MULTISCALE MODEL. SIMUL.
Vol. 11, No. 4, pp. 1071–1096

BINNING FOR EFFICIENT STOCHASTIC MULTISCALE PARTICLE SIMULATIONS*

M. D. MICHELOTTI†, M. T. HEATH†, AND M. WEST‡

Abstract. Gillespie's Stochastic Simulation Algorithm (SSA) is an exact procedure for simulating the evolution of a collection of discrete, interacting entities, such as coalescing aerosol particles or reacting chemical species. The high computational cost of SSA has motivated the development of more efficient variants, such as Tau-Leaping, which sacrifices the exactness of SSA. For models whose interacting entities can be characterized by a continuous parameter, such as a measure of size for aerosol particles, we analyze strategies for accelerating these algorithms by aggregating particles of similar size into bins. We show that for such models an appropriate binning strategy can dramatically enhance efficiency, and in particular can make SSA computationally competitive without sacrificing exactness. These strategies are especially effective for highly multiscale problems. We formulate binned versions of both the SSA and Tau-Leaping algorithms and analyze and demonstrate their performance.

Key words. stochastic simulation algorithm, continuous-time Markov process, Monte Carlo, atmospheric aerosol, coagulation

AMS subject classifications. 60J27, 60J28, 65C05

DOI. 10.1137/130908038

1. Stochastic particle simulation.

Gillespie's Stochastic Simulation Algorithm (SSA) is a Monte Carlo procedure for simulating the evolution of a collection of discrete, interacting entities, such as coalescing aerosol particles [5] or chemically reacting species [6], whose interactions are stochastic. SSA tracks each individual entity in a collection and provides an exact realization of the underlying Markov process, but the resulting computational cost is usually deemed prohibitive for large numbers of particles. The high computational cost of SSA has motivated the development of more efficient variants for the purpose of simulating chemically reacting species. Some of these (e.g., [4, 9, 12]) retain the exactness of SSA.

A popular variant of SSA is Tau-Leaping [7], originally developed for simulating chemically reacting systems, which have a discrete space of reactants and reaction channels. Tau-Leaping achieves greater acceleration by amalgamating interactions over a time interval τ during which interaction propensities are assumed not to vary significantly, thereby "leaping" in time over multiple interactions. Unfortunately, the resulting simulation is no longer exact, and the trade-off between the speed of the algorithm and the accuracy of the approximation depends on the potentially delicate choice of the time step. Tau-Leaping has been improved in various ways, such as improving rules for adaptive selection of the time step τ [8, 14]. However, Tau-Leaping is not directly applicable to the simulation of coalescing aerosol particles, as aerosol particles have a continuous range of sizes rather than occurring in a small number of discrete classes.

Here we explore a different approach to accelerating SSA and Tau-Leaping for

*Received by the editors January 31, 2013; accepted for publication (in revised form) July 5, 2013; published electronically November 5, 2013.
http://www.siam.org/journals/mms/11-4/90803.html
†Department of Computer Science, University of Illinois at Urbana-Champaign, Urbana, IL 61801 (mmichel3@illinois.edu, heath@illinois.edu).
‡Department of Mechanical Science and Engineering, University of Illinois at Urbana-Champaign, Urbana, IL 61801 (mwest@illinois.edu).

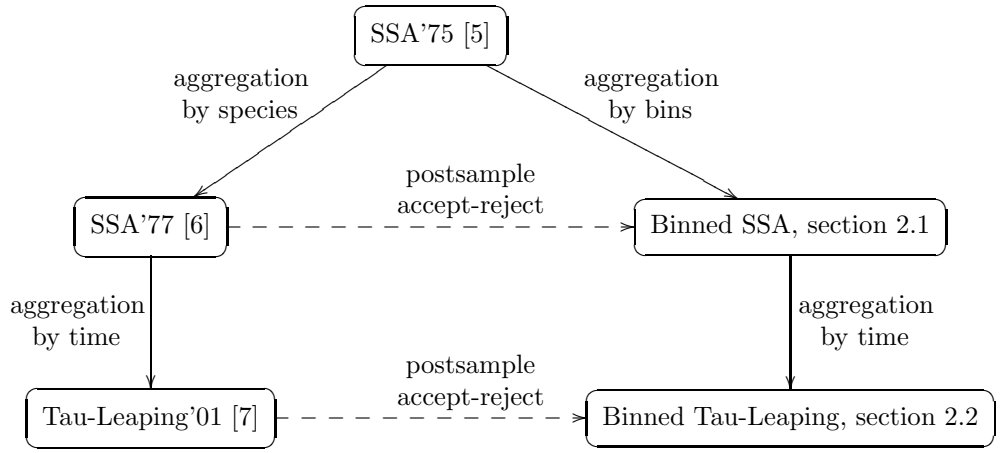

FIG. 1.1. *Conceptual relationship between Gillespie's original SSA method* [5] *(SSA*'75*), his reformulation for chemical kinetic systems* [6] *(SSA*'77*), his Tau-Leaping method* [7] *(Tau-Leaping*'01*), and the binned versions of these methods presented in this paper in sections* 2.1 *and* 2.2.

models whose interacting entities can be characterized by a continuous parameter, such as a measure of size for aerosol particles, in which particles of similar size are aggregated into bins. This approach is inspired by the highly multiscale nature of atmospheric aerosol particles, whose volumes can span many orders of magnitude. We analyze various binning strategies and show that for such models an appropriate binning strategy can dramatically enhance efficiency, and in particular can make SSA competitive with Tau-Leaping *without* sacrificing exactness.

The relationships among the original SSA and Tau-Leaping methods and our new binned versions are shown schematically in Figure 1.1. SSA'77 [6] assumes that the system state consists of many molecules or entities, each of which belongs to one of a relatively small number of species or classes, and is indistinguishable from all other molecules from the same species. This property allows the SSA'75 [5] method to be modified to aggregate operations by species. This approach in turn forms the basis for Tau-Leaping'01 [7], where events are further aggregated over a time step of length τ. The binned methods developed in the present paper can be understood in a similar way, in which the system state consists of many particles that can be grouped together into bins (analogous to species). The difference is that the particles in a bin are merely similar to each other, rather than identical, which requires modified algorithms that include postsample accept-reject stages.

SSA and its variants can simulate a wide variety of Markov processes, but for our main application example we will focus on the simulation of atmospheric aerosols, in which particles may interact by *coagulation*, where two particles coalesce to form a single, larger particle. Particle coagulations are modeled to occur at stochastic rates determined by properties of the coagulating particles. Figure 1.2 depicts the event probability rates for Brownian coagulation [11]. Since the particle volumes are highly multiscale, the Brownian coagulation rates are also highly multiscale.

Stochastic aerosol models were the original motivation for formulating SSA [5], but it has been used relatively little for such problems because of the high cost for realistic numbers of particles. Sectional or modal aerosol models [2, 10, 17, 20] seem more

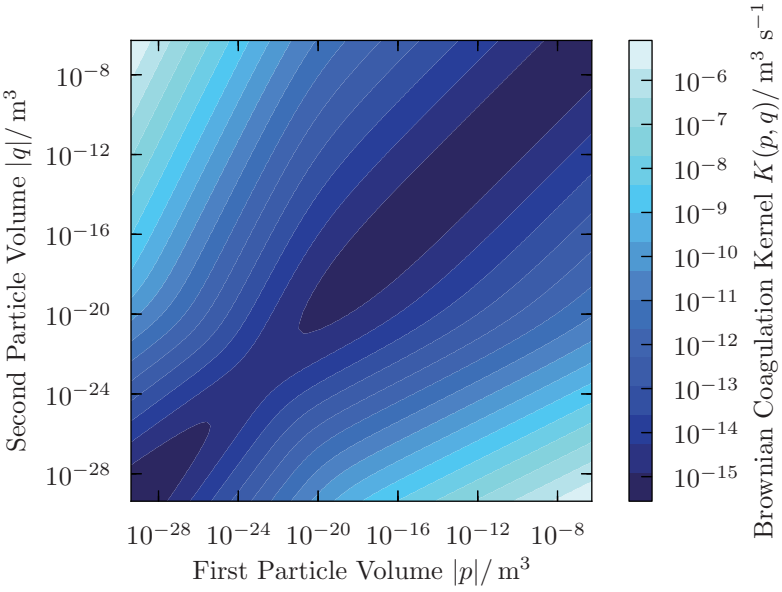

FIG. 1.2. *Brownian kernel coagulation rates between two particles p and q* [11]*, scaled by volume of simulation domain. All particles have density of water,* $1000\,\mathrm{kg/m^3}$.

efficient for such systems, since individual particles need not be tracked. However, such models do not usually capture the multidimensional nature of particle composition. Atmospheric aerosol particles contain a mixture of chemical species, on the order of 20 different species in modern models [19]. For such high-dimensional problems, Monte Carlo methods based on stochastic particle interactions can be competitive. This is the motivation behind the software package PartMC, which uses a particle-resolved aerosol model [3, 15, 16, 18]. For its coagulation routine, PartMC currently uses a binned version of Tau-Leaping, which we will refer to as Binned Tau-Leaping.

This paper is organized as follows. The remainder of section 1 describes the generic stochastic model that we will be simulating and introduces notation for the binning structure. Section 2 formulates binned versions of both SSA and Tau-Leaping. Binned SSA is shown to retain the exactness of SSA. Based on both theoretical analysis and a series of computational test problems, Binned SSA is shown to be competitive in computation time with Binned Tau-Leaping. Section 3 analyzes static binning schemes, justifies the use of logarithmic binning for multiscale aerosol coagulation, and discusses an approach to an adaptive binning scheme. The test problems and testing methodology we use are summarized in the appendix.

1.1. Stochastic event model. Using terminology motivated by aerosol modeling, consider a space of possible "particles" $\mathcal{P}$. Any two particles in $\mathcal{P}$ can undergo a pairwise "event." The specific definition of such an event depends on the application. For example, in the stochastic coalescence model, these events are particle coagulations. It should be noted that the principles of this stochastic event model, as well as the algorithms discussed in this paper, can be applied to events involving one particle, two particles, three particles, and so on. For simplicity and readability, however, we will focus on the two-particle case, considering only pairwise events.

Define a kernel function $K\colon \mathcal{P} \times \mathcal{P} \to \mathbb{R}^+$, where $K(p,q)$ is the stochastic rate at which events occur between particles p and q. For notational simplicity, we assume that no two particles in the simulation are the same element of $\mathcal{P}$. Let $\pi \subset \mathcal{P}$ be the particles present at the current stage of the simulation. Our model defines π as a function of time subject to stochastic pairwise events at rates given by the kernel K. We assume this is a continuous-time Markov process for which we are given the initial value of π at time zero.

Because the events occur stochastically, π is a random variable. Our task is to implement a Monte Carlo simulation to generate a realization of π as a function of time. More sophisticated simulations may involve other processes occurring alongside these events or multiple kinds of events. The algorithms discussed in this paper can be readily coupled with such other processes.

1.2. Binning particles. Performing a stochastic simulation that considers all possible pairwise events can be expensive if done naively. It is useful to bin similar particles together and consider interactions between bins. However, special care must be taken to ensure that a simulation over bins well approximates a simulation over individual particles.

For ease of presentation we assume that the kernel values can be well-bounded by a function of two scalar parameters describing the two particles. This assumption is not essential, however, and algorithms using multidimensional bins can be readily formulated.

In order to bin particles together, we need a way to determine whether two particles are similar to each other in some sense. For this purpose, we employ a *magnitude* operator $|\cdot|\colon \mathcal{P} \to \mathbb{R}$ that characterizes each particle by a real number, which could be its volume, diameter, mass, etc. The choice of magnitude operator will affect the performance of the algorithms to be discussed in this paper, but it will not affect the accuracy of the algorithms. An appropriate magnitude operator may be determined in part by considering how the kernel is implemented. We will need a reasonable upper bound on the kernel given only the magnitudes of the two particles.

In the case of atmospheric aerosol particle coagulations, particle volume is an appropriate choice for the magnitude. The Brownian coagulation kernel depends only on the volume and density of each particle [11], and particle densities do not vary significantly. Therefore, we can achieve a reasonable upper bound for the Brownian kernel knowing only the volume.

The particles in the simulation are grouped into bins based on their magnitudes. Let $\nu_0, \ldots, \nu_m \in \mathbb{R}$ be a strictly increasing sequence that denotes the boundaries of m bins. We will use the following definitions to describe the bins:

$$R(a,b) := \{p \in \mathcal{P} : a < |p| \le b\} \text{ for } a, b \in \mathbb{R}, \tag{1.1}$$

$$r_i := R(\nu_{i-1}, \nu_i), \tag{1.2}$$

$$r(p) := r_i, \text{ where } i \text{ is chosen such that } p \in r_i, \tag{1.3}$$

$$\pi_i := r_i \cap \pi, \tag{1.4}$$

$$K_{\text{up}}(r_i, r_j) \ge \sup\{K(p,q) : p \in r_i, q \in r_j\}. \tag{1.5}$$

Collectively, the sets $\pi_1, \ldots, \pi_m$ partition π according to the bin ranges. The function K_{up} is an upper bound on the value of the kernel over a given pair of bins. Using a tighter upper bound will result in a more efficient algorithm, but the accuracy of the algorithm will remain the same.

We can find an upper bound M_{ij} on the stochastic event rate between any two

bins i and j, with $i \leq j$. Specifically,

$$M_{ij} := \theta_{ij} K_{\text{up}}(r_i, r_j), \tag{1.6}$$

where

$$\theta_{ij} := \begin{cases} |\pi_i|\,|\pi_j|, & i \neq j, \\ \frac{1}{2}|\pi_i|(|\pi_i| - 1), & i = j. \end{cases} \tag{1.7}$$

We will also use the following aggregates:

$$M_i := \sum_{j=i}^{m} M_{ij}, \tag{1.8}$$

$$M_0 := \sum_{i=1}^{m} \sum_{j=i}^{m} M_{ij} = \sum_{i=1}^{m} M_i. \tag{1.9}$$

M_0 is an upper bound on the overall stochastic event rate. We will use these rates to generate "event candidates." Since we are overestimating the rates, some of these candidates will be rejected. We define K_{ij} as the stochastic event rate between bins i and j, and we define K_0 as the overall stochastic event rate:

$$K_{ij} := \sum_{p \in \pi_i} \sum_{q \in \pi_j, q > p} K(p, q), \tag{1.10}$$

$$K_0 := \sum_{i=1}^{m} \sum_{j=i}^{m} K_{ij} = \sum_{p \in \pi} \sum_{q \in \pi, q > p} K(p, q). \tag{1.11}$$

The $>$ operator between two particles, as in (1.11), refers to some ordering on $\mathcal{P}$ that must guarantee that a particle with larger magnitude is greater than a particle with smaller magnitude. From the definitions, M_0 can be written in a form similar to K_0:

$$M_{ij} = \sum_{p \in \pi_i} \sum_{q \in \pi_j, q > p} K_{\text{up}}(r(p), r(q)), \tag{1.12}$$

$$M_0 = \sum_{p \in \pi} \sum_{q \in \pi, q > p} K_{\text{up}}(r(p), r(q)). \tag{1.13}$$

From (1.11) and (1.13), it is immediate that $M_0 \geq K_0$, which is expected since M_0 is an overestimated event rate and K_0 is the exact event rate.

2. Binned algorithms.

2.1. Binned SSA. Ignoring errors due to finite precision arithmetic and pseudo-random number generation, the SSA developed by Gillespie generates an exact realization of the underlying Markov process [5]. Unfortunately, the computation time of SSA makes it prohibitively expensive for large, physically realistic problems. In this section, we present an algorithm that enhances the efficiency of SSA without sacrificing exactness.

Algorithm 1, called the "Binned Stochastic Simulation Algorithm" or "Binned SSA," is a novel approach to simulating a stochastic event model. The approach is similar to SSA, except that it acts over bins of particles rather than individual particles. Event rates are overestimated, so an accept-reject phase is necessary to

Algorithm 1. Binned SSA.

put initial particles into bins $\pi_1, \ldots, \pi_m$
compute and store values of $M_0, M_1, \ldots, M_m$
$t \leftarrow 0$
loop
  $\Delta t \leftarrow \ln(1/\text{rand})/M_0$, where rand is a uniform random number from $[0, 1)$
  $t \leftarrow t + \Delta t$
  if $t > t_{\text{final}}$, break out of loop
  randomly choose i from 1 to m with probability M_i/M_0
  randomly choose j from i to m with probability M_{ij}/M_i
  uniformly randomly choose two distinct particles $p \in \pi_i$ and $q \in \pi_j$
  with probability $K(p, q)/K_{\text{up}}(r_i, r_j)$, process event for (p, q)
  update values $M_0, M_1, \ldots, M_m$ as needed
end loop

retain exactness. The time step Δt is chosen to sample an exponential distribution with rate M_0. Bin pair (π_i, π_j) is chosen with probability M_{ij}/M_0.

Like the original SSA, the Binned SSA generates an exact realization of the Markov process. In other words, the probability distribution of the output of the algorithm is the same as the probability distribution of the stochastic event model. SSA was shown to select the time and particles involved in the next event from a certain joint probability density function [5]. We will show that the same holds for Binned SSA.

LEMMA 2.1. *For Binned SSA (Algorithm 1), let T be the time until the next event is processed, and let P and Q be the particles chosen for that event in the canonical order $P < Q$. Let $\phi(p, q, t)$ be the joint probability density function corresponding to the random variables P, Q, and T. Then*

$$\phi(p, q, t) = K(p, q)e^{-K_0 t}.$$

Thus, Binned SSA generates an exact realization of the underlying Markov process.

Proof. Binned SSA may take multiple loop iterations to generate an event. Let us consider the outcome of a single loop iteration. Either a single event occurs between two particles or no event occurs. Let X denote this outcome, and let $g(x)$ be the probability mass function for this random variable. Then X can be a particle pair (p, q) with $p < q$, or X can be 0, denoting that no event occurred. Consider two particles $p \in \pi_i$ and $q \in \pi_j$ where $p < q$. Particles p and q will generate an event in one iteration if the bin pair (π_i, π_j) is chosen, the particle pair (p, q) is chosen from this bin pair, and the accept-reject test for these particles passes. Thus,

$$g((p, q)) = \frac{M_{ij}}{M_0} \frac{1}{\theta_{ij}} \frac{K(p, q)}{K_{\text{up}}(r_i, r_j)} = \frac{K(p, q)}{M_0}.$$

Since g must sum to 1,

$$g(0) = 1 - \sum_{p \in \pi} \sum_{q \in \pi, q > p} g((p, q)) = 1 - \frac{K_0}{M_0}.$$

Let f be the probability mass function corresponding to random variables P, Q,

and Z, where Z denotes the number of loop iterations needed to process the event:

$$f(p,q,z) = g(0)^{z-1} g((p,q)) = \left(1 - \frac{K_0}{M_0}\right)^{z-1} \frac{K(p,q)}{M_0}.$$

Summing over possible values, we can find the probabilities of obtaining a certain particle pair or achieving a certain number of iterations. We obtain

$$\Pr(P = p, Q = q) = \sum_{z=1}^{\infty} f(p,q,z) = \frac{K(p,q)}{K_0},$$

$$\Pr(Z = z) = \sum_{p \in \pi} \sum_{q \in \pi, q > p} f(p,q,z) = \left(1 - \frac{K_0}{M_0}\right)^{z-1} \frac{K_0}{M_0}.$$

Let ϕ_t be the probability density function for T. Recall that T is the sum of Z independent random variables chosen from an exponential distribution with rate M_0. Also, note that Z follows a geometric distribution with rate K_0/M_0. From this, we conclude that T follows an exponential distribution with rate K_0 [13]:

$$\phi_t(t) = K_0 e^{-K_0 t}.$$

Since P and Q are independent from Z and T, we obtain ϕ by multiplication. Thus,

$$\phi(p,q,t) = \Pr(P = p, Q = q)\phi_t(t) = K(p,q)e^{-K_0 t},$$

which is the desired result. ∎

THEOREM 2.2. *Binned SSA (Algorithm 1) generates an exact realization of the underlying Markov process.*

Proof. Consider a superiteration of the loop in Algorithm 1 to consist of all loop iterations until an event is processed. Lemma 2.1 shows that one superiteration of Binned SSA is equivalent to one loop iteration of SSA. Both algorithms iterate until the simulation time has elapsed. Thus, SSA and Binned SSA are equivalent in the sense that they have the same probability of achieving any given result. Since SSA generates an exact realization of the underlying Markov process, so does Binned SSA. ∎

Since the particle state is constantly changing, it is difficult to evaluate the overall running time of either SSA or Binned SSA. Given the current state π, we can compute the expected computation time per event for the immediate future. This value will be denoted ξ. Let ξ_O be the expected computation time per event for original SSA, and let ξ_B be the expected computation time per event for Binned SSA.

SSA normally takes $O(|\pi|)$ time to process an event. A single loop iteration of Algorithm 1 requires $O(m)$ time, where m is the number of bins. The expected number of loop iterations until an event is reached is 1/Eff, where we define the efficiency as the expected acceptance rate of event candidates:

$$\text{Eff} = \frac{K_0}{M_0}. \tag{2.1}$$

Thus, $\xi_O = O(|\pi|)$ and $\xi_B = O(m/\text{Eff})$. We can typically achieve Eff $> 1/2$ and $m \ll |\pi|$.

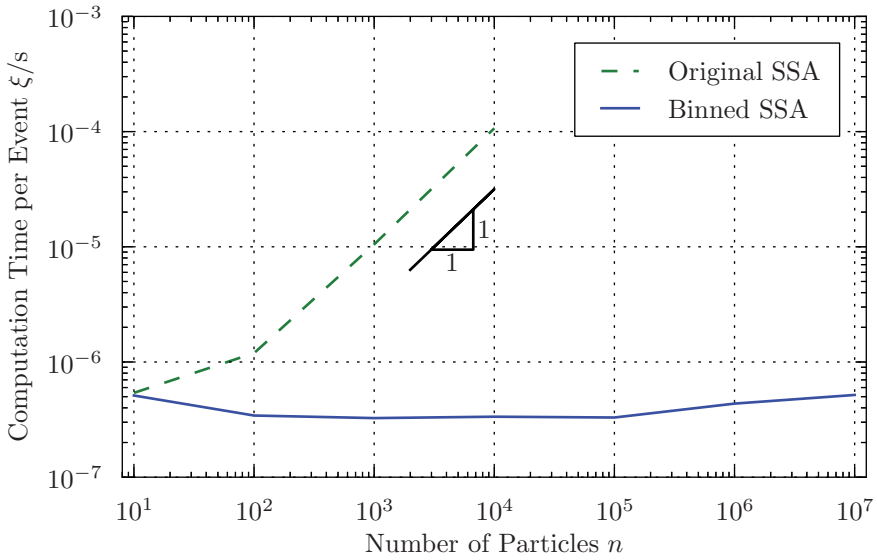

FIG. 2.1. *Computation time per event for Test Problem* 1 *(see Appendix* A*), comparing original SSA with Binned SSA.*

Figure 2.1 compares the computation times of SSA and Binned SSA using Test Problem 1, which uses the additive kernel and has a known analytic solution (descriptions of this and other test problems can be found in Appendix A). As the number of particles increases, the computation time per event for SSA increases linearly, as predicted. However, we do not need to increase the number of bins as the number of particles increases, so the computation time per event for Binned SSA remains roughly constant. Binned SSA outperforms SSA because it does not need to iterate through the list of particles to find the next event pair.

The decrease in startup time is also noteworthy. SSA requires computing K_0 before iterations begin, which is an $O(n^2)$ operation, where n is the initial number of particles (for this and other notation, see Appendix A). Binned SSA requires computing only M_0, an $O(m^2)$ operation, which is much faster since $m \ll n$.

Figure 2.2 compares the accuracy of SSA and Binned SSA. We observe that both algorithms have the same accuracy, as expected from Theorem 2.2. The error bars in Figure 2.2 correspond to 95% confidence intervals based on Student's t-test, as is the case with all error bars in this paper.

2.2. Binned Tau-Leaping. In 2001, Gillespie published the Tau-Leaping algorithm, an approximate accelerated version of original SSA [7]. Tau-Leaping was developed for simulating chemically reacting systems, however, not for particle simulations. The main difference is that there is usually a small, discrete set of possible reactants and reactions in chemical simulations, whereas the space of possible particles in a particle simulation is infinite and continuous. Consequently, Tau-Leaping is not directly applicable to particle simulations, but it can be applied to particle simulations if we first put the particles into bins. This is the approach currently used by the PartMC software [15]. In this section, we will introduce this Binned Tau-Leaping algorithm and analyze its performance.

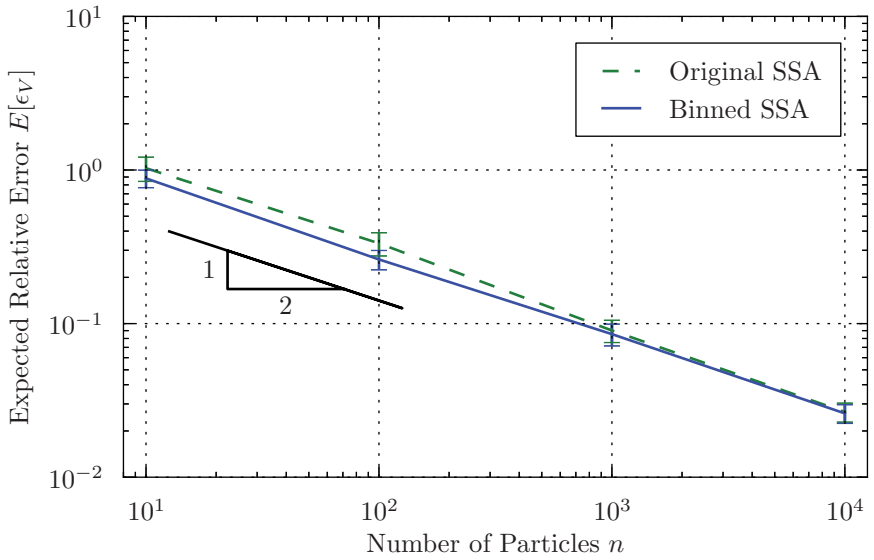

FIG. 2.2. *Relative error for Test Problem* 1*, comparing original SSA with Binned SSA.*

Algorithm 2. Binned Tau-Leaping Algorithm.

put initial particles into bins $\pi_1, \ldots, \pi_m$
$t \leftarrow 0$
while $t \leq t_{\text{final}}$ **do**
  for each bin pair (π_i, π_j) with $i \leq j$ **do**
    $c \leftarrow$ sample Poisson distribution with mean τM_{ij}
    for c iterations **do**
      uniformly randomly choose two distinct particles $p \in \pi_i$ and $q \in \pi_j$
      with probability $K(p,q)/K_{\text{up}}(r_i, r_j)$, process event for (p,q)
    end for
  end for
  $t \leftarrow t + \tau$
end while

Binned SSA is essentially the same as applying original SSA to bins of particles instead of individual particles. Likewise, Binned Tau-Leaping is essentially the same as applying Tau-Leaping to bins of particles instead of individual particles. Since bins are discrete and relatively few in number, Tau-Leaping is applicable and efficient.

Algorithm 2 presents Binned Tau-Leaping. This algorithm uses a predetermined time-step size τ. For each time step, we iterate over all pairs of bins (i, j), where $i \leq j$, and generate a certain number of event candidates from these bins. The number of event candidates is chosen from a Poisson distribution with mean τM_{ij}. We accept or reject these event candidates as we did in Binned SSA. All events processed within a given time step, though ordered, are still deemed to have occurred at time t. Once we have considered all event candidates for this time step, we increment time by τ and proceed to the next time step.

One of the main benefits of the original Tau-Leaping algorithm is that it can

process multiple events at once. However, Binned Tau-Leaping must process one event at a time, because each particle is unique and each event requires its own accept-reject test. Thus, Binned Tau-Leaping does not exhibit the dramatic performance improvement over Binned SSA that Tau-Leaping enjoys over SSA.

Unlike Binned SSA, Binned Tau-Leaping is not exact. Furthermore, the accuracy of Binned Tau-Leaping may be affected by the positions of the bin boundaries if τ is large. However, we can show that Binned Tau-Leaping converges to the correct behavior as τ goes to zero.

LEMMA 2.3. *For Binned Tau-Leaping (Algorithm 2), let T be the time until the next events occur, and let Λ be the set of events that occur in that time step. Let $\Phi_\tau(\lambda, t)$ be the probability that $\Lambda = \lambda$ and $T > t$. Then*

$$\Phi_\tau(\{(p,q)\}, t) = \frac{K(p,q)}{K_0} e^{-tK_0} + O(\tau),$$

where τ is the time-step size of the Tau-Leaping algorithm.

Proof. Let $f_1(i,j)$ be the probability that an event candidate generated from bin pair (i,j) is accepted. To calculate this, we must consider all possible event candidates between these bins. Thus,

$$f_1(i,j) = \sum_{p \in \pi_i} \sum_{q \in \pi_j, q > p} \frac{1}{\theta_{ij}} \frac{K(p,q)}{K_{\text{up}}(r_i, r_j)} = \frac{K_{ij}}{M_{ij}}.$$

Consider the outcome of a single time step. Assuming no events occur from other bin pairs, let $f_2(i,j)$ be the probability that all event candidates generated for bin pair (i,j) are rejected. The number of event candidates comes from a Poisson distribution, so we must consider all outcomes of the Poisson distribution. Therefore,

$$f_2(i,j) = \sum_{k=0}^{\infty} \frac{(\tau M_{ij})^k}{k!} e^{-\tau M_{ij}} (1 - f_1(i,j))^k = e^{-\tau M_{ij}} e^{\tau M_{ij}(1 - f_1(i,j))} = e^{-\tau K_{ij}}.$$

Let f_3 be the probability that all event candidates for a single time step are rejected. We obtain

$$f_3 = \prod_{i=1}^{m} \prod_{j=i}^{m} f_2(i,j) = \prod_{i=1}^{m} \prod_{j=i}^{m} e^{-\tau K_{ij}} = e^{-\tau K_0}.$$

Let $f_4(p,q)$ be the probability that a time step generates event (p,q) and no other events. It is difficult to compute $f_4(p,q)$ exactly, but we can obtain useful upper and lower bounds. Let U be an upper bound on M_{ij} for all i,j. This bound should be valid both before and after the event (p,q). Choose i and j such that $p \in \pi_i$ and $q \in \pi_j$. For a lower bound on f_4, we consider a lower bound on the probability that all bin pairs other than (i,j) obtain a zero Poisson number, that bin pair (i,j) obtains a Poisson number of 1, and that the event candidate (p,q) is chosen and accepted. Therefore,

$$f_4(p,q) \geq (e^{-\tau U})^{m^2} \tau M_{ij} e^{-\tau M_{ij}} \frac{1}{\theta_{ij}} \frac{K(p,q)}{K_{\text{up}}(r_i, r_j)} = \tau K(p,q) e^{-\tau(Um^2 + M_{ij})}.$$

For an upper bound on f_4, we consider the probability that bin pair (i,j) obtains a Poisson number of 1 and that the event candidate (p,q) is chosen and accepted, plus

the probability that bin pair (i, j) obtains a Poisson number greater than 1. Thus,

$$\begin{aligned} f_4(p,q) &\le \tau M_{ij} e^{-\tau M_{ij}} \frac{1}{\theta_{ij}} \frac{K(p,q)}{K_{\text{up}}(r_i, r_j)} + (1 - (1 + \tau M_{ij}) e^{-\tau M_{ij}}) \\ &= \tau K(p,q) e^{-\tau M_{ij}} + (1 - (1 + \tau M_{ij}) e^{-\tau M_{ij}}). \end{aligned}$$

From these two bounds, we observe

$$f_4(p,q) = \tau K(p,q) + O(\tau^2).$$

Let $f_5(p,q)$ be the probability that $\Lambda = \{(p,q)\}$. This is the probability that a time step generates only event (p,q) given that the time step generates at least one event. We obtain

$$f_5(p,q) = \frac{f_4(p,q)}{1 - f_3} = \frac{\tau K(p,q) + O(\tau^2)}{1 - e^{-\tau K_0}} = \frac{K(p,q)}{K_0} + O(\tau).$$

Let $f_6(t)$ be the probability that $T > t$. This is the probability that all of the $\lfloor t/\tau \rfloor$ time steps before t produce no events. Therefore,

$$f_6(t) = f_3^{\lfloor t/\tau \rfloor} = (e^{-\tau K_0})^{\lfloor t/\tau \rfloor} = e^{-tK_0} + O(\tau).$$

Since Λ and T are independent, we obtain Φ_τ by multiplying f_5 and f_6:

$$\begin{aligned} \Phi_\tau(\{(p,q)\}, t) = f_5(p,q) f_6(t) &= \left(\frac{K(p,q)}{K_0} + O(\tau) \right) (e^{-tK_0} + O(\tau)) \\ &= \frac{K(p,q)}{K_0} e^{-tK_0} + O(\tau). \end{aligned}$$

This is the desired result. ∎

THEOREM 2.4. *For a given initial particle distribution π_0 and a given number of events v, let $\boldsymbol{P}, \boldsymbol{Q}, \boldsymbol{T}$ be vectors of random variables corresponding to the first v events generated from SSA. Let $\boldsymbol{P}_\tau, \boldsymbol{Q}_\tau, \boldsymbol{T}_\tau$ be vectors of random variables corresponding to the first v events generated from Binned Tau-Leaping for a given τ. Define*

$$\begin{aligned} D(\boldsymbol{p}, \boldsymbol{q}, \boldsymbol{t}) &:= \Pr(\boldsymbol{P} = \boldsymbol{p}, \boldsymbol{Q} = \boldsymbol{q}, \boldsymbol{T} > \boldsymbol{t}), \\ D_\tau(\boldsymbol{p}, \boldsymbol{q}, \boldsymbol{t}) &:= \Pr(\boldsymbol{P}_\tau = \boldsymbol{p}, \boldsymbol{Q}_\tau = \boldsymbol{q}, \boldsymbol{T}_\tau > \boldsymbol{t}). \end{aligned}$$

Then,

$$D_\tau(\boldsymbol{p}, \boldsymbol{q}, \boldsymbol{t}) = D(\boldsymbol{p}, \boldsymbol{q}, \boldsymbol{t}) + O(\tau).$$

Note that $\boldsymbol{t}$ can be chosen to span the duration of the simulation or longer. Thus, the outcome of Binned Tau-Leaping converges in distribution to the outcome of Binned SSA at a rate of $O(\tau)$.

Proof. Define $\Phi(p, q, t; \pi)$ as the probability of generating an event $P = p, Q = q, T > t$ using SSA with a given particle distribution π. We know this to be

$$\Phi(p, q, t; \pi) = \frac{K(p,q)}{K_0(\pi)} e^{-tK_0(\pi)}.$$

In the above equation, K_0 is written as a function of π. Let p_i, q_i, t_i denote the ith elements of $\boldsymbol{p}, \boldsymbol{q}, \boldsymbol{t}$. Let $\pi_1, \ldots, \pi_v$ be the resulting particle distributions after each of

the events $(p_1, q_1), \ldots, (p_v, q_v)$. To obtain D, we multiply the values of Φ for each event:

$$D(\boldsymbol{p}, \boldsymbol{q}, \boldsymbol{t}) = \prod_{i=1}^{v} \Phi(p_i, q_i, t_i; \pi_{i-1}).$$

Similarly, from Lemma 2.3 we have an expression $\Phi_\tau(\{(p,q)\}, t; \pi)$ for the probability of generating a sole event $P = p, Q = q, T > t$ using Binned SSA with a given particle distribution π (the value π is now written as an explicit parameter):

$$\Phi_\tau(\{(p,q)\}, t; \pi) = \frac{K(p,q)}{K_0(\pi)} e^{-tK_0(\pi)} + O(\tau) = \Phi(p, q, t; \pi) + O(\tau).$$

The probability that Φ_τ gives a set of events λ with size greater than 1 is $O(\tau)$. This can be observed by summing Φ_τ over sets of events λ with size 1:

$$\sum_{p \in \pi} \sum_{q \in \pi, q > p} \Phi_\tau(\{(p,q)\}, 0; \pi) = 1 + O(\tau).$$

We can now obtain D_τ by multiplying the values of Φ_τ for each event, taking into account the $O(\tau)$ possibility of having multiple events in any time step. We obtain

$$\begin{aligned} D_\tau(\boldsymbol{p}, \boldsymbol{q}, \boldsymbol{t}) &= \prod_{i=1}^{v} \Phi_\tau(\{p_i, q_i\}, t_i; \pi_{i-1}) + O(\tau) \\ &= \prod_{i=1}^{v} \Phi(p_i, q_i, t_i; \pi_{i-1}) + O(\tau) \\ &= D(\boldsymbol{p}, \boldsymbol{q}, \boldsymbol{t}) + O(\tau). \end{aligned}$$

This is the desired result. ∎

It should be noted that, like Tau-Leaping, Binned Tau-Leaping becomes exact in the limit $\tau \to 0$, but it requires an infinite amount of computation time in that same limit. The value of τ controls the trade-off between accuracy and computation time.

2.3. Comparison of binned algorithms. Tau-Leaping is usually dramatically faster than SSA, so one might suspect that Binned Tau-Leaping will be dramatically faster than Binned SSA, but this is not usually true. This is because Binned Tau-Leaping must process one event at a time, whereas Tau-Leaping may process many events at once. Thus, it is less clear whether this "approximate accelerated algorithm" is actually faster in the binned formulation.

Like before, we will characterize the cost of Binned Tau-Leaping as the expected computation time per event ξ, which will be compared with that of Binned SSA. In this comparison, we will assume that τ is chosen sufficiently small that the results are sufficiently accurate. Let ξ_τ be the time per event for Binned Tau-Leaping, and let ξ_B be the time per event for Binned SSA.

Each time step of Binned Tau-Leaping has an expected running time of $O(\tau M_0 + m^2)$. The τM_0 term comes from the expected number of event candidates that are generated, and the m^2 term comes from iterating over all pairs of bins. Each time step is expected to generate τK_0 events, so we conclude that $\xi_\tau = O(1/\text{Eff} + m^2/(\tau K_0))$.

Recall that $\xi_B = O(m/\text{Eff})$. It is not clear from this big-O analysis whether ξ_B or ξ_τ will be smaller in a practical simulation, and by how much. These questions depend on many factors involved in a specific problem.

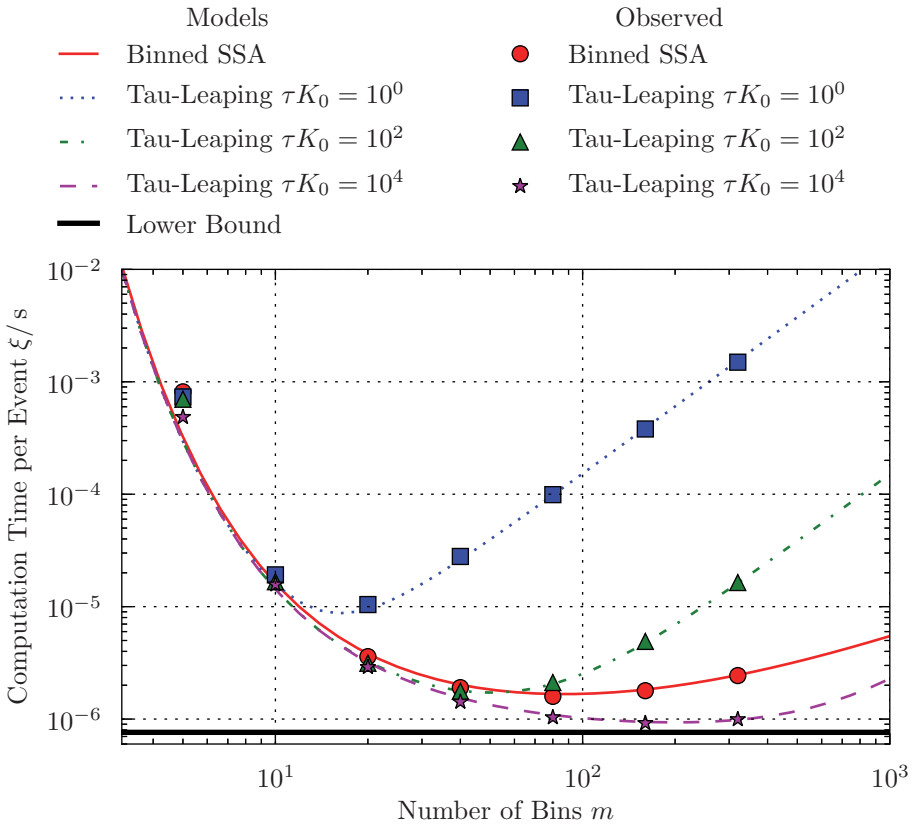

FIG. 2.3. *Computation time per event for Test Problem* 2 *with varying number of bins, comparing Binned SSA with Binned Tau-Leaping for varying* τK_0 *(coagulations per time step), which is controlled by varying* τ. *Curves are based on models in* (2.2), (2.3), (2.4), *and* (3.5). *Plotted symbols are observed values from computer simulations.*

For a more precise analysis, we use the following models for the expected computation time per event:

$$\xi_B := \frac{\alpha_1}{\text{Eff}} + \alpha_2 + \frac{\alpha_3 + \alpha_4 m}{\text{Eff}}, \tag{2.2}$$

$$\xi_\tau := \frac{\alpha_1}{\text{Eff}} + \alpha_2 + \frac{\alpha_5}{\tau K_0}\frac{m(m+1)}{2}, \tag{2.3}$$

where $\alpha_1, \ldots, \alpha_5$ are constants corresponding to the computational time to perform the following operations:

- α_1: generate an event candidate and determine whether to accept it. This involves randomly choosing a pair of particles from two bins and evaluating the kernel.
- α_2: process an event.
- $\alpha_3 + \alpha_4 m$: randomly choose a bin pair, update M_i values, and increase time.
- α_5: iterate to the next bin pair and sample a Poisson distribution.

Figure 2.3 plots ξ_B and ξ_τ for Test Problem 2, a typical coagulation problem using the Brownian kernel. The time parameter α_1 was estimated using the computation

TABLE 2.1
Values of model parameters.

	α_1	α_2	α_3	α_4	α_5
Time(s)	$6.87 \cdot 10^{-7}$	$7.5 \cdot 10^{-8}$	$4.6 \cdot 10^{-8}$	$4.5 \cdot 10^{-9}$	$3.0 \cdot 10^{-8}$

time required to evaluate the Brownian kernel. The remaining parameters were fit to match the observed data in Figure 2.3, and their values are listed in Table 2.1. Because evaluation of the Brownian kernel is a relatively expensive operation, α_1 masks most of the effects of α_2 and α_3. In section 3.1, we will discuss how to model Eff as a function of the number of bins.

Efficiency is always between 0 and 1. Thus, there is a lower bound on ξ for either algorithm:

$$\xi \geq \alpha_1 + \alpha_2, \tag{2.4}$$

as is to be expected. The computation time per event is at least the cost of choosing two particles, evaluating the kernel, and processing an event. This lower bound is plotted in Figure 2.3. It is a valid lower bound regardless of how bin boundaries are chosen.

Now ξ_τ depends on τK_0 (the expected number of coagulations per time step), but ξ_B does not. The value ξ_B depends on the specific particle distribution only through Eff. As we will see in section 3.1, Eff can be insensitive to the particle distribution when many logarithmically spaced bins are used. However, K_0 is sensitive to the particle distribution. Thus, the optimal number of bins for Binned Tau-Leaping may vary over the course of the simulation, whereas the optimal number of bins for Binned SSA remains roughly constant.

Keep in mind that changing τ affects the accuracy of Binned Tau-Leaping. Decreasing τ will decrease the error in the simulation and increase ξ_τ. Binned SSA, on the other hand, incurs no error from time discretization.

Multiple potential sources of error arise in such a stochastic simulation:

$$\epsilon_V \leq \epsilon_\tau + \epsilon_n + \epsilon_u,$$

where ϵ_V is the overall error of the simulation, ϵ_τ is the discretization error due to the choice of time-step size τ, ϵ_n is the modeling error due to using fewer particles than would be present in a physically realistic system, ϵ_u is the error incurred due to the stochastic nature of the simulation, and u is the number of times the simulation is repeated to decrease stochastic error. See Appendix A for formal definitions of these quantities.

When using Binned SSA instead of Binned Tau-Leaping, ϵ_τ is zero. As we can see from Figure 2.3, Binned SSA is never much more expensive than Binned Tau-Leaping (assuming a good choice for the number of bins), and it may be the case that Binned Tau-Leaping is notably more expensive than Binned SSA, depending on the value of τ. Thus, using Binned SSA simultaneously eliminates all doubt about ϵ_τ and guarantees low overhead in computation time.

Figure 2.4 compares the two algorithms applied to Test Problem 3 for various values of n and τ. This test problem involves particles made up of two different species. It is a multi-time-scale problem, which allows Binned SSA to outperform Binned Tau-Leaping with a fixed τ. Using Tau-Leaping with an adaptively chosen τ might be competitive with Binned SSA for this example. However, it still cannot

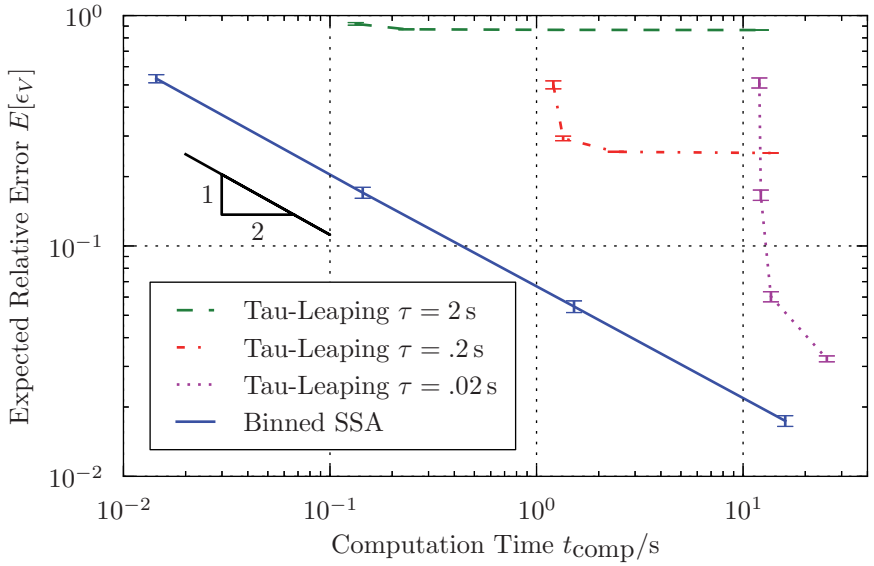

FIG. 2.4. *Error vs. computation time for Test Problem* 3 *comparing Binned SSA with Binned Tau-Leaping, with* n *varying from* 10^4 *to* 10^7 *to produce curves.*

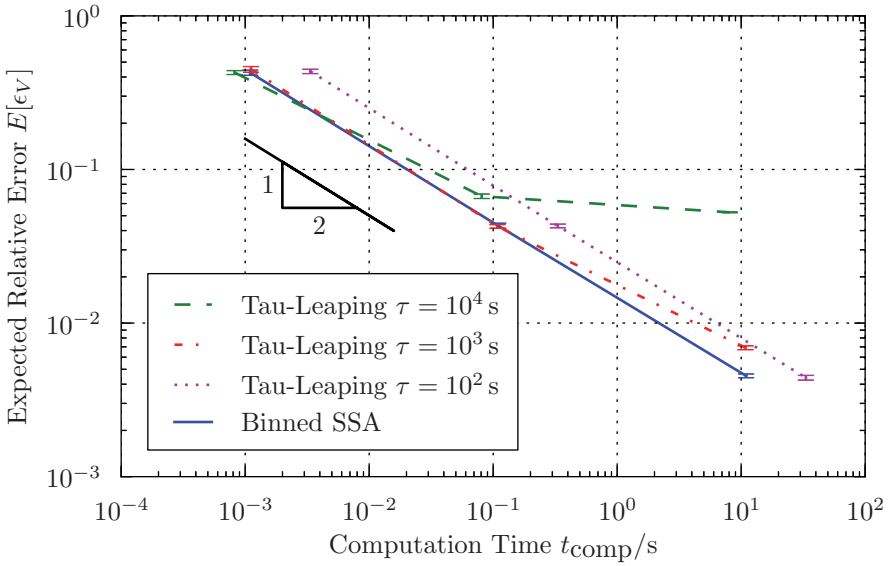

FIG. 2.5. *Error vs. computation time for Test Problem* 4 *comparing Binned SSA with Binned Tau-Leaping, with* u *varied from* 1 *to* 10^4 *to produce curves.*

significantly outperform Binned SSA, as we observe in Figure 2.3. Furthermore, it requires developing a good heuristic that chooses τ to ensure that ϵ_τ is small for a specific application. This is not necessary for Binned SSA, because ϵ_τ is zero.

Let us consider holding n fixed and varying u. The computation time required

increases linearly with both n and u, and ϵ_V decreases according to the law of large numbers with both n and u. However, there are a few reasons why we might choose to increase u rather than n. If the physically realistic system that we want to model is small enough that we can use a physically realistic value of n in our simulation, then there is no point in increasing n further; we would need to increase u in order to increase accuracy. If we have a parallel computer, we can run multiple simulations at the same time and accumulate the results, essentially increasing u without increasing the computation time. When choosing n, we are also limited by the amount of memory on the computer, as well as the nonconstant nature of memory access time as n increases.

Figure 2.5 compares the two algorithms applied to Test Problem 4 for various values of u and τ. In this example, ϵ_n is zero. Test Problem 4 is similar to Test Problem 2 except that it uses fewer particles and simulates a longer time. There are certain desired levels of accuracy for which Binned SSA outperforms Binned Tau-Leaping regardless of the value of τ. Furthermore, even at the lowest level of accuracy, with $u = 1$ and the largest τ, Binned SSA is only slightly more expensive than Binned Tau-Leaping.

3. Binning strategies.

3.1. Analysis of static binning.

Accuracy is insensitive to changes in bin boundary positions. For Binned SSA, accuracy is completely independent of changes in these boundaries. However, the computation time of either algorithm is greatly affected by the bin boundary positions. Thus, we want to choose bin boundaries to minimize ξ, the expected computation time per event.

The easiest scheme to implement is "static" binning, in which bin boundaries remain unchanged throughout the course of the simulation. With a static binning scheme, it is important to make a good choice of bin boundaries at the outset. Our objective is to minimize ξ. Unfortunately, ξ depends not only on bin boundary locations, but also on how many particles are in each bin. Thus, even with a static binning scheme, the value of ξ changes over the course of the simulation as the set of particles π changes.

Because of this, it is desirable to choose a binning scheme that is effective regardless of the set of particles π. Let us require $\pi \subseteq R(\nu_{\text{lo}}, \nu_{\text{hi}})$, where ν_{lo} and ν_{hi} are known values that will serve as the lowest and highest bin boundaries. We are then free to choose interior bin boundaries $\nu_1, \ldots, \nu_{m-1}$, yielding a set of m bins that we write in vector format,

$$\boldsymbol{r} := (r_1, \ldots, r_m). \tag{3.1}$$

The notation $|\boldsymbol{r}|$ denotes the number of bins in $\boldsymbol{r}$. We will express values such as ξ as functions of $\boldsymbol{r}$ and π.

From (2.2) and (2.3), we can see that for fixed $|\boldsymbol{r}|$ and π, minimizing $\xi(\boldsymbol{r}, \pi)$ is the same as maximizing $\text{Eff}(\boldsymbol{r}, \pi)$. Since the particle state is constantly changing, it is desirable to maximize the worst-case efficiency:

$$W(\boldsymbol{r}) := \inf_{\pi} \text{Eff}(\boldsymbol{r}, \pi). \tag{3.2}$$

It is not clear from (3.2) how to compute $W(\boldsymbol{r})$. Fortunately, Theorem 3.1 provides a way to do so.

THEOREM 3.1. *Let* $K_{\min}(r_i, r_j) := \inf\{K(p,q) : p \in r_i, q \in r_j\}$. *Then* $W(\boldsymbol{r})$ *from* (3.2) *is given by*

$$W(\boldsymbol{r}) = \min\left\{\frac{K_{\min}(r_i,r_j)}{K_{\mathrm{up}}(r_i,r_j)} : 1 \le i \le j \le m\right\}. \tag{3.3}$$

Proof. Let $W'(\boldsymbol{r})$ be the right-hand side of (3.3),

$$W'(\boldsymbol{r}) := \min\left\{\frac{K_{\min}(r_i,r_j)}{K_{\mathrm{up}}(r_i,r_j)} : 1 \le i \le j \le m\right\}.$$

We will prove that $W'(\boldsymbol{r}) = W(\boldsymbol{r})$.

Given a set of particles π, note that $W'(\boldsymbol{r}) \le K(p,q)/K_{\mathrm{up}}(r(p), r(q))$ for all $p, q \in \pi$. Thus,

$$\begin{aligned} W'(\boldsymbol{r}) &\le \left(\sum_{p\in\pi}\sum_{q\in\pi, q>p} K(p,q)\right) \Bigg/ \left(\sum_{p\in\pi}\sum_{q\in\pi,q>p} K_{\mathrm{up}}(r(p),r(q))\right) = \frac{K_0(\pi)}{M_0(\boldsymbol{r},\pi)} \\ &= \mathrm{Eff}(\boldsymbol{r},\pi). \end{aligned}$$

Because $W(\boldsymbol{r})$ is an infimum of efficiencies, we have $W'(\boldsymbol{r}) \le W(\boldsymbol{r})$.

Choose bins i, j that achieve the minimum in the equation for $W'(\boldsymbol{r})$. Then choose sequences $p_1, p_2, \ldots \in r_i$ and $q_1, q_2, \ldots \in r_j$ that achieve the infimum in the equation for $K_{\min}(r_i, r_j)$:

$$W'(\boldsymbol{r}) = \frac{K_{\min}(r_i,r_j)}{K_{\mathrm{up}}(r_i,r_j)} = \lim_{k\to\infty}\frac{K(p_k,q_k)}{K_{\mathrm{up}}(r_i,r_j)} = \lim_{k\to\infty}\mathrm{Eff}(\boldsymbol{r},\{p_k,q_k\}).$$

This is a sequence of efficiencies that converges to $W'(\boldsymbol{r})$. Thus, $W'(\boldsymbol{r}) \ge W(\boldsymbol{r})$.

We have shown $W'(\boldsymbol{r}) \le W(\boldsymbol{r})$ and $W'(\boldsymbol{r}) \ge W(\boldsymbol{r})$, and thus $W'(\boldsymbol{r}) = W(\boldsymbol{r})$. ∎

Let $\boldsymbol{l}(m)$ denote a vector of m logarithmically spaced bins. Formally, the entries of $\boldsymbol{l}(m)$ are

$$l_i(m) = R\left(\nu_{\mathrm{lo}}\left(\frac{\nu_{\mathrm{hi}}}{\nu_{\mathrm{lo}}}\right)^{(i-1)/m},\ \nu_{\mathrm{lo}}\left(\frac{\nu_{\mathrm{hi}}}{\nu_{\mathrm{lo}}}\right)^{i/m}\right). \tag{3.4}$$

Figure 3.1 plots the worst-case efficiency $W(\boldsymbol{l}(m))$ for varying m. Observed efficiencies from computer simulations are also plotted, which gives an indication of the difference between worst-case efficiency and "typical" efficiency.

We have determined the worst-case efficiency for a logarithmic binning scheme, but we have not justified that logarithmic binning is a good choice. The following theorem will give an upper bound on W regardless of the binning scheme used. We can then compare the efficiency of logarithmic binning with this upper bound.

THEOREM 3.2. *Assume that the kernel is continuous and* K_{up} *is a tight upper bound. Let* $W_\mu := \max\{W(\boldsymbol{r}) : |\boldsymbol{r}| = \mu\}$. *Then for any set of bins* $\boldsymbol{r}$ *and any positive integer* μ,

$$W(\boldsymbol{r}) \le W_\mu^{1/\lceil|\boldsymbol{r}|/\mu\rceil}.$$

Proof. This result holds trivially for $\mu \ge |\boldsymbol{r}|$, so assume $\mu < |\boldsymbol{r}|$. Let $c = \lceil|\boldsymbol{r}|/\mu\rceil$. Let $\boldsymbol{\rho}$ be a set of μ bins with elements

$$\rho_i := \bigcup_{j=ci-c+1}^{\min\{ci,|r|\}} r_j.$$

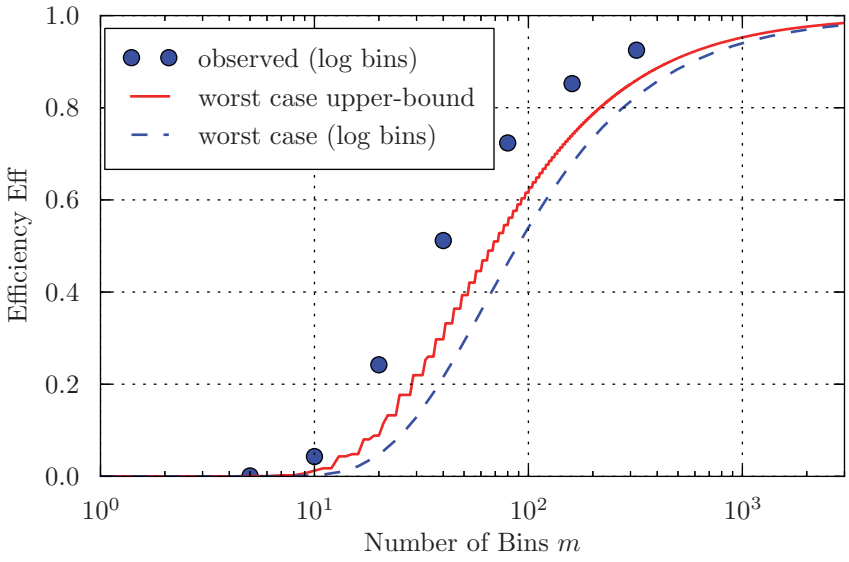

FIG. 3.1. *Efficiency for Test Problem* 2. *Circles are observed efficiencies averaged over duration of simulation using Binned SSA with logarithmically spaced bins* $\boldsymbol{l}(m)$. *Lower curve is worst-case efficiency* $W(\boldsymbol{l}(m))$ *computed from Theorem* 3.1. *Middle curve is upper bound on* W *regardless of bin spacing, computed from Theorem* 3.2 *using* $\mu = 1$ *to* 4.

Consider any pair of bins $y \in \{\rho_i \times \rho_j : 1 \le i \le j \le \mu\}$. Then, choose a sequence of bin pairs $x_1, \ldots, x_c \in \{r_i \times r_j : 1 \le i \le j \le |r| \text{ and } (r_i \times r_j) \subseteq y\}$ such that $K_{\min}(x_1) = K_{\min}(y)$ and $K_{\text{up}}(x_c) = K_{\text{up}}(y)$. We also require that for any $x_k = r_i \times r_j$ and $x_{k+1} = r_{i'} \times r_{j'}$, $|i - i'| \le 1$ and $|j - j'| \le 1$. Since y is made of at most a $c \times c$ square of bin pairs from $\boldsymbol{r}$, it is possible to find such a sequence.

From Theorem 3.1, we know $K_{\min}(x_k) \ge W(\boldsymbol{r}) K_{\text{up}}(x_k)$. Also, note that $\overline{x_k} \cap \overline{x_{k+1}} \ne \emptyset$. Thus, $K_{\text{up}}(x_k) \ge K_{\min}(x_{k+1})$. Applying these two inequalities repeatedly gives

$$K_{\min}(y) = K_{\min}(x_1) \ge W(\boldsymbol{r})^c K_{\text{up}}(x_c) = W(\boldsymbol{r})^c K_{\text{up}}(y).$$

Since this holds for all bin pairs y, we know from Theorem 3.1 that $W_\mu \ge W(\boldsymbol{\rho}) \ge W(\boldsymbol{r})^c$. Thus, $W(\boldsymbol{r}) \le W_\mu^{1/c}$. ∎

COROLLARY 3.3. *Assuming that the kernel is continuous and* K_{up} *is a tight upper bound, then*

$$W(\boldsymbol{r}) \le \left(\frac{K_{\min}(R(\nu_{\text{lo}}, \nu_{\text{hi}}))}{K_{\text{up}}(R(\nu_{\text{lo}}, \nu_{\text{hi}}))} \right)^{1/|r|}.$$

Proof. Apply Theorem 3.2 with $\mu = 1$. ∎

COROLLARY 3.4. *For the additive kernel given in* (A.5), *logarithmic binning achieves optimal worst-case efficiency:*

$$W(\boldsymbol{l}(m)) = W_m = \left(\frac{\nu_{\text{lo}}}{\nu_{\text{hi}}} \right)^{1/m}.$$

Proof. Apply Theorem 3.1 to obtain an expression for $W(\boldsymbol{l}(m))$ and notice that it is the same as the bound in Corollary 3.3. ∎

As seen in Figure 3.1, $W(\boldsymbol{l}(m))$ for the Brownian kernel comes close to the theoretical upper bound on W. Also, Corollary 3.4 tells us that logarithmic binning is optimal for the additive kernel. These facts justify our use of logarithmic binning.

To model typical efficiency, note that the observed efficiency falls roughly along the curve $\sqrt{W(\boldsymbol{l}(m))}$. This is not guaranteed to be a good approximation for every possible particle distribution, but it seems reasonable for modeling purposes, e.g.,

$$\text{Eff}(\boldsymbol{l}(m), \pi) \approx \sqrt{W(\boldsymbol{l}(m))} \approx \sqrt{(5 \cdot 10^{-27})^{1/m}} \quad \text{for Test Problem 2.} \tag{3.5}$$

In Figure 3.1, the worst-case efficiency approaches 1 as the number of bins approaches infinity, but this is not guaranteed to happen. First, recall from (1.5) that K_{up} is not necessarily a tight upper bound, which means that the $K_{\min}/K_{\text{up}}$ ratios may not converge to 1. Even if K_{up} is a tight upper bound, the kernel may depend on more than just the magnitude of each particle, in which case the limit would be

$$\lim_{m\to\infty} W(\boldsymbol{l}(m)) = \min\left\{ \frac{K(p_1, q_1)}{K(p_2, q_2)} : |p_1| = |p_2|, |q_1| = |q_2| \right\}. \tag{3.6}$$

3.2. Adaptive binning strategies. A binning scheme chooses $\boldsymbol{r}$ with the goal of minimizing $\xi(\boldsymbol{r}, \pi)$, the expected computation time per event. Thus far, we have considered only binning schemes that do not change over time. Since the particle set π is changing over time, we cannot optimize the bin structure for a particular particle distribution. An adaptive binning scheme could potentially take advantage of changes in the particle distribution to increase efficiency.

An adaptive binning scheme changes $\boldsymbol{r}$ in response to changes in π. We will allow two operations for changing $\boldsymbol{r}$: bin splitting and bin merging. Bin splitting splits a single bin into two bins. Bin merging combines two consecutive bins into one bin. A split or merge operation is performed only if it decreases the value of $\xi(\boldsymbol{r}, \pi)$.

Suppose bins k and $k+1$ were merged. Let $\boldsymbol{r}^{(\text{M})}$ be the new bins after this merge. Formally,

$$\boldsymbol{r}^{(\text{M})} := (r_1, \ldots, r_{k-1}, s, r_{k+2}, \ldots, r_m), \tag{3.7}$$

where

$$s := r_k \cup r_{k+1}.$$

Computing $\xi(\boldsymbol{r}^{(\text{M})}, \pi) - \xi(\boldsymbol{r}, \pi)$ is expensive because it requires knowing the value of $K_0(\pi)$. The expression $K_0(\pi)(\xi(\boldsymbol{r}^{(\text{M})}, \pi) - \xi(\boldsymbol{r}, \pi))$ is easier to compute, because the $K_0(\pi)$ terms cancel out. If this value is negative, then the merge will decrease the expected computation time per event.

The exact expression for ξ depends on which algorithm is used, along with the coefficients $\alpha_1, \ldots, \alpha_5$ determined by the specific application and computer. Note from (2.2) and (2.3) that ξ depends only on m, M_0, K_0, and τ. As we have already observed, the K_0 terms cancel out in the expression $K_0(\pi)(\xi(\boldsymbol{r}^{(\text{M})}, \pi) - \xi(\boldsymbol{r}, \pi))$. The time-step size τ is a known value, and m decreases by one during the merge. The only difficult term to compute is

$$\Delta_{\text{M}} := M_0(\boldsymbol{r}^{(\text{M})}, \pi) - M_0(\boldsymbol{r}, \pi). \tag{3.8}$$

Expanding this expression, we obtain

$$\Delta_{\text{M}} = \left(\kappa_1 + \kappa_2(0) + \kappa_2(1) + \sum_{i=1; i\neq k, k+1}^{m} (\kappa_3(i, 0) + \kappa_3(i, 1)) \right), \tag{3.9}$$

where

$$\kappa_1 := |\pi_k|\,|\pi_{k+1}|(K_{\mathrm{up}}(s,s) - K_{\mathrm{up}}(r_k, r_{k+1})),$$
$$\kappa_2(j) := \frac{1}{2}|\pi_{k+j}|(|\pi_{k+j}| - 1)(K_{\mathrm{up}}(s,s) - K_{\mathrm{up}}(r_{k+j}, r_{k+j})),$$
$$\kappa_3(i,j) := |\pi_i|\,|\pi_{k+j}|(K_{\mathrm{up}}(r_i,s) - K_{\mathrm{up}}(r_i, r_{k+j})).$$

In the above equations, the partition $\pi_1, \ldots, \pi_m$ corresponds to the original bins $\boldsymbol{r}$.

The same approach works to determine whether splitting bin k is beneficial. However, when performing a split, we must choose a pivot location γ at which to split. Let $\boldsymbol{r}^{(\mathrm{S})}$ be the new bins after this split. Formally,

$$\boldsymbol{r}^{(\mathrm{S})} := (r_1, \ldots, r_{k-1}, s_1, s_2, r_{k+1}, \ldots, r_m), \tag{3.10}$$

where

$$s_1 := r_k \cap R(\nu_{\mathrm{lo}}, \gamma),$$
$$s_2 := r_k \cap R(\gamma, \nu_{\mathrm{hi}}).$$

We want to know the value of $K_0(\pi)(\xi(\boldsymbol{r}^{(\mathrm{S})}, \pi) - \xi(\boldsymbol{r}, \pi))$. As before, the only difficult term to compute is

$$\Delta_{\mathrm{S}} := M_0(\boldsymbol{r}^{(\mathrm{S})}, \pi) - M_0(\boldsymbol{r}, \pi). \tag{3.11}$$

Expanding this expression, we obtain

$$\Delta_{\mathrm{S}} = -\left(\kappa_4 + \kappa_5(1) + \kappa_5(2) + \sum_{i=1; i\neq k}^{m} (\kappa_6(i,1) + \kappa_6(i,2))\right), \tag{3.12}$$

where

$$\kappa_4 := |\pi_k \cap s_1|\,|\pi_k \cap s_2|(K_{\mathrm{up}}(r_k, r_k) - K_{\mathrm{up}}(s_1, s_2)),$$
$$\kappa_5(j) := \frac{1}{2}|\pi_k \cap s_j|(|\pi_k \cap s_j| - 1)(K_{\mathrm{up}}(r_k, r_k) - K_{\mathrm{up}}(s_j, s_j)),$$
$$\kappa_6(i,j) := |\pi_i|\,|\pi_k \cap s_j|(K_{\mathrm{up}}(r_i, r_k) - K_{\mathrm{up}}(r_i, s_j)).$$

Computing Δ_{S} is less straightforward than computing Δ_{M} because we do not know the number of particles in the two portions of π_k in advance. These values appear as $|\pi_k \cap s_1|$ and $|\pi_k \cap s_2|$ in the above equations. Furthermore, if K_{up} is expensive to compute, the program may be implemented such that the values of K_{up} are stored in an array ahead of time. Thus, the values of the K_{up} expressions involving s_1 or s_2 would not be known without extra work.

A few options are available to overcome these difficulties. The pivot γ could be chosen in advance when the bin is first created. For example, γ could be the geometric (or arithmetic) mean of the two boundaries of the bin. This way, we can keep track of how many particles are in each portion of the bin as they are added. If needed, the additional K_{up} values could be stored ahead of time.

A second option is to approximate the number of particles in each portion of the bin by random sampling. The K_{up} values can also be approximated. This allows for more flexibility in the choice of pivot γ. For example, the pivot could be chosen to be an approximate median of the particles in π_k.

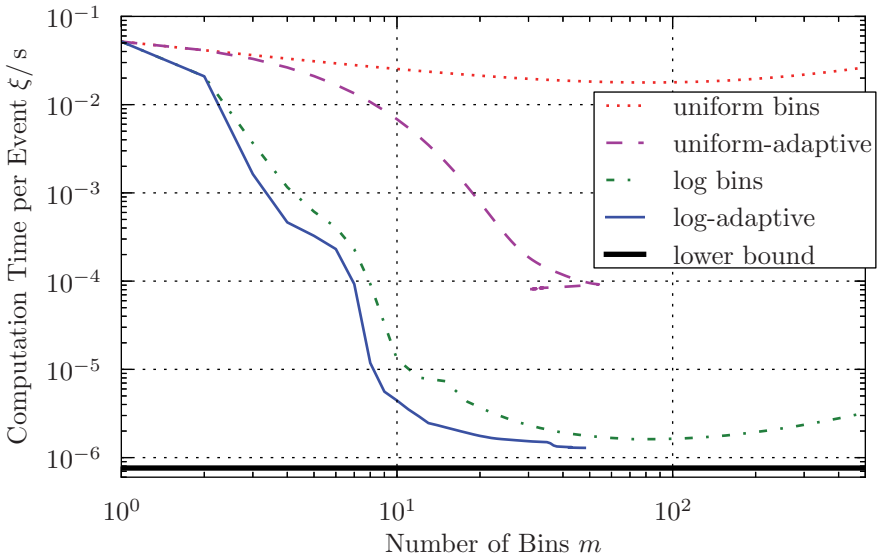

FIG. 3.2. *Computation time per event for Test Problem* 3 *using Binned SSA. Value of* ξ *is computed precisely using* (2.2) *at time* $t = 0$. *Initial distribution* π *was sampled only once. Adaptive binning scheme as described in section* 3.2 *is used, always performing most beneficial split or merge first. Uniform-adaptive and log-adaptive curves are paths that adaptive binning followed to arrive at respective destinations. Uniform-adaptive uses arithmetic mean of bin boundaries as pivot* γ. *Log-adaptive uses geometric mean of bin boundaries as pivot* γ. *Static binning schemes for various* m *are also plotted for reference. Lower bound is from* (2.4).

Now that we know how to test whether a split or merge is beneficial, we need to decide when to perform these tests. Determining whether a split or merge of bin k will be beneficial requires $O(m)$ time. Thus, checking all possible splits and merges requires $O(m^2)$ time. One approach is to check all splits and merges periodically. This could be done every $O(m^2)$ event candidates in order to mask the cost of these checks.

Figure 3.2 shows the paths adaptive binning takes in attempting to minimize ξ. It demonstrates adaptive binning with both arithmetic pivots and geometric pivots, referred to as uniform-adaptive binning and log-adaptive binning, respectively. Uniform-adaptive binning requires more splits and merges than log-adaptive binning, and it does not reach as low a cost as log-adaptive binning does. This further reinforces our earlier conclusion that logarithmic binning and logarithmic refinement of bins are good choices.

We observe that uniform-adaptive binning works considerably better than uniform-static binning, improving ξ by over two orders of magnitude. Log-adaptive binning achieves only a slightly better ξ than log-static binning. As we see from Figure 2.3, Binned SSA and Binned Tau-Leaping with log-static binning come close to the theoretical lower bound if the number of bins is chosen appropriately. This lower bound also applies to adaptive schemes. Thus, there is not much room for improvement. The overhead of adaptivity may or may not be worthwhile, depending on the kernel and the specific application.

4. Conclusion. In this paper, we have presented efficient new algorithms for simulating particle distributions evolving by stochastic events. In order to handle the multiscale nature of the particles efficiently, these algorithms aggregate particles of similar size into bins. We introduced Binned SSA, an algorithm that retains the exactness of Gillespie's Stochastic Simulation Algorithm but is much more efficient computationally for highly multiscale problems. We also introduced Binned Tau-Leaping, a variation of Gillespie's Tau-Leaping algorithm modified for use with particle simulations. Unlike Binned Tau-Leaping, Binned SSA does not require the user to choose a time-step size τ. If it is difficult to choose the value of τ (fixed or adaptively) needed to achieve a desired level of accuracy, then Binned SSA would be a good alternative. Furthermore, there are cases in which Binned SSA outperforms Binned Tau-Leaping regardless of the value of τ chosen.

The running time of either of the binned algorithms is largely determined by the placement of bin boundaries. For a static binning scheme, we showed that logarithmic binning is effective for both the Brownian coagulation kernel and the additive kernel. Adaptive binning can take advantage of the current particle distribution to increase efficiency. The effectiveness of adaptivity will vary depending on the specific problem.

The original formulation of SSA is costly for multiscale problems. By applying logarithmic particle binning to this algorithm, it becomes much more efficient and is able to compete with other algorithms in computational speed while retaining the exactness of the original algorithm.

Appendix A. Test problems and methodology.

A.1. Test problems. In our model problem, the simulation events are aerosol particle coagulations. If two particles coagulate, that means they coalesce to form a single, larger particle with volume equal to the sum of the volumes of the coagulating particles. Certain test problems of this type are referred to throughout this paper. Parameters required to specify each test problem include the following:

- N: number of particles initially present in a physically realistic model.
- n: number of computational particles initially present in the simulation.
- m: number of bins used in the simulation.
- ν_{lo} and ν_{hi}: lowest and highest bin boundaries. Bin boundaries $\nu_{\text{lo}} = \nu_0 < \nu_1 < \cdots < \nu_m = \nu_{\text{hi}}$ are spaced logarithmically unless otherwise specified.
- ν'_{hi}: upper bound on initial particle distribution.
- $F(v)$: probability density function for initial particle distribution, where v is particle volume.
- K: coagulation kernel, which determines the stochastic rate at which events occur between two given particles.
- C: initial concentration, number of particles per unit volume.
- t_{final}: total time period simulated.
- $|V|$: number of histogram buckets to represent the solution (see section A.2 below). Histogram buckets are spaced logarithmically spanning ν_{lo} to ν_{hi} unless otherwise specified.
- u: number of times simulation is repeated to decrease stochastic error by averaging results.

The magnitude operator $|\cdot|$ evaluates the volume of a given particle. Along with the volumes of two given particles, the kernel K may depend on the densities of the two particles and the volume of the simulation domain, denoted by vol. All particles have the density of water, $1000\,\text{kg/m}^3$. The volume of the simulation domain is assumed to remain the same throughout the course of the simulation. This volume

is determined by n and C,

$$\text{vol} = \frac{n}{C}. \tag{A.1}$$

The kernel K is usually the Brownian coagulation kernel K_{brown} [11]. We define this kernel as

$$K_{\text{brown}}(p,q) := \frac{4\pi\left(R_p + R_q\right)\left(D_p + D_q\right)\text{vol}^{-1}}{\frac{R_p+R_q}{R_p+R_q+\sqrt{\delta_p^2+\delta_q^2}} + \frac{4(D_p+D_q)}{\sqrt{\bar{v}_p^2+\bar{v}_q^2}(R_p+R_q)}}, \tag{A.2}$$

where

$$\delta_p := \frac{\left(2R_p + \lambda_p\right)^3 - \left(4R_p^2 + \lambda_p^2\right)^{3/2}}{6R_p\lambda_p} - 2R_p, \tag{A.3}$$

$$\lambda_p := \frac{8D_p}{\pi\bar{v}_p}. \tag{A.4}$$

In the above equations, R_p denotes the radius of p, D_p denotes the particle diffusion coefficient of p, and $\bar{v}_p$ denotes the thermal speed of particle p in air. The open source software package PartMC provides an implementation of the Brownian kernel [15].

Along with the Brownian kernel, the additive kernel

$$K_{\text{add}}(p,q) = \frac{|p| + |q|}{\text{vol}} \cdot 1.0\,\text{s}^{-1} \tag{A.5}$$

is used for one of the test problems.

The initial particle distribution comes from sampling the probability density function $F(v)$, where v is the volume of the particle in cubic meters. These mostly come from a log-normal distribution

$$G(\mu,\sigma;v) := \frac{1}{v\sigma\sqrt{2\pi}}\exp\left(-\frac{(\ln v - \mu)^2}{2\sigma^2}\right). \tag{A.6}$$

The Dirac delta function centered at zero is also used and is denoted by $\delta(v)$. If a volume sampled from the distribution falls outside $(\nu_{\text{lo}}, \nu'_{\text{hi}})$, then it is discarded and resampled.

Parameters for the test problems referenced throughout this paper are given in Table A.1 unless otherwise noted.

A.2. Testing methodology. The result of a simulation is expressed as a histogram. Particles fall into histogram buckets based on their volumes. Unless otherwise specified, the value of each histogram bucket is the number of particles in that bucket divided by n, the total number of particles initially present in the simulation. This histogram uses data only from the final simulation time, $t = t_{\text{final}}$.

We represent the histogram as a vector of real numbers $V(\tau, n, u)$, where τ is the size of the time step ($\tau = 0$ indicates that Binned SSA was used rather than Binned Tau-Leaping), n is the number of particles initially present in the simulation, and u is the number of times the simulation is repeated to decrease stochastic error by averaging results ($u = \infty$ indicates that the expected value is used, i.e., $V(\tau, n, \infty) := E[V(\tau, n, 1)]$).

Let N be the number of particles initially present in the physically realistic model that we wish to simulate. Due to limitations on computational resources, the number

TABLE A.1
Parameters for test problems.

	Test Problem 1	Test Problem 2
N	∞	-
n	10000	$1.5 \cdot 10^5$
m	10	40
ν_{lo} (m^3)	$8 \cdot 10^{-19}$	$4.189 \cdot 10^{-30}$
ν_{hi} (m^3)	$1 \cdot 10^{-14}$	$5.236 \cdot 10^{-7}$
ν'_{hi} (m^3)	$1 \cdot 10^{-14}$	$4.189 \cdot 10^{-9}$
$F(v)$	$\delta(v - 1 \cdot 10^{-18}\,\text{m}^3)$	$G(-40.82, 6.908; v)$
K	K_{add}	K_{brown}
C (m^{-3})	$1 \cdot 10^{15}$	$1 \cdot 10^{10}$
t_{final} (s)	1000	175.8
$\lvert V \rvert$	∞	-
u	1	1
	Test Problem 3	Test Problem 4
N	$1 \cdot 10^7$	1000
n	15000	1000
m	40	40
ν_{lo} (m^3)	$4.189 \cdot 10^{-30}$	$4.189 \cdot 10^{-30}$
ν_{hi} (m^3)	$5.236 \cdot 10^{-7}$	$5.236 \cdot 10^{-7}$
ν'_{hi} (m^3)	$4.189 \cdot 10^{-9}$	$4.189 \cdot 10^{-9}$
$F(v)$	see description	$G(-40.82, 6.908; v)$
K	K_{brown}	K_{brown}
C (m^{-3})	$1 \cdot 10^{11}$	$1 \cdot 10^{11}$
t_{final} (s)	10000	10000
$\lvert V \rvert$	220	220
u	1	1

of computational particles in our simulation, n, may be smaller than N. Let V' be the exact solution to the model. Then V' will match the expected outcome of Binned SSA for $n = N$. Denote the error of the simulation by ϵ_V. Formally,

$$V' = V(0, N, \infty), \tag{A.7}$$

$$\epsilon_V = \frac{||V(\tau, n, u) - V'||_2}{||V'||_2}. \tag{A.8}$$

The error ϵ_V can be broken into three terms,

$$\epsilon_V \leq \underbrace{\frac{||V(\tau, n, u) - V(\tau, n, \infty)||_2}{||V'||_2}}_{\epsilon_u} + \underbrace{\frac{||V(\tau, n, \infty) - V(\tau, N, \infty)||_2}{||V'||_2}}_{\epsilon_n} + \underbrace{\frac{||V(\tau, N, \infty) - V(0, N, \infty)||_2}{||V'||_2}}_{\epsilon_\tau},$$

where ϵ_u is the error incurred due to the stochastic nature of the simulation (which depends on u, the number of times the simulation is repeated), ϵ_n is the modeling error

due to using fewer particles than would be present in a physically realistic system, and ϵ_τ is the discretization error due to the choice of time-step size τ.

In general, the true solution V' is not known exactly. Unless otherwise specified, the solution vector V' will be approximated by $V(0, N, u)$, where u is taken sufficiently large that the plotted values of ϵ_V are accurate.

Additional relevant information on two of the test problems is given below.

- Test Problem 1: There is one histogram bucket for each positive multiple of $1 \cdot 10^{-18}\,\mathrm{m}^3$. This test problem has a known analytical solution [1], which is used to compute error.
- Test Problem 3: This problem uses a bimodal distribution,

$$F(v) = \frac{1}{2}G(-40.82, 6.908; v) + \frac{1}{2}G(-61.55, 0.9671; v).$$

  Particles from the first mode are made of species 1, and particles from the second mode are made of species 2. Once coagulations begin, particles may consist of multiple species. The value of a histogram bucket in V is the mass of species 2 in that bucket divided by n.

To test the algorithms computationally, SSA, Binned SSA, and Binned Tau-Leaping were implemented in C++, and tests were run on a standard serial desktop computer. The computation time needed to run a given simulation is denoted by t_{comp}. In the plots throughout this paper, error bars correspond to 95% confidence intervals based on Student's t-test. If horizontal or vertical error bars are absent from a plot, that means they are negligible.

REFERENCES

[1] D. J. Aldous, *Deterministic and stochastic models for coalescence (aggregation and coagulation): A review of the mean-field theory for probabilists*, Bernoulli, 5 (1999), pp. 3–48.

[2] S. E. Bauer, D. L. Wright, D. Koch, E. R. Lewis, R. McGraw, L. S. Chang, S. E. Schwartz, and R. Ruedy, *MATRIX (multiconfiguration aerosol tracker of mixing state): An aerosol microphysical module for global atmospheric models*, Atmos. Chem. Phys., 8 (2008), pp. 6003–6035.

[3] R. E. L. DeVille, N. Riemer, and M. West, *The weighted flow algorithm (WFA) for stochastic particle coagulation*, J. Comput. Phys., 230 (2011), pp. 8427–8451.

[4] M. A. Gibson and J. Bruck, *Efficient exact stochastic simulation of chemical systems with many species and many channels*, J. Phys. Chem. A, 104 (2000), pp. 1876–1889.

[5] D. T. Gillespie, *An exact method for numerically simulating the stochastic coalescence process in a cloud*, J. Atmos. Sci., 32 (1975), pp. 1977–1989.

[6] D. T. Gillespie, *Exact stochastic simulation of coupled chemical reactions*, J. Phys. Chem., 81 (1977), pp. 2340–2361.

[7] D. T. Gillespie, *Approximate accelerated stochastic simulation of chemically reacting systems*, J. Chem. Phys., 115 (2001), pp. 1716–1733.

[8] D. T. Gillespie and L. R. Petzold, *Improved leap-size selection for accelerated stochastic simulation*, J. Chem. Phys., 119 (2003), 8229.

[9] D. T. Gillespie, M. Roh, and L. R. Petzold, *Refining the weighted stochastic simulation algorithm*, J. Chem. Phys., 130 (2009), 174103.

[10] G. A. Grell, S. E. Peckham, R. Schmitz, and S. A. McKeen, *Fully coupled online chemistry within the WRF model*, Atmos. Environ., 39 (2005), pp. 6957–6975.

[11] M. Z. Jacobson, *Fundamentals of Atmospheric Modeling*, 2nd ed., Cambridge University Press, Cambridge, UK, 2005.

[12] H. Kuwahara and I. Mura, *An efficient and exact stochastic simulation method to analyze rare events in biochemical systems*, J. Chem. Phys., 129 (2008), 165101.

[13] R. Nelson, *Probability, Stochastic Processes, and Queueing Theory: The Mathematics of Computer Performance Modeling*, Springer-Verlag, New York, 1995.

[14] M. Rathinam, L. R. Petzold, Y. Cao, and D. T. Gillespie, *Stiffness in stochastic chemically reacting systems: The implicit tau-leaping method*, J. Chem. Phys., 119 (2003), 12784.

[15] N. Riemer, M. West, R. A. Zaveri, and R. C. Easter, *Simulating the evolution of soot mixing state with a particle-resolved aerosol model*, J. Geophys. Res., 114 (2009), D09202.
[16] N. Riemer, M. West, R. A. Zaveri, and R. C. Easter, *Estimating black carbon aging time-scales with a particle-resolved aerosol model*, J. Aerosol Sci., 41 (2010), pp. 143–158.
[17] P. Stier, J. Feichter, S. Kinne, S. Kloster, E. Vignati, J. Wilson, L. Ganzeveld, I. Tegen, M. Werner, Y. Balkanski, M. Schulz, O. Boucher, A. Minikin, and A. Petzold, *The aerosol-climate model ECHAM5-HAM*, Atmos. Chem. Phys., 5 (2005), pp. 1125–1156.
[18] R. A. Zaveri, J. C. Barnard, R. C. Easter, N. Riemer, and M. West, *Particle-resolved simulation of aerosol size, composition, mixing state, and the associated optical and cloud condensation nuclei activation properties in an evolving urban plume*, J. Geophys. Res., 115 (2010), D17210.
[19] R. A. Zaveri, R. C. Easter, J. D. Fast, and L. K. Peters, *Model for simulating aerosol interactions and chemistry (MOSAIC)*, J. Geophys. Res., 113 (2008), D13204.
[20] Y. Zhang, C. Seigneur, J. H. Seinfeld, M. Jacobson, and F. S. Binkowski, *Simulation of aerosol dynamics: A comparative review of algorithms used in air quality models*, Aerosol Sci. Tech., 31 (1999), pp. 487–514.