the $L_2$-error (over the first three non-trivial moments) surface illustrated in Figure 9. Notice that when the number of cells is sufficiently large, just a few particles suffice to capture the moments with small error. In particular, when $N = 1$, we can reduce the error by simply tracking the particle through many cells. This is because we are using the individual particles in all cells to represent our overall distribution, so the steady-state PSD is effectively represented by all particles in the entire domain.

### 4.2 Time-Dependent Case

The algorithm introduced in this section effectively solves a one-dimensional advection equation in steady-state. Notice that if we solve this equation in a uniform background velocity of $u = 1$, then under the interchange of space and time coordinates, the equation describes moment transport in a time-dependent zero-dimensional system. We may therefore use our quasi-one dimensional steady-state stochastic solver to resolve the time evolution in a zero-dimensional system. Essentially, we are considering our grid to discretise physical time, and we reach a steady-state in “pseudo time” at each physical time step as outlined in Smith et al. [39]. This idea could
be very useful in coupling to CFD, for example, it could be used to solve particle
dynamics in complex geometries by post-processing streamlines calculated in CFD
[1, 4]. The particle size distributions in a full three-dimensional geometry could then
by reconstructed using statistical techniques, for example, kernel density estimation.

5 An Improved Algorithm

5.1 Improving the Convergence Behaviour

We have seen that the algorithm certainly seems to converge to the steady-state
solution, but as Figure 6 illustrated, the character of that convergence is somewhat
erratic. In order to establish the reason for this behaviour, we studied the algo-

rithm in detail and attempted to introduce additional terms into (7) to improve the
convergence properties whilst still reducing to (2) in the steady-state limit.

The reason for the poor convergence of the sample paths can be understood by
starting from the steady-state distribution in (7) (i.e., the distribution which solves
the limiting case). The algorithm takes a single particle from \( m^\text{in} \) and looks at events
affecting it. If any field particle is replaced, then the jump is \( O(1/N) \), whilst over
the lifetime of the particle, the change in the distribution is \( O(1) \). The situation for
\( \Phi \) is potentially even more dramatic. The updated \( \Phi \) is likely to be \( Q_{\text{in}} \beta(x)/\alpha \), i.e.,
the jumps are very erratic depending upon the particle being considered.

We found that by introducing additional jump processes to describe field particle
sampling, \( M \)- and \( \Phi \)-update events, in addition to the usual test particle inflow,
coagulation and field particle replacement events, we were able to improve the con-
vergence behaviour. The convergence was further enhanced by adapting the rates of
particle swap, sample and \( \Phi \)-update events to depend upon a discrete time param-
eter \( n_t \), in such a way that the probability of such events occurring tends to zero as
\( n_t \to \infty \).

The mathematical analysis which led to these findings and the derivation of the
jumps are given in §7.1. Here we present a distilled summary of the equations. Let
\( Z = (x, P, M, \Phi) \) and consider the generator

\[
G(\Psi, t)(Z) = \frac{\Phi}{N} \sum_{i=1}^{N} \frac{K(x, z_i)}{z_i} (\Psi(J^r_i(Z)) - \Psi(Z)) \\
+ \frac{Q_{\text{in}}}{M \alpha} \int_{\mathbb{R}} (\Psi(J_{\text{in}}(Z, w)) - \Psi(Z)) \frac{m^\text{in}(dw)}{Q_{\text{in}}} \\
+ \frac{M}{\Phi \tau n_t} \sum_{i=1}^{N} (\Psi(J^r_i(Z)) - \Psi(Z)) \\
+ \frac{1}{\beta(x)n_t} \sum_{i=1}^{N} (\Psi(J^\beta_i(Z)) - \Psi(Z))
\]
\[
+ \frac{1}{\beta(x)} (\Psi(J_M(Z)) - \Psi(Z)) \\
+ \frac{N}{\tau n_t} (\Psi(J_\Phi(Z)) - \Psi(Z))
\]

with jumps \( J \) defined by

\[
\begin{align*}
J^c_i(Z) &= (x + z_i, P, M, \Phi), \\
&\text{(coagulation)} \\
J^{in}_i(Z, w) &= (w, P, M, \Phi), \\
&\text{(inflow)} \\
J^s_i(Z) &= (x, P + \frac{1}{N}(\delta_x - \delta_{zi}), M, \Phi), \\
&\text{(swap)} \\
J^\phi_i(Z) &= (z_i, P, M, \Phi), \\
&\text{(sample)} \\
J_M(Z) &= (x, P, Q_{in}\beta(x)/\alpha, \Phi), \\
&\text{(}M\text{ update}) \\
J_\Phi(Z) &= (x, P, M, (1 - 1/N)\Phi + M/N). \\
&\text{(}\Phi\text{ update})
\end{align*}
\]

The dynamics of our stochastic process is then given by

\[
\Psi(Z_t) = \Psi(Z_0) + \int_0^t \mathcal{G}(\Psi, s)(Z_s) \, ds.
\]  \hspace{1cm} (21)

The total rate is given by

\[
\varrho(z, t) = \frac{\Phi}{N} \sum_{i=1}^N \frac{K(x, z_i)}{z_i} + \frac{Q_{in}}{M\alpha} + \frac{MN}{\Phi\tau n_t} + \frac{N}{\beta(x)n_t} + \frac{1}{\tau} + \frac{N}{\tau n_t}.
\]

This can be used to define a transition measure and to construct a Martingale, enabling the powerful theorems of functional analysis [16] to be brought to bear on the convergence of the stochastic process. We will not pursue this path here, however, but will instead construct the alternative algorithm to which these jumps and corresponding rates naturally lead.

It should be noted that we have some degree of freedom over the actual implementation. For example, rather than doing a large number of little jumps, we could instead do one large jump. Alternatively, we may use under-relaxation since

\[
\frac{d\Phi}{dt} = \frac{1}{\tau} (M - \Phi) = \frac{1}{\varepsilon_1\tau} (\varepsilon_1 M + \varepsilon_2 \Phi) - \Phi,
\]

where \( \varepsilon_1 + \varepsilon_2 = 1 \). Hence, instead of doing a jump \( \Phi \mapsto M \) with rate \( 1/\tau \), we can perform jumps \( \Phi \mapsto \varepsilon_1 M + \varepsilon_2 \Phi \) at the more frequent rate of \( 1/\varepsilon_1\tau \).

### 5.2 Updated Numerical Results

The jumps and rates of the previous section, suitably modified to account for the freedom of choice, suggests a version of the single particle method with improved convergence behaviour. This improved version is given in Algorithm 4.
Algorithm 4: The Single Particle Method Improved (SPMIm)

input: \(N, L, K, \text{iterMax}, m^n, \alpha, \beta(x), \tau, \) initial particle distribution \((x_1, \ldots, x_N)\);
output: Steady-state PSD \(\{x_i, i \in N\}\), first \(K\) moments \(m_k\);

begin
\(x \leftarrow 1, \Phi \leftarrow Q_{\text{in}}(x)/\alpha, M \leftarrow \Phi, n_0 \leftarrow 1;\)
iter \(\leftarrow 0;\)
while iter < iterMax do
\(\) calculate: rates
\(\rho_{\text{coag}} \leftarrow \frac{\sum_{i=1}^{N} K(x, x, z_i)\Phi}{M N};\)
\(\rho_{\text{M}} \leftarrow \frac{1}{M N};\)
\(\rho_{\text{Φ}} \leftarrow \frac{1}{\beta(x)[x]|M|};\)
\(\rho_{\text{swap}} \leftarrow \frac{M Q_{\text{in}}}{N};\)
\(\rho_{\text{samp}} \leftarrow \frac{N}{M N};\)
\(\rho_{\text{in}} \leftarrow Q_{\text{in}}/N;\)
and the total rate
\(\varphi \leftarrow \rho_{\text{coag}} + \rho_{\text{M}} + \rho_{\text{Φ}} + \rho_{\text{swap}} + \rho_{\text{samp}} + \rho_{\text{in}};\)
Choose a stochastic event \(e\) with probability \(\varphi_e/\varphi\) and goto \(e \in \{\text{Coagulation, M-Jump, Φ-Jump, Swap, Sample, Inflow}\};\)
Coagulation
Two particles collide and coagulate. Choose a field particle index \(i\) (collision partner) with probability
\(K(x, x, z_i)\Phi/\varphi_{\text{coag}} z_i;\)
and coagulate with the test particle: \(x \leftarrow x + z_i;\)
M-Jump
update \(M \leftarrow Q_{\text{in}}(x)/\alpha;\)
Φ-Jump
update \(\Phi \leftarrow \Phi + \frac{M}{N};\)
Swap
Replace a uniformly chosen field particle \(i\) with \(x: z_i \leftarrow x;\)
Sample
Replace \(x\) with a uniformly chosen field particle \(i: x \leftarrow z_i;\)
Inflow
A test particle enters the system. Generate a test particle \(x \sim m^n;\)
update iter \(\leftarrow \text{iter} + 1;\)

As illustrated in Figure 10, Algorithm 4 shows considerably less erratic behaviour, and generally has smoother convergence properties than the original algorithm. We show the full convergence behaviour in Table 2.

Table 2: Numerical convergence study for the improved single particle method (SPMIm). CPU times are in seconds \((N \times L = 2^{13}10^2 \sim 8.1 \times 10^6)\).

<table>
<thead>
<tr>
<th>(N)</th>
<th>(c_{\text{stoch}})</th>
<th>(c_{\text{tot}})</th>
<th>(t_{\text{CPU}}/L)</th>
<th>(t_{\text{CPU}}/N \times 10^6)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.02341</td>
<td>0.1495</td>
<td>0.003759</td>
<td>37.59</td>
</tr>
<tr>
<td>2</td>
<td>0.02765</td>
<td>0.145</td>
<td>0.0063</td>
<td>31.50</td>
</tr>
<tr>
<td>4</td>
<td>0.01886</td>
<td>0.09982</td>
<td>0.003221</td>
<td>8.053</td>
</tr>
<tr>
<td>8</td>
<td>0.009553</td>
<td>0.05333</td>
<td>0.003261</td>
<td>4.077</td>
</tr>
<tr>
<td>16</td>
<td>0.004705</td>
<td>0.03261</td>
<td>0.005334</td>
<td>3.334</td>
</tr>
<tr>
<td>32</td>
<td>0.004481</td>
<td>0.029</td>
<td>0.003256</td>
<td>1.018</td>
</tr>
<tr>
<td>64</td>
<td>0.001742</td>
<td>0.01096</td>
<td>0.003203</td>
<td>0.5005</td>
</tr>
<tr>
<td>128</td>
<td>0.0006832</td>
<td>0.004622</td>
<td>0.008438</td>
<td>0.6592</td>
</tr>
<tr>
<td>256</td>
<td>0.0003181</td>
<td>0.004294</td>
<td>0.01564</td>
<td>0.6111</td>
</tr>
<tr>
<td>512</td>
<td>0.0003323</td>
<td>0.002267</td>
<td>0.04882</td>
<td>0.9535</td>
</tr>
<tr>
<td>1024</td>
<td>8.898e-05</td>
<td>0.002891</td>
<td>0.1080</td>
<td>1.055</td>
</tr>
</tbody>
</table>

We illustrate this behaviour graphically in comparison to the original algorithm in Figures 11a & 11b. Notice that both algorithms exhibit very similar performance in terms of convergence order, but the improved algorithm has a higher computational efficiency, in addition to its smoother convergence, as illustrated by the steeper
Figure 10: Illustration of the convergence behaviour of the improved single particle method (SPMIm) in comparison to the original method. The simulation has been run with a low number of stochastic particles ($N = 2^4$) and a single run so as not to obfuscate the nature of the convergence.

Figure 11: Comparison between computational efficiencies of the improved single particle method (SPMIm) and with the original method (SPM). The straight lines in Fig. 11a have slope $-1$. The lines in 11b are those of best fit.
gradient of the solid (green) line in Figure 11b, which shows that error reduces more rapidly with increasing CPU time than the for the original method.

6 Conclusions

This paper described an existing stochastic algorithm which is capable of resolving population balances with a large number of dimensions. The mathematical foundations of the algorithm were developed and its convergence behaviour was investigated under a range of conditions. The algorithm was extended to one dimensional geometries with non-uniform flow fields. Whilst it seems that the algorithm converges to the solution in expectation, the actual convergence of the sample paths were found to be of high variance. This observation inspired the introduction of additional stochastic jump processes to smooth the size of these jumps and so improve the convergence character and overall efficiency of the algorithm.

In CFD applications, the efficiency of the algorithms which solve the underlying PBEs is paramount, because they will typically be called a great many times throughout each CFD fluid flow iteration. Studies were performed in order to establish values of key parameters to strike the correct balance between computational efficiency and numerical resolution to render the algorithm practicable for CFD coupling. It was found that in typical applications, where particles are tracked through a very large number of cells, it sufficed to have very few stochastic particles per cell (sometimes just one) in order to accurately resolve the particle size distributions, making the method highly efficient, and therefore potentially of considerable use in modelling particle processes in complex geometries.

7 Appendix

7.1 Mathematical Analysis of The SPM

We have seen that the algorithm certainly seems to converge to the steady-state solution, but as Figure 6 illustrated, the character of that convergence is somewhat erratic. We shall now attempt to establish why this could be. Let us revisit (7). We introduce a measure $\mu$ and a point $\omega$, which will serve as a combined source and sink, corresponding to a place which particles can come from or go to, whilst preserving the total mass of the system. We then recast the equation into the following form

$$\frac{\partial m(x,t)}{\partial t} = \int K(x, x-y) \phi(y,t)m(x-y, t)\mu(dy) - \int K(x,y) \phi(y,t)m(x, t)\mu(dy) - \frac{m(x,t)}{\theta(x)} + \left( \delta_\omega(x) + \frac{\phi(x,t)}{\Phi(t)} - \frac{m(x,t)}{\theta(t)} \right) \int m(y,t) \frac{\mu(dy)}{\theta(y)}$$
\[+ \frac{m_{in}(x)}{V} - \frac{Q_{in}}{V} \delta_\omega(x).\]  

(22)

Notice this differs by the addition of the last three terms. The coagulation and in and outflow rates of our new point are zero: \(1/\theta(\omega) = K(\omega, \cdot) = m_{in}(\omega) = 0\). Noting also that \(\delta_\omega(\omega) = 1\) and evaluating (22) at \(x = \omega\), we find

\[
\frac{dm(\omega, t)}{dt} = \left(1 + \frac{\phi(\omega, t)}{\Phi(t)} - \frac{m(\omega, t)}{M(t)}\right) \int_N \frac{m(y, t)}{\theta(y)} \mu(dy) - \frac{Q_{in}}{V},
\]

(23)

and upon integration

\[
\frac{dM}{dt} = \frac{Q_{in}}{V} - \int_N \frac{m(y, t)}{\theta(y)} \mu(dy).
\]

(24)

We now augment the set over which we have been integrating with the new point \(\omega\) and call the new set \(\mathcal{N} = \mathcal{N} \cup \{\omega\}\). We then have

\[
\mathcal{M} = \int_{\mathcal{N}} m(x, t) \mu(dx) = \int_{\mathcal{N}} m(x, t) \mu(dx) + m(\omega, t) = M + m(\omega, t),
\]

and thus, using (23) and (24), we find

\[
\frac{d\mathcal{M}}{dt} = \frac{dM}{dt} + \frac{dm(\omega, t)}{dt} = \left(\frac{\phi(\omega, t)}{\Phi(t)} - \frac{m(\omega, t)}{M(t)}\right) \int_N \frac{m(y, t)}{\theta(y)} \mu(dy).
\]

If we now define

\[
\phi(\omega, t) = \frac{m(\omega, t) \Phi(t)}{M(t)},
\]

then \(d\mathcal{M}/dt = 0\), so \(\mathcal{M}\) is constant, which we may take to be 0. Therefore, \(m(\omega, t) = -M(t)\) and \(\phi(\omega, t) = -\Phi(t)\). We therefore have

\[
\int_{\mathcal{N}} \phi(x, t) \mu(dx) = \int_{\mathcal{N}} \phi(x, t) \mu(dx) + \phi(\omega, t) = \Phi(t) + \phi(\omega, t) = 0,
\]

and similarly

\[
\int_{\mathcal{N}} m(x, t) \mu(dx) = 0.
\]

Let us define two new processes

\[
X_t(x) = \frac{m(x, t)}{M(t)} \quad \text{and} \quad \psi_t(x) = \frac{\phi(x, t)}{\Phi(t)}.
\]

We find \(X_t(\omega) = -1 = \psi_t(\omega)\) and

\[
\int_{\mathcal{N}} X_t(x) \mu(dx) = 0 = \int_{\mathcal{N}} \psi_t(x) \mu(dx).
\]
We wish to establish the PDE satisfied by $X_t$. Appealing to (22) and (24), we find

\[
\frac{\partial X_t}{\partial t} = \frac{\partial}{\partial t} \left( \frac{m(x,t)}{M(t)} \right) = \frac{1}{M} \frac{\partial m}{\partial t} - \frac{X_t}{M} \frac{dM}{dt}
\]

\[
= \int_N \frac{K(x,x-y)}{y} \phi(y,t) X_t(x-y) \mu(dy) - \int_N \frac{K(x,y)}{y} \phi(y,t) X_t(x) \mu(dy)
\]

\[
- \frac{X_t(x)}{\theta(x)} + \left( \delta_\omega(x) + \frac{\phi(x,t)}{\Phi(t)} - X_t(x) \right) \int_N \frac{X_t(y)}{\theta(y)} \mu(dy)
\]

\[
+ \frac{m^{in}(x)}{MV} - \frac{Q_{in}}{MV} \delta_\omega(x) - X_t \left( \frac{Q_{in}}{MV} - \int_N \frac{X_t(y)}{\theta(y)} \mu(dy) \right)
\]

\[
= \Phi \int_N \frac{K(x,x-y)}{y} \psi_t(y) X_t(x-y) \mu(dy) - \Phi \int_N \frac{K(x,y)}{y} \psi_t(y) X_t(x) \mu(dy)
\]

\[
- \frac{X_t(x)}{\theta(x)} + \left( \delta_\omega(x) + \psi_t(x) \right) \int_N \frac{X_t(y)}{\theta(y)} \mu(dy) - \frac{X_t(x)}{\theta(x)}
\]

\[
+ \frac{m^{in}(x)}{MV} - \frac{Q_{in}}{MV} \left( \frac{m^{in}}{Q_{in}} - \delta_\omega - X_t \right). \tag{25}
\]

Notice that the terms quadratic in $X_t$ have cancelled, which justifies the addition of the $-m(x,t)/M(t) \int_N m(y,t)/\theta(y) \mu(dy)$ term to (22). We also see that when $x = \omega$ the right hand side is 0 and the left hand side is $\partial X_t(\omega)/\partial t = \partial/\partial t(-1) = 0$, so the equation is consistent.

Now, when $x \in \mathbb{N}$, we have

\[
\frac{\partial X_t}{\partial t} = \Phi \int_N \frac{K(x,x-y)}{y} \psi_t(y) X_t(x-y) \mu(dy) - \Phi \int_N \frac{K(x,y)}{y} \psi_t(y) X_t(x) \mu(dy)
\]

\[
- \frac{X_t(x)}{\theta(x)} + \psi_t(x) \int_N \frac{X_t(y)}{\theta(y)} \mu(dy) - \frac{X_t(x)}{\theta(x)} + \frac{m^{in}(x)}{MV} - \frac{Q_{in}}{MV} \left( \frac{m^{in}}{Q_{in}} - X_t \right), \tag{26}
\]

and

\[
\frac{dM}{dt} = \frac{Q_{in}}{V} - M \int_N \frac{X_t(y)}{\theta(y)} \mu(dy). \tag{27}
\]

We need to ensure that $\Phi(t) - M(t) \to 0$ and $\psi_t - X_t \to 0$ as $t \to \infty$. With (8) in mind, let us suppose that

\[
\frac{\partial \phi(x,t)}{\partial t} = \frac{1}{\tau} \left( m(x,t) - \phi(x,t) \right).
\]
Then when \( x = \omega \), we find
\[
\frac{d\Phi}{dt} = \frac{1}{\tau} (M - \Phi),
\]
recalling that \( \phi(\omega, t) = -\Phi(t) \) and \( m(\omega, t) = -M(t) \). We therefore find that
\[
\frac{\partial \psi}{\partial t} = \frac{1}{\Phi} \frac{\partial \phi}{\partial t} - \psi = \frac{1}{\tau} \frac{M}{\Phi} (X_t - \psi_t),
\]
(29)
The term \( \tau \Phi/M \) is the average time between events, so is the rate at which we must update the field particles mass density. If there exists a steady-state solution to the problem, then \( \phi(x, t) \to m(x) \) as \( t \to \infty \). Substitution into (22) shows that, for \( x \in \mathbb{N} \)
\[
\int_{\mathbb{N}} \frac{K(x, x-y)}{y} m(y)m(x-y)\mu(dy) - \int_{\mathbb{N}} \frac{K(x,y)}{y} m(y)m(x)\mu(dy) + \frac{m_{in}}{V} - \frac{m(x)}{\theta(x)} = 0
\]
as required. This explains why the convergence of the sample paths is quite poor (qv. Figure 6). Suppose our initial distribution \( m(x, 0) \) solves the limiting case exactly. The current algorithm takes a single particle from the source distribution and looks at events affecting it. If any field particle is replaced (which happens at a rate proportional to the residence time, and therefore with high probability), then we have a jump of order \( 1/N \), though the change in the distribution whilst the test particle is active is \( O(1) \). This is clearly an unsatisfactory situation.

The case for the field particle mass density, \( \Phi \), is potentially even worse. The change in \( \Phi \) is likely to be of order \( Q_{in}\theta(x)/V \), which could be very erratic (depending upon the particle being considered).

To overcome these limitations, we have two options: (i) We can content ourselves with convergence of the expectation, or (ii) we can introduce an averaging procedure, e.g., under-relaxation, or we can track
\[
\tilde{\Phi}_{n+1} = \frac{n\tilde{\Phi}_n + \Phi_n}{n+1},
\]
(30)
so that
\[
\tilde{\Phi}_n = \frac{1}{n} \sum_{i=1}^{n-1} \Phi_i.
\]
For \( \psi_t \), we define sequences of Markov jump processes
\[
\tilde{P}_n = \frac{1}{N} \sum_{i=1}^{n} \delta_{y_i} \quad \text{and} \quad P_n = \frac{1}{N} \sum_{i=1}^{n} \delta_{z_i},
\]
and in a similar way track,
\[
\tilde{P}_{n+1} = \frac{n\tilde{P}_n + P_n}{n+1},
\]
(31)
then
\[
\tilde{P}_{n+1} - \tilde{P}_n = \frac{P_n - \tilde{P}_n}{n+1} = \frac{1}{N(n+1)} \sum_{i=1}^{n} (\delta_{z_i} - \delta_{y_i}),
\]
which indicates that we should replace $y_i$ by a uniformly chosen $z_j$ with probability $1/(N(n + 1))$. This construction will ensure that the sample paths of $\bar{\Phi}_n \bar{P}_n$ will converge to the measure-valued solution to our original steady-state equation (2). As in §2, we wish to construct the generators and jumps of the new processes, so that we can modify Algorithm 1 to account for them.

The analysis explains why the convergence of the sample paths is quite poor. Suppose our initial distribution $m(x, 0)$ solves the limiting case exactly. The SPM takes a single particle from the source distribution and looks at events affecting it. If any field particle is replaced (which happens at a rate proportional to the residence time, and therefore with high probability), then we have a jump of order $1/N$, though the change in the distribution whilst the test particle is active is $O(1)$. This is clearly an unsatisfactory situation. The case for the field particle mass density, $\Phi$, is potentially even worse. The change in $\Phi$ is likely to be of order $Q_{in} \beta/\alpha$, which is likely very erratic (depending upon the particle being considered).

To overcome these limitations, we have two options: (i) We can content ourselves with convergence of the expectation, or (ii) we can introduce an averaging procedure, e.g., under-relaxation, or we can track

$$\bar{\Phi}_{n+1} = \frac{n\bar{\Phi}_n + \Phi_n}{n + 1},$$

so that

$$\bar{\Phi}_n = \frac{1}{n} \sum_{i=1}^{n-1} \Phi_i.$$  

For $\psi$, we define sequences of Markov jump processes

$$\bar{P}_n = \frac{1}{N} \sum_{i=1}^{n} \delta_{y_i} \quad \text{and} \quad P_n = \frac{1}{N} \sum_{i=1}^{n} \delta_{z_i},$$

and in a similar way track,

$$\bar{P}_{n+1} = \frac{n\bar{P}_n + P_n}{n + 1},$$

then

$$\bar{P}_{n+1} - \bar{P}_n = \frac{P_n - \bar{P}_n}{n + 1} = \frac{1}{N(n + 1)} \sum_{i=1}^{n} (\delta_{z_i} - \delta_{y_i}),$$

which indicates that we should replace $y_i$ by a uniformly chosen $z_j$ with probability $1/(N(n + 1))$. This construction will ensure that the sample paths of $\bar{\Phi}_n \bar{P}_n$ will converge to the measure-valued solution to our original steady-state equation (2). As in §2, we wish to construct the generators and jumps of the new processes, so that we can modify Algorithm 1 to account for them.

### 7.2 Jumps & Rates

We establish the weak form of (26) in the usual way by multiplying by an arbitrary compactly supported test function $f_1 \in C^\infty$ and integrating. Using a similar method
to that used to establish (3), and recalling that \( \langle \mu, f \rangle = \int f(y) \mu(dy) = \mu(f) \), we find
\[
\frac{d}{dt} \langle \delta_x, f_1 \rangle = \left\langle \Phi \int \frac{K(x, y)}{y} P_t(dy)(\delta_{x+y} - \delta_x) + \frac{P_t - \delta_x}{\theta(x)} + \frac{Q_{in}}{MV} \left( \frac{m^{in}}{Q_{in}} - \delta_x \right), f_1 \right\rangle,
\]

with \( m^{in} \) now regarded as a measure. We also have the weak forms of (27), (28), (29), (32), (33):
\[
\begin{align*}
\frac{d}{dt} \langle M, f_2 \rangle &= \left\langle \frac{Q_{in} - M}{\theta(x)}, f_2 \right\rangle, \\
\frac{d}{dt} \langle \Phi, f_3 \rangle &= \left\langle \frac{1}{\tau} (M - \Phi), f_3 \right\rangle, \\
\frac{d}{dt} \langle P, f_4 \rangle &= \left\langle \frac{1}{\tau} \Phi (\delta_x - P), f_4 \right\rangle, \\
\frac{d}{dt} \langle \tilde{\Phi}, f_5 \rangle &= \left\langle \frac{\Phi - \tilde{\Phi}}{t}, f_5 \right\rangle, \\
\frac{d}{dt} \langle \tilde{P}, f_6 \rangle &= \left\langle \frac{P - \tilde{P}}{t}, f_6 \right\rangle.
\end{align*}
\]

Notice that \( f_2, f_3 \) and \( f_5 \) are constants, so \( \langle \cdot, f_{2,3,5} \rangle \) simply corresponds to scalar multiplication.

From (34), we see
\[
\frac{d\tilde{\Phi}}{dt} = \frac{\Phi - \tilde{\Phi}}{t},
\]

which, using \( t \) as an integrating factor gives
\[
\tilde{\Phi}(t) = \frac{1}{t} \int_0^t \Phi(s) \, ds.
\]

Expressing \( \tilde{P}_t \) and \( P_t \) as a sum of delta functions gives us
\[
\begin{align*}
\frac{d}{dt} \langle \delta_x, f_1 \rangle &= \left\langle \Phi \frac{N}{\theta(x)} \sum_{i=1}^N K(x, z_i) (\delta_{x+z_i} - \delta_x) + \frac{1}{N\theta(x)} \sum_{i=1}^N (\delta_{z_i} - \delta_x) + \frac{Q_{in}}{MV} \left( \frac{m^{in}}{Q_{in}} - \delta_x \right), f_1 \right\rangle, \\
\frac{d}{dt} \langle P, f_4 \rangle &= \left\langle \frac{1}{N} \sum_{i=1}^N (\delta_x - \delta_{z_i}), f_4 \right\rangle, \\
\frac{d}{dt} \langle \tilde{P}, f_6 \rangle &= \left\langle \frac{1}{N^2} \sum_{i=1}^N \sum_{j=1}^N (\delta_{z_i} - \delta_{y_j}), f_6 \right\rangle.
\end{align*}
\]

Let us denote the weak form of \( F \) by \( \tilde{F} \), and consider
\[
\tilde{F}(Z) = \langle \delta_x, f_1 \rangle + \langle M, f_2 \rangle + \langle \Phi, f_3 \rangle + \langle P, f_4 \rangle + \langle \tilde{\Phi}, f_5 \rangle + \langle \tilde{P}, f_6 \rangle.
\]
\[ f_1(x) + Mf_2 + \Phi f_3 + \frac{1}{N} \sum_{i=1}^{N} f_4(z_i) + \tilde{\Phi} f_5 + \frac{1}{N} \sum_{i=1}^{N} f_6(y_i). \]

The new generator is then

\[ \mathcal{G}(F, t)(Z) = \frac{\Phi}{N} \sum_{i=1}^{N} \frac{K(x, z_i)}{z_i} (F(J^c_i(Z)) - F(Z)) \]

\[ + \frac{1}{N \theta(x)} \sum_{i=1}^{N} \left( F(J^s_i(Z)) - F(Z) \right) \]

\[ + \frac{Q_{in}}{MV} \int_{N} (F(J_{in}(Z, w)) - F(Z)) \frac{m_{in}(dw)}{Q_{in}} \]

\[ + \frac{M}{N \Phi} \sum_{i=1}^{N} (F(J^r_i(Z)) - F(Z)) \]

\[ + \frac{1}{\theta(x)} (F(J_M(Z)) - F(Z)) \]

\[ + \frac{1}{\tau} (F(J_{\Phi}(Z)) - F(Z)) \]

\[ + (F(J_{\Phi}(Z, t)) - F(Z)) \]

\[ + \frac{1}{Nt} \sum_{i=1}^{N} \sum_{j=1}^{N} (F(J_{i,j}(Z)) - F(Z)), \]

with jumps \( J \) defined by

\[ J^c_i(Z) = (x + z_i, P, M, \Phi, \tilde{\Phi}, \tilde{P}) \quad \text{(coagulation)} \]

\[ J^s_i(Z) = (z_i, P, M, \Phi, \tilde{\Phi}, \tilde{P}) \quad \text{(swap)} \]

\[ J_{in}(Z, w) = (w, P, M, \Phi, \tilde{\Phi}, \tilde{P}) \quad \text{(inflow)} \]

\[ J^r_i(Z) = (x, P + \frac{1}{N}(\delta_x - \delta_{z_i}), M, \Phi, \tilde{\Phi}, \tilde{P}) \quad (-) \]

\[ J_M(Z) = (x, P, Q_{in}\theta(x)/V, \Phi, \tilde{\Phi}, \tilde{P}) \quad \text{(M update)} \]

\[ J_{\Phi}(Z) = (x, P, M, \Phi, \tilde{\Phi}, \tilde{P}) \quad \text{(\( \Phi \) update)} \]

\[ J_{\Phi}(Z, t) = (x, P, M, \Phi, \tilde{\Phi} + \frac{\Phi - \tilde{\Phi}}{t}, \tilde{P}) \quad \text{((\( \Phi \) update)} \]

\[ J_{i,j}(Z) = (x, P, M, \Phi, \tilde{\Phi}, \tilde{P} + \frac{1}{N}(\delta_x - \delta_{y_j})). \quad \text{(\( \tilde{P} \) update)} \]

Hence

\[ \tilde{F}(Z_t) = \tilde{F}(Z_0) + \int_{0}^{t} \mathcal{G}(\tilde{F}, s)(Z_s) \, ds \quad (35) \]

and we have established the dynamics of the stochastic process.
The total rate is given by

\[ \varrho(z, t) = \frac{\Phi}{N} \sum_{i=1}^{N} \frac{K(x, z_i)}{z_i} + \frac{2}{\theta(x)} + \frac{Q_{in}}{MV} + \frac{MN}{\Phi \tau} + \frac{1}{\tau} + 1 + \frac{N}{t}. \]

This can be used to define a transition measure and to construct a Martingale, enabling the powerful theorems of functional analysis [16] to be brought to bear on the convergence of the stochastic process. We will not pursue this path here, however, but will instead construct the alternative algorithm to which these rates and jumps naturally lead.

It should be noted that we have some degree of freedom over the actual implementation. For example, rather than doing a large number of little jumps, we could instead do one large jump. Alternatively, we may use under-relaxation since

\[ \frac{d\Phi}{dt} = \frac{1}{\tau} (M - \Phi) = \frac{1}{\varepsilon_1 \tau} ((\varepsilon_1 M + \varepsilon_2 \Phi) - \Phi), \]

where \( \varepsilon_1 + \varepsilon_2 = 1 \). Hence, instead of doing a jump \( \Phi \rightarrow M \) with rate \( 1/\tau \), we can perform jumps \( \Phi \rightarrow \varepsilon_1 M + \varepsilon_2 \Phi \) at the more frequent rate of \( 1/\varepsilon_1 \tau \) (under-relaxation also enables us to remove the \( 1/t \) dependence, replacing it by \( \varepsilon_2 \varrho(z, t) \), so it occurs for all events, and no longer appears in the total rate expression). We can also use this freedom to modify some of the jumps so that they are \( N \) dependent, in such a way that they tend to 0 as \( N \) tends to infinity. For example, one realisation of this is to take \( \tau \sim N \), so

\[ J_\Phi(Z) = \Phi + \frac{M - \Phi}{N}. \]
References


