Asynchronous τ -leaping

Zbigniew Jędrzejewski-Szmek^{a)} and Kim T. Blackwell^{b)}

Krasnow Institute for Advanced Study, George Mason University, Fairfax, Virginia 22030, USA

(Received 20 November 2015; accepted 6 March 2016; published online 28 March 2016)

Stochastic simulation of cell signaling pathways and genetic regulatory networks has contributed to the understanding of cell function; however, investigation of larger, more complicated systems requires computationally efficient algorithms. τ -leaping methods, which improve efficiency when some molecules have high copy numbers, either use a fixed leap size, which does not adapt to changing state, or recalculate leap size at a heavy computational cost. We present a hybrid simulation method for reaction-diffusion systems which combines exact stochastic simulation and τ -leaping in a dynamic way. Putative times of events are stored in a priority queue, which reduces the cost of each step of the simulation. For every reaction and diffusion channel at each step of the simulation the more efficient of an exact stochastic event or a τ -leap is chosen. This new approach removes the inherent trade-off between speed and accuracy in stiff systems which was present in all τ -leaping methods by allowing each reaction channel to proceed at its own pace. Both directions of reversible reactions and diffusion are combined in a single event, allowing bigger leaps to be taken. This improves efficiency for systems near equilibrium where forward and backward events are approximately equally frequent. Comparison with existing algorithms and behaviour for five test cases of varying complexity shows that the new method is almost as accurate as exact stochastic simulation, scales well for large systems, and for various problems can be significantly faster than τ -leaping. © 2016 AIP Publishing LLC. [<http://dx.doi.org/10.1063/1.4944575>]

I. INTRODUCTION

Simulation of cell signaling pathways and genetic regulatory networks^{1–5} can provide tremendous insight into how extracellular stimuli control cell activity and gene expression. Such models are immensely complex and can easily contain hundreds of molecular species and reactions, resulting in long computations. Noise inherent in signaling networks with low copy numbers of molecules also influences the behaviour of the networks,^{6,7} resulting in a need for stochastic simulation techniques. Many systems are spatially inhomogeneous, requiring techniques for modeling diffusion. Designing efficient and accurate stochastic methods is challenging because of the complexity of models of biological systems.

The exact stochastic simulation algorithm (SSA)⁸ is one of the milestones of modeling reaction networks. The SSA simulates the evolution of the system using a Monte Carlo approach. Each execution of the SSA generates a random walk in the state space, following the chemical master equation. It provides an unbiased realization of the evolution of the system, in a simple conceptual framework. The disadvantage is that the execution time scales linearly with the number of molecules multiplied by the number of reactions, which makes it very expensive both for high copy numbers, and for the more complicated models that are often seen in realistic biological models.⁹

The need to perform stochastic simulations more efficiently has lead to many attempts to speed up the SSA. Three main directions are algorithmic improvements,^{10–12}

hybrid models which partition the system to treat fast and slow reactions in different manners,^{13–19} and forgoing exact stochastic simulation when copy numbers are high and multiple reaction events will be executed rapidly in essentially the same environment.^{20–26} Algorithmic improvements have been insufficient because of the sheer number of events happening during simulation of large models, which are getting larger as we model more and more complicated systems. Partitioning assumes a large difference between slow and fast reactions and is problematic when propensities are smoothly distributed. Forgoing exact stochastic simulation seems to be the most promising in achieving the tremendous speedups required for efficient simulation of large systems.

τ -leaping is the most commonly used approach for simulating models with high copy numbers: the state of the system is updated in steps (τ leaps) during which every reaction can fire multiple times. Step size is constrained so that the change of propensities in each step is minuscule;^{24,27,28} the number of firings is randomized in order to preserve the mean number of firings and variance. For many models τ -leaping leads to efficient simulation. Unfortunately, τ -leaping methods do not deal well with “stiff” systems, which have a large disparity between propensities of different reactions.^{25,29,30} Existing algorithms calculate a common leap size for all reactions. In effect the leap size is dictated by the most constraining reaction, and simulation of large systems is rarely efficient.

In complex biological networks many different reactions can produce or consume the same molecular species. This needs to be taken into account in the leap size calculation, because the combined firing of multiple reactions can lead

^{a)}zjedrzej@gmu.edu

^{b)}kblackw1@gmu.edu

to propensity changes which exceed the specified limits.^{21,22} Binomial τ -leaping was created to deal with this issue: the number of firings of multiple reactions in a leap are drawn from a joint distribution. Both papers also introduce the idea of drawing the number of firings from a Binomial distribution, which is more accurate than the Poisson distribution for larger leap times.

Originally exact stochastic simulation and τ -leaping were used to describe spatially uniform systems but have been extended to non-uniform systems by dividing them into subvolumes, each of which has its own population of molecules.^{21,31–33} In the spatial variant of exact stochastic simulation, reactions within a single subvolume are described in the same way as previously, but molecules are allowed to diffuse between subvolumes.¹¹ Diffusion events are described in a manner identical to reaction events, and become another reaction channel, which “transforms” a molecule of a certain species in one subvolume into a molecule of the same species in a neighbouring subvolume. The subvolumes should be small enough to satisfy the well-stirredness requirement,⁸ but significantly larger than the reaction radius.³⁴ Diffusion of molecules is usually faster than reactive molecular collisions, and with small subvolumes diffusion becomes the fastest “reaction,” which makes SSA exceedingly slow.

τ -leaping also was adapted to multiple subvolumes and diffusion of molecules^{23,35} which helps to regain simulation efficiency but brings additional computational challenges. τ -leaping algorithms iterate over all reactions in all subvolumes, so the simulation time is proportional to the number of reactions times the number of subvolumes. When calculating the leap size constraints, fast diffusion leads to very small τ -leaps. In a typical system, adjacent subvolumes have comparable concentrations of molecules, which means that diffusion to and fro tends to cancel itself out. Considering just the net flux allows bigger leaps to be taken.^{25,26}

Over time the state of the system can change significantly; therefore when we compare two different points in time, the allowed leap size also may change significantly. There are two variants of τ -leaping: those recalculating the allowed leap size at every step, and those using a fixed leap size. The latter deal ineffectively with changing state: for the leap approximation to be valid, the leap size must be dictated by the smallest allowed leap size over the course of simulation. Leaping algorithms which recalculate leap size are better in this regard, but recalculating the leap size is a heavy cost which can negate the benefits of leaping.^{24,25}

When considering cell dynamics, a signal from outside of the cell will be introduced as a high inflow of a certain molecule during a pre-defined narrow time window. When such injection starts, reaction propensities change quickly and reactions which were slow may become fast, either temporarily or permanently. Therefore, there is no fixed hierarchy of fast and slow reactions. This means that any partitioning or ordering of the system would have to be dynamically adjusted during the simulation, which makes this approach inherently complicated.

To simulate realistic biological networks and to deal with all the problems described so far we need a stochastic simulation algorithm that will deal well with millions of

“reaction” channels, large differences in reaction propensities, and dynamically changing conditions. Simulation speed becomes a constraining factor in creation of really large models, and every increase in simulation efficiency will allow us to create more complicated and biologically relevant models. Existing algorithms do not scale well enough with problem size and are not optimal for large, spatially inhomogeneous models. In this article we describe a new algorithm which combines exact stochastic simulation with τ -leaping, without partitioning the system. It is a very good approximation for exact stochastic simulations, it is significantly faster than existing τ -leaping algorithms for the complicated models describing cell reaction networks.

In Section II we describe existing exact stochastic simulation and τ -leaping algorithms because we will be using portions of those algorithms in our asynchronous algorithm. In Section III we provide formulae which describe the distribution of the number of events in a leap. In Section IV we describe the new algorithm. Section V discusses the results and compares our new algorithm to previous approaches.

II. EXACT STOCHASTIC METHODS AND τ -LEAPING

The description used here is similar to existing work,^{8,20} so only a brief restatement of notation and the most important equations will be given.

The system is divided into subvolumes. There are N species, and the state of the system at a given point in time is fully described by the population of each species in each of the subvolumes — the molecular population vector $X = X(t)$. Molecules undergo M reactions and also diffuse between subvolumes. As far as the simulation is concerned, each species in each subvolume is distinct, with diffusion being a special “reaction” which transforms a molecule in one subvolume into a molecule of the same species in the second subvolume. Therefore we can use the same formalism used to describe non-spatial stochastic systems, with the understanding that the index i of “species” iterates over both species and subvolumes, and the index j of “reaction” iterates over all reaction channels in all subvolumes and in addition over all possible channels of diffusion between neighboring subvolumes for all species.

An essential assumption is that the subvolumes are small enough, compared to the diffusion length, for the concentration gradient to be neglected inside of each subvolume, and at the same time, subvolumes are large enough to be bigger than the reaction radius. Diffusion randomizes the positions of molecules, making their individual identities and locations unimportant. Succinctly, our subvolumes are “well-stirred.” This is the basis which allows SSA to describe a real system with veracity.^{8,34}

Within this framework, each reaction channel is described by the propensity function $a_j(X)$, with $a_j(X)dt$ being the probability that reaction j will occur within the infinitesimal time interval $[t, t + dt)$. The effect of each reaction on the system is described by the stoichiometry,

$$v_{ji} = (\text{the change in population of species } i \text{ in reaction } j). \quad (1)$$

We assume that the propensity of reactions depends only on the number of molecules of each species and some reaction-specific constants. Following standard “mass-action kinetics” the propensity is given by

$$a_j = r_j \prod_{\text{species } i} X_i^{n_{ji}}, \quad (2)$$

where X^n is the “falling factorial” $X^n = X(X-1)\cdots(X-n+1)$, and n_j is the “order vector” for reaction j .

This formulation seems to be the most common.^{23,25,36,37} Some authors use a different convention in which the rate constant is divided by the number of combinations for reactions consuming multiple particles of the same species (r_j becomes $r_j/n!$ in (2)).^{8,38,39} Care must be taken to properly adjust r_j to match the convention used.

A. Exact stochastic simulation

In the exact stochastic simulation algorithm, as formulated by Gillespie,⁸ there is a single subvolume, and at each step, the time and index of the next reaction are generated based on the propensities.

When the propensity of a reaction is a , the probability density of the next event being at time t is given by the exponential distribution,

$$\mathcal{E}(t) = ae^{-at}. \quad (3)$$

Such numbers can be efficiently generated from uniform variates by using the cumulative distribution function.

In the first reaction method, which was described by Gillespie only for illustration, an exponential random variate is generated for *each* reaction using (3) with the propensity a_j from (2). Those are the *putative* times until each of the reactions, but only the earliest one is used, and all other are discarded. Though slow, this method is amenable to algorithmic improvements, as described next.

B. The next reaction method

An efficient way to implement SSA was described by Gibson and Bruck.¹⁰ It is similar to the first reaction method, but instead of iterating over the list of possible reactions at each step, reactions are kept in an indexed priority queue, with the putative firing times t_j used as keys. This gives constant-time access to the event which will fire first.

The first element in the priority queue is always the reaction with the earliest firing time. After it is executed, a new putative time is generated, and its position in the queue is updated. Then the dependency graph is used to review all dependent reactions, updating their putative firing times and possibly updating their position in the queue.

The dependency graph, shown in Fig. 1, connects those reactions whose propensity will change after the firing of a reaction and is used to find the reactions whose putative times and position in the priority queue need to be updated. The graph is created before the simulation starts by considering each pair of reactions (j, k) , the stoichiometry vector v_j , and the “order vector” n_k (Eqs. (1) and (2)). Reaction j influences reaction k , which leads to an edge in the graph, when $v_{ji}n_{ki} \neq 0$ for any species i .

Logarithmic time is required to update the priority queue after each putative time change. Thus, the next reaction method reduces the computational complexity from $O(YM)$ to $O(Y \log M)$, where Y is the number of events and M is the number of reaction channels.

Gibson and Bruck describe another optimization, whereby the putative times of dependent reactions are not generated anew but instead are updated by setting

$$t_{j,\text{new}} = \frac{a_{j,\text{old}}}{a_{j,\text{new}}}(t_{j,\text{old}} - t) + t, \quad (4)$$

where t is the time at which the last reaction fired, and t_j is the putative time of event j . This extra twist avoids the generation

FIG. 1. A graph of dependency relationships between events for a two subvolume system with two species A, B, and one reaction ($2A \leftrightarrow B$). Each reaction event in each subvolume and each diffusion event between subvolumes are nodes. Edges (red arrows) go from an event to all events whose propensity changes when it fires. For the diffusion of particle A from subvolume 0 to subvolume 1, graph edges must lead to all diffusion events of particle A from subvolumes 0 and 1, and all reactions in subvolumes 0 and 1 in which A is the reactant. Reaction $B \rightarrow 2A$ in subvolume 0 changes the propensity of diffusion of A from subvolume 0 to subvolume 1, but not the propensity of diffusion in the opposite direction. Similar considerations hold for B. Most nodes have an edge pointing to itself, since the execution of a certain reaction usually influences the propensity of that reaction. Dashed lines (black solid line) signify the relationships which turn the dependency graph into a priority queue (each child has time t larger than the parent). During simulation those edges are implied by the position in the array, as shown in Fig. 3.

of one random number, thus providing a minor optimization. The authors show that the modified procedure gives event timings which are statistically equivalent to the original Gillespie SSA.

C. The τ -leap approximation

Even with the algorithmic improvements proposed by Gibson and Bruck, exact stochastic simulation still requires time proportional to the number of events Y . This leads to poor performance, especially for spatial systems. To deal with this problem, various approximations that trade some accuracy for computational efficiency have been proposed. In the Poisson τ -leaping method²⁰ instead of considering each single firing individually, a time interval τ is selected and the number of firings of each reaction channel happening in this interval is generated by taking sample values from the Poisson distribution $\mathcal{P}$, using the mean $a_j\tau$.

Populations are updated using the stoichiometry from (1)

$$X(t + \tau) = X(t) + \sum_{\text{reaction } j} v_j \mathcal{P}(a_j \tau). \quad (5)$$

This approximation is very close to exact simulation when τ is small enough to satisfy the leap condition, which states that “change in the state will be so slight that no propensity function will suffer an appreciable (i.e., macroscopically noninfinitesimal) change in its value.”²⁰

The effect of this approximation on a single reaction in a single leap is clear: the change of propensity and thus the fraction of the population that could react during a single leap is bounded. But the systems that are being modeled are non-linear and include multiple reactions, thus, though ε still has the same meaning, when multiple leaps are combined the effect can be both bigger and smaller than ε . Though epsilon is called an accuracy control parameter, and the accuracy of the simulation is indeed monotonic with ε , there is no way to calculate the cumulative accuracy for the general case from ε .

In order to satisfy the leap condition, Gillespie-Petzold τ -leaping puts two constraints on the leap length τ .²⁷ Both the mean and the standard deviation of the change of propensity of any reaction are separately bounded to be at most a fraction ε of the total propensity $a_0 = \sum_j a_j$. This still allows for large propensity changes in individual reactions, but²⁴ updates this to bound the relative propensity of every reaction

$$\tau < \min_{\text{reaction } j} \left\{ \varepsilon \frac{a_j}{|\mu_j|}, \varepsilon^2 \frac{a_j^2}{\sigma_j^2} \right\}, \quad (6)$$

where ε is the accuracy control parameter which determines how closely the exact stochastic simulation will be approximated, and

$$f_{jk} = \sum_{\text{species } i} \frac{\partial a_j}{\partial X_i} v_{ki}, \quad (7a)$$

$$\mu_j \tau = \sum_{\text{reaction } k} f_{jk} a_k \tau = \text{mean change in } a_j, \quad (7b)$$

$$\sigma_j^2 \tau = \sum_{\text{reaction } k} f_{jk}^2 a_k \tau = \text{variance of change in } a_j. \quad (7c)$$

With those conditions, the mean and the standard deviation of the change are separately bounded by εa_j . Rare events that change propensity by more than a few ε are still possible, but now the probability is low enough for this to not be a problem in practice.⁴⁵

Let us consider the simple system with four species A, B, C, D shown in Fig. 2(a). In the initial part of the simulation the system changes quickly, and either very small steps or exact stochastic simulation is necessary to recreate proper dynamics of the initial rise of C. At this stage exact stochastic simulation is the most efficient and accurate approach. After about 30 ms leaping may start without sacrificing accuracy.⁴⁶ Comparison of the allowed leap lengths of the three reactions in Fig. 2(b) reveals a problem. Between 30 and 500 ms, conditions are favourable for leaping but allowed leaps are different. Existing τ -leaping algorithms uses the smallest allowed leap duration; the effective leap sizes are small, Fig. 2(c). As the system grows, this problem becomes more pronounced, and at any point in time there are likely to be reactions which can leap efficiently, other reactions which can leap but with smaller jumps, and finally, reactions for which leaping is inefficient, and exact events would be preferred.

D. Spatial inhomogeneity

By dividing the system into subvolumes, we allow the system to be inhomogeneous, which makes the stochastic simulation approach applicable to many more problems, in particular to biological systems in which some molecules diffuse slowly and reactions are often compartmentalized. As stated at the beginning of this section, the state vector $X(t)$ stores $N \cdot Z$ numbers, for N species and Z subvolumes. Reactions within a single subvolume are described in the same way as previously, but we also allow molecules to diffuse between subvolumes. The propensity of diffusion between subvolumes p and q is given by

$$a_{i,p \rightarrow q} = \frac{D_i}{\delta z^2}, \quad (8)$$

where δz is the distance between the subvolumes, and D_i is the diffusion coefficient for species i .²³ Reactions in separate subvolumes become effectively separate, and in addition we have to consider diffusion. The number of reaction channels becomes approximately

$$M' = MZ + 2dZ, \quad (9)$$

where d is the dimensionality of the system.

In exact methods and existing τ -leaping methods computational complexity is proportional to the number of reaction channels. For any exact algorithm it is also linear in the number of firings Y . If we assume that the number of firings scales proportionally to the size of the system, the scaling is approximately $O(M'^2)$ for the direct method, $O(M' \log M')$ for the next reaction method (a more efficient SSA implementation, Ref. 10) and $O(M')$ for all existing τ -leaping methods.

τ -leaping scales only linearly with the number of subvolumes. By design, the number of steps in τ -leaping

FIG. 2. τ -leaping is limited by the constraint that all leaps have the same length. Reactions: $A + B \xrightarrow{10^{-5}/\text{ms/nM}}$ C, $C + C \xrightleftharpoons[10^{-5}/\text{ms}]{10^{-6}/\text{ms/nM}}$ D. (a) Molecular populations as a function of time in exact stochastic simulation. (b) Leap lengths allowed under leaping constraints (6), at any given point in time, separately for each reaction. Actual leaps are made according to the *minimum* of all three bounds. (c) The effective mean number of firings in each leap, calculated as $a_j \cdot \tau$, after the leap is chosen as $\tau = \min_j \tau_j$.

is largely independent of the number of molecules, at least if the number of molecules is large enough. This means that τ -leaping should scale well for large spatial problems. The caveat is that the number of molecules must be large, across all species, and across all subvolumes. If some species have low numbers, the variance of the change of propensity becomes significant even for small leaps, requiring small time steps to observe the leap condition.

III. POPULATION CHANGE IN A LEAP

To calculate the leap conditions, we must calculate the effect of a leap on propensities of any other reaction. Two relations are needed: the first generates the number of firings for a given leap of length τ , and the second calculates the effect of one reaction firing on the propensity of a dependent reaction.

We will be using the formulae for number of firings introduced by Refs. 21 and 22 for binomial leaping. The number of *possible* firings is determined by the least populous species

$$L_j = \min_{v_{ji} < 0} X_i / v_{ji}, \quad (10)$$

and the actual number of firings is drawn from the Binomial distribution

$$\mathcal{B}(L_j, a_j \tau / L_j), \quad (11)$$

which gives a mean number of firings proportional to propensity (2)

$$y_j = a_j \tau \quad (12a)$$

and variance

$$\sigma_{y_j}^2 = a_j \tau (1 - a_j \tau / L_j). \quad (12b)$$

In certain cases, simply taking the predicted reaction rate for a given time grossly overestimates the change to the number of molecules. Probably the most important case, from the simulation efficiency perspective, is the diffusion between two subvolumes that have approximately the same number of molecules. We know that approximately the same numbers will diffuse forward and back, but only the net difference causes observable changes. Taking this into account to optimize the leap time was described by Ref. 25. As noted in Ref. 26, calculating the expected number of firings based on the difference of populations gives the correct mean, but incorrect variance. If we ignore repeated diffusion of the same molecule, the net change is exactly described by the Skellam distribution, and the mean and variance are given by $a_1 \tau - a_2 \tau$ and $a_1 \tau + a_2 \tau$, respectively.⁴⁰ The simplest way to generate leap sizes with the right mean and variance is to generate the forward j and backward j' leaps using (11) and use the difference as the effective leap size. The expected number of firings is

$$y'_j = (a_j - a_{j'}) \tau = a'_j \tau \quad (13a)$$

and the corresponding variance

$$\sigma_{y_j}'^2 = \sigma_{y_j}^2 + \sigma_{y_j'}^2. \quad (13b)$$

For the second relation, let us first consider a reaction with single species with population X . If the reaction is first order

$$a \sim rX, \quad \frac{\partial a}{\partial X}/a = \frac{1}{X}, \quad (14)$$

whereas, for higher orders

$$a \sim rX^n, \quad \frac{\partial a}{\partial X}/a = \frac{n}{X} + O\left(\frac{1}{X^2}\right). \quad (15)$$

Since the leap limit is approximate anyway, all but the first term can be ignored. When there are other substrates, their concentrations appear in the same form in the numerator and the denominator of (14) and (15) and cancel each other. So

the formula for any reaction of order n_i is

$$\frac{\partial a}{\partial X_i}/a = \frac{n_i}{X_i}. \quad (16)$$

The relationship between the number of firings of a reaction and the propensity changes in a dependent reaction must take into account the effect of changing the population of more than one reactant at once. This is relatively simple in the linear approximation, but Ref. 24 and other references use separate formulae for different orders and how many molecules of each species are consumed. We want to allow reactions up to (at least) the fourth order, which would lead to a large number of cases, so a general formula is derived for our new algorithm.

To calculate the leap condition we need to know the relative propensity change as a result of another reaction firing. In the linear approximation,

$$\begin{aligned} \frac{\Delta a_k}{a_k} &= \frac{\sum_i \frac{\partial a_k}{\partial X_i} \Delta X_i}{a_k} && \text{where sum is over all reactants } i \text{ of reaction } k \\ &= \sum_i n_{ki} \frac{\Delta X_i}{X_i} && \text{by virtue of (16)} \\ &= \sum_i \frac{n_{ki} v_{ji}}{X_i} y_j && \text{where } y_j \text{ is the number of firings of reaction } j. \end{aligned} \quad (17)$$

Turning this around, if we ask how big we can make y_j without changing dependent reactions by more than ε , we have to consider all dependent reactions k and take the smallest answer,

$$y_j^{\max} \leq \min_k \frac{\varepsilon}{\left| \sum_i \frac{n_{ki} v_{ji}}{X_i} \right|}. \quad (18)$$

The final step is to combine the limit for the maximum number of firings with the formulae for the number of firings as a function of time. Following the approach used in Refs. 24 and 27 we apply two separate limits, and similarly to (6), require τ to be such that the mean given by (12a) or (13a) and the standard deviation given by (12b) or (13b) are each smaller than the limit specified by Eq. (18)

$$\max \{y_j, \sigma_{y_j}\} < y_j^{\max}. \quad (19)$$

The limit for variance is especially important for bidirectional reactions which can have a large variance even if the mean change is zero.

A. When is leaping slower than exact?

It is common for the sum in (18) to consist of only one summand; for diffusion it is always the case. The limit on mean change then simplifies to

$$y_j^{\max} \leq \varepsilon \min_k \frac{X_i}{n_{ki} |v_{ji}|}. \quad (20)$$

For an exact event, $y_j = 1$ by definition. Thus leaping is more efficient when y_j given by (20) is larger than one. For uni-directional first order reactions which influence other first order reactions, X_i needs to be at least $1/\varepsilon$ for leaping to be advantageous, and even more for higher order reactions.

For bi-directional reactions similar considerations hold, except that a_j has to be substituted with a_j' from (13a), which makes leaps longer, and there are *two* exact events to consider; leaping is advantageous when the leap is longer than the faster of the two reactions.

For reactions which have multiple non-zero elements in the $n_j v_k$ vector, the situation is more complicated, because the sum in (20) can have both positive and negative summands.

IV. ASYNCHRONOUS τ -LEAPING METHOD

The algorithm described here combines pieces from most of the methods described previously, but the biggest gain for computational efficiency comes from using the indexed priority queue for all event channels. In Ref. 11 a priority queue was used for exact stochastic events, both reactions and diffusion. We do the same, but in addition we combine exact events and leaps in the same queue. Fig. 1 shows the conceptual view of the queue, but in the actual implementation it is simply an array, Fig. 3.

In contrast to previous algorithms, the duration of each leap is set separately, and for each event channel we consider leaping or scheduling an exact event. Each time an event

FIG. 3. A next event queue as actually stored in memory — adjusting positions in the queue exchanges the pointers in the array only. The children of a node at position j are in locations $2j+1$ and $2j+2$.

has been executed, and propensities have been recalculated, we pick the more advantageous option. Handling each event channel separately means that we only have to consider the constraints on that single event when picking the leap time. This allows both bigger leaps to be taken and is less computationally expensive.

A. Reusing event times

When propensity of an event changes, the putative time of the next occurrence also changes. For exact events, we could either generate the time anew using an exponential variate or reuse the “old” time using formula (4). As others have written (e.g., Ref. 12), especially on today’s hardware this is a minor optimization. In our implementation of the plain next reaction method, the impact is about 10%, so it is still useful.⁴⁷

In asynchronous leaping we extend the idea of updating putative times to leaps and when their propensity changes also update the putative time without changing the scheduled number of firings. We do this by assuming that the y_j events which happen during the leap happen with constant speed before the propensity change, and then again with constant speed after the propensity change. This approximation is valid because we constrain the leaps so that the propensity changes “infinitesimally.” When no propensity change occurred, the total leap time is proportional to the propensity (Eq. (12a)). If the propensity changes at time t (Fig. 4),

$$\tau_{j,\text{new}} = t + y_j \frac{\tau_{j,\text{old}} - t}{\tau_{j,\text{old}}} / a_{j,\text{new}} = \frac{a_{j,\text{old}}}{a_{j,\text{new}}} (\tau_{j,\text{old}} - t) + t, \quad (21)$$

which is the same formula as Gibson’s and Bruck’s original update formula (4).

Updating the putative time of exact events and leaps is the simplest approach computationally, because we can treat all events in the same way. In addition the efficiency

FIG. 4. Updating leap time after propensity changes at time t .

gain compared to generating a new event is bigger for leaps, because generation of a binomial variate for the number of firings is more costly than generation of an exponential variate for the putative firing time.

When calculating leap sizes, we need to consider the issue of multiple reactions which share the same dependent reaction and potentially all increase or decrease the propensity of that dependent reaction. In the original τ -leaping algorithm²⁷ as well as binomial τ -leaping proposed by Refs. 21 and 22 all reactions “fire” simultaneously (i.e., leaps are synchronized), and all populations and all propensities would be recalculated in each step. Therefore it would be possible for the propensity of that reaction to change by a multiple of ε , violating the leap constraint.

We sidestep this issue by always executing a single reaction. If two reactions modify the propensity of the same dependent reaction, the putative firing time of this dependent reaction is updated twice, using (4) or (21). In fact, it is possible for the *same* reaction to fire multiple times, causing the propensity to change and the same putative time to be repeatedly updated.

B. Efficient calculation of leap limits

Eq. (20) describes how dependent reactions limit the allowed number of firings. It is possible for the denominator in (20) to be identical for two different dependent events k . For example, in the case of diffusion, the sum always has one element, so the denominator is the same for all reactions of the same order. To take advantage of different dependent reactions being described by the same formula, before the simulation starts, we calculate the set K_j for each reaction. Each element of the set is a list of pairs $K_{jk} = (i, n_{ki} v_{ji})$ for any species i for every dependent event k . We then remove those elements of K_j which are duplicates or have the same elements but with opposite signs. Leap constraint (18) simplifies to

$$y_j^{\max} = \min_{K_{jk} \in K_j} \frac{1}{\left| \sum_{(i,z) \in K_{jk}} z / X_i \right|}. \quad (22)$$

We use this form in (19) to reduce the number of elements over which minimization is performed.

C. Description of the algorithm

A pseudo-code description of the method is presented in Algorithm 1. The priority queue is initialized by calculating

the propensity a_j for every event and generating a putative time for an exact event $t_j = \text{exponential_variate}(a_j)$, leap size $y_j = 1$. Once the queue is initialized by sorting the array of events by putative time, simulation may begin.

To reposition an element in the priority queue means to keep exchanging it with the child with smallest putative time until all children have a bigger putative time. This is the usual priority queue UPDATE procedure from Gibson and Bruck.^{10,48}

The leap length is compared with the expected time for an exact event. If the ratio is suitably large (a user specified parameter, typically between 2 and 10), a leap is queued, and if not, the time for an exact event is generated and the exact event is queued. For unidirectional leaps the leap size is non-negative, but for bidirectional leaps it may also be negative. It should be noted that no random numbers are used until after the decision whether to leap or not has been made. This way we avoid bias against short leaps.

When a leap for a bidirectional event is scheduled, we generate a number which describes the firings in both directions. The “forward” reaction has just fired, so it does not have a putative time. But the “backward” reaction has a putative exact event scheduled. We cancel that event when `reverse_is_leaping` flag is set. The propensity used in the leap properly includes the probability of “backward” reaction, so there is no bias introduced. This is similar to the case where we would regenerate all putative firing times of future reactions, as was done in the first reaction method.²⁰

V. RESULTS

There are many metrics that can be used when comparing speed. For a scientist using given software to explore a biological model, only the total run time matters. But

ALGORITHM 1.

```

current time  $t = 0$ 
while  $t < \text{desired simulation time}$ :
     $j = \text{the event at position } 0 \text{ in the priority queue}$ 
     $t = t_j$ 
    execute  $j$  by changing state  $X$  by stoichiometry  $v_j$  times  $y_j$ 
    recalculate propensity  $a_j$ 
    if event  $j$  has reverse event  $k$ :
        recalculate propensity  $a_k$ 
    if  $a_j == 0$ :
        assign putative time  $t_j = \infty$ 
    else:
        Reschedule the event either as another exact stochastic event or a leap:
        calculate leap limit  $\tau_j$  using (12) or (13) with (19) and (22)
        if  $\tau_j > 1/a_j$ :
             $t_j = t + \tau_j$ 
            generate  $y_j$  using Eq. (11)
            if event  $j$  has reverse event  $k$ :
                 $t_k = \infty$ 
                reposition  $k$  in the priority queue
        else:
             $t_j = t + \text{exponential\_variate}(a_j)$ 
             $y_j = 1$ 
            if event  $j$  has reverse event  $k$  and  $a_k > 0$ :
                 $t_k = \text{exponential\_variate}(a_k)$ 
                reposition  $k$  in the priority queue
    reposition  $j$  in the priority queue
    for every event  $k$  dependent on  $j$  except the reverse:
        old_propensity =  $a_k$ 
        calculate new  $a_k$ 
        if  $a_k == 0$ :
             $t_k = \infty$ 
        else if old_propensity == 0:
             $t_k = \text{exponential\_variate}(a_k)$ 
        else:
            update  $t_k$  using Eq. (4) or (21)
            reposition  $k$  in the priority queue
The simulation runs for a predefined time, but we usually also want to save
the state of the system periodically to the output file:
 $t_1 = \text{the putative time of the event now at position } 0$ 
if  $t_1 > \text{the time to report population } T_n$ :
    save_population( $X, T_n$ )

```

FIG. 5. Degradation reaction $A \rightarrow \phi$, with one subvolume: (a) particle numbers vs. time; (b) the difference with respect to the deterministic solution $X(t) = X_0 e^{-at}$. When leaping, the outcome is consistently biased towards leaps that are too large (because the initial propensity at the beginning of the leap is used for the whole leap); the fixed- τ and asynchronous leaping trajectories are below the deterministic one. The accuracy control parameter is set to a large value of $\varepsilon = 0.10$ for illustration purposes, and results in a maximum discrepancy of about 5% of the correct number of molecules, marked with an arrow on the graph (b). Leaps of 0.06 ms change the population by exactly 10%, so the asynchronous and fixed- τ approaches have the exact same leap sizes.

when evaluating the software itself, other measures which relate more to the computational complexity and less the quality of implementation are equally important. Therefore in comparisons shown in the rest of this article, we will be showing numbers of events (e.g., τ -leaps or “exact” events in the SSA), in addition to the simulation time. For example, for the simple simulation shown in Fig. 5 and Table I, the time to actually write the results to a file is greater than the simulation time, and to meaningfully compare various methods, simulation times are reported for executions performed without writing any results. Event counts are superior in this regard, as they are not influenced by technical details like writing results to disk, and they are also independent of the hardware, implementation quality (as long as the algorithm is executed correctly), and entirely repeatable (as long as the random seed is known).⁴⁹

In the following examples, we compare our new, asynchronous τ -leaping with the next reaction method,¹⁰ and with previous version of NeuroRD implementing fixed step τ -leaping.^{33,50} Our new method is implemented in the latest version of the NeuroRD simulator,⁵¹ which allows for easy comparisons of speed and accuracy with an existing, verified implementation, as well as with the next reaction method implementation of the SSA, available in NeuroRD by disabling leaping.

Examples range from very simple to relatively large models.⁵² If we strictly wanted to preserve the same accuracy, we should run the fixed-step τ -leaping method with the lowest value of τ -leap used in the present algorithm. This is not feasible, so we use leap sizes which give similar results.

A. Example I — Single molecule degradation

A simple model of degradation of a single molecule is shown in Fig. 5. This model has a known solution and is a “best case scenario” for fixed- τ leaping: because the reaction is first order, the leap size does not change over time. Thus, the leap size can be chosen to achieve the specified accuracy. Timing results are shown in Table I. The number of molecules is large, so exact simulation is relatively slow, and τ -leaping is fairly accurate. Note that it would be possible to “manipulate” the relative speed by simply extending the simulation time. Once the number of molecules reaches 0, exact and asynchronous leaping reach infinity at once, while fixed- τ implementation dutifully steps along at constant speed.

Asynchronous leaping and fixed- τ leaping are effectively the same in this case, executing almost the same number of leaps, so this example demonstrates the overhead of leap size calculations. Each step in the asynchronous method is about six times slower than in fixed- τ leaping, and thus

TABLE I. Comparison of simulation approaches for $A \rightarrow \phi$ shown in Fig. 5. The last column shows the biggest difference from deterministic solution, as a fraction of the population at that point (black arrow on Fig. 5). For $\tau = 0.0006$ ms and $\varepsilon = 0.001$ accuracy becomes comparable with the noise in the exact stochastic simulation. Reported times are given for a simulation run without the initial setup and the time necessary to write results to disk. Time is given for the fastest execution, and numbers of events are given for the average of 1000 runs.

Method	Leaps	Exact events	Runtime (ms)	Diff. from deterministic (%)
Exact stochastic	—	$1.2 \cdot 10^6$	390	0.1
Asynchronous $\varepsilon = 0.001$	3322	3305 ± 60	406	0.2
Asynchronous $\varepsilon = 0.01$ (Fig. 5)	333	3263 ± 50	361	0.8
Asynchronous $\varepsilon = 0.1$	34	2272 ± 51	344	5.2
Fixed $\tau = 0.0006$ ms	3334	—	64	0.4
Fixed $\tau = 0.006$ ms	334	—	52	1.0
Fixed $\tau = 0.06$ ms	34	—	51	4.9

FIG. 6. Leaps are independent in asynchronous leaping, so it is possible to make bigger steps while still satisfying the limit on propensity changes. Reactions are the same as in Fig. 2. (a) Leap lengths allowed under the leaping constraints in asynchronous leaping method at any given point in time with accuracy control parameter $\varepsilon = 0.01$. Both directions in reaction $2C \leftrightarrow D$ are handled together, so there are only two events to consider. (b) The mean number of firings (size) of each leap for the leaps shown in panel (a). Leap sizes can be compared with the dotted line corresponding to exact events which are always size 1. When leap sizes are above the dotted line, leaps are more effective than exact events. (c) Relative propensity changes of reactions after each leap from panel (b), calculated as the fraction of the propensity before and after each leap is executed. Propensity changes are, as expected, below the specified limit. We show each unidirectional reaction separately: there are two dependent reactions which constrain reaction $A + B \rightarrow C$, and only one for $2C \leftrightarrow D$.

simulation time for the asynchronous method is similar to SSA.

B. Example II — Two-step production

A simple model with three reactions was shown in Fig. 2. Fig. 6 shows the leap limits that are allowed in asynchronous leaping, for comparison with the leap sizes in “synchronous” leaping shown in Fig. 2(c). This example demonstrates the flexibility of asynchronous leaping. The population in this case is small, and asynchronous leaping with small values of

the accuracy control parameter is the same as pure SSA. For higher values of ε asynchronous leaping does significantly fewer steps than other methods, and runtimes are similar, (Table II). We take the advantage of the simplicity of this model to show asynchronous leaps: in Fig. 7 the model is simulated with $\varepsilon = 0.1$ and different step lengths are visible.

The accuracy of different methods is compared by running multiple simulations and looking at the distribution of populations at several points in time. Fig. 8 shows that the distribution of the asynchronous method is the same as

TABLE II. Comparison of timings for exact stochastic simulation, fixed- τ , and asynchronous leaping for Example II: two-step production. Time for the fastest out of 1000 runs is reported for consistency with the previous example; in this case the runtime is so small that the timings are dominated by startup overhead. The last four columns show the root of the mean square of the difference between the mean for exact and the other method divided by the S.E.M. for exact. Note that the difference between exact and either asynchronous with $\varepsilon = 0.001$ or asynchronous with $\varepsilon = 0.01$ are similar, even though no leaps are taken with $\varepsilon = 0.001$. This suggests no loss of accuracy with this example for $\varepsilon = 0.01$.

Method	Leaps	Exact events	Runtime (ms)	Diff. from exact			
				A	B	C	D
Exact stochastic (Fig. 2(a))	—	1582 ± 10	56	—	—	—	—
Asynchronous $\varepsilon = 0.001$	0	1583 ± 11	80	1.6	1.3	0.9	1.0
Asynchronous $\varepsilon = 0.01$ (Fig. 7(a))	428 ± 3	1150 ± 11	82	1.1	1.2	1.3	1.5
Asynchronous $\varepsilon = 0.1$ (Fig. 7(b))	1535 ± 9	43 ± 2	80	11	13	13	23
Fixed $\tau = 0.5$ ms	4000	—	63	2.1	2.5	1.9	2.4
Fixed $\tau = 2.5$ ms	800	—	49	10.2	11.1	7.8	5.9

FIG. 7. Demonstration of asynchronous leaping: for a suitably large value of ε the leaps become visible, slightly different for each reaction.

the exact method when $\varepsilon = 0.001$, whereas the fixed- τ has a different distribution for molecule C at 50 ms (Fig. 8(b)) even for $\tau = 0.5$ ms. Thus, even for this small model, the asynchronous algorithm exhibits an improvement in speed over the exact method and an improvement in accuracy over the fixed τ -leap method. However, the strength of the algorithm is more apparent from simulations of larger models.

FIG. 8. Distribution of results at four different points for the example in Figs. 2 and 6. Histograms of population at different time points for (a) species A, ((b) and (c)) species C, (d) species D, calculated from 2^{13} simulations for each method. The asynchronous algorithm is as accurate as the exact algorithm with $\varepsilon = 0.001$. Table II contains the averages over the whole simulation.

C. Example III — A + B annihilation

This example has been used several times to test exact stochastic simulation and different leaping approaches.^{11,25,31} Molecules A and B diffuse, and their reaction annihilates both of them. As expected, asynchronous leaping with $\varepsilon = 0.01$ is very similar to exact stochastic simulation, Fig. 9.

In Table III the initial numbers of molecules of species A and B are scaled between 10^2 and 10^7 to explore the domain in which asynchronous leaping provides computational efficiency. For very small number of molecules asynchronous leaping is *slower* than exact simulation, because of the additional computation overhead which gives no advantages. The case shown in Fig. 9 allows a tiny amount of leaping, again without significant advantages. For higher molecule numbers exact stochastic simulation time grows slightly slower than linearly, while asynchronous leaping becomes faster.

D. Example IV — Production of cAMP stimulated by Da

More biologically relevant is the following model describing dopamine-stimulated (Da) production of cyclic adenosine monophosphate (cAMP) via adenylyl cyclase (AC), and the feedback loop limiting cAMP elevation: cAMP activation of protein kinase A (PKA) with enhancement of phosphodiesterase (PDE10) activity caused by PKA phosphorylation. This is a subset of the reactions involved in the induction of synaptic plasticity in the striatum.^{41–43}

FIG. 9. Comparison of exact stochastic simulation and asynchronous leaping in the A+B annihilation model. Molecules can both react and diffuse, and accurate simulation of both event types is necessary. Asynchronous leaping (right column) with accuracy control parameter $\varepsilon = 0.01$ gives very similar results to the exact stochastic simulation algorithm (left column). The system consists of a 1D domain of length 40, divided into 100 subvolumes, shown on the x axis. Time is shown on the y axis. Initially $N = 1000$ molecules of A (upper row) are evenly distributed, and N molecules of B (lower row) are located in the leftmost subvolume. Molecules annihilate each other in the reaction $A + B \xrightarrow{10} \phi$ and diffuse with the diffusion coefficient $D = 5$. An average of 1000 runs is shown, individual simulations are much more noisy. The R.M.S. of the difference between the mean for asynchronous and exact divided by S.E.M. of exact is 1.2 for A, 1.4 for B.

TABLE III. Comparison of timings for exact stochastic simulation and asynchronous leaping for Example III. The number of molecules is varied to explore how exact events are replaced by leaping.

Number of molecules	Exact		Asynchronous $\varepsilon = 0.01$		
	Exact events	Runtime	Leaps	Exact events	Runtime (s)
10^2	$2.6 \cdot 10^5$	0.5 s	0	$2.6 \cdot 10^5$	0.7
10^3 (Fig. 9)	$2.5 \cdot 10^6$	3 s	23	$2.5 \cdot 10^6$	3
10^4	$2.5 \cdot 10^7$	27 s	$3.8 \cdot 10^4$	$2.4 \cdot 10^7$	31
10^5	$2.5 \cdot 10^8$	225 s	$2.2 \cdot 10^6$	$4.9 \cdot 10^7$	63
10^5	$2.5 \cdot 10^9$	35 min	$6.3 \cdot 10^5$	$1.0 \cdot 10^7$	13
10^6	$2.5 \cdot 10^{10}$	321 min	$6.5 \cdot 10^5$	$1.4 \cdot 10^7$	14

The morphology consists of three subvolumes, $1 \times 1 \times 0.25 \mu\text{m}^3$, $1 \times 1 \times 1 \mu\text{m}^3$, $1 \times 1 \times 0.25 \mu\text{m}^3$. Diffusion constants are $k_{\text{Da}} = 10 \mu\text{m}^2/\text{s}$, $k_{\text{ATP}} = 74.8 \mu\text{m}^2/\text{s}$, $k_{\text{cAMP}} = k_{\text{AMP}} = 86.4 \mu\text{m}^2/\text{s}$, $k_{\text{PKAc}} = 0.836 \mu\text{m}^2/\text{s}$; other species do not diffuse. Initial concentrations are $[\text{Da}] = 10 \text{ nmol/l}$, $[\text{ATP}] = 2 \cdot 10^6 \text{ nmol/l}$, $[\text{cAMP}] = 36 \text{ nmol/l}$, $[\text{PDE10}] = 120 \text{ nmol/l}$, $[\text{PDE2}] = 600 \text{ nmol/l}$, $[\text{D1R}] = 0.2 \text{ nmol/m}^2$, $[\text{Gsabg}] = 2 \text{ nmol/m}^2$, $[\text{AC}] = 1 \text{ nmol/m}^2$, $[\text{PDE10}] = 0.1 \text{ nmol/m}^2$, $[\text{PKA}] = 1 \text{ nmol/m}^2$, where volume concentrations apply to all three subvolumes, while surface concentrations apply only to the two surface subvolumes, and the number of molecules in a subvolume is given by the side area of $1 \mu\text{m}^2$ multiplied by density. At time $t = 10^5 \text{ ms}$ Da is injected into the middle subvolume for 10 ms at the speed of 361.4 particles/ms.

In this model, one of the dominant features is the large number of ATP and cAMP molecules. Diffusion of ATP significantly slows exact stochastic simulation; without

leaping for the most frequent events, simulation speed suffers greatly. On the opposite end of the spectrum, molecules such as pPDE10 and its complexes with PKA and cAMP, which appear with counts in single or double digits in each subvolume, make τ -leaping ineffective. Timing results are shown in Table IV.

This model is highly non-linear, with multiple positive and negative feedback loops, which means that accurate simulation of some of those species can influence the result drastically. With the accuracy control parameter $\varepsilon = 0.01$, we get about 11 200-fold speed up over exact stochastic simulation, while still producing similar results, Fig. 10.

To explore the influence of the model size on simulation speed we grew the morphology in one dimension, thus changing the number of subvolumes. Since reactions are repeated in every subvolume, the number of reaction channels (Fig. 1) grows linearly with the number of subvolumes. The runtime of different simulation algorithms scales differently:

TABLE IV. Production of cAMP via AC stimulated by Da: comparison of timings for exact stochastic simulation, fixed- τ , and asynchronous leaping with three subvolumes and 100 s of simulation. Time for the fastest out of 12 runs is reported without initialization overhead (except for exact stochastic simulation, where only one run was performed). The last column shows the average number of position adjustments per exact firing or leap for algorithms using the priority queue.

Method	Number of leaps	Number of exact events	Runtime	Swaps/event
Exact stochastic	—	$9.2 \cdot 10^{11}$	168 h	1.12
Asynchronous $\varepsilon = 0.001$	$4.1 \cdot 10^7$	$1.2 \cdot 10^8$	261 s	2.20
Asynchronous $\varepsilon = 0.01$	$7.9 \cdot 10^5$	$3.8 \cdot 10^7$	54 s	2.30
Asynchronous $\varepsilon = 0.1$	$1.8 \cdot 10^5$	$2.1 \cdot 10^7$	37 s	2.65
Fixed $\tau = 0.005 \text{ ms}$	$1.15 \cdot 10^{10}$	—	787 s	
Fixed $\tau = 0.05 \text{ ms}$	$1.15 \cdot 10^9$	—	78 s

FIG. 10. Production of cAMP via AC stimulated by Da: comparison of populations in exact stochastic simulation, fixed- τ , and asynchronous leaping, for a subset of more interesting particles. For all three algorithms a smoothed mean based on 12 simulations is shown, and additionally for exact stochastic simulation the standard deviation of the mean is shown as a light green range. Y axis shows molecule numbers.

FIG. 11. Scaling of the simulation time as a function of size for the model of Fig. 10. Model size M' is defined as the number of reaction channels (including diffusion). For asynchronous leaping this is equal to the size of the dependency graph.

FIG. 12. Bistable model which switches stochastically between two stable solutions. The top panel shows exact stochastic simulation, the bottom panel shows asynchronous leaping. In both panels, upstates are marked with a solid gray background. For this accuracy setting ($\varepsilon = 0.05$), it is difficult to observe the minor deviation in statistical properties, such as the decrease in upstate lifetime shown in Fig. 13(e).

fixed- τ leaping repeatedly loops over all reaction channels, so runtime grows linearly with the model size; SSA depends on the actual number of firings; asynchronous leaping depends on both the number of events and the number of reaction channels. When leaping is effective, i.e., when there are enough molecules for allowed leaps to be bigger than the average exact event length, the runtime of asynchronous leaping is proportional to the number of *active* reaction channels. When leaping is not effective, asynchronous leaping “degrades” to exact stochastic simulation.

In Fig. 11 the initial number of molecules increases proportionally to the morphology size, and the number of actual events in the simulation is approximately proportional to the model size. The runtime of asynchronous leaping grows slightly faster than the number of events, while fixed- τ leaping scales exactly proportionally to the model size. If some event channels were less active and the number of events grew more slowly, asynchronous leaping would become more efficient.

E. Example V — Bistable production–annihilation model

This model was introduced by Ref. 44 and was analyzed in detail in Refs. 24 and 37. A single-subvolume with the species A undergoes the reactions $A \xrightleftharpoons[2200/s]{37.5/s} \phi$, $3A \xrightleftharpoons[0.18/s/nM]{0.00025/s/nM/nM} 2A$. The stochastic solution exhibits two stable regions, around 128 and 455 molecules, and chaotic switching between them, Fig. 12. The bistability is dependent on stochastic noise, and thus this is an excellent test case to check the accuracy of the

new asynchronous algorithm. Fig. 13 shows that asynchronous and exact simulation exhibit similar dynamics for small leap sizes and deviate as the leaps get bigger. The population in this model is low, and there are only four reactions to consider, so asynchronous leaping is not expected to be particularly fast. The timing results show that leaping does not lead to increased efficiency, but neither does it degrade performance significantly, and for customary values of the accuracy control parameter, the results do not deviate from exact stochastic simulation.

VI. DISCUSSION

We describe a new asynchronous τ -leaping stochastic simulation algorithm, which is designed to deal with stiff systems efficiently. It combines leaps and exact events in a single priority queue and chooses the correct way to handle a given reaction (or diffusion) event based on the population of its substrates, and the influence it will have on the dependent propensities. This could be described as “dynamic partitioning,” though there is no explicit partitioning of the system. Previous approaches explored the idea that some event channels are better treated as exact stochastic events,^{15,24} but the leap events and exact stochastic events still used the same time step. By allowing them to proceed asynchronously, our algorithm deals with both kinds of stiffness described in the introduction: first, some populous species can diffuse in leaps of a million particles between two subvolumes, while slow or sensitive reactions are simulated exactly; second, some reactions can be simulated exactly while its substrates have

FIG. 13. Comparison of exact stochastic simulation (green), asynchronous leaping (red) as a function of the accuracy control parameter ε , and fixed- τ leaping (blue) as a function of the leap size τ , in the Schlögl bistable model. Plot (a) shows how average event length (calculated as the simulated time of 10^5 s divided by the number of exact events and leaps) depends on the accuracy control parameter ε . It should be noted that the values of ε above 0.01 are normally not recommended, and values up to 0.05 are used here for illustration purposes. In all plots the (single) value for exact stochastic simulation is shown as a horizontal band. Plot (b) shows that asynchronous leaping is *slower* than exact simulation for small values of ε . This is a result of the additional overhead of calculating leap constraints, even though the population is low and no leaping occurs. For higher ε , leaping increases simulation efficiency. The runtime for fixed- τ leaping, not shown, is inversely proportional to the size of the step, and for the points shown on the graph, varies between 190 min for 0.01 ms steps and 6 min for 0.3 ms steps. Plots (c) and (d) show the average population in the two semi-stable states. For fixed- τ leaping, the average event length is simply the leap size, and for asynchronous leaping, the average event length from plot (a) is used. The average *leap* length for asynchronous leaping is actually much higher than the average event length. Using this measure allows us to compare both leaping approaches. For small ε values asynchronous leaping has results very similar to exact. As the leap size increases, both approaches deviate from exact stochastic simulation, up to the point (not shown) where the oscillation between the two solutions ceases completely. Plots (e) and (f) show the average lifetime. When the population in the upper state shown in (c) is greater, the average lifetime is bigger too. Similarly, when population in the lower state is larger, making it easier to jump to the other state, the average lifetime in the downstate is smaller.

low copy numbers, and can seamlessly segue into leaping once the conditions are right.

Several timing tests show that asynchronous τ -leaping performs almost as efficiently as fixed- τ leaping in conditions that are optimal for fixed- τ leaping (Example I) and performs better than fixed- τ leaping and significantly better than exact stochastic simulation in conditions in which exact stochastic simulation is necessary to properly simulate some of the reactions and τ -leaping is necessary to efficiently simulate others (Example IV).

The use of a priority queue allows larger systems to be simulated. The time required for each step is approximately logarithmic in the number of active reaction channels. For complex biological systems like Example IV this is crucial, because the simulation does not suffer a slow down for each diffusing species or reaction that is added.

Example IV consists of 149 reaction channels (including 20 diffusion channels). When simulation starts, the propen-

sities of only 25 of them are non-zero. Over the course of simulation, most of those other channels are “activated,” and move up in the priority queue; but until that happens they do not contribute to the computational cost. The number of “swaps,” i.e., changes of event order in the priority queue, is an indicator of how much overhead is incurred by updates of the priority queue. The queue depth is 150, so about $\log_2 150/2 \approx 3.5$ swaps would occur for each event if the swaps were between random events. Instead 2.3 swaps occur per event for $\varepsilon = 0.01$, because the propensity of events does not change rapidly, and they tend to stay at the same depth of the priority queue. On the other hand, this number is much larger than in the next reaction method implementation of SSA. In this model, ATP diffusion, which causes exact stochastic simulation to be so slow, also causes the number of swaps per event to be ≈ 1 , because the ATP diffusion event permanently occupies the top of the priority queue. Effectively, in asynchronous leaping, we can expect

scaling that is better than logarithmic in the number of events.

We can compare this result with the optimized direct method.¹² In this set of algorithmic optimizations to the direct method, performance was significantly increased by presorting reactions by their propensity. One step of the direct method computes a running sum of propensities $\sum_{j'=0}^J a_{j'}$, which is at each step compared with a random number to determine the reaction to fire. By sorting according to propensity, the highest $a_{j'}$ were summed first, and only a smaller number of components had to be summed. As a result, the time required for this substep was reduced, and instead of being proportional to the total number of reaction channels, it was comparable to (half) the average count of most frequent reactions. With asynchronous leaping and its priority queue something similar happens, except that the speedup is from logarithmic to sub-logarithmic and not from linear to sub-linear. A second difference is that the queue is dynamic, so sorting is constantly updated during the simulation and works even if reaction propensities change significantly.

Let us put computational complexity considerations aside and turn to ease of use. An important aspect of using a simulation algorithm for actual research is how much knowledge about the model dynamics is required, and how many parameters have to be adjusted. In the asynchronous leaping algorithm, the accuracy control parameter ε is the only control knob. This is superficially similar to standard τ -leaping, where either ε or the fixed τ is specified, but asynchronous leaping should be more robust, because exact stochastic simulation is used when necessary. Also, there is no population threshold below which reactions are treated differently. During simulation, the time until an exact event and the allowed leap are compared. Choosing one or the other affects only the speed of simulation, since very short leaps simply result in having $\ll 1$ events per leap. This is possible because we remove the inherent trade-off between speed and accuracy in stiff systems which was present in all τ -leaping methods, whether fixed or variable, by allowing each reaction channel to proceed at its own pace. A simulation algorithm which is simpler to use while scaling better for large problems will allow more realistic and thereby more interesting scientific problems to be solved. For large biological networks our implementation is more efficient than the existing algorithms, without sacrificing accuracy.

In the future we hope to extend our approach to scale to even bigger models. In the current approach, τ -leaping constraints and the randomized numbers of firings are based on a linear approximation. Using higher order approximations would allow bigger leaps. This non-linearity would complicate the analysis of interactions between different reactions as well as updating leap duration as a result of changes of propensity.

Automatic determination of the accuracy control parameter ε should make the algorithm easier to use. The value of ε has a fairly general meaning which is not tied to the specific model as much as the actual leap sizes are. For simple problems ε can be used directly to estimate the accuracy of results, but for more complicated problems the relationship

is less clear. We intend to replace ε with a control measure which would determine the allowed leap size based on the estimated variances of reactions.

ACKNOWLEDGMENTS

This work was supported by the NIH-NSF program on Collaborative Research in Computational Neuroscience through NIH Grant No. R01 AA16022. The authors are grateful to Jane Wendelin for providing financial management of the project.

- ¹P. Jain and U. S. Bhalla, *PLoS One* **9**, e95154 (2014).
- ²D. Barik, W. T. Baumann, M. R. Paul, B. Novak, and J. J. Tyson, *Mol. Syst. Biol.* **6**, 405 (2010).
- ³Y.-Y. Tseng, S. M. Hunt, C. Heintzen, S. K. Crosthwaite, and J.-M. Schwartz, *PLoS Comput. Biol.* **8**, e1002437 (2012).
- ⁴D. Wang, L. Cardelli, A. Phillips, N. Piterman, and J. Fisher, *BMC Syst. Biol.* **3**, 118 (2009).
- ⁵R. Srivastava, E. L. Haseltine, E. Mastny, and J. B. Rawlings, *J. Chem. Phys.* **134**, 154109 (2011).
- ⁶C. V. Rao, D. M. Wolf, and A. P. Arkin, *Nature* **420**, 231 (2002).
- ⁷M. J. Lawson, B. Drawert, M. Khammash, L. Petzold, and T.-M. Yi, *PLoS Comput. Biol.* **9**, e1003139 (2013).
- ⁸D. T. Gillespie, *J. Comput. Phys.* **22**, 403 (1976).
- ⁹J. Intosalmi, T. Manninen, K. Ruohonen, and M.-L. Linne, *BMC Bioinf.* **12**, 252 (2011).
- ¹⁰M. A. Gibson and J. Bruck, *J. Phys. Chem. A* **104**, 1876 (2000).
- ¹¹J. Elf and M. Ehrenberg, *Syst. Biol.* **1**, 230 (2004).
- ¹²Y. Cao, H. Li, and L. Petzold, *J. Chem. Phys.* **121**, 4059 (2004).
- ¹³T.-H. Ahn, Y. Cao, and L. T. Watson, in *Proceedings of the 2008 International Conference on Bioinformatics and Computational Biology, BIOCOMP'08* (Las Vegas, NV, 2008), pp. 431–436.
- ¹⁴C. V. Rao and A. P. Arkin, *J. Chem. Phys.* **118**, 4999 (2003).
- ¹⁵Y. Cao, D. T. Gillespie, and L. R. Petzold, *J. Chem. Phys.* **123**, 054104 (2005).
- ¹⁶H. Salis and Y. Kaznessis, *J. Chem. Phys.* **122**, 054103 (2005).
- ¹⁷A. Ciliberto, F. Capuani, and J. J. Tyson, *PLoS Comput. Biol.* **3**, e45 (2007).
- ¹⁸S. L. Cotter, K. C. Zygalakis, I. G. Kevrekidis, and R. Erban, *J. Chem. Phys.* **135**, 094102 (2011).
- ¹⁹J. S. Hogg, L. A. Harris, L. J. Stover, N. S. Nair, and J. R. Faeder, *PLoS Comput. Biol.* **10**, e1003544 (2014).
- ²⁰D. T. Gillespie, *J. Chem. Phys.* **115**, 1716 (2001).
- ²¹T. Tian and K. Burrage, *J. Chem. Phys.* **121**, 10356 (2004).
- ²²A. Chatterjee, D. G. Vlachos, and M. A. Katsoulakis, *J. Chem. Phys.* **122**, 024112 (2005).
- ²³K. T. Blackwell, *J. Neurosci. Methods* **157**, 142 (2006).
- ²⁴Y. Cao, D. T. Gillespie, and L. R. Petzold, *J. Chem. Phys.* **124**, 044109 (2006).
- ²⁵W. Koh and K. T. Blackwell, *J. Chem. Phys.* **134**, 154103 (2011).
- ²⁶W. Koh and K. T. Blackwell, *J. Chem. Phys.* **137**, 154111 (2012).
- ²⁷D. T. Gillespie and L. R. Petzold, *J. Chem. Phys.* **119**, 8229 (2003).
- ²⁸C. A. Yates and K. Burrage, *J. Chem. Phys.* **134**, 084109 (2011).
- ²⁹X. Cai and Z. Xu, *J. Chem. Phys.* **126**, 074102 (2007).
- ³⁰Y. Xu and Y. Lan, *J. Chem. Phys.* **137**, 204103 (2012).
- ³¹S. Lampoudi, D. T. Gillespie, and L. R. Petzold, *J. Chem. Phys.* **130**, 094104 (2009).
- ³²G. Kalantzis, *Comput. Biol. Chem.* **33**, 205 (2009).
- ³³R. F. Oliveira, A. Terrin, G. Di Benedetto, R. C. Cannon, W. Koh, M. Kim, M. Zaccolo, and K. T. Blackwell, *PLoS One* **5**, e11725 (2010).
- ³⁴S. A. Isaacson and D. Isaacson, *Phys. Rev. E* **80**, 066106 (2009).
- ³⁵K. A. Iyengar, L. A. Harris, and P. Clancy, *J. Chem. Phys.* **132**, 094101 (2010).
- ³⁶C. Capellos and B. H. Bielski, *Kinetic Systems: Mathematical Description of Chemical Kinetics in Solution* (Wiley-Interscience, New York, 1972).
- ³⁷R. Erban and S. J. Chapman, *Phys. Biol.* **4**, 16 (2007).
- ³⁸D. A. McQuarrie, *J. Chem. Phys.* **38**, 433 (1963).
- ³⁹D. T. Gillespie, *J. Chem. Phys.* **113**, 297 (2000).
- ⁴⁰J. G. Skellam, *J. R. Stat. Soc.* **109**, 296 (1946).
- ⁴¹P. Calabresi, P. Gubellini, D. Centonze, B. Picconi, G. Bernardi, K. Chergui, P. Svenningsson, A. A. Fienberg, and P. Greengard, *J. Neurosci.* **20**, 8443 (2000).

⁴²E. Fino, V. Paille, Y. Cui, T. Morera-Herreras, J.-M. Deniau, and L. Venance, *J. Physiol.* **588**, 3045 (2010).

⁴³S. L. Hawes, F. Gillani, R. C. Evans, E. A. Benkert, and K. T. Blackwell, *J. Neurophysiol.* **110**, 2027 (2013).

⁴⁴F. Schlögl, *Z. Phys.* **253**, 147 (1972).

⁴⁵The equations limiting τ given in Ref. 24 contain an additional *lower* bound on the term on the right side which allows each leap to have at least one firing. In our new algorithm described in Sec. IV we simply execute a single exact event in such conditions, so this lower bound is not necessary.

⁴⁶For illustration purposes, let's set the accuracy control parameter to a fairly large value $\varepsilon = 0.1$ and compare exact stochastic events and τ -leaps. Despite the high value of ε , expected time until the first reaction is 0.048 ms, but the maximum allowed leap size is 0.0048 ms. This means that we would need to do about 10 leaps before the first C molecule is produced. After that happens, the expected time until the second reaction drops infinitesimally, and the maximum allowed leap size grows to 0.0096 ms, which still means we need to do multiple leaps before second C is produced. Only when we have more than 20 molecules of C, we expect more than one firing in each leap.

⁴⁷It can be disabled with the `stochdiff.neq.update_times` boolean setting.

⁴⁸“UPDATE_AUX” has an error in line 7: by the time this line is executed, the swap has been performed and “minimum child(n)” is not necessarily the one that should be acted upon. The algorithm should read as

```
UPDATE_AUX(node n)
If Value(n) < Value(parent(n))
    SWAP(n, parent(n))
    UPDATE_AUX(parent(n))
Else If Value(n) > minimum value(children(n))
    m = minimum child(n)
    SWAP(n, m)
UPDATE_AUX(m).
```

⁴⁹Results shown in graphs are from trials which *do* actually write results to disk.

⁵⁰In NeuroRD source repository, in the `src/main/java/org/textensor/stochdiff/numeric/grid` subdirectory, `SteppedStochasticGridCalc.java` file contains the implementation of fixed- τ leaping, and `AdaptiveGridCalc.java` together with `NextEventQueue.java` contain the implementation of asynchronous leaping and next reaction methods. In the XML files describing simulations, `<calculation/>` element can be set to `GRID_STEPPED_STOCHASTIC`, `GRID_ASYNC`, and `GRID_EXACT`, for fixed- τ , asynchronous leaping, and SSA, respectively.

⁵¹This article describes NeuroRD version 3.0; sources are available at <https://github.com/neurord/stochdiff/releases/tag/v3.0>, under the GNU Public License version 2 or any later. This software is implemented in Java and can be compiled in the standard manner using Maven.

⁵²The model files for examples used in this article can be downloaded from <https://github.com/neurord/jcp2016-examples>.