Advances in Water Resources 117 (2018) 115–119

Contents lists available at ScienceDirect

Advances in Water Resources

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

Short communication

On the accuracy of simulating mixing by random-walk particle-based mass-transfer algorithms☆

Michael J. Schmidt[a], Stephen D. Pankavich[a], David A. Benson[b,*]

[a] Applied Mathematics and Statistics, Colorado School of Mines, Golden, CO, 80401, USA
[b] Hydrologic Science and Engineering, Colorado School of Mines, Golden, CO, 80401, USA

ARTICLE INFO

Keywords:
Particle methods
Diffusion-reaction equation
Advection-diffusion-reaction equation
Numerical methods

ABSTRACT

Several algorithms have been used for mass transfer between particles undergoing advective and macro-dispersive random walks. The mass transfer between particles is required for general reactions on, and among, particles. The mass transfer is shown to be diffusive, and may be simulated using implicit, explicit, or mixed methods. All algorithms investigated are accurate to $\mathcal{O}(\Delta t)$. For N particles, the implicit and semi-implicit methods require inverse matrix solutions and $\mathcal{O}(N^3)$ calculations. The explicit methods use forward matrix solves and require only $\mathcal{O}(N^2)$ calculations. Practically, this means that naïve implementations with more than about 5000 particles run more reliably using explicit methods.

1. Introduction

The random-walk particle-tracking (RWPT) method was originally developed to simulate advective and dispersive transport of conservative or simply (linearly, instantaneously reversible) sorbing solutes (Labolle et al., 1996; Salamon et al., 2006). The method is attractive because it does not suffer from numerical dispersion or negative concentrations. The method was extended (Benson and Meerschaert, 2008) to nonlinearly interacting (bimolecular) chemical reactions by sequentially calculating the product of the probabilities of particle collision and thermodynamic reaction. The actual reactions were then performed using Monte Carlo methods and particles were "born" or "killed" by a comparison of reaction probability to randomly-generated numbers. The method was originally restricted to one, or a series of, bimolecular reactions (Benson et al., 2017; Bolster et al., 2016; Ding et al., 2012; Ding and Benson, 2015; Ding et al., 2017; Paster et al., 2013; 2014), because any particle was composed of only one chemical species. If the reaction is viewed as a mixing process, which may be denoted $2pA + 2qA \rightarrow (p+q)A + (p+q)A$, then particles can carry as many species as desired, and mass transfer of all species occurs between particles (Benson and Bolster, 2016). The mass transfer still only occurs between particles with some probability of collision, and these probabilities may be viewed as the weights associated with mass transfer. Benson and Bolster (2016) suggested that this collision-weighted mass transfer process follows a diffusion equation, although this was not shown rigorously. Furthermore, those authors chose a particular explicit mass transfer scheme, while later studies used an implicit scheme (Engdahl et al., 2017). Because both explicit and implicit schemes appear to work, it is plausible that a combination of these, similar to the Crank–Nicolson (C-N) algorithm, may increase accuracy. The purpose of this paper is to first develop a framework to investigate whether the "action" of the mass-transfer algorithm proposed by Benson and Bolster (2016) is actually diffusive. Once this diffusive nature is shown, the convergence rates of the several algorithms that immediately present themselves can be demonstrated.

2. Semi-implicit scheme

Among a total of N particles located at positions x_i, the collision-weighted mass exchange over a time step Δt is written

$$m_j^{k+1} - m_j^k = \sum_{i=1}^{N} \frac{1}{2}\left(m_i^{k+\ell} - m_j^{k+\ell}\right) P(|x_i - x_j|; \Delta t), \tag{1}$$

where the superscript denotes timestep (i.e., $m_j^k = m_j(k\Delta t)$), $\ell = 0, 1$, and $P_{ij} = P(|x_i - x_j|; \Delta t)$ is the probability of particle collision. This collision probability is shown to depend only upon the distance between particles, though it may have a more complicated form if non-isotropic or position dependent diffusion/dispersion paradigms are considered. Nonetheless, while the functional form of P may change, the mass transfer algorithm would be unaltered. For particles undergoing Brownian motion, this is the convolution of each particle's Gaussian location density, which is also Gaussian (see Benson and Meerschaert, 2008; Paster et al., 2014). If $\ell = 1$, the calculation is implicit, and if $\ell = 0$,

☆ The authors were supported by the National Science Foundation under awards EAR-1417145 and DMS-1614586.
* Corresponding author.
E-mail address: dbenson@mines.edu (D.A. Benson).

https://doi.org/10.1016/j.advwatres.2018.05.003
Received 2 March 2018; Received in revised form 3 May 2018; Accepted 5 May 2018
Available online 9 May 2018

the calculation is explicit (which may take several forms, for example, sequentially calculated or simultaneously calculated). A semi-implicit form is reminiscent of the Crank–Nicolson scheme and uses equal amounts of k and $k+1$ masses, so that we may write (1) as

$$m_j^{k+1} - m_j^k = \alpha \sum_{i=1}^{N} \frac{1}{2}\left(m_i^{k+1} - m_j^{k+1}\right)P_{ij} + (1-\alpha)\sum_{i=1}^{N}\frac{1}{2}\left(m_i^k - m_j^k\right)P_{ij}, \quad (2)$$

which uses $\alpha = 1$, $1/2$, and 0 for implicit, semi-implicit, and explicit formulations respectively. Now denote the masses as a vector, i.e., $\boldsymbol{m} = [m_1, \ldots, m_N]^T$, and if one constructs a matrix of particle collision probabilities $\boldsymbol{P}$ with entries P_{ij}, then (2) can be expressed as

$$\left[\boldsymbol{I} + \frac{\alpha}{2}(\mathrm{diag}\,(\boldsymbol{1P}) - \boldsymbol{P})\right]\boldsymbol{m}^{k+1} = \left[\boldsymbol{I} - \frac{1-\alpha}{2}(\mathrm{diag}\,(\boldsymbol{1P}) + \boldsymbol{P})\right]\boldsymbol{m}^k \quad (3)$$

where $\boldsymbol{A} = \mathrm{diag}(\boldsymbol{x})$ denotes a diagonal matrix, $\boldsymbol{A}$ with the entries of vector $\boldsymbol{x}$ along the main diagonal and $\boldsymbol{1}$ is an $1 \times N$ vector of ones.

3. Explicit schemes

Clearly, setting $\alpha = 0$ in (3) results in an explicit forward matrix calculation. We call this matrix-explicit. All of the masses used to calculate the transfer magnitudes are from the beginning of the timestep. Another method sequentially calculates (2) for $j = 1, \ldots, N$. After the j^{th} particle is updated, its new mass can be used on the right side of the equation for subsequent calculations. If the sum is calculated using one fixed value for m_j, then we call this vector-explicit, calculated as followed (employing pseudo-code, where $\circ$ denotes the entry-wise, or Hadamard, product)

$$\begin{aligned}
&\texttt{for } j = 1 : N\\
&\quad \boldsymbol{\Delta m} = \frac{1}{2}\left(\boldsymbol{m}(t) - m_j(t)\right)\circ \boldsymbol{P}_{(:,j)}\\
&\quad \boldsymbol{m}(t+\Delta t) = \boldsymbol{m}(t) - \boldsymbol{\Delta m}\\
&\quad m_j(t+\Delta t) = m_j(t) + \sum \boldsymbol{\Delta m}\\
&\texttt{end}.
\end{aligned} \quad (4)$$

Furthermore, if the sum is expanded, then each calculation may use an updated m_j accounting for all previous terms in the sum. We call this explicit-sequential, and it is calculated as follows

$$\begin{aligned}
&\texttt{for } i = 1 : N\\
&\quad \texttt{for } j = 1 : N\\
&\qquad \Delta m = \frac{1}{2}\left(m_i(t) - m_j(t)\right)\boldsymbol{P}_{(i,j)}\\
&\qquad m_i(t+\Delta t) = m_i(t) - \Delta m\\
&\qquad m_j(t+\Delta t) = m_j(t) + \Delta m\\
&\quad \texttt{end}\\
&\texttt{end}.
\end{aligned} \quad (5)$$

This method has a computational advantage in that there is no matrix multiplication required (just two loops over particle numbers), and hence it can accommodate huge particle numbers. It turns out that the vector-explicit algorithm is unstable for all ranges of parameters tested here and will not be explored further.

4. Accuracy as a function of repeated operation

In general, the particle positions change due to non-uniform and potentially unsteady mean velocity. The particles are also typically given a random component to represent diffusion and hydrodynamic dispersion; therefore, each simulation in an ensemble has subtle differences (Labolle et al., 1996). This is one advantage of the method: the evolving particle spacings (controlled by the number of particles) and masses represent the heterogeneity of concentrations—as defined by evolving auto- and cross-correlation functions—and the resulting mixing process (Benson et al., 2017; Bolster et al., 2016; Paster et al., 2014). However, in order to check accuracy and convergence in this paper, we must artificially remove the randomness of simulations. This is done by eliminating the random movements of particles and spacing them evenly on the interval (0,1), where the number of particles dictates the size of the constant spacing. This also allows us to construct the classical Eulerian implicit finite-difference (FD) approximation of diffusion using a 3-point space stencil for comparison. (We stress that our particle collision method may not be the most efficient way to simulate diffusion on a fixed grid of points, but the method will continue to work no matter how "mixed-up" the particle positions become.)

We track errors over time as functions of N, Δt [T], and total time $k\Delta t$. In all simulations we choose a diffusion coefficient $D = 10^{-3}$ [$\mathrm{L}^2\,\mathrm{T}^{-1}$] and a total simulation time of 10 s (unless specified otherwise). For an initial condition (IC) we choose a Heaviside function to represent the most unmixed (and error-inducing) possible state. We also choose a Gaussian IC to determine if errors remain more stable over time. Our measure of error between simulations and analytic solutions uses the root-mean-square error (RMSE),

$$\mathrm{RMSE}(\boldsymbol{s} - \boldsymbol{a}) = \left(\frac{1}{N}\sum_{j=1}^{N}(s_j - a_j)^2\right)^{1/2},$$

where s_j and a_j denote simulated and analytic solutions at spatial point j. We also utilized the infinity norm, $\max_j(|s_j - a_j|)$, which showed similar scaling and is not shown here for brevity.

To illustrate the motivation for this technical note, for $N = 50$ we see that all solutions appear diffusive by visual inspection of the plots of $m(x, t = 10)$ (Fig. 1(a)). On the other hand, considering the various solution methods after one time step (here $\Delta t = 0.1$), it is clear that the methods differ significantly in their "one-step" approximation of diffusion. To isolate error incurred by time discretization, we first fix $\Delta t = 0.1$ and vary the number of particles (Fig. 2). The errors are similar for $N = 500, 1000$, and 5000, indicating that, as long as a sufficient, minimum number of particles is used, increasing particle number does not appreciably decrease error. In subsequent simulations we use $N = 1000$ for consistency. All methods achieve their greatest error at the beginning of the simulation, due to the unmixed, or infinite gradient, IC. Repeated applications of the operators result in reduced error. In other words, repeated application of the matrix operations converges to a true diffusive operator. This is discussed further in Section 6. Also evident on the plot is the relatively poor performance of both implicit and semi-implicit particle methods, relative to the explicit matrix particle method that tends to converge quickly to the accuracy of the Eulerian finite-difference solution to which we compare.

5. Accuracy as a function of Δt

For a given number of particles (here $N = 1000$), the overall errors of all methods decrease over repeated application. However, we note that the mass-transfer algorithm, investigated here, is only one component of a particle-tracking simulation that may involve other processes like diffusive random walks, advective motion, and chemical reaction. If these other processes are included, it may negate this property. One might expect that, similar to the Crank–Nicolson time-stencil in an FD implementation, the semi-implicit solutions would improve as Δt decreases, relative to the explicit and implicit methods, but this is not the case. All methods tested here have errors approximately proportional to Δt (Fig. 1(b)).

To better understand the relation between error and Δt, we wish to find the power p such that $\mathcal{E}_A := \mathrm{RMSE}(\boldsymbol{s} - \boldsymbol{a}) < c(\Delta t)^p = \mathcal{O}(\Delta t^p)$, given the simulated and analytic solution vectors ($\boldsymbol{s}$ and $\boldsymbol{a}$) and some constant c. Conducting a convergence analysis for a one-second simulation and refining Δt by successive halves, we compute an experimental value of