SIAM J. SCI. COMPUT. © 2016 Society for Industrial and Applied Mathematics
Vol. 38, No. 4, pp. A2091–A2117

A MULTILEVEL ADAPTIVE REACTION-SPLITTING SIMULATION METHOD FOR STOCHASTIC REACTION NETWORKS[*]

ALVARO MORAES[†], RAUL TEMPONE[†], AND PEDRO VILANOVA[†]

Abstract. In this work, we present a novel multilevel Monte Carlo method for kinetic simulation of stochastic reaction networks characterized by having simultaneously fast and slow reaction channels. To produce efficient simulations, our method adaptively classifies the reactions channels into fast and slow channels. To this end, we first introduce a state-dependent quantity named level of activity of a reaction channel. Then, we propose a low-cost heuristic that allows us to adaptively split the set of reaction channels into two subsets characterized by either a high or a low level of activity. Based on a time-splitting technique, the increments associated with high-activity channels are simulated using the tau-leap method, while those associated with low-activity channels are simulated using an exact method. This path simulation technique is amenable for coupled path generation and a corresponding multilevel Monte Carlo algorithm. To estimate expected values of observables of the system at a prescribed final time, our method bounds the global computational error to be below a prescribed tolerance, TOL, within a given confidence level. This goal is achieved with a computational complexity of order $O(TOL^{-2})$, the same as with a pathwise-exact method, but with a smaller constant. We also present a novel low-cost control variate technique based on the stochastic time change representation by Kurtz, showing its performance on a numerical example. We present two numerical examples extracted from the literature that show how the reaction-splitting method obtains substantial gains with respect to the standard stochastic simulation algorithm and the multilevel Monte Carlo approach by Anderson and Higham.

Key words. error estimates, error control, control variates, weak approximation, hybrid algorithms, multilevel Monte Carlo, Chernoff tau-leap, adaptive reaction splitting

AMS subject classifications. 60J75, 60J27, 65G20, 92C40

DOI. 10.1137/140972081

1. Introduction.

Stochastic reaction networks (SRNs) are a class of Markovian pure jump processes, $X : [0,T] \times \Omega \to \mathbb{Z}_+^d$, frequently used for modeling stochastic biochemical reaction networks. Consider a biochemical system of d species, $(S_1, S_2, \dots, S_d)$, interacting through J different reaction channels, $(\mathcal{R}_1, \mathcal{R}_2, \dots, \mathcal{R}_J)$, where each reaction channel is a pair, $\mathcal{R}_j = (\nu_j, a_j(x))$, with $\nu_j \in \mathbb{Z}^d$ and $a_j : \mathbb{R}_+^d \to \mathbb{R}_+$. In the biochemical literature, the vectors ν_j are known as stoichiometric vectors and the functions a_j as propensities. Let $X_i(t) \equiv X_i(t,\omega)$ be the number of particles of species S_i in the system at time t. The evolution of the state vector, $X(t) = (X_1(t), \dots, X_d(t)) \in \mathbb{Z}_+^d$, is modeled as a continuous-time Markov chain starting at a deterministic, known state $x_0 = X(0) \in \mathbb{Z}_+^d$. When the reaction channel $\mathcal{R}_j$ fires, the current state of the system, x jumps to $x + \nu_j$. The probability that reaction $\mathcal{R}_j$ fires during the small interval $(t, t+\mathrm{d}t)$ is given by

$$\mathrm{P}\left(X(t+\mathrm{d}t) = x + \nu_j \,|\, X(t) = x\right) = a_j(x)\mathrm{d}t + o\left(\mathrm{d}t\right). \tag{1.1}$$

Due to the stochastic mass action kinetics principle [11], we often consider polynomial propensity functions, a_j. To preclude the existence of states with negative components

[*]Submitted to the journal's Methods and Algorithms for Scientific Computing section June 10, 2014; accepted for publication (in revised form) April 27, 2016; published electronically July 7, 2016. This work was supported by King Abdullah University of Science and Technology (KAUST).
http://www.siam.org/journals/sisc/38-4/97208.html

[†]King Abdullah University of Science and Technology (KAUST), Computer, Electrical and Mathematical Science and Engineering, Thuwal, Saudi Arabia (alvaro.moraesgutierrez@kaust.edu.sa, raul.tempone@kaust.edu.sa, pedro.guerra@kaust.edu.sa).

we set $a_j(x){=}0$ for those x such that $x{+}\nu_j \notin \mathbb{Z}_+^d$. The process X admits the following random time change representation by Kurtz [12]:

$$X(t) = X(0) + \sum_{j=1}^{J} \nu_j Y_j \left(\int_0^t a_j(X(s))\,\mathrm{d}s \right), \tag{1.2}$$

where $Y_j : \mathbb{R}_+ {\times} \Omega \to \mathbb{Z}_+$ are independent unit-rate Poisson processes.

Among the phenomena usually modeled by SRNs we mention biochemical reactions at thermal equilibrium and constant volume, spread of epidemic diseases in homogeneously mixed populations, natural and artificial neural networks, pharmacokinetics, virus kinetics, and dynamics of social networks (see [26], [4], [28], [17]).

The main goal of this work is to estimate the expected value, $\mathrm{E}[g(X(T))]$, where $g : \mathbb{R}_+^d \to \mathbb{R}$ is a given real observable of X. In principle, it is possible to approximate $\mathrm{E}[g(X(T))]$ using the Monte Carlo method by generating a number of independent pathwise-exact simulations of X up to time T. Pathwise-exact simulations of X can be obtained using the stochastic simulation algorithm (SSA) [15] or the modified next reaction method (MNRM) [2], among other algorithms. However, pathwise-exact realizations of X may be computationally expensive in those systems where the total propensity, $a_0(x){:=}\sum_j a_j(x)$, becomes large in a significant proportion of the states, $x \in \mathbb{Z}_+^d$, visited by the process X. The high computational cost of generating pathwise-exact samples is due to the cost of simulating all interarrival times in the system, which are exponential random variables with rate $a_0(x)$. For that reason, in [16] (and independently in [5]), the tau-leap method is proposed to approximate the SSA by evolving the process X by fixed time steps while freezing the propensity functions a_j at the beginning of each time step. A drawback of the tau-leap method is that the simulated paths may take negative values, which is a nonphysical consequence of the approximation and not a qualitative feature of the original process. In [23], we proposed a Chernoff-based hybrid method that switches adaptively between the tau-leap and an exact method. This switching strategy allows us to control the probability of reaching negative values while keeping the computational work substantially smaller than the work of an exact method. To reduce the computational work of the method presented in [23], a hybrid multilevel Monte Carlo (MLMC) method (see the seminal work by Giles [14]) is proposed in [24].

In the present work, we show how the hybrid method developed in [23, 24] can be successfully extended to simulating systems with two types of reaction channels that in the literature of biochemical reactions are usually referred as *fast and slow channels*. To adaptively divide the set of reaction channels, $\mathcal{R}$, into fast and slow channels, we start by defining a state-dependent quantity named the *level of activity* of channel $\mathcal{R}_j$ and noted as $\tilde{a}_j(x)$. Briefly, the level of activity is intuitively defined as the product of the propensity, $a_j(x)$, multiplied by a factor that takes into account the probability of reaching a negative state by taking a tau-leap step exclusively in the direction of the jth vector, ν_j. Then, after sorting $\mathcal{R}$ at each time step by the level of activity, we split $\mathcal{R}$ into two subsets: $\mathcal{R}_{\mathrm{TL}}$ and $\mathcal{R}_{\mathrm{MNRM}}$, fast and slow channels, respectively. The splitting is designed to approximately minimize[1] the expected computational work of

[1]In principle, there are 2^J possible splittings out of the set $\mathcal{R}$. By sorting $\mathcal{R}$ according to the level of activity, we obtain only $J+1$ possible splittings that may or may not contain the optimal one (but according to our numerical experiments, we obtain a near optimal splitting). In our reaction-splitting method, we even reduce the number of splittings from $J+1$ to 3 by assuming that the optimal partition does not change dramatically during two consecutive steps of the algorithm.

taking a Chernoff tau-leap step with the reactions in $\mathcal{R}_{\mathrm{TL}}$ and then taking an exact step with the reactions in $\mathcal{R}_{\mathrm{MNRM}}$. The exact step is computed using the MNRM. We name the paths simulated by this algorithm as *reaction-splitting paths*. Our resulting reaction-splitting method is amenable to MLMC and as in our previous work [24], it achieves a computational complexity of order $\mathcal{O}\left(TOL^{-2}\right)$.

Reaction-splitting methods for simulating SRNs are treated, for instance, in [18, 26, 19, 25], but our work is, to the best of our knowledge, the first that (i) achieves the computational complexity of a pathwise-exact method like the SSA (but with a smaller constant) by using the MLMC paradigm, (ii) explicitly uses a decomposition of the global error to provide all the simulation parameters needed to achieve our goal with minimal computational effort, (iii) actually controls the global probability of reaching negative populations with the tau-leap method, and (iv) needs only two user-defined parameters that are natural quantities—the maximum allowed relative global error or tolerance and the confidence level.

In [18], the authors propose an adaptive reaction-splitting scheme that considers not only the exact and tau-leap methods but also the Langevin and mean field ones. Their primary goal is to obtain fast hybrid simulated paths and they do not try to control the global error as in this work. The efficiency of their method is measured a posteriori using smoothed frequency histograms that should be close to the exact ones according to the distance defined by Cao and Petzold in [9]. In their work, the tau-leap step is chosen according to the "leap condition" (as in [8]) but they do not perform global discretization-error control. To avoid negative populations, the authors reverse the population updates if any value is found to be negative after accounting for all the reactions. Then, the tau-lep step size is decremented and the path simulation is restarted. It is known that this approach introduces bias into the estimations and for that reason postleap techniques have been proposed [3]. Our Chernoff-based bound [23] is a fast and accurate procedure to obtain a tau-leap step size that actually bounds the one-step exit probability. Finally, the method in [18] needs to define three parameters that quantify the speed of the reaction channels, which, in principle, are not trivial to determine for a given problem.

Puchalka and Kierzek's approach [26] seems to be closest to ours in spirit since they also explore the idea of adaptively splitting the set of reaction channels using the tau-leap method for the fast channels and an exact method for the slow ones. They seek to simulate fast approximate paths while maintaining qualitative features of the system. The quantitative features are checked a posteriori against an exact method. Regarding their tau-leap step size selection, Puchalka and Kierzek consider a user-defined maximal time step that is empirically chosen by numerical tests instead of controlling the discretization error. Their classification rule is applied individually to each reaction channel. It takes into account both the percentage of individual activity and the abundance of the species consumed. In a certain sense, it can be seen as a way of controlling the probability of negative populations and an ad hoc method to split the reaction channels by optimizing the computational work.

In [19] and [25], the reaction-splitting issue is addressed but the partition method is not adaptive, i.e., fast and slow reaction channels are identified offline and are inputs to the algorithms. We note that these works do not provide any measure or control of the resulting global error. Furthermore, they do not control the probability of attaining negative populations.

In [6], systems having fast and slow reaction channels are addressed by extending the drift-implicit tau-leap idea to the multilevel setting. This estimator couples drift-implicit paths at the coarser discretization levels until a certain interface level is reached. Then it couples explicit tau-leap paths. This method is an alternative approach to problems with the presence of slow and fast reaction channels.

In the remainder of this section, we introduce the mathematical model and the path simulation techniques used in this work. In section 2, we present our reaction-splitting method. Then, inspired by the ideas of Anderson and Higham [1], we propose an algorithm that couples two reaction-splitting paths. This algorithm uses four building blocks that result from the combination of the MNRM and the Chernoff tau-leap method. In section 3, we propose an MLMC estimator based on our reaction-splitting method. Next, we introduce the global error decomposition and show a new method for estimating the global exit error. Then, we show the automatic procedure that estimates our quantity of interest within a given prescribed relative tolerance, up to a given confidence level. Next, in section 4, we introduce a novel control variate based on the random time change representation (1.2). In section 5, we apply our method to a system with fast and slow channels where both the SSA and the MLMC version of the tau-leap method [1] demand very high amounts of computational work to produce accurate results. We also illustrate the variance reduction capability of the control variate introduced in section 4. Finally, section 6 presents our conclusions.

1.1. The MNRM. The MNRM, introduced in [2] and based on the next reaction method [13], is a pathwise-exact simulation algorithm like Gillespie's SSA that explicitly uses representation (1.2) for simulating exact paths and generates only one exponential random variable per iteration. The reaction times are modeled with firing times of Poisson processes, Y_j, with internal times given by the integrated propensity functions. The randomness is now separated from the state of the system and is encapsulated in the Y_j's. Computing the next reaction and its time is equivalent to computing how much time passes before one of the Poisson process, Y_j, fires, and which process fires at that particular time, by taking the minimum of such times.

It is important to mention that the MNRM is used to simulate correlated pathwise-exact/tau-leap paths as well as nested tau-leap/tau-leap paths, as in [24, 1]. In section 2.5, we use this feature for coupling two reaction-splitting paths.

1.2. The tau-leap approximation. In this section, we define $\bar{X}$, the tau-leap approximation of the process, X, which follows from applying the forward Euler approximation to the integral term in the random time change representation (1.2).

The tau-leap method was proposed in [16] to avoid the computational drawback of the exact methods, i.e., when many reactions occur during a short time interval. The tau-leap process, $\bar{X}$, starts from $X(0)$ at time 0, and given that $\bar{X}(t)=\bar{x}$ and a time step $\tau>0$, we have that $\bar{X}$ at time $t+\tau$ is generated by

$$\bar{X}(t+\tau) = \bar{x} + \sum_{j=1}^{J} \nu_j \mathcal{P}_j\left(a_j(\bar{x})\tau\right),$$

where $\{\mathcal{P}_j(\lambda_j)\}_{j=1}^{J}$ are independent Poisson distributed random variables with parameter λ_j, used to model the number of times that the reaction j fires during the $(t, t+\tau)$ interval. Again, this is nothing but a forward Euler discretization of the stochastic differential equation formulation of the pure jump process (1.2), realized by the Poisson random measure with state-dependent intensity (see, e.g., [22]).

In the limit, when τ tends to zero, the tau-leap method gives the same solution as the exact methods [22]. The total number of firings in each channel is a Poisson distributed stochastic variable depending only on the initial population, $\bar{X}(t)$. The error thus comes from the variation of $a(X(s))$ for $s \in (t, t+\tau)$.

1.3. The Chernoff preleap method. In [23], we derived an algorithm for selecting the size of the tau-leap step based on a Chernoff-type bound that allows us to guarantee that the one-step exit probability inherent in the tau-leap method is less than a predefined quantity, $\delta{>}0$. The idea is to find the largest possible time step, τ, such that the next step of the approximate process, $\bar{X}$, does not exit $\mathbb{Z}_+^d$ with probability greater than $1{-}\delta$. This can be achieved by solving d auxiliary problems, one for each x-coordinate, $\bar{X}_i(t)$, $i = 1, 2, \ldots, d$, as follows. Find the largest possible $\tau_i \geq 0$ such that

$$\mathrm{P}\left(\bar{X}_i(t) + \sum_{j=1}^{J} \nu_{ji} \mathcal{P}_j\left(a_j\left(\bar{X}(t)\right)\tau_i\right) < 0 \;\middle|\; \bar{X}(t)\right) \leq \delta_i, \tag{1.3}$$

where $\delta_i{=}\delta/d^-$ and d^- is the number of species consumed by one or more reaction channels. Here ν_{ji} is the ith coordinate of the jth reaction channel, ν_j. Finally, select $\tau{:=}\min\{\tau_i : i = 1, 2, \ldots, d\}$.

2. Generating reaction-splitting paths.

In this section, we first introduce the concept of the level of activity of a reaction channel. Then, we present a splitting heuristic that indicates how to adaptively partition the set of reaction channels according to their respective levels of activity. Using this partition, the next state of the approximate process is computed as the present state plus the sum of two increments. The first increment is obtained by applying the Chernoff tau-leap to the high-activity channels while the second increment is obtained by applying an exact method to the low-activity channels. Using this heuristic iteratively, we produce reaction-splitting paths. Finally, we show how to couple two reaction-splitting paths, that is, how to produce highly correlated paths associated with two consecutive discretization meshes. These coupled paths are used to compute an MLMC estimator of $\mathrm{E}[g(X(T))]$.

2.1. The level of activity of a reaction channel. The level of activity of a reaction channel, $\tilde{a}_j(x)$, is an auxiliary quantity designed to adaptively split the set of reaction channels, $\mathcal{R}$, into two disjoint sets: $\mathcal{R}_{\mathrm{TL}}(x)$ and $\mathcal{R}_{\mathrm{MNRM}}(x)$. As mentioned in the introduction, the increment associated with those reactions in $\mathcal{R}_{\mathrm{TL}}(x)$ is computed using the Chernoff tau-leap, while the increment associated with those reactions in $\mathcal{R}_{\mathrm{MNRM}}(x)$ is computed using the MRNM.

First, observe that (1.1) implies that the number of firings of the channel, $\mathcal{R}_j$, in an interval of size one is approximately proportional to $a_j(x)$. Therefore, at first sight, $a_j(x)$ gives a natural measure for the level of activity of the jth channel at state x.

Observe that it would be desirable to avoid including in $\mathcal{R}_{\mathrm{TL}}(x)$ those reaction channels that produce small Chernoff tau-leap step sizes. To this end, let t and x be the current time and state, respectively, of the reaction-splitting path. Consider a (not necessarily uniform) time-mesh of the interval $[0, T]$, required to control the time-discretization error associated with the tau-leap method, and let $\tilde{T} > t$ be the next mesh point. Define I_j as the set of indices with negative components of the vector ν_j; that is, $I_j{:=}\{i : \nu_{j,i} < 0\}$, where $\nu_{j,i}$ is the ith component of the vector ν_j. It is easy to see that if $I_j \neq \emptyset$, then the largest value of $k \in \mathbb{N} := \{1, 2, 3, \ldots\}$ such that $x{+}k\,\nu_j \in \mathbb{Z}_+^d$ is $k_j(x){:=}\lfloor \min\{-x_i/\nu_{j,i} : i \in I_j\}\rfloor$, where $\lfloor y \rfloor$ is the greatest integer that is less than or equal to the real number, y. This observation leads us to define θ_j as follows: let $\kappa \sim \mathcal{P}(a_j(x)(\tilde{T}{-}t))$ (a Poisson random variable with rate $a_j(x)(\tilde{T}{-}t)$).

Then, define

$$\theta_j := \begin{cases} \mathrm{P}\left(\kappa > k_j(x) \,\middle|\, x\right) & \text{if } I_j \neq \emptyset, \\ 0 & \text{otherwise.} \end{cases} \tag{2.1}$$

Therefore, θ_j is the one-step exit probability due to a single tau-leap step in the direction of ν_j with the tau-leap size equal to the distance to the next time-mesh point, $\tilde{T} - t$. A large value of θ_j indicates that $\mathcal{R}_j$ should not be included in $\mathcal{R}_{\mathrm{TL}}$. Then, the penalty weight for $a_j(x)$ is $1{-}\theta_j$. Finally, we define the *level of activity* of the channel, $\mathcal{R}_j$, at the state x as $\tilde{a}_j(x){:=}(1{-}\theta_j)a_j(x)$. Notice that we are omitting in the notation the role of $\tilde{T}$.

2.2. The splitting heuristic. Hereafter, for the sake of brevity in the notation, we name the jth reaction channel, $\mathcal{R}_j$, simply j. We now explain how we split the set of reaction channels, $\mathcal{R}{:=}\{1,\dots,J\}$, into $\mathcal{R}_{\mathrm{TL}}$ and $\mathcal{R}_{\mathrm{MNRM}}$.

Let (t, x) be the current time and state of the approximate process, $\bar{X}$, and let $\tilde{T} > t$ be the smallest time-mesh greater than t. We want to split $\mathcal{R}$ into two subsets, $\mathcal{R}_{\mathrm{MNRM}}$ and $\mathcal{R}_{\mathrm{TL}}$, such that the expected computational work of reaching $\tilde{T}$, starting at t, is minimal for all possible splittings. Ideally, we want to solve the problem of minimizing the sum $\mathrm{Work}(\mathcal{R}_{\mathrm{TL}}, x, t) + \mathrm{Work}(\mathcal{R}_{\mathrm{MNRM}}, x, t)$ among all the 2^J possible splittings of the set $\mathcal{R}$, but this is unfeasible even for a moderate number of reaction channels.

The main idea of this section is as follows. First, we define a linear order on $\mathcal{R}$, based on the basic principle that we want to use tau-leap for the jth reaction only if its level of activity is high enough. This linear order determines $J{+}1$ possible splittings, out of 2^J. The linear order is then a permutation, σ, over $\mathcal{R}$ such that

$$\tilde{a}_{\sigma(j)}(x) \geq \tilde{a}_{\sigma(j+1)}(x),\ \ j{=}1,\dots,J{-}1.$$

Second, we find among the $J{+}1$ partitions the one with optimal work. This is the computational work incurred when performing one step of the algorithm using the tau-leap method with the reactions in $\mathcal{R}_{TL}$ and the MNRM with the reactions in $\mathcal{R}_{\mathrm{MNRM}}$. The work corresponding to $\mathcal{R}_{\mathrm{TL}}$ is

$$\mathrm{Work}(\mathcal{R}_{\mathrm{TL}}, x, t) := C_s + \frac{\tilde{T}{-}t}{\min\{\tau_{Ch}, \tilde{T}{-}t\}} \sum_{j \in \mathcal{R}_{\mathrm{TL}}} C_P(a_j(x)\tau_{Ch}), \tag{2.2}$$

where C_s is the work of computing the split (see section 2.3) and $C_P(\lambda)$ is the work of generating a Poisson random variate with rate λ. The factor $(\tilde{T}{-}t)/\min\{\tau_{Ch}, \tilde{T}{-}t\}$ takes into account the number of steps required to reach $\tilde{T}$ from t (at least one step). In practice, it is possible to estimate b_i, $i{=}1, 2, 3, 4$, using Monte Carlo sampling and a least-squares fit. For more details, we refer to [23].

Similarly, the work corresponding to $\mathcal{R}_{\mathrm{MNRM}}$ is

$$\mathrm{Work}(\mathcal{R}_{\mathrm{MNRM}}, x, t) := \frac{\tilde{T}{-}t}{\min\{\tau_{\mathrm{MNRM}}, \tilde{T}{-}t\}} C_{\mathrm{MNRM}},$$

where the constant C_{MNRM} is the work of an MNRM step. Here, τ_{MNRM} is defined as $(\sum_{j \in \mathcal{R}_{\mathrm{MNRM}}} a_j(x))^{-1}$.

2.3. Reducing the computational cost of the splitting rule. The work required to perform the splitting includes the work required to determine Work($\mathcal{R}_{\mathrm{TL}}$) and Work($\mathcal{R}_{\mathrm{MNRM}}$), both defined in section 2.2. The linear order previously defined determines $J{+}1$ possible splittings, $(\mathcal{S}_0, \mathcal{S}_1, \ldots, \mathcal{S}_J)$. The splitting $\mathcal{S}_j$ denotes the partition $(\mathcal{R}_{\mathrm{TL}}, \mathcal{R}_{\mathrm{MNRM}})$, where $\mathcal{R}_{\mathrm{TL}}$ contains the reaction channels with the j-highest levels of activity. Note that when $j = 0$, $\mathcal{R}_{\mathrm{TL}} = \emptyset$. The cost of computing each of the $J{+}1$ splits is dominated by the cost of determining the Chernoff tau-leap step size, τ_{Ch} (see (2.2)). As we observe in [23], the work of computing a single τ_{Ch} is linear on J. Then, to avoid a complexity of order J^2 in the splitting rule, we implement a local search instead of computing J τ_{Ch}s, to keep the complexity of the split procedure linear on J.

The key idea is to keep track of the index of the last split at each decision time, say, κ, and assume that the propensities do not vary widely between two consecutive steps of the algorithm. If that is the case, we compute the new set of levels of activity and just evaluate at most three splits: $\mathcal{S}_\kappa$, and its neighbors, $\mathcal{S}_{\kappa-1}$ and $\mathcal{S}_{\kappa+1}$ (there are only two splits when $\kappa{=}0$ or $\kappa{=}J$). Then, the cost of the splitting rule is on the order of three computations of a Chernoff step size. It turns out that this local search is very accurate for the examples we worked on. In order to avoid being trapped in local minima, a randomization rule may be applied.

To avoid unnecessary evaluations of the splitting procedure, a thinning technique can be applied as follows. The key idea is that, from one decision time to the next one, the index of the last split changes only if the level of activity changes too. How significant this change should be is not known a priori but it can be modeled by using a scalar parameter. Then, after computing the penalized propensities and before evaluating the splitting rule, a comparison between the current and the previous level of activity is made. Only if the change in the activity level is significant is the splitting procedure evaluated. This thinning technique can be implemented in linear order with respect to the number of reaction channels, J, so no significant overhead is introduced.

2.4. The reaction-splitting algorithm. Before describing the reaction-splitting path algorithm (Algorithm 2), we would like to make two comments. First, like in [23], we introduce a time-mesh (not necessarily uniform) in the interval $[0, T]$ for controlling the time-discretization error coming from the tau-leap method. Second, we consider that a mesh point is a time, t, in which the propensities, a_j, are evaluated in order to make a decision about the split; we call these points *horizons*. The proposed time-mesh can be modified by two procedures: (i) augmentation, that is, we add a new horizon as a consequence of the Chernoff preleap; (ii) deletion, that is, we jump over one or more of the proposed time-mesh points to the next reaction time when the decision algorithm decides to take a purely exact step.

Now we proceed to explain how to generate one reaction-splitting path. The main building block for simulating a reaction-splitting path is Algorithm 1, which computes one step of the reaction-splitting algorithm. A whole reaction-splitting path is simply the result of applying a sequence of reaction-splitting steps. Let $x{=}\bar{X}(t)$ be the current state of the approximate process, $\bar{X}$. The expected time step of the MNRM is therefore given by $1/a_0(x)$. To move one step forward using the reaction-splitting Chernoff tau-leap method, we first need to compute the split and then to compute the tau-leap increments for the reactions in the tau-leap set, $\mathcal{R}_{\mathrm{TL}}$. Finally we need to compute the MNRM steps for the reactions in the set $\mathcal{R}_{\mathrm{MNRM}}$, as discussed in section 2.3.

Algorithm 1. The one-step mixing rule. Inputs: the current state of the approximate process, $\bar{X}(t)$, the current time, t, the values of the propensity functions evaluated at $\bar{X}(t)$, $(a_j(\bar{X}(t)))_{j=1}^{J}$, the one-step exit probability bound δ, the next time-mesh point, $\tilde{T}$, and the index of the previous optimal split, κ. Outputs: the index of the new approximate optimal split, κ^*, and its corresponding partition, $\mathcal{R}_{\mathrm{TL}}$ and $\mathcal{R}_{\mathrm{MNRM}}$.

```
Require: $a_0 \leftarrow \sum_{j=1}^{J} a_j > 0$
1: if $K_1/a_0 < \tilde{T} - t$ then
2:     Compute $\theta_j$, $j$=1,…,$J$ (see (2.1))
3:     $\tilde{a}_{\sigma(j)} \leftarrow$ Compute the levels of activity for $j$=1,…,$J$ (see (2.1))
4:     if $\tilde{a}$ changed significantly since the previous decision time then
5:         $(\mathcal{S}_{\kappa-1}, \mathcal{S}_{\kappa}, \mathcal{S}_{\kappa+1}) \leftarrow$ Compute three splits (at most) taking into account the
           levels of activity $\tilde{a}_{\sigma(j)}$ and the index of the previous optimal split $\kappa$
6:         $\kappa^* \leftarrow$ Evaluate the works corresponding to $(\mathcal{S}_{\kappa-1}, \mathcal{S}_{\kappa}, \mathcal{S}_{\kappa+1})$ and take the
           index of the minimum work split
7:         return $(\kappa^*, \mathcal{R}_{\mathrm{TL}}, \mathcal{R}_{\mathrm{MNRM}})$
8:     else
9:         return previous $(\kappa^*, \mathcal{R}_{\mathrm{TL}}, \mathcal{R}_{\mathrm{MNRM}})$
10:    end if
11: else
12:    return $(\kappa^*, \emptyset, \mathcal{R})$ corresponding to $\mathcal{R}_{\mathrm{TL}} = \emptyset$
13: end if
```

To avoid the overhead caused by an unnecessary computation of the split, we first estimate the computational work of moving forward from the current time, t, to the next time-mesh point, $\tilde{T}$, by using the MNRM only. If this work is less than the work of computing the split, we take an exact step.

To compare the mentioned computational costs, we define K_1 as the ratio between the cost of computing the split, C_s, and the cost of computing one step using the MNRM.

To generate one reaction-splitting path, we introduce Algorithm 2, which combines the approximate Chernoff tau-leap method and the exact MNRM to generate a whole reaction-splitting path. This algorithm automatically and adaptively partitions the reactions into two subsets, $\mathcal{R}_{\mathrm{TL}}$ and $\mathcal{R}_{\mathrm{MNRM}}$, using a computational work criterion. Since a reaction-splitting path consists of a certain number of exact/approximate steps, it may also exit the lattice, except in those steps in which the tau-leap method is not applied, that is, when $\mathcal{R}_{\mathrm{TL}} = \emptyset$. The idea of this algorithm is to apply, at each decision point, the one-step mixing rule (Algorithm 1) to determine sets $\mathcal{R}_{\mathrm{TL}}$ and $\mathcal{R}_{\mathrm{MNRM}}$ and then to apply the corresponding method.

In the first line of Algorithm 2, we set the indices and the states. Here $\bar{Z}$ is an auxiliary state. Then, we compute, for this unique time, all possible $J+1$ splits and obtain the index of the optimal splitting, κ. Inside the main loop, we move forward using the partition provided by Algorithm 1. If $\mathcal{R}_{\mathrm{TL}} \neq \emptyset$, we take an approximate step of size $\tau_{Ch}(\mathcal{R}_{\mathrm{TL}})$ (which is never greater than the distance to the next time-mesh point, $\tilde{T}$). This step size sets the horizon, H. On the other hand, if $\mathcal{R}_{\mathrm{TL}} = \emptyset$, the system will evolve until the next exact reaction time, H, which could be greater than $\tilde{T}$. As we mentioned, the Chernoff preleap controls the one-step exit probability but does not preclude the possibility of jumping outside the lattice, $\mathbb{Z}_+^d$. If we do not exit the lattice, and $\mathcal{R}_{\mathrm{MNRM}} \neq \emptyset$, then we evolve the system by taking exact steps until

Algorithm 2. The reaction-splitting-path algorithm. Inputs: the initial state, $X(0)$, the propensity functions, $(a_j)_{j=1}^J$, the stoichiometric vectors, $\nu{=}(\nu_j)_{j=1}^J$, the final time, T, and the one-step exit probability bound, δ. Outputs: a sequence of states, $(\bar{X}(t_k))_{k=0}^K$. Notes: given the current state, $next_{\mathrm{MNRM}}$ computes the next state using the MNRM method. Here, t_i denotes the current time at the i-th step, and $\tau_{Ch}(\mathcal{R}_{\mathrm{TL}})$ is the Chernoff step size associated with $\mathcal{R}_{\mathrm{TL}}$.

```
1: $i \leftarrow 0,\ t_i \leftarrow t_0, \bar{X}(t_i) \leftarrow X(0), \bar{Z} \leftarrow X(0)$
2: $(\mathcal{S}_0, \mathcal{S}_1, \ldots, \mathcal{S}_J) \leftarrow$ Compute the complete set of splits
3: $\kappa \leftarrow \arg\min_j \mathrm{Work}(\mathcal{S}_j)$ Index of the optimal split
4: while $t_i < T$ do
5:     $\tilde{T} \leftarrow$ next time-mesh point greater than $t_i$
6:     $(\kappa, \mathcal{R}_{\mathrm{TL}}, \mathcal{R}_{\mathrm{MNRM}}) \leftarrow$ Update the partition of $\mathcal{R}$ using Algorithm 1 with parameters $(\bar{Z}, t_i, (a_j(\bar{Z}))_{j=1}^J, \delta, \tilde{T}, \kappa)$
7:     if $\mathcal{R}_{\mathrm{TL}} \neq \emptyset$ then
8:         $\bar{Z} \leftarrow \bar{Z} + \sum_{j\in\mathcal{R}_{\mathrm{TL}}} \nu_j\, \mathcal{P}_j(a_j(\bar{Z})\tau_{Ch}(\mathcal{R}_{\mathrm{TL}}))$
9:         $H \leftarrow t_i + \tau_{Ch}(\mathcal{R}_{\mathrm{TL}})$
10:    else
11:        $H \leftarrow$ from MNRM
12:    end if
13:    if $\bar{Z} \in \mathbb{Z}_+^d$ then
14:        $t_{i+1} \leftarrow H$
15:    else
16:        return $((\bar{X}(t_k))_{k=0}^i)$
17:    end if
18:    if $\mathcal{R}_{\mathrm{MNRM}} \neq \emptyset$ then
19:        while $t_i < H$ do
20:            $(\bar{Z}, t_i) \leftarrow next_{\mathrm{MNRM}}(\bar{Z}, \mathcal{R}_{\mathrm{MNRM}}, t_i, H)$
21:        end while
22:    end if
23:    $i \leftarrow i + 1$
24:    $\bar{X}(t_i) \leftarrow \bar{Z}$
25: end while
26: return $((\bar{X}(t_k))_{k=0}^i)$
```

the horizon, H. Finally, we update the parameters and proceed iteratively until we reach the final time, T.

Remark 2.1 (computational complexity of determining the κth level of activity). In order to determine the three splits $\mathcal{S}_{\kappa-1}, \mathcal{S}_\kappa, \mathcal{S}_{\kappa+1}$ we need to be able to determine the levels of activity $\tilde{a}_{\kappa-1}$, $\tilde{a}_\kappa$, and $\tilde{a}_{\kappa+1}$. It is possible to use the median-of-medians algorithm [7], which has not only average complexity of $\mathcal{O}(J)$ but also the same worst-case complexity. This algorithm has large constants, and thus it should be used only when the number of reaction channels, J, is large enough. We estimate that this number should be at most $1E5$.

Remark 2.2 (comparison with the one-step hybrid rule). In [23], we developed a hybrid method, which, at each decision point, determines which method, exact or tau-leap, is cheaper to apply to the whole set of reactions. That is, in the hybrid method, we have either $\mathcal{R}_{\mathrm{TL}} = \emptyset$ and $\mathcal{R}_{\mathrm{MNRM}} = \mathcal{R}$ or $\mathcal{R}_{\mathrm{TL}} = \mathcal{R}$ and $\mathcal{R}_{\mathrm{MNRM}} = \emptyset$. Then, the reaction-splitting method can be seen as a generalization of the hybrid one.

The key difference is in the cost of the decision rule, which, as we saw in section 2.3, for the reaction-splitting method is on the order of three times the cost of computing a Chernoff tau-leap step size. This difference can be significant in some problems. A Pareto splitting rule may be able to recover the cost of the hybrid one-step decision rule.

2.5. Coupling two reaction-splitting paths. Coupling two reaction-splitting paths is essential for the multilevel estimator. The four algorithms that are the building blocks of the coupling algorithm were already presented in [24]. The novelty here comes from the fact that the coupled reaction-splitting algorithm may have to run the four algorithms concurrently in the sense of the time of the process, t. In this section, we denote with a bar $\bar{\cdot}$ and a double bar $\bar{\bar{\cdot}}$ coarse and fine grid-related quantities, respectively.

Before describing how to couple two reaction-splitting paths, we would like to make a few comments. First, the coupled paths, like in [24], evolve through a system of horizons. The horizons are the times in which each path applies Algorithm 1 to choose the splitting, and this is done independently of the state or decision of the other path. This is crucial for the telescoping property inherent in the MLMC method. Second, two nested time meshes are proposed for controlling the discretization errors at the coarse and fine levels (this is discussed in the next section). A mesh point is a time, t, in which the propensities, a_j, are evaluated to make a decision about the split, that is, about the horizons. During the coupling process, these two proposed time meshes are modified by two procedures: (i) augmentation, that is, we add points to the time-mesh as a consequence of the Chernoff preleap; (ii) deletion, that is, we ignore points of the proposed time-mesh when the decision algorithm takes purely exact steps (steps such that $\mathcal{R}_{\text{TL}} = \emptyset$) and the next reaction time is greater than the distance to the next proposed time-mesh point. As a result, the actual coarse and fine time-meshes are nonnested, i.e., the coarse mesh may contain evaluation points not appearing in the fine mesh. Third, when an approximate path exits the lattice (say, the coarse path), the other process (the fine path) evolves according to Algorithm 2.

We now briefly describe the reaction-splitting Chernoff coupling algorithm, i.e., Algorithm 3. Let $\bar{X}$ and $\bar{\bar{X}}$ be two reaction-splitting paths corresponding to two nested proposed time discretizations (see the previous paragraph), called coarse and fine, respectively. Assume that the current time is t and that we know the states, $\bar{X}(t)$ and $\bar{\bar{X}}(t)$, the next grid points at each level, $\bar{t}$, $\bar{\bar{t}}$, and the corresponding one-step exit probabilities, $\bar{\delta}$ and $\bar{\bar{\delta}}$. Based on this knowledge, we have to determine the four sets $(\bar{\mathcal{R}}_{\text{TL}}, \bar{\mathcal{R}}_{\text{MNRM}}, \bar{\bar{\mathcal{R}}}_{\text{TL}}, \bar{\bar{\mathcal{R}}}_{\text{MNRM}})$ that correspond to the four algorithms, B1, B2, B3, and B4, that we use as building blocks. Table 2.1 summarizes them. To do that, the algorithm computes, independently, the sets $\mathcal{R}_{\text{TL}}$ and $\mathcal{R}_{\text{MNRM}}$ for each level and the time until the next decision is taken, H, using Algorithm 4. Next, it computes concurrently the increments due to each one of the sets (storing the results in $\Delta\bar{X}$ and $\Delta\bar{\bar{X}}$ for the coarse and fine grid, respectively). We note that the only case in which we use a Poisson random variate generator for the tau-leap method is in Algorithm B1 (Algorithm 5). For Algorithms B2, B3, and B4, the Poisson random variables are simulated by adding independent exponential random variables with the same rate, λ, until exceeding a given time final time, T. This is because in those three blocks at least one level is pathwise exact. The only difference in the latter blocks is the time point at which the propensities, a_j, are computed. For B2, the coarse propensities are frozen at time t, whereas for B3 the finer are frozen at t. In B4, the propensities are computed at each time step. After arriving at time H, the four sets

Algorithm 3. Coupled reaction-splitting path. Inputs: the initial state, $X(0)$, the final time, T, the propensity functions, $(a_j)_{j=1}^J$, the stoichiometric vectors, $(\nu_j)_{j=1}^J$, and two time-meshes, one coarser, $(t_i)_{i=0}^N$, such that $t_N{=}T$, and one finer, $(s_j)_{j=0}^{N'}$, such that $s_0{=}t_0$, $s_M{=}t_N$, and $(t_i)_{i=0}^N \subset (s_j)_{j=0}^{N'}$. Outputs: a sequence of states evaluated at the coarse grid, $(\bar{X}(t_k))_{k=0}^K \subset \mathbb{Z}_+^d$, such that $t_K \le T$, a sequence of states evaluated at the fine grid $(\bar{\bar{X}}(s_l))_{l=0}^{K'} \subset \mathbb{Z}_+^d$, such that $\bar{X}(t_K) \in \mathbb{Z}_+^d$ or $\bar{\bar{X}}(s_{K'}) \in \mathbb{Z}_+^d$. If $t_K < T$, both paths exit the $\mathbb{Z}_+^d$ lattice before the final time, T. It also returns the number of times the tau-leap method is successfully applied at the fine level and at the coarse level and the number of exact steps at the fine level and at the coarse level. For the sake of simplicity, we omit sentences involving the recording of current state variables, counting of the number of steps, checking if the path jumps out of the lattice, updating of the current split, κ, and the return sentence.

```
1: $t \leftarrow t_0$; $\bar{X} \leftarrow X(0)$; $\bar{\bar{X}} \leftarrow X(0)$
2: $\bar{t} \leftarrow$ next grid point in $(t_i)_{i=0}^N$ larger than $t$
3: $(\bar{H}, \bar{\mathcal{R}}_{\mathrm{TL}}, \bar{\mathcal{R}}_{\mathrm{MNRM}}, \bar{a}) \leftarrow$ Algorithm 4 with $(\bar{X}, t, \bar{t}, T, \bar{\delta})$
4: $\bar{\bar{t}} \leftarrow$ next grid point in $(s_i)_{i=0}^N$ larger than $t$
5: $(\bar{\bar{H}}, \bar{\bar{\mathcal{R}}}_{\mathrm{TL}}, \bar{\bar{\mathcal{R}}}_{\mathrm{MNRM}}, \bar{\bar{a}}) \leftarrow$ Algorithm 4 with $(\bar{\bar{X}}, t, \bar{\bar{t}}, T, \bar{\bar{\delta}})$
6: while $t < T$ do
7:     $H \leftarrow \min\{\bar{H}, \bar{\bar{H}}\}$
8:     $(B_1, B_2, B_3, B_4) \leftarrow$ split building blocks from $(\bar{\mathcal{R}}_{\mathrm{TL}}, \bar{\mathcal{R}}_{\mathrm{MNRM}}, \bar{\bar{\mathcal{R}}}_{\mathrm{TL}}, \bar{\bar{\mathcal{R}}}_{\mathrm{MNRM}})$
9:     Algorithm 5 (compute state changes due to block $B_1$)
10:    Initialize internal clocks $R, P$ if needed (see [23, 24])
11:    for $\mathcal{B} = B_2, B_3, B_4$ do
12:        $t_r \leftarrow t$
13:        while $t_r < H$ do
14:            update $P_{j\in\mathcal{B}}$
15:            switch $\mathcal{B}$
16:                case $B_2$:
17:                    $\bar{d} \leftarrow \bar{a}_{j\in\mathcal{B}}$
18:                    $\bar{\bar{d}} \leftarrow a_{j\in\mathcal{B}}(\bar{\bar{X}})$
19:                    $\tau_r \leftarrow$ Compute the Chernoff tau-leap step size using $(\bar{X}, \bar{a}_{j\in\mathcal{B}}, H, \bar{\delta})$
20:                case $B_3$:
21:                    $\bar{d} \leftarrow a_{j\in\mathcal{B}}(\bar{X})$
22:                    $\bar{\bar{d}} \leftarrow \bar{\bar{a}}_{j\in\mathcal{B}}$
23:                    $\tau_r \leftarrow$ Compute the Chernoff tau-leap step size using $(\bar{\bar{X}}, \bar{\bar{a}}_{j\in\mathcal{B}}, H, \bar{\bar{\delta}})$
24:                case $B_4$:
25:                    $\bar{d} \leftarrow a_{j\in\mathcal{B}}(\bar{X})$
26:                    $\bar{\bar{d}} \leftarrow a_{j\in\mathcal{B}}(\bar{\bar{X}})$
27:                    $\tau_r \leftarrow \infty$
28:            end switch
29:            $A_1 \leftarrow \min(\bar{d}, \bar{\bar{d}})$
30:            $A_2 \leftarrow \bar{d} - A_1$; $A_3 \leftarrow \bar{\bar{d}} - A_1$
31:            $H_r \leftarrow \min\{H, t_r{+}\tau_r\}$
32:            $(t_r, \bar{X}, \bar{\bar{X}}, R_{j\mathcal{B}}, P_{j\in\mathcal{B}}) \leftarrow$ Algorithm 6 with $(t_r, H_r, \bar{X}, \bar{\bar{X}}, R_{j\in\mathcal{B}}, P_{j\in\mathcal{B}}, A)$
33:        end while
34:    end for
35:    $t \leftarrow H$
36:    if $t < T$ then
37:        if $\bar{H} = \bar{\bar{H}}$ then
38:            $\bar{t} \leftarrow$ next grid point in $(t_i)_{i=0}^N$ larger than $t$
39:            $(\bar{H}, \bar{\mathcal{R}}_{\mathrm{TL}}, \bar{\mathcal{R}}_{\mathrm{MNRM}}, \bar{a}) \leftarrow$ Algorithm 4 with $(\bar{X}, t, \bar{t}, T, \bar{\delta})$
40:        end if
41:        if $\bar{H} = \bar{\bar{H}}$ then
42:            $\bar{\bar{t}} \leftarrow$ next grid point in $(s_j)_{j=0}^{N'}$ larger than $t$
43:            $(\bar{\bar{H}}, \bar{\bar{\mathcal{R}}}_{\mathrm{TL}}, \bar{\bar{\mathcal{R}}}_{\mathrm{MNRM}}, \bar{\bar{a}}) \leftarrow$ Algorithm 4 with $(\bar{\bar{X}}, t, \bar{\bar{t}}, T, \bar{\bar{\delta}})$
44:        end if
45:    end if
46: end while
```

Table 2.1
Building blocks for simulating two coupled reaction-splitting Chernoff tau-leap paths. Algorithms B1 *and* B2 *are presented as Algorithms* 2 *and* 3 *in* [1]*. Algorithms* B3 *and* B4 *can be directly obtained from Algorithm* B2 *(see* [24]*).*

	$\bar{\mathcal{R}}_{\text{TL}}$	$\bar{\mathcal{R}}_{\text{MNRM}}$
$\bar{\bar{\mathcal{R}}}_{\text{TL}}$	B1	B3
$\bar{\bar{\mathcal{R}}}_{\text{MNRM}}$	B2	B4

Algorithm 4. Compute the next time horizon. Inputs: the current state, $\tilde{X}$, the current time, t, the next grid point, $\tilde{t}$, the final time, T, the one-step exit probability bound, $\tilde{\delta}$, and the propensity functions, $a{=}(a_j)_{j=1}^J$. Outputs: the next horizon, H, the set of reaction channels to which the Tau-leap method should be applied, $\tilde{\mathcal{R}}_{\text{TL}}$, the set of reaction channels to which MNRM should be applied, $\tilde{\mathcal{R}}_{\text{MNRM}}$, and current propensity values, $\tilde{a}$.

```
1: $\tilde{a} \leftarrow a(\tilde{X})$
2: $(\tilde{\mathcal{R}}_{\text{TL}}, \tilde{\mathcal{R}}_{\text{MNRM}}) \leftarrow$ Algorithm 1 with $(\bar{X}, t, (a_j(\bar{X}))_{j=1}^J, \tilde{\delta}, \tilde{t}, \kappa)$
3: if $\tilde{\mathcal{R}}_{\text{TL}} \neq \emptyset$ then
4:     $\tilde{H} \leftarrow \min\{\tilde{t}, t{+}\tau(\tilde{\mathcal{R}}_{\text{TL}}), T\}$
5: else
6:     $\tilde{H} \leftarrow \min\{t{+}\tau(\tilde{\mathcal{R}}_{\text{TL}}), T\}$
7: end if
8: return $(\tilde{H}, \tilde{\mathcal{R}}_{\text{TL}}, \tilde{\mathcal{R}}_{\text{MNRM}}, \tilde{a})$
```

Algorithm 5. Compute building block 1. This algorithm is part of Algorithm 3.

```
1: $t_r \leftarrow t$
2: while $t_r < H$ do
3:     $\bar{\tau}_r \leftarrow$ Compute the Chernoff tau-leap step size using $(\bar{X}, \bar{a}_{j\in B_1}, H, \bar{\delta})$
4:     $\bar{\bar{\tau}}_r \leftarrow$ Compute the Chernoff tau-leap step size using $(\bar{\bar{X}}, \bar{\bar{a}}_{j\in B_1}, H, \bar{\bar{\delta}})$
5:     $H_r \leftarrow \min\{H, t_r{+}\bar{\tau}_r, t_r{+}\bar{\bar{\tau}}_r\}$
6:     $A_1 \leftarrow \min(\bar{a}_{j\in B_1}, \bar{\bar{a}}_{j\in B_1})$
7:     $A_2 \leftarrow \bar{a}_{j\in B_1} - A_1$
8:     $A_3 \leftarrow \bar{\bar{a}}_{j\in B_1} - A_1$
9:     $\Lambda \leftarrow \mathcal{P}(A\cdot(H_r{-}t_r))$
10:    $\bar{\bar{X}} \leftarrow \bar{\bar{X}} + (\Lambda_1{+}\Lambda_2)\nu_{j\in B_1}$
11:    $\bar{X} \leftarrow \bar{X} + (\Lambda_1{+}\Lambda_3)\nu_{j\in B_1}$
12:    $t_r \leftarrow H_r$
13: end while
```

$(\bar{\mathcal{R}}_{\text{TL}}, \bar{\mathcal{R}}_{\text{MNRM}}, \bar{\bar{\mathcal{R}}}_{\text{TL}}, \bar{\bar{\mathcal{R}}}_{\text{MNRM}})$ and the time until the next decision is taken, H, are determined again, and then all procedures are repeated until the simulation reaches the final time, T.

3. The multilevel estimator and global error decomposition.

In this section, we first present the MLMC estimator. We then analyze and control the computational global error, which is decomposed into three error components: the discretization error, the global exit error, and the Monte Carlo statistical error. Upper bounds for each of the three components are given. Finally, we briefly describe the automatic estimation procedure that allows us to estimate our quantity of interest within a given prescribed relative tolerance and up to a given confidence level.

Algorithm 6. The auxiliary function used in Algorithm 3. Inputs: current time, t, current time horizon, $\bar{\bar{T}}$, current system state at a coarser level and a finer level, $\bar{X}$, $\bar{\bar{X}}$, respectively, the internal clocks R and P, the values, A, and the current building block, B. Outputs: updated time, t, updated system states, $\bar{X}$, $\bar{\bar{X}}$, and updated internal clocks, R_i, P_i, i=1, 2, 3.

```
1: $\Delta t_i \leftarrow (P_i - R_i)/A_i$ for $i = 1, 2, 3$
2: $\Delta \leftarrow \min_i\{\Delta t_i\}$
3: $\mu \leftarrow \text{argmin}_i\{\Delta t_i\}$
4: if $t + \Delta > \bar{\bar{T}}$ then
5:     $R \leftarrow R + A\cdot(\bar{\bar{T}} - t)$
6:     $t \leftarrow \bar{\bar{T}}$
7: else
8:     update $\bar{X}$ and $\bar{\bar{X}}$ using $\nu_{j\in B}$
9:     $R \leftarrow R + A\Delta$
10:    $r \leftarrow \text{uniform}(0, 1)$
11:    $P_\mu \leftarrow P_\mu + \log(1/r)$
12:    $t \leftarrow t + \Delta$
13: end if
14: return $(t, \bar{X}, \bar{\bar{X}}, R, P)$
```

3.1. The MLMC estimator. We proceed now to discuss and implement an MLMC [20, 14, 27] estimator for the reaction-splitting method. Consider a hierarchy of nested meshes of the time interval $[0, T]$, indexed by $\ell = 0, 1, \ldots, L$. Let Δt_0 be the size of the coarsest time mesh that corresponds to level ℓ=0. The size of the time mesh at level $\ell \geq 1$ is given by $\Delta t_\ell = R^{-\ell}\Delta t_0$, where R>1 is a given integer constant. As in [24], this system of nested time-meshes is proposed to control the discretization error and the meshes may change during the coupling process by adding and deleting points. The actual meshes are therefore nonnested.

Let $\{\bar{X}_\ell(t)\}_{t\in[0,T]}$ be a reaction-splitting Chernoff tau-leap process with a time-mesh of size Δt_ℓ and a one-step exit probability bound δ_ℓ, and let $g_\ell := g(\bar{X}_\ell(T))$ be our quantity of interest computed with a mesh of size Δt_ℓ. Let A_L be the event in which $\bar{X}_L$ arrives at the final time, T, without exiting the state space of X.

Consider the following telescopic decomposition:

$$\mathrm{E}[g_L \mathbf{1}_{A_L}] = \mathrm{E}[g_0 \mathbf{1}_{A_0}] + \sum_{\ell=1}^{L} \mathrm{E}[g_\ell \mathbf{1}_{A_\ell} - g_{\ell-1}\mathbf{1}_{A_{\ell-1}}], \tag{3.1}$$

where $\mathbf{1}_A$ is the indicator function of the set A. This motivates the definition of our MLMC estimator of $\mathrm{E}[g(X(T))]$:

$$\mathcal{M}_L := \frac{1}{M_0}\sum_{m=1}^{M_0} g_0 \mathbf{1}_{A_0}(\omega_{m,0}) + \sum_{\ell=1}^{L}\frac{1}{M_\ell}\sum_{m=1}^{M_\ell}[g_\ell \mathbf{1}_{A_\ell} - g_{\ell-1}\mathbf{1}_{A_{\ell-1}}](\omega_{m,\ell}). \tag{3.2}$$

Remark 3.1. For some models, the following estimator, (3.3), could be a better option in terms of computational cost. If the strong error due to the exit error is not significant, the exit error control may be enforced only in the finest level and for bias purposes only. Thus in such cases we use instead the telescopic decomposition,

$$\mathrm{E}[g_L \mathbf{1}_{A_L}] = \mathrm{E}[g_0] + \sum_{\ell=1}^{L-1}\mathrm{E}[g_\ell - g_{\ell-1}] + \mathrm{E}[g_L \mathbf{1}_{A_L} - g_{L-1}],$$

and the corresponding MLMC estimator,

$$\mathcal{M}_L := \frac{1}{M_0}\sum_{m=1}^{M_0} g_0(\omega_{m,0}) + \sum_{\ell=1}^{L-1}\frac{1}{M_\ell}\sum_{m=1}^{M_\ell}[g_\ell - g_{\ell-1}](\omega_{m,\ell}) + \frac{1}{M_L}\sum_{m=1}^{M_L}[g_L\mathbf{1}_{A_L} - g_{L-1}](\omega_{m,L}). \tag{3.3}$$

Whenever a reaction-splitting path exits the lattice, we project its negative components to zero in all the levels but the deepest one. In the deepest level an exited path is taken into account in our estimator but, due to the effect of an indicator function, it does not contribute to the approximation of the quantity of interest.

3.2. Global error decomposition. In this section we show how the computational global error $\mathcal{E}_L$ can be naturally decomposed into three components: the discretization error, $\mathcal{E}_{I,L}$, and the exit error, $\mathcal{E}_{E,L}$, both coming from the tau-leap part of the reaction-splitting method, and the Monte Carlo statistical error, $\mathcal{E}_{S,L}$. We also give upper bounds for each one of the three components.

The computational global error, $\mathcal{E}_L$, is defined as

$$\mathcal{E}_L := \mathrm{E}[g(X(T))] - \mathcal{M}_L$$

and can be decomposed into

$$\begin{aligned}\mathrm{E}[g(X(T))] - \mathcal{M}_L &= \mathrm{E}[g(X(T))(\mathbf{1}_{A_L} + \mathbf{1}_{A_L^c})] - \mathrm{E}[g_L\mathbf{1}_{A_L}] + \mathrm{E}[g_L\mathbf{1}_{A_L}] - \mathcal{M}_L\\ &= \underbrace{\mathrm{E}[g(X(T))\mathbf{1}_{A_L^c}]}_{=:\mathcal{E}_{E,L}} + \underbrace{\mathrm{E}[(g(X(T))-g_L)\,\mathbf{1}_{A_L}]}_{=:\mathcal{E}_{I,L}} + \underbrace{\mathrm{E}[g_L\mathbf{1}_{A_L}]-\mathcal{M}_L}_{=:\mathcal{E}_{S,L}}.\end{aligned}$$

For each reaction-splitting path, $(\bar{X}_\ell(t_{n,\ell},\bar{\omega}))_{n=0}^{N(\bar{\omega})}$, we define the sequence of dual weights, $(\varphi_{n,\ell}(\bar{\omega}))_{n=1}^{N(\bar{\omega})}$, backward as follows:

$$\begin{aligned}\varphi_{N(\bar{\omega}),\ell} &:= \nabla g(\bar{X}_\ell(t_{N(\bar{\omega}),\ell},\bar{\omega})),\\ \varphi_{n,\ell} &:= \left(Id + \Delta t_{n,\ell}\mathbb{J}_a^T(\bar{X}_\ell(t_{n,\ell},\bar{\omega}))\,\nu^T\right)\varphi_{n+1,\ell}, \quad n = N(\bar{\omega})-1,\dots,1,\end{aligned} \tag{3.4}$$

where $\Delta t_{n,\ell}{:=}t_{n+1,\ell}{-}t_{n,\ell}$, ∇ is the gradient operator, and $\mathbb{J}_a(\bar{X}_\ell(t_{n,\ell},\bar{\omega}))\equiv[\partial_i a_j(\bar{X}_\ell(t_{n,\ell},\bar{\omega}))]_{j,i}$ is the Jacobian matrix of the propensity function, a_j, for $j{=}1\dots J$ and $i{=}1\dots d$. We then approximate $\mathcal{E}_{I,L}$ by $\mathcal{A}\left(\mathcal{E}_{I,L}(\bar{\omega});\cdot\right)$, where

$$\mathcal{E}_{I,L}(\bar{\omega}) := \sum_{n=1}^{N(\bar{\omega})}\left(\gamma_{n,\mathrm{TL}} + \gamma_{n,\mathrm{SP}}\right)(\bar{\omega}),$$

where the terms $\gamma_{n,\mathrm{TL}}$ and $\gamma_{n,\mathrm{SP}}$ that take into account the contributions of the tau-leap approximation and the reaction splitting, respectively, are given by

$$\begin{aligned}\gamma_{n,\mathrm{TL}} &:= \frac{\Delta t_{n,L}}{2}\sum_{j=1}^{J}\mathbf{1}_{j\in\mathcal{R}_{\mathrm{TL}}(n)}\left(\nu_j\cdot\varphi_{n,L}\right)\left(a_j(\bar{X}_L(t_{n+1,\ell}))-a_j(\bar{X}_L(t_{n,\ell}))\right),\\ \gamma_{n,\mathrm{SP}} &:= \left(\varphi_{n,L}\cdot\mathrm{E}[\tilde{e}_{n,L}]\right),\end{aligned}$$

where $\mathcal{A}\left(X;M\right){:=}\frac{1}{M}\sum_{m=1}^{M}X(\omega_m)$ and $\mathcal{S}^2\left(X;M\right){:=}\mathcal{A}\left(X^2;M\right)-\mathcal{A}\left(X;M\right)^2$ denote the sample mean and the sample variance of the random variable, X, respectively, and the term $\mathrm{E}[\tilde{e}_{n,L}]$ is discussed in section 3.3. Here $\mathbf{1}_{j\in\mathcal{R}_{\mathrm{TL}}(n)}=1$ if and only if, at time $t_{n,\ell}$, the tau-leap method was used for reaction channel j. We denote by Id the $d\times d$ identity matrix.

The variance of the statistical error, $\mathcal{E}_{S,L}$, is given by $\sum_{\ell=0}^{L}\frac{\mathcal{V}_\ell}{M_\ell}$, where $\mathcal{V}_0 := \mathrm{Var}\left[g_0\mathbf{1}_{A_0}\right]$ and $\mathcal{V}_\ell := \mathrm{Var}\left[g_\ell\mathbf{1}_{A_\ell} - g_{\ell-1}\mathbf{1}_{A_{\ell-1}}\right]$, $\ell\geq 1$. To estimate $\mathcal{V}_\ell$, $\ell\geq 1$ we use the following estimator:

$$\hat{\mathcal{V}}_\ell := \mathcal{S}^2\left(\sum_n \mathrm{E}[\varphi_{n+1}\cdot e_{n+1}\,\big|\,\mathcal{F}](\bar{\omega}); M_\ell\right) + \mathcal{A}\left(\sum_n \mathrm{Var}\left[\varphi_{n+1}\cdot e_{n+1}\,\big|\,\mathcal{F}\right](\bar{\omega}); M_\ell\right),$$

where $\mathcal{F}$ is a suitable chosen sigma algebra such that $(\varphi_n(\bar{\omega}))_{n=1}^{N(\bar{\omega})}$ is measurable, with $N(\bar{\omega})$ being the total number of steps given by Algorithm 3. In this way, the only randomness in $\mathrm{E}[\varphi_{n+1}\cdot e_{n+1}\,|\,\mathcal{F}]$ and $\mathrm{Var}\left[\varphi_{n+1}\cdot e_{n+1}\,|\,\mathcal{F}\right]$ comes from the local errors, $(e_n)_{n=1}^{N(\bar{\omega})}$, defined as $e_{n+1} := \tilde{X}_{\ell,n+1} - \tilde{X}_{\ell-1,n+1}$, where the processes $\tilde{X}_\ell$ and $\tilde{X}_{\ell-1}$ are pure tau-leap processes corresponding to the meshes Δt_ℓ and $\Delta t_{\ell-1}$, respectively, and both starting from the same initial point $\tilde{X}_{\ell-1,n}$. In the aforementioned work, we derived exact and approximate formulas for computing $\mathrm{E}[\varphi_{n+1}\cdot e_{n+1}\,|\,\mathcal{F}]$ and $\mathrm{Var}\left[\varphi_{n+1}\cdot e_{n+1}\,|\,\mathcal{F}\right]$. Details and full explanations for the formulae presented here can be found in [24].

3.3. Local error splitting. Let us briefly comment on the structure of the local error, e_{n+1}, under our splitting assumption. Consider a nontrivial partition $\{\mathcal{R}_A, \mathcal{R}_B\}$ of the set, $\mathcal{R}$, of the reaction channels and its respective reaction networks, X^A, X^B, and X. According to the Kurtz representation in the time interval $[0,t]$ (1.2), we have

$$\begin{aligned} X(t) &= x_0 + \sum_{j\in\mathcal{R}_A}\nu_j Y_j\left(\int_0^t a_j(X_s)ds\right) + \sum_{j\in\mathcal{R}_B}\nu_j Y_j\left(\int_0^t a_j(X_s)ds\right),\\ X^A(t) &= x_0 + \sum_{j\in\mathcal{R}_A}\nu_j Y_j\left(\int_0^t a_j(X_s^A)ds\right),\\ X^B(t) &= x_0 + \sum_{j\in\mathcal{R}_B}\nu_j Y_j\left(\int_0^t a_j(X_s^B)ds\right), \end{aligned}$$

where, as usual, $Y_j : \mathbb{R}_+\times\Omega\to\mathbb{Z}_+$ are independent unit-rate Poisson processes. Under our splitting assumption we are locally replacing the process X by $\tilde{X}$ defined as follows:

$$\tilde{X}(t) = x_0 + (X^A(t)-x_0) + (X^B(t)-x_0).$$

As a consequence, the local splitting error in the interval $[0,t]$ is

$$\begin{aligned} X(t)-\tilde{X}(t) = &\sum_{j\in\mathcal{R}_A}\nu_j Y_j\left(\int_0^t \left(a_j(X_s)-a_j(X_s^A)\right)ds\right)\\ &+\sum_{j\in\mathcal{R}_B}\nu_j Y_j\left(\int_0^t \left(a_j(X_s)-a_j(X_s^B)\right)ds\right). \end{aligned}$$

To approximate the expected value $\mathrm{E}[X(t)-\tilde{X}(t)]$ for a small time t, we (i) approximate $a_j(X_s)-a_j(X_s^A)$ by $\nabla a_j(X_s^A)\cdot(X_s-X_s^A)$ and (ii) apply the trapezoidalrule to approximate $\int_0^t \left(a_j(X_s) - a_j(X_s^A)\right) ds$ by $t/2\left(\nabla a_j(X_t^A)\cdot(X_t - X_t^A)\right)$. We apply the same approximations in the case of X^B. Let us now consider a deterministic approximation of $X_t - X_t^A$ by one forward Euler step applied to their associated reaction rate ODEs. After few simple calculations we obtain

$$X_t - X_t^A \approx t \sum_{k\in\mathcal{R}_B} \nu_k a_k(x_0),$$

and as a consequence

$$\begin{aligned}\int_0^t \left(a_j(X_s) - a_j(X_s^A)\right) ds &\approx t/2\left(\nabla a_j(X_t^A)\cdot(X_t - X_t^A)\right)\\ &\approx t^2/2 \sum_{k\in\mathcal{R}_B} a_k(x_0)(\nabla a_j(X_t^A)\cdot \nu_k),\end{aligned}$$

where we can approximate $X_t^A \approx x_0 + t\sum_{j\in\mathcal{R}_A}\nu_j a_j(x_0)$. We proceed analogously with the approximation X_t^B.

In this way, the expected value $\mathrm{E}\left[X(t) - \tilde{X}(t)\right]$ can be approximated by

(3.5)
$$t^2/2\left(\sum_{j\in\mathcal{R}_B}\nu_j \sum_{k\in\mathcal{R}_B} a_k(x_0)(\nu_k\cdot\nabla a_j(X_t^A)) + \sum_{j\in\mathcal{R}_B}\nu_j\sum_{k\in\mathcal{R}_A} a_k(x_0)(\nu_k\cdot\nabla a_j(X_t^B))\right).$$

To approximate the term $\mathrm{E}[\tilde{e}_{n,L}]$ introduced in the previous section we use the expression (3.5) in our dual-weighted error formulae replacing $[0,t]$ by $[t_n, t_{n+1}]$ and recalling that our partitions $\mathcal{R}_A$ and $\mathcal{R}_B$ are computed adaptively. A similar expression can be obtained for the variance of the local error.

Remark 3.2 (backward Euler). In (3.4), we have that the weights $\varphi_{n,\ell}$ can be computed by a backward Euler formula when excessively fine time meshes are required for stability, i.e., by means of recursion, $\varphi_{n,\ell} := \left(Id - \Delta t_{n,\ell}\mathbb{J}_a^T(\bar{X}_\ell(t_{n,\ell},\bar{\omega}))\,\nu^T\right)^{-1}\varphi_{n+1,\ell}$.

3.4. An optimized upper bound for the global exit probability. We showed in [23] that by choosing adequately the one-step exit probability bound, δ, the exit error, $\mathcal{E}_{E,L}$, satisfies $|\mathcal{E}_{E,L}| \le |\mathrm{E}[g(X(T))]|\,\mathrm{P}(A^c) \le TOL^2$. We now discuss how to build an approximate upper bound for $\mathrm{P}(A^c)$ in the context of reaction-splitting paths.

Given τ and the current state of the approximate process, $\bar{X}$, at time t, consider a tau-leap step taking into account the reaction channels, $\mathcal{R}_j \in \mathcal{R}_{\mathrm{TL}}(x)$. We would like to find an upper bound for the following probability:

$$\mathrm{P}\left(x + \sum_j \nu_j \mathcal{P}_j(a_j(x)\tau) \notin \mathbb{Z}_+^d \,\middle|\, \bar{X}(t) = x\right). \tag{3.6}$$

Let us consider the ith component of x and the case in which, for some j, we have that $\nu_{j,i} < 0$; otherwise, the probability (3.6) is trivially zero. In this case, the Chernoff inequality (obtained by means of the Markov inequality and the moment generating function of a Poisson random variable) gives

$$\mathrm{P}\left(x_i + \sum_j \nu_{j,i}\mathcal{P}_j(a_j(x)\tau) < 0 \,\middle|\, \bar{X}(t) = x\right) < \inf_{s>0} F_i(s), \tag{3.7}$$

where $F_i(s) := \exp(-sx_i + \tau \sum_j a_j(x)(\exp(-s\nu_{j,i}) - 1))$. Given $i \in \{1, 2, \ldots, d\}$, consider the two following conditions:

(I) $\exists j : \nu_{j,i} < 0$ and

(II) $-F_i'(0) = x_i + \tau \sum_j a_j(x)\nu_{j,i} > 0$.

It is easy to see that if (I) and (II) are true, then $\inf_{s>0} F_i(s)$ has a nontrivial solution, but if (II) is true and (I) is false, then $\inf_{s>0} F_i(s) = 0$. Finally, if (II) is false, then $\inf_{s>0} F_i(s) = 1$. From these considerations, we can obtain that

$$\mathrm{P}\left(x + \sum_j \nu_j \mathcal{P}_j(a_j(x)\tau) \notin \mathbb{Z}_+^d \,\middle|\, \bar{X}(t) = x \right) < \max_i \inf_{s>0} F_i(s). \tag{3.8}$$

Given x and τ, we can compute using any standard iterative method the value $\eta_i(x, \tau) := \inf_{s>0} F_i(s)$. We define $\eta(x, \tau) := \max_i \eta_i(x, \tau)$.

Here it is worth recalling that the Chernoff tau, $\tau_{Ch}(x, \delta)$, is computed in such a way that the one-step exit probability in (3.6) is less than δ. Observe now that each time we take a reaction-splitting step three things can happen: (i) $\mathcal{R}_{\mathrm{TL}} \neq \emptyset$ and $\tau_{Ch}(x, \delta) \leq \tilde{T} - t$, (ii) $\mathcal{R}_{\mathrm{TL}} \neq \emptyset$ and $\tau_{Ch}(x, \delta) > \tilde{T} - t$, and (iii) $\mathcal{R}_{\mathrm{TL}} = \emptyset$. In (i), the one-step exit probability is less than δ; in (ii), this probability is less than $\eta(x, \tau)$, which can be much less than δ; in (iii), the one-step exit probability is obviously zero.

Summing up, the probability for one path to reach the final time, T, without exiting the lattice is greater than the product of factors $(1 - \delta)$, $(1 - \eta)$ (and 1 in case (iii) of course). We can compute this product, π, for each generated reaction-splitting path and conclude that $\mathrm{P}(A^c) \lessapprox 1 - \mathcal{A}(\pi_m; M)$ for M sufficiently large.

This approximate bound for $\mathrm{P}(A^c)$ is much less stringent than the one we proposed in [23], namely, $\delta \mathrm{E}[N_{\mathrm{TL}}]$, where N_{TL} is the number of tau-leap steps in one approximate path.

3.5. Estimation procedure. In this section, we briefly describe the automatic procedure that estimates $\mathrm{E}[g(X(T))]$ within a given prescribed relative tolerance, $TOL{>}0$, up to a given confidence level. Up to minor changes, it is the same as the one presented in [24]. It is important to remark that the minimal user intervention is required to obtain the parameters needed to simulate the reaction-splitting paths and, subsequently, to compute the estimations using (3.2). Once the reaction network is given (stoichiometric matrix ν and J propensity functions, a_j), the user only needs to set the required maximum allowed relative global error or tolerance, TOL, and the confidence level, α. This process has three phases:

Phase I. Calibration of virtual machine-dependent quantities. In this phase, we estimate the quantities C_{MNRM}, C_{TL}, C_s and the function C_P that allow us to model the expected computational work, measured in runtime.

Phase II. Solution of the work optimization problem. We obtain the total number of levels, L, and the sequences, $(\delta_\ell)_{\ell=0}^L$ and $(M_\ell)_{\ell=0}^L$, i.e., the one-step exit probability bounds and the required number of simulations at each level. In this phase, given a relative tolerance, $TOL{>}0$, we solve the work optimization problem

$$\begin{cases} \min_{\{\Delta t_0, L, (M_\ell, \delta_\ell)_{\ell=0}^L\}} \sum_{\ell=0}^L \psi_\ell M_\ell \\ \text{s.t.} \\ \mathcal{E}_{E,L} + \mathcal{E}_{I,L} + \mathcal{E}_{S,L} \leq TOL. \end{cases} \tag{3.9}$$

An algorithm to efficiently compute the solution of this optimization problem is given in [24]. Our objective function is the expected total work of the MLMC estimator, $\mathcal{M}_L$, i.e., $\sum_{\ell=0}^{L} \psi_\ell M_\ell$, where L is the deepest level, ψ_0 is the expected work of a single-level path at level 0, and ψ_ℓ, for $\ell \geq 1$, is the expected computational work of two coupled paths at levels $\ell-1$ and ℓ. Finally, M_0 is the number of single-level paths at level 0, and M_ℓ, for $\ell \geq 1$, is the number of coupled paths at levels $\ell-1$ and ℓ. We now describe the quantities $(\psi_\ell)_{\ell=0}^{L}$. First, ψ_0 is the expected work of a single reaction-splitting path (simulated by Algorithm 2),

$$\begin{aligned} \psi_0 &:= C_{\mathrm{MNRM}} \mathrm{E}[N_{\mathrm{MNRM}}(\Delta t_0, \delta_0)] + C_{\mathrm{TL}} \mathrm{E}[N_{\mathrm{TL}}(\Delta t_0, \delta_0)] \\ &\quad + \int_{[0,T]} \mathrm{E}[\sum_{j \in \mathcal{R}_{\mathrm{TL}}(s)} C_P(a_j(\bar{X}_0(s)) \tau_{Ch}(\bar{X}_0(s), \delta_0)) ds], \end{aligned} \tag{3.10}$$

where Δt_0 is the size of the time mesh at level 0, δ_0 is the exit probability bound at level 0, and $\mathcal{R}_{\mathrm{TL}} = \mathcal{R}_{\mathrm{TL}}(t)$ is the tau-leap set, which depends on time (and also the current state of the process). The $\mathcal{R}_{\mathrm{TL}}$ set is determined at each decision step by Algorithm 1. Therefore, the expected work at level 0 is $\psi_0 M_0$, where M_0 is the total number of single reaction-splitting paths. For $\ell \geq 1$, we use Algorithm 3 to generate M_ℓ-coupled paths that couple levels $\ell-1$ and ℓ. The expected work of a pair of coupled reaction-splitting paths at levels ℓ and $\ell - 1$ is

$$\begin{aligned} \psi_\ell &:= C_{\mathrm{MNRM}} \mathrm{E}[N^{(c)}_{\mathrm{MNRM}}(\ell)] + C_{\mathrm{TL}} \mathrm{E}[N^{(c)}_{\mathrm{TL}}(\ell)] \\ &\quad + \int_{[0,T]} \mathrm{E}[\sum_{j \in \mathcal{R}_{\mathrm{TL},\ell}(s)} C_P(a_j(\bar{X}_\ell(s)) \tau_{Ch}(\bar{X}_\ell(s), \delta_\ell)) ds] \\ &\quad + \int_{[0,T]} \mathrm{E}[\sum_{j \in \mathcal{R}_{\mathrm{TL},\ell-1}(s)} C_P(a_j(\bar{X}_{\ell-1}(s)) \tau_{Ch}(\bar{X}_{\ell-1}(s), \delta_{\ell-1})) ds], \end{aligned} \tag{3.11}$$

where

$$\begin{aligned} N^{(c)}_{\mathrm{MNRM}}(\ell) &:= N_{\mathrm{MNRM}}(\Delta t_\ell, \delta_\ell) + N_{\mathrm{MNRM}}(\Delta t_{\ell-1}, \delta_{\ell-1}), \\ N^{(c)}_{\mathrm{TL}}(\ell) &:= N_{\mathrm{TL}}(\Delta t_\ell, \delta_\ell) + N_{\mathrm{TL}}(\Delta t_{\ell-1}, \delta_{\ell-1}). \end{aligned}$$

Phase III. Estimation of $\mathrm{E}[g(X(T))]$.

Remark 3.3 (on the computational complexity of the reaction-splitting method). Using the same arguments employed in the proof of the computational complexity of the multilevel hybrid Chernoff tau-leap method [24], it is possible to show that the computational complexity of the reaction-splitting method is also of order $\mathcal{O}(TOL^{-2})$. At the heart of the proof lies the fact that, by construction, Algorithm 1 selects a partition with nearly minimal expected cost among a set of partitions that include the pure pathwise-exact paths. As a consequence, the expected cost of a reaction-splitting path is bounded by a constant K multiplied by the expected cost of an exact path. The constant K is a virtual machine quantity associated with the cost of computing the splits (in our numerical experiments K is never greater than 3).

Remark 3.4. Note that in the work [1], the proposed unbiased time-discretization scheme can achieve a computational complexity of order $\mathcal{O}(TOL^{-2})$ by choosing a fixed number of levels or, more precisely, a number of levels independent of TOL. Instead, the authors in [1] obtain a computational complexity of order $\mathcal{O}(TOL^{-2}\log(TOL)^2)$ since they overrefine the time-mesh until $\Delta t_L = \mathcal{O}(TOL)$.

4. A novel control variate based on a deterministic time change.

In this section, we motivate a novel control variate for the random variable $X(T,\omega)$ defined by the random time change representation,

$$X(T,\omega) = x_0 + \sum_j \nu_j Y_j \left(\int_0^T a_j(X(s))\, ds, \omega \right).$$

First, we replace each of the independent Poisson processes, $(Y_j(s,\omega))_{s\geq 0}$, by the identity function. This defines the deterministic mean field, namely,

$$Z(T) = x_0 + \sum_j \nu_j \int_0^T a_j(Z(s))\, ds.$$

Next, we consider the random variable

$$\tilde{X}(T,\omega) = x_0 + \sum_j \nu_j Y_j \left(\int_0^T a_j(Z(s))\, ds, \omega \right),$$

which uses the same realizations of $(Y_j(s,\omega))_{s\geq 0}$ that define $X(T,\omega)$ but a different, deterministic clock for each Y_j based on Z. In this way, we expect some correlation between $X(T)$ and $\tilde{X}(T)$. Since $\mathrm{E}[\tilde{X}(T)] = Z(T)$ is a quantity which we can approximate very efficiently with a deterministic method, we have that $\tilde{X}(T)$ is a potential control variate for $X(T)$ obtained at almost negligible extra computational cost.

We have that $\tilde{X}(T,\omega)$ can be considered as a deterministic time change approximation of $X(T,\omega)$.

To implement this idea, we first consider the sequence Z_k, defined as a forward Euler discretization of the mean field over a suitable mesh, $\{t_0{=}0, t_1, \ldots, t_K{=}T\}$, $\Delta t_k := t_{k+1}{-}t_k$, $k{=}0,1,\ldots,K{-}1$, that is,

$$\begin{cases} Z_{k+1} = Z_k + \sum_j \nu_j a_j(Z_k)\Delta t_k, & k{=}0,\ldots,K{-}1, \\ Z_0 = x_0. \end{cases}$$

The sequence Z_k allows us to define another sequence, $\hat{\Lambda}_{j,k}$, by

$$\begin{cases} \hat{\Lambda}_{j,k+1} = \hat{\Lambda}_{j,k} + a_j(Z_k)\Delta t_k, & k{=}1,\ldots,K{-}1, \\ \hat{\Lambda}_{j,0} = 0, \end{cases}$$

where $\hat{\Lambda}_{j,K}$ clearly approximates $\int_0^T a_j(Z(s))\, ds$.

Then, for each realization of $\bar{X}(T,\omega)$, which is an approximation of $X(T,\omega)$, we compute the control variate:

$$\hat{X}_K = x_0 + \sum_j \nu_j Y_j \left(\hat{\Lambda}_{j,K} \right), \tag{4.1}$$

which is the corresponding approximation of $\tilde{X}(T,\omega)$ and has the computable expectation

$$\mu_K := \mathrm{E}[\hat{X}_K] = x_0 + \sum_j \nu_j \hat{\Lambda}_{j,K}.$$

Now, we consider the random sequence, $\{\bar{X}_n(\omega)\}_{n=0}^{N(\omega)}$, generated by any approximation algorithm. Here, $\bar{X}_{N(\omega)}(\omega)$ is an approximation of $X(T,\omega)$. The sequence of random times, $\{\bar{\Lambda}_{j,n}(\omega)\}$, is defined by

$$\begin{cases} \bar{\Lambda}_{j,n+1} = \bar{\Lambda}_{j,n} + a_j(\bar{X}_n(\omega))\Delta s_n, \quad n = 0,\ldots,N(\omega)-1, \\ \bar{\Lambda}_{j,0} = 0, \end{cases}$$

over the mesh $\{s_0 = 0, s_1, \ldots, s_{N(\omega)} = T\}$, $\Delta s_n := s_{n+1} - s_n$, $n = 0, 1, \ldots, N(\omega)-1$.

At this point, it is crucial to observe that we can keep track of the values $Y_j(\bar{\Lambda}_{j,n}, \omega)$, since at each step of the approximation algorithm, we are sampling the increments of the processes, Y_j. From now on, we omit ω in our notation for the sake of simplicity.

The values $Y_j(\hat{\Lambda}_{j,K})$, required in (4.1), can by obtained by sampling the process Y_j as follows. For each realization of $\bar{X}$, we have only two scenarios:

1. For some n, $\bar{\Lambda}_{j,n} < \hat{\Lambda}_{j,K} < \bar{\Lambda}_{j,n+1}$. Since $Y_j\left(\bar{\Lambda}_{j,n}\right)$ and $Y_j\left(\bar{\Lambda}_{j,n+1}\right)$ are known, we sample a Poissonian bridge (binomial), i.e.,

$$\begin{aligned} Y_j(\hat{\Lambda}_{j,K}) \,\big|\, \left(Y_j\left(\bar{\Lambda}_{j,n}\right), Y_j\left(\bar{\Lambda}_{j,n+1}\right)\right) &\sim Y_j\left(\bar{\Lambda}_{j,n}\right) + \mathcal{B}, \\ \mathcal{B} &\sim \text{binomial}\left(Y_j\left(\bar{\Lambda}_{j,n+1}\right) - Y_j\left(\bar{\Lambda}_{j,n}\right), \frac{\hat{\Lambda}_{j,K} - \bar{\Lambda}_{j,n}}{\bar{\Lambda}_{j,n+1} - \bar{\Lambda}_{j,n}}\right). \end{aligned}$$

2. $\hat{\Lambda}_{j,K} > \bar{\Lambda}_{j,N}$. Since we know the value $Y_j\left(\bar{\Lambda}_{j,N}\right)$, we simply have to sample a Poisson random variate as follows:

$$\begin{gathered} Y_j(\hat{\Lambda}_{j,K}) \,\big|\, Y_j\left(\bar{\Lambda}_{j,N}\right) \sim Y_j\left(\bar{\Lambda}_{j,N}\right) + \mathcal{P}, \\ \mathcal{P} \sim \text{Poisson}(\hat{\Lambda}_{j,K} - \bar{\Lambda}_{j,N}). \end{gathered}$$

Finally, using the aforementioned control variate, we can estimate $\mathrm{E}[g(\bar{X}(T))]$ with

$$\frac{1}{M}\sum_{m=1}^{M} g(\bar{X}_N(\omega_m)) - \frac{1}{M}\sum_{m=1}^{M}\left(g(\hat{X}_K(\omega_m)) - g(\mu_K)\right) \tag{4.2}$$

for any linear functional, g, since $\mathrm{E}\,[g(\hat{X}_K)] = g(E[\hat{X}_K]) = g(\mu_K)$.

Remark 4.1 (nonlinear observables). Observe in (4.1) that $\hat{X}_K$ is a linear combination of independent Poisson random variables. For that reason, we can exactly compute the expected value of any polynomial or exponential function of $\hat{X}_K$. Let $\mathcal{C}$ be the class of functions spanned by the family of polynomials and exponential functions on $\hat{X}_K$. Let g be any nonlinear function and $\tilde{g}$ a suitable approximation from the class $\mathcal{C}$. Consider now the following telescoping sum:

$$\mathrm{E}[g(\bar{X}_N)] = \mathrm{E}[(g-\tilde{g})(\bar{X}_N)] + \mathrm{E}[\tilde{g}(\bar{X}_N) - \tilde{g}(\hat{X}_K)] + \mathrm{E}[\tilde{g}(\hat{X}_K)].$$

Observe that the random variable $(g-\tilde{g})(\bar{X}_N)$ has small variance since g is well approximated by $\tilde{g}$ while the variance of $\tilde{g}(\bar{X}_N) - \tilde{g}(\hat{X}_K)$ is small since $\hat{X}_K$ is coupled to $\bar{X}_N$.

This observation leads us to approximate $\mathrm{E}[g(\bar{X}_N)]$ by

$$\frac{1}{M}\sum_{m=1}^{M}\left(g(\bar{X}_N(\omega_m)) - \tilde{g}(\hat{X}_K(\omega_m)\right) + \mathrm{E}[\tilde{g}(\hat{X}_K))],$$

where the last term can be exactly computed.

Remark 4.2 (reducing the variance at the coarsest level). Consider the trivial decomposition

$$g(\bar{X}_0(T)) = g(\tilde{X}(T)) + \left(g(\bar{X}_0(T)) - g(\tilde{X}(T))\right),$$

where $\tilde{X}(T)$ is the deterministic time change control variate. Then,

$$\mathrm{E}[g(\bar{X}_0(T))] = \mathrm{E}[g(\tilde{X}(T))] + \mathrm{E}[g(\bar{X}_0(T)) - g(\tilde{X}(T))].$$

We assume that we can compute exactly $\mathrm{E}[g(\tilde{X}(T))]$ (if not, we can use a $\tilde{g}$ as in Remark 4.1), we simply have to estimate $\mathrm{E}[g(\bar{X}_0(T)) - g(\tilde{X}(T))]$ instead of $\mathrm{E}[g(\bar{X}_0(T))]$ in our multilevel scheme.

The computational gain lies in the fact that Var $[g(\bar{X}_0(T)) - g(\tilde{X}(T))]$ could be substantially smaller than Var $\left[g(\bar{X}_0(T))\right]$.

Remark 4.3 (computational cost). An advantage of this control variate is that the computational cost is almost negligible because we only need to store two scalars, $\bar{\Lambda}_{j,n}$ and $\bar{\Lambda}_{j,n+1}$, for each reaction, j. These values are determined at each step by $a_j(\bar{X}_n)$, which is a quantity that is already computed at each time step of the reaction-splitting algorithm. Also, for each realization of the control variate, at most one additional Poisson random variate is needed for each reaction channel.

5. Numerical examples.

In this section, we present three examples that illustrate the reaction-splitting method. The first one is proposed to show explicitly an example where the reaction-splitting method has a clear advantage over the pure tau-leap strategy by Anderson and Higham [1]. The second example compares the reaction splitting method against the hybrid approach presented in [24]. Also, for this example it is possible to empirically verify that the Phase II algorithm computes parameters such that the prescribed tolerance is achieved within the required confidence. The third example shows the performance of the novel control variate technique.

5.1. Repressilator.

This model was first presented in the work [10] and then adapted to mass action kinetics in the work [21]. It has 6 species and 15 reactions,

$$\begin{array}{rlrlrl}
\emptyset & \xrightarrow{0.1} m_A, & \emptyset & \xrightarrow{0.1} m_B, & \emptyset & \xrightarrow{0.1} m_C,\\
m_A & \xrightarrow{50} m_A + p_A, & m_B & \xrightarrow{50} m_B + p_B, & m_C & \xrightarrow{50} m_C + p_C,\\
m_A & \xrightarrow{0.01} \emptyset, & m_B & \xrightarrow{0.01} \emptyset, & m_C & \xrightarrow{0.01} \emptyset,\\
m_A + p_B & \xrightarrow{50} p_B, & m_B + p_C & \xrightarrow{50} p_C, & m_C + p_A & \xrightarrow{50} p_A,\\
p_A & \xrightarrow{0.01} \emptyset, & p_B & \xrightarrow{0.01} \emptyset, & p_C & \xrightarrow{0.01} \emptyset.
\end{array}$$

In this model, the state vector $X(t){=}(m_A(t), m_B(t), m_C(t), p_A(t), p_B(t), p_C(t))$ corresponds to the number of mRNA's and proteins, respectively. The propensity functions are given by stochastic mass action kinetics. The quantity of interest is $g(X(t)){:=}p_A(t)$, the initial state is $X_0 = (10, 500, 0, 0, 0, 0)$, and the final time is $T{=}4750$. The initial time mesh is uniform and its size is $T/2^{11}$.

In this example, it is not possible to apply a pure tau-leap multilevel strategy, because the coarsest uniform grid required for implementing such a strategy renders the computational work of simulating a tau-leap path larger than the work of an exact path. The expected number of exact steps for one MNRM trajectory is 2.5e+05 ($\approx 2^{21}{+}2^{19}$). Using this exact method, we estimate $\mathrm{E}[g(X(T))]$ using 1.0e+03 samples

Table 5.1
Results of estimating the quantity of interest using a pure tau-leap scheme for $M = 1000$ and $\Delta t = T/2^{19}$.

	TL steps	$\hat{\mathrm{E}}\left[g(\bar{X}(T))\right]$	$\sqrt{\widehat{\mathrm{Var}}\left[g(\bar{X}(T))\right]/M}$
Δt	2^{19}	8.05e+01	1.75e+01
$\Delta t/2$	2^{20}	4.20e+02	5.29e+01
$\Delta t/4$	2^{21}	1.77e+03	1.64e+02
$\Delta t/8$	2^{22}	5.50e+03	3.19e+02
$\Delta t/16$	2^{23}	1.49e+04	5.02e+02
$\Delta t/32$	2^{24}	2.10e+04	5.78e+02
$\Delta t/64$	2^{25}	2.51e+04	6.14e+02
$\Delta t/128$	2^{26}	2.78e+04	6.22e+02

Table 5.2
Details of the ensemble run of the Phase II *algorithm.*

TOL	2.00e-01	1.00e-01	5.00e-02	2.50e-02
$\frac{\hat{W}_{ML}}{\hat{W}_{\mathrm{SSA}}}$	4.67e-02	4.15e-02	4.86e-02	9.05e-02

and the result is 2.8e+04 with a standard error of 5.8e+02. The reaction-splitting method, for an initial grid of size $T/2^{11}$, gives an estimation of 2.9e+04 with a standard error of 5.9e+02. On average, the reaction-splitting method took 5907 exact steps and only 2097 ($\approx 2^{11}$) tau-leap steps. The required number of steps in the pure tau-leap method to obtain an estimation close to 2.9e+04 is 2^{25}. In Table 5.1 we show, for different grid sizes, the estimation using the tau-leap method with a fixed number of samples.

We now analyze an ensemble of 30 independent runs of the Phase II algorithm (see section 3.5), using different relative tolerances. In Table 5.2 we show the quotient between the total computational work for the multilevel reaction-splitting method and the SSA method, for δ=0.6. It is worth mentioning that by selectively applying the exact method to a subset of reaction channels, the reaction-splitting algorithm is able to substantially reduce the exit error without imposing a significant small control parameter δ. In fact, for $\delta = 0.6$, the empirical one-step exit probability in this example turns out to be 6.5e-05 on average.

5.2. Intracellular virus kinetics. This model, first developed in the work [28], has four species and six reactions:

- $E \xrightarrow{1} E{+}G$, the viral template (E) forms a viral genome (G),
- $G \xrightarrow{0.025} E$, the genome generates a new template,
- $E \xrightarrow{1000} E{+}S$, a viral structural protein (S) is generated,
- $G{+}S \xrightarrow{7.5\text{e-}06} V$, the virus (V) is produced,
- $E \xrightarrow{0.25} \emptyset$, $S \xrightarrow{2} \emptyset$ degradation reactions.

The propensity functions are given by stochastic mass action kinetics, that is,

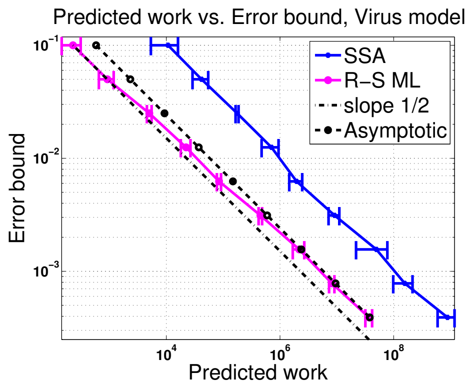
$$a(X) = (E,\ 0.025\,G,\ 1000\,E,\ 7.5\text{e-}06\,G\,S,\ 0.25\,E,\ 2\,S)\,.$$

In this model, $X(t)$=$(G(t), S(t), E(t), V(t))$ and $g(X(t))$:=$V(t)$ is the number of viruses produced. The initial state is X_0=$(0, 0, 10, 0)$ and the final time is T=20. The initial time mesh is uniform and its size is 0.5. This example is interesting because (i) it shows a clear separation of time scales, (ii) our previous hybrid Chernoff method has no computational work gain with respect to an exact method, and (iii) in the work [1], the authors take an alternative approach treating a priori the fast reactions 3 and 5 separately.

We now analyze an ensemble of 10 independent runs of the Phase II algorithm (see section 3.5), using different relative tolerances. In Figure 5.1, we show the total predicted work (runtime) for the multilevel mixed method and for the SSA method in relation to the estimated error bound. We also show the estimated asymptotic work of the multilevel mixed method. We remark that the computational work of the multilevel *hybrid* method (presented in [24]) is the same as the work of the SSA. For all the levels, $\delta = 0.01$ was obtained. In Figure 5.2, we can observe how the estimated weak error, $\hat{\mathcal{E}}_{I,\ell}$, and the estimated variance of the difference of the functional between two consecutive levels, $\hat{\mathcal{V}}_\ell$, decrease linearly as we refine the time mesh, which corresponds to a tau-leap-dominated regime. This linear relationship for the variance starts at level 1 and when the MNRM-dominated regime is reached, both quickly converge to zero as expected. The estimated total path work, $\hat{\psi}_\ell$, increases as we refine the time mesh. We observe that it increases until it reaches a plateau, which corresponds to the pure MNRM regime where the computational cost is independent of the discretization time-mesh size. In Figure 5.3, we show TOL versus the actual computational error. It can be seen that the prescribed tolerance is achieved with the required confidence of 95%, since C_A=1.96 for all the tolerances. The QQ-plot in the right part of Figure 5.3 was obtained as follows: (i) for the range of tolerances specified in the first column of Table 5.1, we ran the Phase II algorithm five times, (ii) for each output of the calibration algorithm, we sampled the multilevel estimator, $\mathcal{M}_L$, defined in 3.2, 100 times. This plot reaffirms our assumption about the Gaussian distribution of the statistical error.

Remark 5.1. In the simulations, we observe that, as we refine TOL, the optimal number of levels approximately increases logarithmically, which is a desirable feature. We fit the model $L^* = a\log(TOL^{-1}) + b$, obtaining a=1.47 and b=3.56.

Remark 5.2. We empirically observe that by controlling the delta parameter, i.e., the one-step exit probability for all the levels we are indirectly controlling the strong error by generating a high proportion of coupled paths that never leave the lattice.



TOL	L^*	$\frac{\hat{W}_{ML}}{\hat{W}_{\text{SSA}}}$
	(avg, min, max)	(avg, min, max)
1.00e-01	(1.2, 1, 2)	(3, 2, 4)e-02
5.00e-02	(2.6, 2, 3)	(2, 2, 2)e-02
2.50e-02	(3.8, 3, 4)	(3, 3, 3)e-02
1.25e-02	(4.6, 4, 5)	(3, 3, 5)e-02
6.25e-03	(5.4, 5, 6)	(5, 3, 6)e-02
3.13e-03	(6.6, 6, 8)	(5, 4, 6)e-02
1.56e-03	(8.4, 7, 9)	(5, 3, 7)e-02
7.81e-04	(9.0, 9, 9)	(6, 4, 8)e-02
3.91e-04	(8.4, 8, 9)	(5, 3, 7)e-02

FIG. 5.1. *Left: Predicted work (runtime) versus the estimated error bound, with* 95% *confidence intervals. The multilevel reaction-splitting method is preferred over the SSA and the multilevel hybrid method for all the tolerances. Right: Details of the ensemble run of the Phase* II *algorithm.*

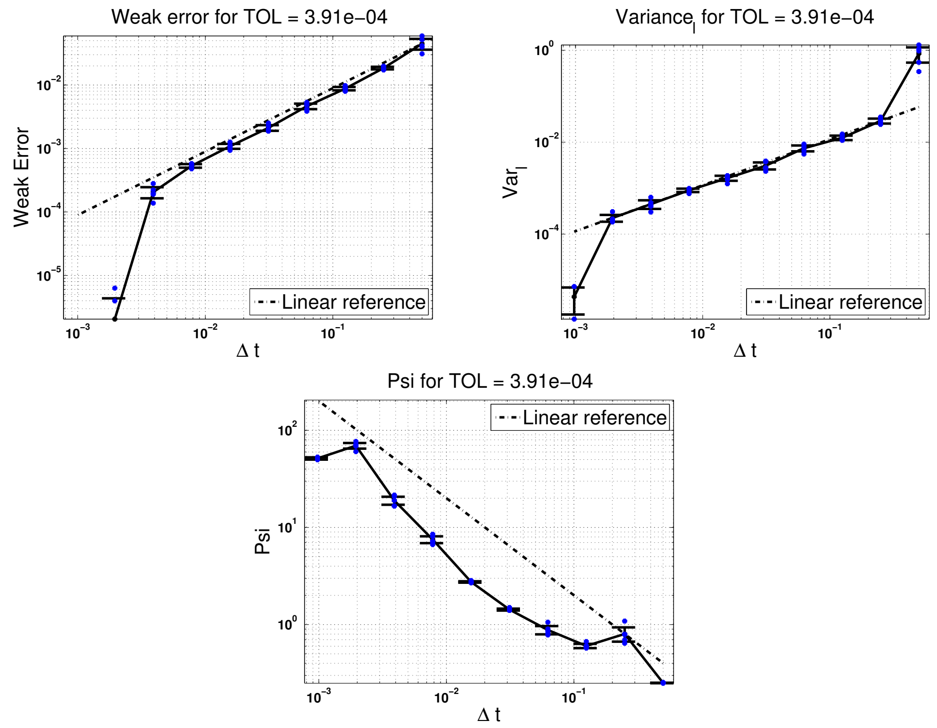


FIG. 5.2. *Upper left: the estimated weak error, $\hat{\mathcal{E}}_{I,\ell}$, as a function of the time mesh size, h. Upper right: the estimated variance of the difference between two consecutive levels, $\hat{\mathcal{V}}_\ell$, as a function of h. Bottom: the estimated path work, $\hat{\psi}_\ell$, as a function of h. In all cases, we show the data points generated in the 10 runs, its average, and the corresponding 95% confidence intervals.*

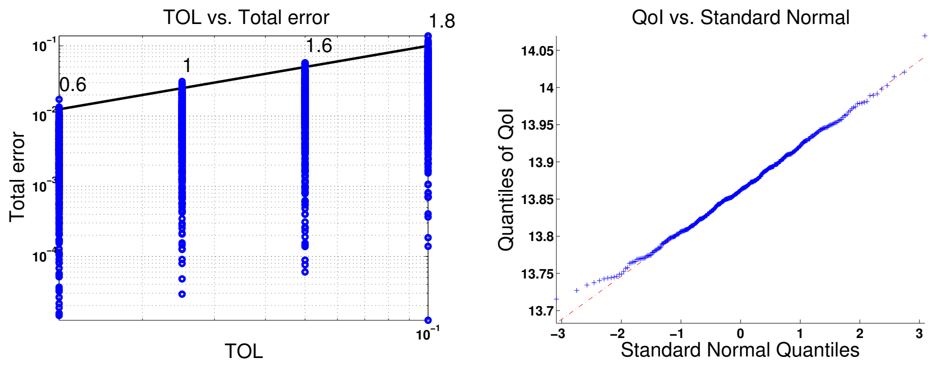


FIG. 5.3. *Left: TOL versus the actual computational error. The numbers above the straight line show the percentage of runs that had errors larger than the required tolerance. We observe that, in all cases, the computational error follows the imposed tolerance with the expected confidence of 95%. Right: quantile-quantile plot based on realizations of $\mathcal{M}_L$.*

5.3. Time-deterministic random change control variate. Here we consider a closed economy formed by one holding company, N small business units, and S startup companies. Each business unit obtains its funds by trading with the environment (represented by the empty set) and it makes net transfers to the holding company according to its current money level. Also, some business units may have an entrepreneurial venture with a startup company to increase its own wealth. At the same time, the holding company pays dividends to the environment and, from time to time, its money level is increased by its own investments. We define the full set of parameters for this example as follows:

the number of business units is $N = 200$; number of startup companies $S = 20$ the initial money levels are $X(0) = (1, \dots, 1, 1.0e{+}05, 1, \dots, 1) \in \mathbb{R}^{N+1+S}$; each business unit obtains its funds from the environment at a constant rate and makes net transfers to the holding company at a rate proportional to its current money level. In the language of biochemical reactions, we have

$$\begin{gathered}\emptyset \xrightarrow{k} X_1, \dots, \emptyset \xrightarrow{k} X_N,\\ X_1 \xrightarrow{c} Y, \dots, X_N \xrightarrow{c} Y.\end{gathered}$$

In addition, some business units may have a venture with the startup companies

$$Z_i + X_i \xrightarrow{d} 3X_i \;\text{ for } i{=}1, \dots, S.$$

The holding company pays dividends to the environment and receives returns from its own investments,

$$Y \xrightarrow{a} \emptyset,\; Y \xrightarrow{b} 50\,Y.$$

This last reaction should be interpreted as

$$\mathrm{P}\left(Y(t + \mathrm{d}t) = y + 49 \,\middle|\, Y(y) = y\right) = b\,y\,\mathrm{d}t + o\left(\mathrm{d}t\right),$$

that is, with a rate proportional to its current money level, the holding company increases its money level by 49 units. As a consequence, the corresponding stoichiometric matrix of size $(2(N{+}1) \times N{+}1)$ is

$$\nu = \left(\begin{array}{cccccccc}
1 & 0 & \dots & 0 & 0 & 0 & \dots & 0\\
0 & 1 & \dots & 0 & 0 & 0 & & 0\\
\vdots & & \dots & \vdots & 0 & \vdots & & \vdots\\
-1 & 0 & \dots & 1 & 0 & 0 & \dots & 0\\
0 & -1 & \dots & 1 & 0 & 0 & \dots & 0\\
\vdots & & \dots & \vdots & & \vdots & & \vdots\\
0 & 0 & \dots & -1 & 0 & 0 & \dots & 0\\
0 & 0 & \dots & 49 & 0 & 0 & \dots & 0\\
2 & 0 & \dots & 0 & -1 & 0 & \dots & 0\\
0 & 2 & \dots & 0 & 0 & -1 & \dots & 0\\
\vdots & & \dots & \vdots & & \vdots & & \vdots\\
0 & 0 & \dots & 0 & 0 & 0 & \dots & -1
\end{array}\right)^{tr},$$

and the associated propensity functions are

$$a(X) = (\underbrace{k, \dots, k}_{N \text{ times}}, c\,X_1, \dots, c\,X_N, a\,Y, b\,Y, d\,Z_1X_1, \dots, d\,Z_SX_S).$$

In our numerical examples, the set of coefficient rates are $k = 1$, $c = 1$, $d = 1$, $a = 0.5$, $b = 0.01$. The final simulation time is $T = 3$.

Table 5.3
Variance comparison between a raw estimation and using the control variate technique for $M = 1000$ path samples and a grid of size Δt=0.125.

$Y(0)$	$\hat{\mathrm{E}}\left[Y(T)\right]$	$\widehat{\mathrm{Var}}[Y(T)]$	$\widehat{\mathrm{Var}}[Y(T)-\tilde{Y}(T)]$	$\frac{\widehat{\mathrm{Var}}[Y(T)-\tilde{Y}(T)]}{\widehat{\mathrm{Var}}\left[Y(T)\right]}$
1.00e+05	9.77e+04	7.01e+06	8.40e+04	1.19e-02
5.00e+03	5.50e+03	4.01e+05	1.90e+04	4.75e-02
5.00e+01	7.12e+02	2.62e+04	5.17e+03	1.96e-01
5.00e+00	6.71e+02	2.30e+04	4.79e+03	2.08e-01

In Table 5.3 we show the variance reduction of $Y(T)$ by applying the control variate technique presented in section 4 depending on the initial state of $Y(0)$. We denote by $\tilde{Y}(T)$ the control variate of $Y(T)$. We can see that the variance reduction depends essentially on the number of particles in the system, which means that the error in approximating the random time using the corresponding reaction rate ODEs becomes smaller when the number of particles increases. Still, we observe that the approximation is good even in the case of a relatively small particle count. In this example, we observed that the variance reduction effect is essentially independent of the grid size.

6. Conclusions.

Given an SRN, X, with fast and slow channels, and a given suitable observable of X, in this work we addressed the problem of approximating, $\mathrm{E}[g(X(T))]$, within a given prescribed relative tolerance, $TOL{>}0$, and up to a given confidence level at near-optimal computational work. To this end, we developed an adaptive MLMC (reaction-splitting method) that extends the hybrid method presented in [23, 24] using a low-cost reaction-splitting heuristic. Our mixed method achieves the same order of computational complexity of an exact method, namely, $\mathcal{O}\left(TOL^{-2}\right)$, but with a smaller constant. We also present a new method for estimating the global exit error associated with the use of the tau-leap method. Additionally, based on Kurtz's random time change representation, we introduced a novel control variate for $g(X(T))$, which requires negligible computational cost when simulating an approximate path of X.

Acknowledgments. The authors are members of the SRI Center for Uncertainty Quantification in Computational Science and Engineering at KAUST. The authors would like to thank Prof. Mike Giles for very interesting discussions.

REFERENCES

[1] D. Anderson and D. Higham, *Multilevel Monte Carlo for continuous Markov chains, with applications in biochemical kinetics*, SIAM Multiscale Model. Simul., 10 (2012), pp. 146–179.
[2] D. F. Anderson, *A modified next reaction method for simulating chemical systems with time dependent propensities and delays*, J. Chem. Phys., 127 (2007), pp. 1–10.
[3] D. F. Anderson, *Incorporating postleap checks in tau-leaping*, J. Chem. Phys., 128 (2008).
[4] H. Andersson and T. Britton, *Stochastic Epidemic Models and Their Statistical Analysis*, Lecture Notes in Statist. 151, Springer, New York, 2000.
[5] J. P. Aparicio and H. Solari, *Population dynamics: Poisson approximation and its relation to the Langevin processs*, Phys. Rev. Lett., 86 (2001), pp. 4183–4186.
[6] C. Ben Hammouda, A. Moraes, and R. Tempone, *Multilevel Drift-Implicit Tau-Leap*, arXiv:1512.00721v1, 2015.
[7] M. Blum, R. W. Floyd, V. Pratt, R. L. Rivest, and R. E. Tarjan, *Time bounds for selection*, J. Comput. System Sci., 7 (1973) pp. 448–461.

[8] Y. Cao, D. T. Gillespie, and L. R. Petzold, *Efficient step size selection for the tau-leaping simulation method*, J. Chem. Phys., 124 (2006) 044109.

[9] Y. Cao and L. Petzold, *Accuracy limitations and the measurement of errors in the stochastic simulation of chemically reacting systems*, J. Comput. Phys., 212 (2006) pp. 6–24.

[10] M. B. Elowitz and S. Leibler, *A synthetic oscillatory network of transcriptional regulators*, Nature, 403 (2000) pp. 335–338.

[11] P. Érdi and G. Lente, *Stochastic Chemical Kinetics*, Springer, New York, 2014.

[12] S. N. Ethier and T. G. Kurtz, *Markov Processes: Characterization and Convergence*, Wiley Ser. Probab. Stat., Wiley, Hoboken, NJ, 2005.

[13] 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.

[14] M. Giles, *Multi-level Monte Carlo path simulation*, Oper. Res., 53 (2008), pp. 607–617.

[15] D. T. Gillespie, *A general method for numerically simulating the stochastic time evolution of coupled chemical reactions*, J. Comput. Phys., 22 (1976), pp. 403–434.

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

[17] J. Goutsias and G. Jenkinson, *Markovian dynamics on complex reaction networks*, Phys. Rep., 529 (2013), pp. 199–264.

[18] L. Harris and P. Clancy, *A partitioned leaping approach for multiscale modeling of chemical reaction dynamics*, J. Chem. Phys., 125 (2006).

[19] E. Haseltine and J. Rawlings, *Approximate simulation of coupled fast and slow reactions for stochastic chemical kinetics*, J. Chem. Phys., 117 (2002).

[20] S. Heinrich, *Multilevel Monte Carlo methods*, in Large-Scale Scientific Computing, Lecture Notes in Comput. Sci. 2179, Springer, Berlin, 2001, pp. 58–67.

[21] B. Hepp, A. Gupta, and M. Khammash, *Adaptive hybrid simulations for multiscale stochastic reaction networks*, J. Chem. Phys., 142 (2015).

[22] T. Li, *Analysis of explicit tau-leaping schemes for simulating chemically reacting systems*, Multiscale Model. Simul., 6 (2007), pp. 417–436.

[23] A. Moraes, R. Tempone, and P. Vilanova, *Hybrid Chernoff tau-leap*, Multiscale Model. Simul., 12 (2014) pp. 581–615.

[24] A. Moraes, R. Tempone, and P. Vilanova, *Multilevel hybrid Chernoff tau-leap*, BIT, 56 (2016), pp. 189–239.

[25] S. Plyasunov, *Averaging methods for stochastic dynamics of complex reaction networks: Description of multi-scale couplings*, arXiv:physics/0510054v1, 2005.

[26] J. Puchalka and A. Kierzek, *Bridging the gap between stochastic and deterministic regimes in the kinetic simulations of the biochemical reaction networks*, Biophys. J., 86 (2004), pp. 1357–1372.

[27] A. Speight, *A multilevel approach to control variates*, J. Comput. Finance, 12 (2009), pp. 1–25.

[28] R. Srivastava, L. You, J. Summers, and J. Yin, *Stochastic vs. deterministic modeling of intracellular viral kinetics*, J. Theoret. Biol., 218 (2002), pp. 309–321.