A MULTILEVEL ADAPTIVE REACTION-SPLITTING SIMULATION METHOD FOR STOCHASTIC REACTION NETWORKS

ALVARO MORAES*, RAUL TEMPONE†, AND PEDRO VILANOVA‡

Abstract. Stochastic modeling of reaction networks is a framework used to describe the time evolution of many natural and artificial systems, including, biochemical reactive systems at the molecular level, viral kinetics, the spread of epidemic diseases, and wireless communication networks, among many other examples. In this work, we present a novel multilevel Monte Carlo method for kinetic simulation of stochastic reaction networks that is specifically designed for systems in which the set of reaction channels can be adaptively partitioned into two subsets characterized by either "high" or "low" activity. Adaptive in this context means that the partition evolves in time according to the states visited by the stochastic paths of the system. 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 is achieved with a computational complexity of order $\mathcal{O}\left(TOL^{-2}\right)$, the same as with an exact method, but with a smaller constant. We also present a novel control variate technique based on the stochastic time change representation by Kurtz, which may dramatically reduce the variance of the coarsest level at a negligible computational cost. Our numerical examples show substantial gains with respect to the standard Stochastic Simulation Algorithm (SSA) by Gillespie and also our previous hybrid Chernoff tau-leap method.

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

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

1. Introduction.

Stochastic reaction networks (SRN) are mathematical models that employ Markovian dynamics to describe the time evolution of interacting particle systems where one particle interact with the others through a finite set of reaction channels. Typically, there is a finite number of interacting chemical species $(S_1, S_2, \ldots, S_d)$ and a stochastic process, X, such that its i-th coordinate is a non-negative integer number $X_i(t)$ that keeps track of the abundance of the i-th species at time t. Therefore, the state space of the process X is the lattice $\mathbb{Z}_+^d$.

Our main goal is to estimate the expected value $\mathrm{E}\left[g(X(T))\right]$, where X is a non-homogeneous Poisson process describing a SRN, and $g : \mathbb{R}^d \to \mathbb{R}$ is a given real observable of X at a final time T. Pathwise realizations can be simulated exactly using the Stochastic Simulation Algorithm (SSA), introduced by Gillespie in [10] (also known as Kinetic Monte Carlo among physicists, see [4] and references therein), or the Modified Next Reaction Method (MNRM) introduced by Anderson in [3], among other methods. Although these algorithms generate exact realizations of X, they may be computationally expensive for systems that undergo high activity. For that reason, Gillespie proposed in [11] the tau-leap method to approximate the SSA by evolving the process with fixed time steps while freezing the propensity functions 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

*Mathematical and Computer Sciences and Engineering Division, King Abdullah University of Science and Technology (KAUST), Thuwal, Saudi Arabia (alvaro.moraesgutierrez@kaust.edu.sa).

†Mathematical and Computer Sciences and Engineering Division, King Abdullah University of Science and Technology (KAUST), Thuwal, Saudi Arabia (raul.tempone@kaust.edu.sa).

‡Mathematical and Computer Sciences and Engineering Division, King Abdullah University of Science and Technology (KAUST), Thuwal, Saudi Arabia (pedro.guerra@kaust.edu.sa).

qualitative feature of the original process. For that reason, we proposed in [16, 17] a Chernoff-based hybrid method that switches adaptively between the tau-leap and an exact method. This allows us to control the probability of reaching negative values while keeping the computational work substantially smaller than the work of an exact method. The hybrid method developed in [16, 17] can be successfully applied to systems where the state space, $\mathbb{Z}_+^d$, can be decomposed into two regions according to the activity of the system; where all the propensities are uniformly low or uniformly high, *i.e.*, non-stiff systems. To handle stiff systems, we first measure the total activity of the system at a certain state by the total sum of the propensity functions evaluated at this state. The activity of the system is low when all the propensities are uniformly low, but a high level of activity can be the result of a high activity level in one single channel. This observation suggests that to reduce computational costs, we should adaptively split the set of reaction channels into two subsets according to the individual high and low activity levels. It is natural to evolve the system in time by applying the tau-leap method to the high activity channels and an exact method to the low activity ones. This is the main idea we develop in this work.

Reaction-splitting methods for simulating stochastic reaction networks are treated for instance in [13, 19, 14, 18], but our work is, to the best of our knowledge, the first that i) achieves the computational complexity of an exact method like the SSA by using the multilevel Monte Carlo 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) effectively 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 [13], 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 main goal is to obtain fast hybrid simulated paths, and they do not try to control the global error. 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 [7]. In their work, the tau-leap step is chosen according to the "leap condition" (as in [6]) but they do not perform a rigorous control of the global discretization error. In order to avoid negative populations, the authors reverse 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. This approach introduces bias in the estimations, and even by controlling the small reactant populations, a tau-leap step always may lead to negative populations subsequently increasing its computational work. Our Chernoff -based bound is a fast and accurate procedure to obtain the correct tau-leap step size. Finally, the method in [13] 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 [19] seems to be closest to our approach in spirit since they also explore the idea of adaptively splitting the set of reaction channels using the tau-leap method for the fast ones 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 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 manner to split the reaction channels by optimizing the computational work.

In [14] and [18], 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 of 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 the remaining of this section, we introduce the mathematical model and the path simulation techniques used in this work. In Section 2, we present an algorithm to generate mixed trajectories; that is, the algorithm generates a trajectory using an exact method for the low activity channels and the Chernoff tau-leap method for the high activity ones. Then, inspired by the ideas of Anderson and Higham [2], we propose an algorithm for coupling two mixed Chernoff tau-leap paths. This algorithm uses four building blocks that result from the combination of the MNRM and the tau-leap methods. In Section 3, we propose a mixed MLMC estimator. Next, we introduce a global error decomposition and show that the computational complexity of our method is of order $\mathcal{O}\left(TOL^{-2}\right)$. Finally, 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 present a novel control variate technique to reduce the variance of the quantity of interest at level 0. In Section 5, the numerical examples illustrate the advantages of the mixed MLMC method over the hybrid MLMC method presented in [17] and to the SSA. Finally, Section 6 presents our conclusions.

1.1. A Class of Markovian Pure Jump Processes. In this section, we describe the 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 interacting through J different reaction channels. For the sake of brevity, we write $X(t,\omega) \equiv X(t)$. Let $X_i(t)$ be the number of particles of species i in the system at time t. We study the evolution of the state vector, $X(t) = (X_1(t), \ldots, X_d(t)) \in \mathbb{Z}_+^d$, modeled as a continuous-time Markov chain starting at $X(0) \in \mathbb{Z}_+^d$. Each reaction can be described by the vector $\nu_j \in \mathbb{Z}^d$, such that, for a state vector $x \in \mathbb{Z}_+^d$, a single firing of reaction j leads to the change $x \to x + \nu_j$. The probability that reaction j will occur during the small interval $(t, t+\mathrm{d}t)$ is then assumed to be

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

for a given non-negative polynomial propensity function, $a_j : \mathbb{R}^d \to \mathbb{R}$. 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 [8]:

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

where $Y_j : \mathbb{R}_+ \times \Omega \to \mathbb{Z}_+$ are independent unit-rate Poisson processes. Hence, X is a non-homogeneous Poisson process.

1.2. The Modified Next Reaction Method (MNRM). The MNRM, introduced in [3], and based on the Next Reaction Method [9], is an 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 exact/tau-leap paths as well as nested tau-leap/tau-leap paths, as in [17, 2]. In Section 2.5, we use this feature for coupling two mixed paths.

1.3. 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 [11] 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 else than 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.*, [15]).

In the limit, when τ tends to zero, the tau-leap method gives the same solution as the exact methods [15]. 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 $s \in (t, t+\tau)$.

1.4. The Hybrid Chernoff Tau-leap Method. In [16], we derived a Chernoff-type bound that allows us to guarantee that the one-step exit probability 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, with high probability, in the next step, the approximate process, $\bar{X}$, will take a value in the lattice, $\mathbb{Z}_+^d$, of non-negative integers. 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 ν_{ji} is the i-th coordinate of the j-th reaction channel, ν_j. Finally, we let $\tau := \min\{\tau_i : i = 1, 2, \ldots, d\}$. Using the exact pre-leap method we developed in [16, 17] for single-level and multilevel hybrid schemes, allows us to switch adaptively

between the tau-leap and an exact method. By construction, the probability that one hybrid path exits the lattice, $\mathbb{Z}_+^d$, can be estimated by

$$\mathrm{P}\left(A^c\right) \leq \mathrm{E}\left[1-(1-\delta)^{N_{\mathrm{TL}}}\right] = \delta \mathrm{E}\left[N_{\mathrm{TL}}\right] - \frac{\delta^2}{2}\left(\mathrm{E}\left[N_{\mathrm{TL}}^2\right] - \mathrm{E}\left[N_{\mathrm{TL}}\right]\right) + o(\delta^2),$$

where $\bar{\omega} \in A$ if and only if the whole hybrid path, $(\bar{X}(t_k,\bar{\omega}))_{k=0}^{K(\bar{\omega})}$, belongs to the lattice, $\mathbb{Z}_+^d$, $\delta{>}0$ is the one-step exit probability bound, and $N_{\mathrm{TL}}(\bar{\omega}){\equiv}N_{\mathrm{TL}}$ is the number of tau-leap steps in a hybrid path. Here, A^c is the complement of the set A.

To simulate a hybrid path, given the current state of the approximate process, $\bar{X}(t)$, we adaptively determine whether to use an exact or the tau-leap method for the next step. This decision is based on the relative computational cost of taking an exact step versus the cost of taking a Chernoff tau-leap step. Instead, in the present work, at each time step, we adaptively determine which reactions are suitable for using the exact method and which reactions are suitable for the Chernoff tau-leap method.

2. Generating Mixed Paths. In this section we explain how mixed paths are generated. First, we present the splitting heuristic; that is, we discuss how to partition the set of reaction channels at each decision time. Then, we present the one-step mixing rule, which is the main building block for constructing a mixed path. Finally, we show how to couple two mixed paths.

2.1. The Splitting Heuristic. In this section, we explain how we partition the set of reaction channels, $\mathcal{R}{:=}\{1,...,J\}$, 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 H be the next decision (or synchronization) time (given by the Chernoff tau-leap step size $\tau_{Ch} = \tau_{Ch}(x,\delta)$ and the time mesh). 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 H, starting at t, is minimal for all possible splittings.

The idea goes 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 j-th reaction if its activity is high. This linear order determines $J{+}1$ possible splittings, out of 2^J. In order to measure the activity, it turns out that using only the propensity functions evaluated at x, that is, $a_j(x)$, is not enough. This is because the j-th reaction could affect components of x with small values. If this is the case, this determines small Chernoff tau-leap step sizes. In order to avoid this scenario, we penalize the j-th reaction channel if it has a high exit probability. We approximate this exit probability using a Poisson distribution for each dimension of x. For example, let ν_{ji} be the i-th component of the j-th reaction channel. If $\nu_{ji} < 0$, then the probability that a Poisson distributed random variable with rate $a_j(x)(H{-}t)$ is greater than x_i/ν_{ji} measures how likely species x_i can become negative in the interval $H{-}t$, independently of reactions $j'{\in}\mathcal{R}$, $j{\neq}j'$. Let $I_j{:=}\{i : \nu_{ji} < 0\}$,

$$\theta_j := \begin{cases} \mathrm{P}\left(\mathcal{P}(a_j(x)(H{-}t)) > \min_{i\in I_j}\{-\frac{x_i}{\nu_{ji}}\} \,\middle|\, x\right) & \text{if } I_j \neq \emptyset \\ 0 & \text{otherwise} \end{cases}. \tag{2.1}$$

Then, the penalty weight for $a_j(x)$ is $1{-}\theta_j$. We define $\tilde{a}_j(x){:=}(1{-}\theta_j)a_j(x)$. The linear order is then a permutation, σ, over $\mathcal{R}$ such that

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

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

tau-leap for the reactions $\mathcal{R}_{TL}$ and the MNRM for the reactions $\mathcal{R}_{\text{MNRM}}$. The work corresponding to $\mathcal{R}_{\text{TL}}$ is

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

where C_s is the work of computing the split (see Section 2.2), and $C_P(\lambda)$ is the work of a Poisson random variate with rate λ. The factor $\frac{H-t}{\min\{\tau_{Ch}, H-t\}}$ takes into account the number of steps required to reach $H = H(t)$ from t. For the Gamma simulation method developed by Ahrens and Dieter in [1], which is the one used by MATLAB, C_P is defined as

$$C_P(\lambda) := \begin{cases} b_1 + b_2 \ln \lambda & \text{for } \lambda > 15 \\ b_3 + b_4 \lambda & \text{for } \lambda \leq 15 \end{cases}.$$

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 [16].

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

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

where the constant C_{MNRM} is the work of an MNRM step and $\tau_{\text{MNRM}} = \left(\sum_{j \in \mathcal{R}_{\text{MNRM}}} a_j(x) \right)^{-1}$.

2.2. On the Work required to the Splitting Heuristic, C_s=$C_s(J)$. The work required to perform the splitting includes the work required to determine Work($\mathcal{R}_{\text{TL}}$) and Work($\mathcal{R}_{\text{MNRM}}$), both defined in Section 2.1. The linear order previously defined determines J+1 possible splittings, $\mathcal{S}_i$, i=0, ..., J, as follows:

	$\mathcal{R}_{\text{TL}}$	$\mathcal{R}_{\text{MNRM}}$
$\mathcal{S}_0$	$\emptyset$	$\mathcal{R}$
$\mathcal{S}_1$	$\{\sigma^{-1}(1)\}$	$\{\sigma^{-1}(2), .., \sigma^{-1}(J)\}$
$\mathcal{S}_2$	$\{\sigma^{-1}(1), \sigma^{-1}(2)\}$	$\{\sigma^{-1}(3), .., \sigma^{-1}(J)\}$
$\vdots$		
$\mathcal{S}_J$	$\mathcal{R}$	$\emptyset$

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 [16], the work of computing a single τ_{Ch} is linear on J. Then, in order to avoid J^2 complexity of the splitting rule, we implement a local search instead of computing J τ_{Ch}'s, to keep the complexity of C_s linear on J. The main idea is to keep track of the last split at each decision time, assuming that the propensities do not vary widely between. If that is the case, we can just evaluate the previous split, $\mathcal{S}_\kappa$, and its neighbors, κ−1 and κ+1. 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.

REMARK 2.1 (Pareto Splitting rule). *Instead of computing a cost-based splitting at each decision time, the following rule can be applied:*

$$\mathcal{R}_{TL} \text{ is defined s.t. } \frac{\sum_{j \in \mathcal{R}_{TL}} \tilde{a}_{\sigma(j)}}{\sum_{k=1}^{J} \tilde{a}_k} \geq \nu,$$

where ν is a problem-dependent threshold, which can be estimated using the cost-based splitting rule. The idea is to use the tau-leap method for a $(100 \times \nu)\%$ of the penalized activity (measured as before using the $\tilde{a}_j$'s), and an exact method for the other channels.

This rule is adaptive because it depends on the current state of the process, but it does not take into account the computational cost of the resulting partition of $\mathcal{R}$. The advantage of this rule is that it is three times faster than the previous one. For the examples we worked on, the overall average gain in terms of computational work in a whole mixed path is about 45% of the total work.

2.3. The one-step Mixing Rule. In this section we present the main building block for simulating a mixed path. Let $x=\bar{X}(t)$ be the current state of the approximate process, $\bar{X}$. Therefore, the expected time step of the MNRM is given by $1/a_0(x)$. To move one step forward using the MNRM, we should compute at least $a_0(x)$ and sample a uniform random variable. On the other hand, to move one step forward using the mixed Chernoff tau-leap method, we need first to compute the split, then compute the tau-leap increments for the reactions in the tau-leap set, $\mathcal{R}_{\rm TL}$, and finally compute the MNRM steps for the reactions in the set $\mathcal{R}_{\rm MNRM}$, as discussed in Section 2.2.

To avoid the overhead caused by unnecessary computation of the split, we first estimate the computational work of moving forward from the current time, t, to the next grid 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. In order to compare the mentioned

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 grid point, $\tilde{T}$, and the previous optimal split, κ. Outputs: the tau-leap set, $\mathcal{R}_{\rm TL}$, the exact set, $\mathcal{R}_{\rm MNRM}$, and the new optimal split κ.

Require: $a_0 \leftarrow \sum_{j=1}^J a_j > 0$
1: **if** $K_1/a_0 < \tilde{T} - t$ **then**
2: Compute θ_j, j=1, .., J (see (2.1))
3: $\tilde{a}_{\sigma(j)} \leftarrow$ Sort$\{(1-\theta_j)a_j\}$ descending, j=1, .., J
4: $\mathcal{S}_i \leftarrow$ Compute the splits taking into account the previous optimal split κ
5: $(\mathcal{R}_{\rm TL}, \mathcal{R}_{\rm MNRM}, \kappa) \leftarrow$ Take the minimum work split
6: **return** $(\mathcal{R}_{\rm TL}, \mathcal{R}_{\rm MNRM}, \kappa)$
7: **else**
8: **return** $(\emptyset, \mathcal{R}, \kappa)$
9: **end if**

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.

REMARK 2.2 (Comparison with the one-step hybrid rule). *In [16] 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 $\mathcal{R}_{TL} = \emptyset$ and $\mathcal{R}_{MNRM} = \mathcal{R}$ or $\mathcal{R}_{TL} = \mathcal{R}$ and $\mathcal{R}_{MNRM} = \emptyset$. Then, the mixed 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 2.2, in the mixed method is on the order of three times the computation of the Chernoff 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.

2.4. The Mixed-Path Algorithm. In this section, we present a novel algorithm (Algorithm 2) that combines the approximate Chernoff tau-leap method and the exact MNRM to generate a whole hybrid path. This algorithm combines the advantages of an exact method (expensive but exact) and the tau-leap method (may be cheaper but has a discretization error and a positive probability of exiting the lattice). This algorithm automatically and adaptively partitions the reactions into two subsets, $\mathcal{R}_{\rm TL}$ and $\mathcal{R}_{\rm MNRM}$, using a computational work criterion. Since a mixed 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 $\mathcal{R}_{\rm TL}$ is empty. The idea of this algorithm is to apply, at each decision point, the one-step mixing rule (Algorithm 1) to determine the sets $\mathcal{R}_{\rm TL}$ and $\mathcal{R}_{\rm MNRM}$, and then to apply the corresponding method.

2.5. Coupled Mixed Paths. In this section, we explain how to couple two mixed paths. This is essential for the multilevel estimator. The four algorithms that are the building blocks of the coupling algorithm were already presented in [17]. The novelty here comes from the fact that the coupled mixed 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.

We now briefly describe the mixed Chernoff coupling algorithm, *i.e.*, Algorithm 3. Let $\bar{X}$ and $\bar{\bar{X}}$ be two mixed paths, corresponding to two nested time discretizations, called coarse and fine, respectively. Assume that the current time is t, and 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 $(\bar{\mathcal{R}}_{\rm TL}, \bar{\mathcal{R}}_{\rm MNRM}, \bar{\bar{\mathcal{R}}}_{\rm TL}, \bar{\bar{\mathcal{R}}}_{\rm MNRM})$, that correspond to four algorithms, B1, B2, B3 and B4, that we use as building blocks. Table 2.1 summarizes them. In order

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

TABLE 2.1

Building blocks for simulating two coupled mixed Chernoff tau-leap paths. Algorithms B1 and B2 are presented as Algorithms 2 and 3 in [2]. Algorithms B3 and B4 can be directly obtained from Algorithm B2 (see [17]).

to do that, the algorithm computes, independently, the sets $\mathcal{R}_{\rm TL}$ and $\mathcal{R}_{\rm MNRM}$ 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 variates 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. The only difference in the latter blocks are the time points 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

Algorithm 2 The mixed-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, $(\bar{X}(t_k))_{k=0}^K$, and the number of times, $N_{\rm TL}$, that the tau-leap method was successfully applied (*i.e.*, $\bar{X}(t_k) \in \mathbb{Z}_+^d$, we applied the tau-leap method and we obtained an $\bar{X}(t_{k+1}) \in \mathbb{Z}_+^d$). Notes: given the current state, $next_{\rm MNRM}$ computes the next state using the MNRM method. Here, t_i denotes the current time at the i-th step, and $\tau_{Ch}(\mathcal{R}_{\rm TL})$ is the Chernoff step size associated with $\mathcal{R}_{\rm TL}$.

$i \leftarrow 0$, $t_i \leftarrow t_0, \bar{X}(t_i) \leftarrow X(0), \bar{Z} \leftarrow X(0)$
$\mathcal{S}_j \leftarrow$ Compute splits, j=0, ..., J
$\kappa \leftarrow \arg\min_j \text{Work}(\mathcal{S}_j)$
while $t_i < T$ **do**
  $\tilde{T} \leftarrow$ next grid point greater than t_i
  $(\mathcal{R}_{\rm TL}, \mathcal{R}_{\rm MNRM}, \kappa) \leftarrow$ Algorithm 1 with $(\bar{Z}, t_i, (a_j(\bar{Z}))_{j=1}^J, \delta, \tilde{T}, \kappa)$
  if $\mathcal{R}_{\rm TL} \neq \emptyset$ **then**
    $\Delta_{\rm TL} \leftarrow \mathcal{P}(a_j(\bar{Z})\tau_{Ch}(\mathcal{R}_{\rm TL}))\nu_j$, for $j \in \mathcal{R}_{\rm TL}$
    $H \leftarrow t_i + \tau_{Ch}(\mathcal{R}_{\rm TL})$
  else
    $H \leftarrow \min\{t_i - \log(r)/\sum_j a_j, T\}$, $r \sim$ Unif$(0,1)$
  end if
  if $\mathcal{R}_{\rm MNRM} \neq \emptyset$ **then**
    while $t_i < H$ **do**
      $(\bar{Z}, t_i) \leftarrow next_{\rm MNRM}(\bar{Z}, \mathcal{R}_e, t_i, H)$
    end while
  end if
  $\bar{Z} \leftarrow \bar{Z} + \Delta_{\rm TL}$
  if $\bar{Z} \in \mathbb{Z}_+^d$ **then**
    $N_{\rm TL} \leftarrow N_{\rm TL} + 1$
    $t_{i+1} \leftarrow H$
  else
    return $((\bar{X}(t_k))_{k=0}^i, N_{\rm TL})$
  end if
  $i \leftarrow i + 1$
  $\bar{X}(t_i) \leftarrow \bar{Z}$
end while
return $((\bar{X}(t_k))_{k=0}^i, N_{\rm TL})$

four sets $(\bar{\mathcal{R}}_{\rm TL}, \bar{\mathcal{R}}_{\rm MNRM}, \bar{\bar{\mathcal{R}}}_{\rm TL}, \bar{\bar{\mathcal{R}}}_{\rm MNRM})$ 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 Total Error Decomposition. In this section, we first show the multilevel Monte Carlo 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 one 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, up to a given confidence level.

3.1. The MLMC Estimator. In this section, we discuss and implement a multilevel Monte Carlo estimator for the mixed Chernoff tau-leap case. 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 the 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. Let $\{\bar{X}_\ell(t)\}_{t\in[0,T]}$ be a mixed Chernoff tau-leap process with a time mesh of size Δt_ℓ and a one-step exit probability bound δ, and let $g_\ell := g(\bar{X}_\ell(T))$ be our quantity of interest computed with a mesh of size Δt_ℓ. We can simulate paths of $\{\bar{X}_\ell(t)\}_{t\in[0,T]}$ by using Algorithm 2. We are interested in estimating $\mathrm{E}[g_L]$, and we can simulate correlated pairs, $(g_\ell, g_{\ell-1})$ for $\ell = 1, \ldots, L$, by using Algorithm 3. Let A_ℓ be the event in which the ℓ-th grid level path, $\bar{X}_\ell$, arrives at the final time, T, without exiting the state space of X.

Consider the following telescopic decomposition:

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

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

$$\mathcal{M}_L := \frac{1}{M_0}\sum_{m=1}^{M_0} g_0 \mathbf{1}_{A_0}(\omega_{m,0}) + \sum_{\ell=1}^{L}\frac{1}{M_\ell}\sum_{m=1}^{M_\ell}[g_\ell \mathbf{1}_{A_\ell} - g_{\ell-1}\mathbf{1}_{A_{\ell-1}}](\omega_{m,\ell}). \tag{3.1}$$

Computational Complexity. A key property of our multilevel estimator is that the computational work is a function of the given relative tolerance, TOL, is of the order of TOL^{-2}. The optimal work is is given by

$$w_L^*(TOL) = \left(\frac{C_A}{\theta}\sum_{\ell=0}^{L}\sqrt{\mathcal{V}_\ell \psi_\ell}\right)^2 TOL^{-2}.$$

From the fact that the sum $\sum_{\ell=0}^{\infty}\sqrt{\mathcal{V}_\ell\psi_\ell}$ converges, because $\psi_\ell = \mathcal{O}(\psi_{\mathrm{MNRM}})$, we conclude that $\sup_L\{\sum_{\ell=0}^{L}\sqrt{\mathcal{V}_\ell\psi_\ell}\}$ is bounded and, therefore, the expected computational complexity of the multilevel mixed Chernoff tau-leap method is $w_L^*(TOL)=\mathcal{O}\left(TOL^{-2}\right)$.

3.2. Global Error Decomposition. In this section, we define the computational global error, $\mathcal{E}_L$, and show how it 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 mixed 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 as

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

We showed in [16] that by choosing adequately the one-step exit probability bound, δ, the exit error, $\mathcal{E}_{E,L}$, satisfies $|\mathcal{E}_{E,L}| \leq |\mathrm{E}[g(X(T))]\,|\,\mathrm{P}(A_L^c) \leq TOL^2$.

An efficient procedure for accurately estimating $\mathcal{E}_{I,L}$ in the context of the tau-leap method is described in [17]. For each mixed path, $(\bar{X}_\ell(t_{n,\ell},\bar{\omega}))_{n=0}^{N(\bar{\omega})}$, we define the sequence of dual weights, $(\varphi_{n,\ell}(\bar{\omega}))_{n=1}^{N(\bar{\omega})}$, backwards as follows:

$$\begin{aligned}(3.2)\qquad \varphi_{N(\bar{\omega}),\ell} &:= \nabla g(\bar{X}_\ell(t_{N(\bar{\omega}),\ell},\bar{\omega}))\\ \varphi_{n,\ell} &:= \left(Id + \Delta t_{n,\ell}\mathbb{J}_a^T(\bar{X}_\ell(t_{n,\ell},\bar{\omega}))\,\nu^T\right)\varphi_{n+1,\ell},\quad n = N(\bar{\omega})-1,\ldots,1,\end{aligned}$$

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 $j{=}1\ldots J$ and $i{=}1\ldots d$. We then approximate $\mathcal{E}_{I,L}$ by $\mathcal{A}\left(\mathcal{E}_{I,L}(\bar{\omega});\cdot\right)$, where

$$\mathcal{E}_{I,L}(\bar{\omega}) := \sum_{n=1}^{N(\bar{\omega})}\left(\frac{\Delta t_{n,L}}{2}\varphi_{n,L}\sum_{j=1}^{J}\mathbf{1}_{j\in\mathcal{R}_{\mathrm{TL}}(n)}\nu_j^T\left(a_j(\bar{X}_L(t_{n+1,\ell})){-}a_j(\bar{X}_L(t_{n,\ell}))\right)\right)(\bar{\omega}),$$

$\mathcal{A}\left(X;M\right){:=}\frac{1}{M}\sum_{m=1}^{M}X(\omega_m)$ 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. 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, and 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 $\mathcal{V}_\ell := \mathrm{Var}\left[g_\ell\mathbf{1}_{A_\ell}-g_{\ell-1}\mathbf{1}_{A_{\ell-1}}\right]$, $\ell\geq 1$. In [17], we presented an efficient and accurate method for estimating $\mathcal{V}_\ell$, $\ell\geq 1$ using the formula

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

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 $\mathrm{E}\left[\varphi_{n+1}\cdot e_{n+1}\,\middle|\,\mathcal{F}\right]$ and $\mathrm{Var}\left[\varphi_{n+1}\cdot e_{n+1}\,\middle|\,\mathcal{F}\right]$ comes from the local errors, $(e_n)_{n=1}^{N(\bar{\omega})}$, defined as $e_n := X_{\ell,n}-X_{\ell-1,n}$. In the aforementioned work, we derived exact and approximate formulas for computing $\mathrm{E}\left[\varphi_{n+1}\cdot e_{n+1}\,\middle|\,\mathcal{F}\right]$ and $\mathrm{Var}\left[\varphi_{n+1}\cdot e_{n+1}\,\middle|\,\mathcal{F}\right]$.

Remark 3.1 (Backward Euler). *In (3.2), we have that $\varphi_{n,\ell}$ can be computed by a backward Euler formula when too fine time meshes are required for stability,* i.e.*,* $\varphi_{n,\ell} := \left(Id-\Delta t_{n,\ell}\mathbb{J}_a^T(\bar{X}_\ell(t_{n,\ell},\bar{\omega}))\,\nu^T\right)^{-1}\varphi_{n+1,\ell}$.

3.3. Estimation Procedure. In this section, we briefly describe the automatic procedure that estimates $\mathrm{E}\left[g(X(T))\right]$ 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 [17]. It is important to remark that the minimal user intervention is required to obtain the parameters needed to simulate the mixed paths, and subsequently, to compute the estimations using (3.1). 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

$$\begin{cases} \min_{\{\Delta t_0, L, (M_\ell,\delta_\ell)_{\ell=0}^L\}} \sum_{\ell=0}^L \psi_\ell M_\ell \\ \text{s.t.} \\ \mathcal{E}_{E,L} + \mathcal{E}_{I,L} + \mathcal{E}_{S,L} \leq TOL \end{cases}. \tag{3.3}$$

An algorithm to efficiently compute the solution of this optimization problem is given in [17]. 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 hybrid path (simulated by Algorithm 2),

$$\begin{aligned} \psi_0 := C_{\text{MNRM}} \mathrm{E}\left[N_{\text{MNRM}}(\Delta t_0, \delta_0)\right] + C_{\text{TL}} \mathrm{E}\left[N_{\text{TL}}(\Delta t_0, \delta_0)\right] \\ + \int_{[0,T]} \mathrm{E}\left[\sum_{j \in \mathcal{R}_{\text{TL}}(s)} C_P(a_j(\bar{X}_0(s)) \tau_{Ch}(\bar{X}_0(s), \delta_0)) ds\right], \end{aligned} \tag{3.4}$$

where Δt_0 is the size of the time mesh at level 0 and δ_0 is the exit probability bound at level 0, and $\mathcal{R}_{\text{TL}} = \mathcal{R}_{\text{TL}}(t)$ is the tau-leap set, which depends on time (and also the current state of the process). The set $\mathcal{R}_{\text{TL}}$ 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 hybrid 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 hybrid paths at levels ℓ and $\ell-1$ is

$$\begin{aligned} \psi_\ell := C_{\text{MNRM}} \mathrm{E}\left[N^{(c)}_{\text{MNRM}}(\ell)\right] + C_{\text{TL}} \mathrm{E}\left[N^{(c)}_{\text{TL}}(\ell)\right] \\ + \int_{[0,T]} \mathrm{E}\left[\sum_{j \in \mathcal{R}_{\text{TL},\ell}(s)} C_P(a_j(\bar{X}_\ell(s)) \tau_{Ch}(\bar{X}_\ell(s), \delta_\ell)) ds\right] \\ + \int_{[0,T]} \mathrm{E}\left[\sum_{j \in \mathcal{R}_{\text{TL},\ell-1}(s)} C_P(a_j(\bar{X}_{\ell-1}(s)) \tau_{Ch}(\bar{X}_{\ell-1}(s), \delta_{\ell-1})) ds\right], \end{aligned} \tag{3.5}$$

where

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

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

4. A 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 the independent Poisson processes, $(Y_j(s,\omega))_{s\geq 0}$, by the identity function. This defines the deterministic mean field,

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

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 $X(T,\omega)$. In this way, we expect some correlation between $X(T)$ and $\tilde{X}(T)$. Since $\mathrm{E}\left[\tilde{X}(T)\right] = Z(T)$ is a computable quantity, 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 allow 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 $\hat{\Lambda}_{j,K}$ 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

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

Now, we consider the random sequence, $\{\bar{X}_n(\omega)\}_{n=0}^{N(\omega)}$, generated in this case by the mixed algorithm. Here, $\bar{X}(\omega)_{N(\omega)}$ is an approximation of $X(T,\omega)$. The sequence of mixed 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.

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

$$Y_j(\hat{\Lambda}_{j,K}) \sim Y_j\left(\bar{\Lambda}_{j,n}\right) + \text{binomial}\left(Y_j\left(\bar{\Lambda}_{j,n+1}\right) - Y_j\left(\bar{\Lambda}_{j,n}\right), \frac{\hat{\Lambda}_{j,K} - \bar{\Lambda}_{j,n}}{\bar{\Lambda}_{j,n+1} - \bar{\Lambda}_{j,n}}\right).$$

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

$$Y_j(\hat{\Lambda}_{j,K}) \sim Y_j\left(\bar{\Lambda}_{j,N}\right) + \text{Poisson}(a_j(\bar{X}_N)(\hat{\Lambda}_{j,K} - \bar{\Lambda}_{j,N})).$$

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

$$\frac{1}{M}\sum_{m=1}^{M} g(\bar{X}_N(\omega_m)) - \beta\frac{1}{M}\sum_{m=1}^{M}(g(\hat{X}_K(\omega_m)) - g(\mu_K)),$$

for any linear functional, g. For polynomial observables, g, this estimator can be easily extended by Taylor expansions in such way that we can estimate $\mathrm{E}\left[g(\tilde{X}(T))\right]$ by powers, $g\left(\left(\mathrm{E}\left[\tilde{X}(T)\right]\right)^k\right)$.

REMARK 4.1 (Reducing the variance at the coarsest level). *The main application of the deterministic time change control variate, $\tilde{X}(T)$, in this work is at the coarsest level of our multilevel hierarchy. Consider the trivial decomposition*

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

Therefore,

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

Since we can compute exactly $\mathrm{E}\left[g(\tilde{X}(T))\right]$*, we just have to estimate* $\mathrm{E}\left[g(\bar{X}_0(T)) - g(\tilde{X}(T))\right]$ *instead of* $\mathrm{E}\left[g(\bar{X}_0(T))\right]$ *in our multilevel scheme. The computational gain lies in the fact that* $\mathrm{Var}\left[g(\bar{X}_0(T)) - g(\tilde{X}(T))\right]$ *could be substantially lower than* $\mathrm{Var}\left[g(\bar{X}_0(T))\right]$.

REMARK 4.2 (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 mixed algorithm. Also, for each realization of the control variate, at most one Poisson random variate is needed for each reaction channel.*

REMARK 4.3 (Empirical Time Change). *We can also compute the final times, $\hat{\Lambda}_{j,K}$, using a sample average of mixed paths instead of the mean field. We found no significant improvements when using that approach, which requires a lot more computational work. We conjecture that, for settings in which the mean field is not representative, this approach is the only reasonable option.*

5. Numerical Examples. In this section, we present two examples to illustrate the performance of our proposed method, and we compare the results with the hybrid MLMC approach given in [17]. For benchmarking purposes, we use Gillespie's Stochastic Simulation Algorithm (SSA) instead of the Modificed Next Reaction Method (MNRM) because the former is widely used in the literature.

Intracellular Virus Kinetics. This model, first developed in [20], 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,
- $G+S \xrightarrow{7.5\times10^{-6}} V$, the virus (V) is produced,
- $E \xrightarrow{0.25} \emptyset$, $S \xrightarrow{2} \emptyset$ degradation reactions.

Its stoichiometric matrix and its propensity functions, $a_j : \mathbb{Z}_+ \to \mathbb{R}$, are given by

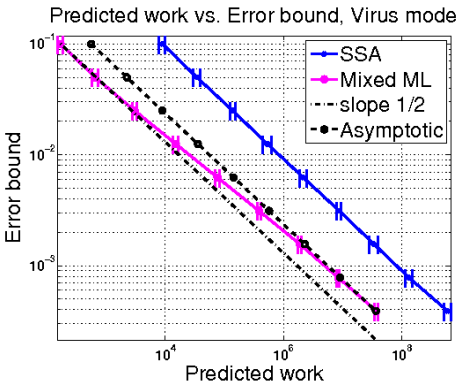
$$\nu = \begin{pmatrix} 1 & 0 & 0 & 0 \\ -1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ -1 & -1 & 0 & 1 \\ 0 & 0 & -1 & 0 \\ 0 & -1 & 0 & 0 \end{pmatrix}^{tr} \quad \text{and } a(X) = \begin{pmatrix} E \\ 0.025\, G \\ 1000\, E \\ 7.5\times10^{-6} G\, S \\ 0.25\, E \\ 2\, S \end{pmatrix},$$

respectively.

In this model, $X(t) = (G(t), S(t), E(t), V(t))$, and $g(X(t)) = V(t)$, the number of viruses produced. The initial condition is X_0=(0, 0, 10, 0) and the final time is T=20. This example is interesting because i) it shows a clear separation of time scales, ii) our previous hybrid Chernoff method has no compuational work gain with respect to an exact method, and iii) in [2] the authors take an alternative approach, not using the multilevel aspect of their paper.

We now analyze an ensemble of 10 independent runs of the phase II algorithm (see Section 3.3), 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, versus 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 is the same as the work of the SSA.

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, $\hat{\mathcal{V}}_\ell$,



TOL	L^*	$\frac{\hat{W}_{ML}}{\hat{W}_{\mathrm{SSA}}}$
1.00e-01	1.0	0.02 $\pm$0.001
5.00e-02	1.0	0.02 $\pm$0.001
2.50e-02	1.2 $\pm$0.261	0.02 $\pm$0.001
1.25e-02	2.2 $\pm$0.261	0.03 $\pm$0.002
6.25e-03	3.4 $\pm$0.320	0.04 $\pm$0.004
3.13e-03	4.6 $\pm$0.320	0.04 $\pm$0.002
1.56e-03	5.8 $\pm$0.261	0.06 $\pm$0.008
7.81e-04	7.4 $\pm$0.433	0.07 $\pm$0.006
3.91e-04	8.6 $\pm$0.320	0.06 $\pm$0.007

FIG. 5.1. *Left: Predicted work (runtime) versus the estimated error bound, with 95% confidence intervals. The multilevel mixed method is preferred over the SSA and the multilevel hybrid method for all the tolerances. Right: Details for the ensemble run of the phase II algorithm. Here,* $\hat{W}_{ML} = \sum_{\ell=0}^{L^*} \hat{\psi}_\ell M_\ell$ *and* $\hat{W}_{SSA} = M_{SSA}\, C_{SSA}\, \mathcal{A}(N_{SSA*};\cdot)$. *As an example, the fourth row of the table tells us that, for a tolerance* TOL=$1.25 \cdot 10^{-2}$, *2.2 levels are needed on average. The work of the multilevel hybrid method is, on average, 3% of the work of the SSA and the multilevel hybrid method. Confidence intervals at 95% are also provided.*

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, as expected. 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. Observe that it increases linearly for the coarser grids, until it reaches a plateau, which corresponds to the pure MNRM case where the computational cost is independent of the grid size. In the lower right panel, we show the total computational work, only for the cases in which $\hat{\mathcal{E}}_{I,\ell} < TOL - TOL^2$.

In Figure 5.4, we show the main outputs of the phase II algorithm, δ_ℓ and M_ℓ for $\ell = 0, ..., L^*$, for the smallest considered tolerance. In this example, L^* is 8 or 9, depending on the run. We observe that the number of realizations decreases slower than linearly, from levels 1 to L^*-1, until it drops, due to the change to a MNRM dominated regime.

In Figure 5.5, 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 5.5 was obtained as follows: i) for the range of tolerances specified in the first column of Table 5, we ran the phase II algorithm 5 times, ii) for each output of the calibration algorithm, we sampled the multilevel estimator $\mathcal{M}_L$, defined in 3.1, 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 (Pareto rule). *Using the cost-based rule (see remark 2.1), we estimate the threshold for the Pareto rule, obtaining* $\nu = 0.95419$. *It turns out that, for this example,* $\hat{W}_{MixPareto}/\hat{W}_{Mix}$ *ranges from* 0.6 *to* 0.75 *(for most* TOL*s). This shows that it is possible to increase the computational work gains further in some*

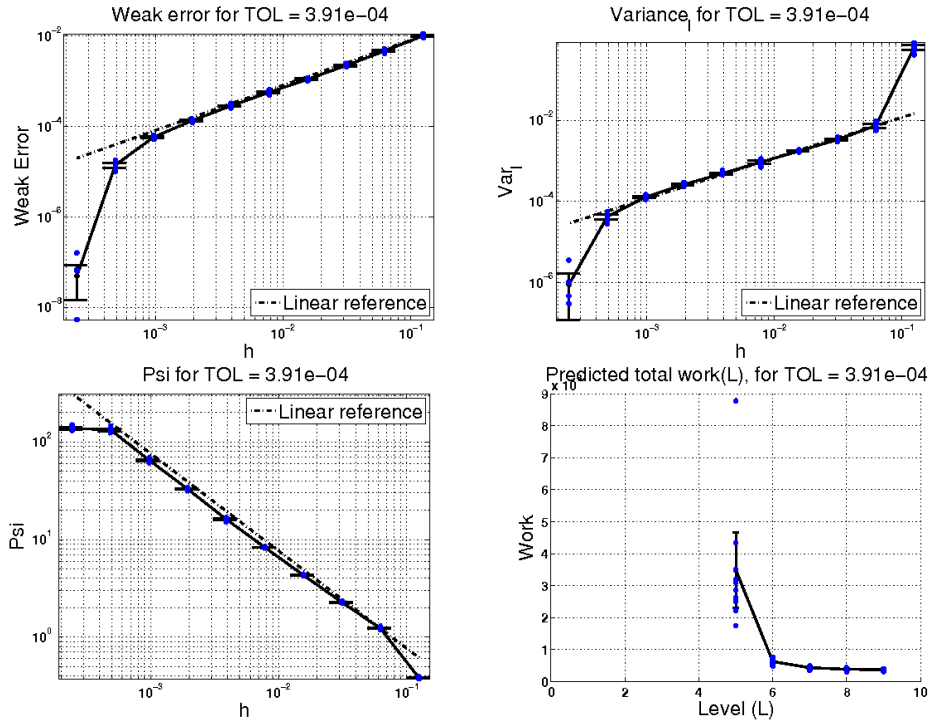


FIG. 5.2. *Upper left: estimated weak error, $\hat{\mathcal{E}}_{I,\ell}$, as a function of the time mesh size, h. Upper right: estimated variance of the difference between two consecutive levels, $\hat{\mathcal{V}}_\ell$, as a function of h. Lower left: estimated path work, $\hat{\psi}_\ell$, as a function of h. Lower right: estimated total computational work, $\sum_{l=0}^{L} \hat{\psi}_l M_l$, as a function of the level, L.*

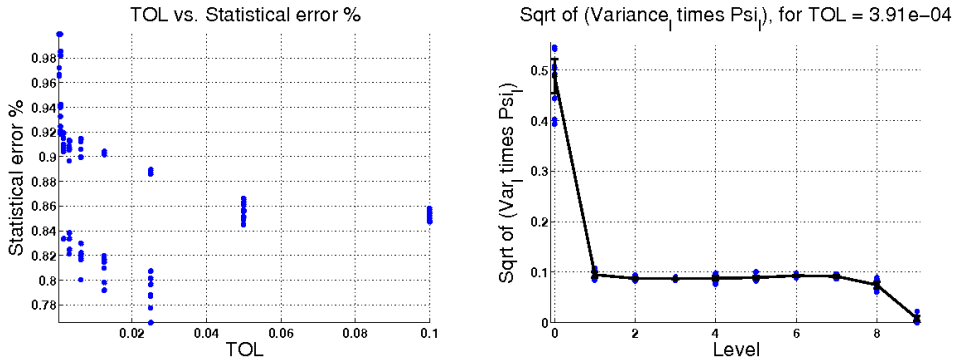


FIG. 5.3. *Left: Percentage of the statistical error over the total error. As we mentioned in Section 3.1, it is well above 0.5 for all the tolerances. Right: $\sqrt{\hat{\mathcal{V}}_\ell \hat{\psi}_\ell}$, as a function of ℓ, for the smallest tolerance, which decreases as the level increases. Observe that the contribution of level 0 is less than 50% of the sum of the other levels.*

examples.

REMARK 5.3. *The savings in computational work when generating Poisson random variables heavily depend on MATLAB's performance capabilities. In fact, we would expect better results from our method if we were to implement our algorithms in more performance-oriented languages or if we were to sample Poisson random*

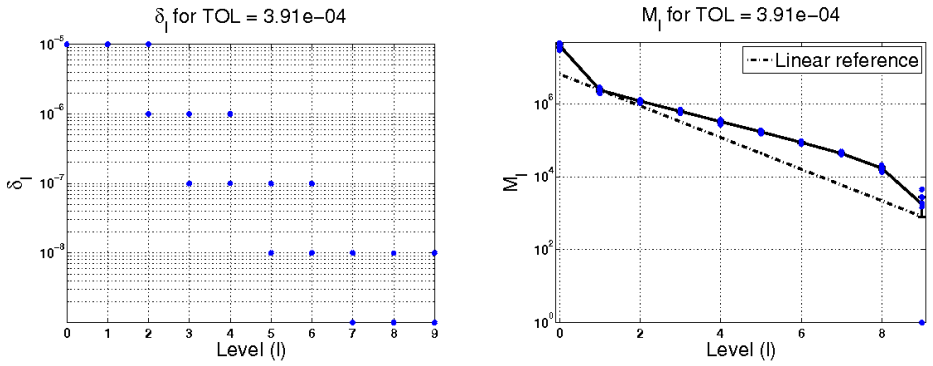


FIG. 5.4. *The one-step exit probability bound, δ_ℓ, and M_ℓ for ℓ=0, 1, ..., L^*, for the smallest tolerance.*

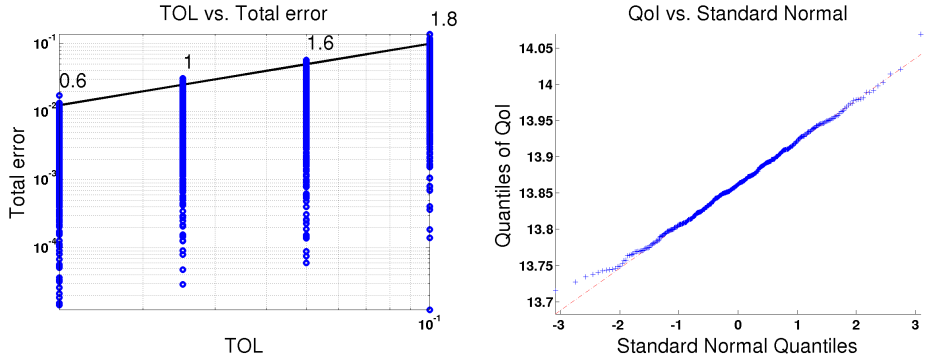


FIG. 5.5. *Left: TOL versus the actual computational error. The numbers above the straight line show the percentage of runs that had errors larger than the required tolerance. We observe that in all cases, the computational error follows the imposed tolerance with the expected confidence of 95%. Right: quantile-quantile plot based on realizations of $\mathcal{M}_L$.*

variables in batches.

A Simple Stiff System. This model, adapted from [5], has three species and a mixture of fast and slow reaction channels,

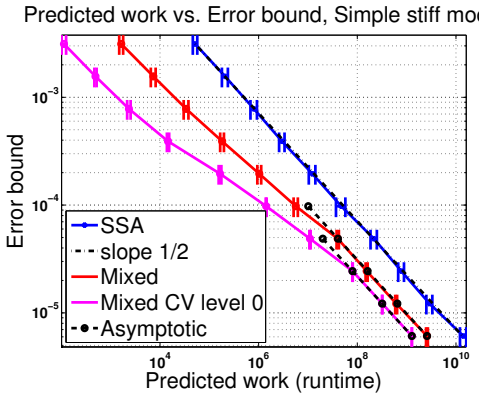
$$X_1 \underset{c_2}{\overset{c_1}{\rightleftharpoons}} X_2 \xrightarrow{c_3} X_3 \xrightarrow{c_4} \emptyset, \quad c_2 \gg c_3 > c_4.$$

Its stoichiometric matrix and propensity functions, $a_j : \mathbb{Z}_+ \to \mathbb{R}$, are given by

$$\nu = \begin{pmatrix} -1 & 1 & 0 \\ 1 & -1 & 0 \\ 0 & -1 & 1 \\ 0 & 0 & -1 \end{pmatrix}^{tr} \quad \text{and } a(X) = \begin{pmatrix} c_1 X_1 \\ c_2 X_2 \\ c_3 X_2 \\ c_4 X_3 \end{pmatrix},$$

respectively, where $g(X(t)) = X_3(t)$. In this model, successive firings of the reaction $X_2 \to X_3$ are separated by many reversible firings between X_1 and X_2, which takes a lot of computational work in a standard SSA run. In [12], Gillespie et al. claim that this inefficiency cannot be addressed using ordinary tau-leaping because of the stiffness of the system. We show here that we have substantial gains using our mixed method,

which also controls the global error. In this example, we also show the performance of the control variate idea, presented in Section 4. We analyze 10 independent runs of the phase II algorithm (see Section 3.3), using different relative tolerances. In Figure 5.6, we show the total predicted work (runtime) for the multilevel mixed method with and without a control variate at level 0 and for the SSA method versus the estimated error bound. We also show the estimated asymptotic work of the multilevel mixed method. Observe that, for practical tolerances, the computational work gains with respect to the SSA method, when using the control variate, are of a factor of 500 times. Without using the control variate, computational gains are also substantial.



TOL	L^*	$\frac{\hat{W}_{ML_{cv}}}{\hat{W}_{\mathrm{SSA}}}$	$\frac{\hat{W}_{ML}}{\hat{W}_{\mathrm{SSA}}}$
3.13e-03	1.0	0.002 ±0.0004	0.03 ±0.001
1.56e-03	1.0	0.003 ±0.0004	0.04 ±0.001
7.81e-04	1.0	0.003 ±0.0010	0.04 ±0.002
3.91e-04	1.0	0.004 ±0.0004	0.06 ±0.003
1.95e-04	2.0	0.013 ±0.0015	0.09 ±0.008
9.77e-05	3.0	0.027 ±0.0040	0.13 ±0.016
4.88e-05	4.0	0.065 ±0.0146	0.19 ±0.025
2.44e-05	6.0	0.100 ±0.0136	0.21 ±0.020
1.22e-05	6.0	0.109 ±0.0299	0.22 ±0.029
6.10e-06	6.0	0.108 ±0.0168	0.19 ±0.020

FIG. 5.6. *Left: Predicted work (runtime) versus the estimated error bound, with 95% confidence intervals, for the simple stiff model with and without using the control variate at level 0, as described in Section 4. Right: Details of the ensemble run of the phase II algorithm using the control variate (third column) and without using the control variate (fourth column). As an example, the fifth row of the table tells us that, for a tolerance* TOL=$1.95 \cdot 10^{-4}$, *2 levels are needed on average. The work of the multilevel mixed method using the control variate at level 0 is, on average, 1% of the work of the SSA. When not using the control variate, it is 9%. Confidence intervals at 95% are also provided.*

6. Conclusions. In this work, we addressed the problem of approximating the quantity of interest $\mathrm{E}\left[g(X(T))\right]$, where X is a non-homogeneous Poisson process that describes a stochastic reactions network, and g is a given suitable observable of X, within a given prescribed relative tolerance, TOL>0, up to a given confidence level at near-optimal computational work.

We developed an automatic, adaptive reaction-splitting multilevel Monte Carlo method, based on our Chernoff tau-leap mehthod [16, 17]. Its computational complexity is $\mathcal{O}\left(TOL^{-2}\right)$. This method can be therefore seen as a variance reduction of the SSA, which has the same complexity. In our numerical examples, we obtained substantial gains with respect to SSA and, for systems in which the set of reaction channels can be adaptively partitioned into "high" and "low" activity, over our previous multilevel hybrid Chernoff tau-leap method.

We also presented a novel control variate for $g(X(T))$, which adds negligible computational cost when simulating a path of $X(T)$, and it may lead to additional dramatic cost reductions.

Acknowledgments. The research reported here was supported by King Abdullah University of Science and Technology (KAUST). The authors are members of the SRI Center for Uncertainty Quantification in Computational Science and Engineering

at KAUST.

REFERENCES

[1] J. Ahrens and U. Dieter. Computer methods for sampling from gamma, beta, Poisson and bionomial distributions. *Computing*, 12:223–246, 1974.

[2] D. Anderson and D. Higham. Multilevel Monte Carlo for continuous Markov chains, with applications in biochemical kinetics. *SIAM Multiscal Model. Simul.*, 10(1), 2012.

[3] D. F. Anderson. A modified next reaction method for simulating chemical systems with time dependent propensities and delays. *The Journal of Chemical Physics*, 127(21), 2007.

[4] C. C. Battaile. The kinetic monte carlo method: Foundation, implementation, and application. *Computer Methods in Applied Mechanics and Engineering*, Volume 197(41-42):33863398, 2008.

[5] Y. Cao, D. T. Gillespie, and L. R. Petzold. The slow-scale stochastic simulation algorithm. *The Journal of Chemical Physics*, 122(1):–, 2005.

[6] Y. Cao, D. T. Gillespie, and L. R. Petzold. Efficient step size selection for the tau-leaping simulation method. *The Journal of Chemical Physics*, 124(4):044109, 2006.

[7] Y. Cao and L. Petzold. Accuracy limitations and the measurement of errors in the stochastic simulation of chemically reacting systems. *Journal of Computational Physics*, 212(1):6–24, 2006.

[8] S. N. Ethier and T. G. Kurtz. *Markov Processes: Characterization and Convergence (Wiley Series in Probability and Statistics)*. Wiley-Interscience, 2nd edition, 9 2005.

[9] M. A. Gibson and J. Bruck. Efficient exact stochastic simulation of chemical systems with many species and many channels. *The Journal of Physical Chemistry A*, 104(9):1876–1889, 2000.

[10] D. T. Gillespie. A general method for numerically simulating the stochastic time evolution of coupled chemical reactions. *Journal of Computational Physics*, 22:403–434, 1976.

[11] D. T. Gillespie. Approximate accelerated stochastic simulation of chemically reacting systems. *Journal of Chemical Physics*, 115:1716–1733, July 2001.

[12] D. T. Gillespie, A. Hellander, and L. R. Petzold. Perspective: Stochastic algorithms for chemical kinetics. *The Journal of Chemical Physics*, 138(17):–, 2013.

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

[14] E. Haseltine and J. Rawlings. Approximate simulation of coupled fast and slow reactions for stochastic chemical kinetics. *J. Chem. Phys*, 117(15), 2002.

[15] T. Li. Analysis of explicit tau-leaping schemes for simulating chemically reacting systems. *Multiscale Model. Simul.*, 6(2):417–436 (electronic), 2007.

[16] A. Moraes, R. Tempone, and P. Vilanova. Hybrid Chernoff tau-leap. *To appear in SIAM Multiscale Modeling and Simulation*, 2014.

[17] A. Moraes, R. Tempone, and P. Vilanova. Multilevel hybrid Chernoff tau-leap. *arXiv:1403.2943*, 2014.

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

[19] J. Puchalka and A. Kierzek. Bridging the gap between stochastic and deterministic regimes in the kinetic simulations of the biochemical reaction networks. *Biophysical Society Biophysical Journal*, 86(3):1357–1372, 2004.

[20] R. Srivastava, L. You, J. Summers, and J. Yin. Stochastic vs. deterministic modeling of intracellular viral kinetics. *Journal of Theoretical Biology*, 218(3):309–321, 2002.

Algorithm 3 Coupled mixed 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 a finer one, $(s_j)_{j=0}^{N'}$, 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 $t_K \leq T$, 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 was 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, the updating of the current split, κ, and the return sentence.

$t \leftarrow t_0$; $\bar{X} \leftarrow X(0)$; $\bar{\bar{X}} \leftarrow X(0)$
$\bar{t} \leftarrow$ next grid point in $(t_i)_{i=0}^N$ larger than t
$(\bar{H}, \bar{\mathcal{R}}_{\mathrm{TL}}, \bar{\mathcal{R}}_{\mathrm{MNRM}}, \bar{a}) \leftarrow$ Algorithm 4 with $(\bar{X},t,\bar{t},T,\bar{\delta})$
$\bar{\bar{t}} \leftarrow$ next grid point in $(s_i)_{i=0}^N$ larger than t
$(\bar{\bar{H}}, \bar{\bar{\mathcal{R}}}_{\mathrm{TL}}, \bar{\bar{\mathcal{R}}}_{\mathrm{MNRM}}, \bar{\bar{a}}) \leftarrow$ Algorithm 4 with $(\bar{\bar{X}},t,\bar{\bar{t}},T,\bar{\bar{\delta}})$
while $t < T$ **do**
  $H \leftarrow \min\{\bar{H}, \bar{\bar{H}}\}$
  $(B_1, B_2, B_3, B_4) \leftarrow$ split building blocks from $(\bar{\mathcal{R}}_{\mathrm{TL}}, \bar{\mathcal{R}}_{\mathrm{MNRM}}, \bar{\bar{\mathcal{R}}}_{\mathrm{TL}}, \bar{\bar{\mathcal{R}}}_{\mathrm{MNRM}})$
  Algorithm 5 (compute state changes due to block B_1)
  Initialize internal clocks R, P if needed (see [16, 17])
  $\Delta\bar{X} \leftarrow 0$; $\Delta\bar{\bar{X}} \leftarrow 0$
  for $\mathcal{B} = B_2, B_3, B_4$ **do**
    $t_r \leftarrow t$
    $\bar{X}_r \leftarrow \bar{X}$; $\bar{\bar{X}}_r \leftarrow \bar{\bar{X}}$
    while $t_r < H$ **do**
      update $P_{j\in\mathcal{B}}$
      switch $\mathcal{B}$
        case B_2**:**
          $\bar{d} \leftarrow \bar{a}_{j\in\mathcal{B}}$
          $\bar{\bar{d}} \leftarrow a_{j\in\mathcal{B}}(\bar{\bar{X}})$
          $\tau_r \leftarrow$ Compute the Chernoff tau-leap step size using $(\bar{X}_r, \bar{a}_{j\in\mathcal{B}}, H, \bar{\delta})$
        case B_3**:**
          $\bar{d} \leftarrow a_{j\in\mathcal{B}}(\bar{X})$
          $\bar{\bar{d}} \leftarrow \bar{\bar{a}}_{j\in\mathcal{B}}$
          $\tau_r \leftarrow$ Compute the Chernoff tau-leap step size using $(\bar{\bar{X}}_r, \bar{\bar{a}}_{j\in\mathcal{B}}, H, \bar{\bar{\delta}})$
        case B_4**:**
          $\bar{d} \leftarrow a_{j\in\mathcal{B}}(\bar{X})$
          $\bar{\bar{d}} \leftarrow a_{j\in\mathcal{B}}(\bar{\bar{X}})$
          $\tau_r \leftarrow \infty$
      end switch
      $A_1 \leftarrow \min(\bar{d}, \bar{\bar{d}})$
      $A_2 \leftarrow \bar{d} - A_1$; $A_3 \leftarrow \bar{\bar{d}} - A_1$
      $H_r \leftarrow \min\{H, t_r{+}\tau_r\}$
      $(t_r, \bar{X}_r, \bar{\bar{X}}_r, R_{j\mathcal{B}}, P_{j\in\mathcal{B}}) \leftarrow$ Algorithm 6 with $(t_r, H_r, \bar{X}_r, \bar{\bar{X}}_r, R_{j\in\mathcal{B}}, P_{j\in\mathcal{B}}, A)$
    end while
    $\Delta\bar{X} \leftarrow \Delta\bar{X} + (\bar{X}_r{-}\bar{X})$; $\Delta\bar{\bar{X}} \leftarrow \Delta\bar{\bar{X}} + (\bar{\bar{X}}_r{-}\bar{\bar{X}})$
  end for
  $\bar{X} \leftarrow \bar{X} + \Delta\hat{X} + \Delta\bar{X}$; $\bar{\bar{X}} \leftarrow \bar{\bar{X}} + \Delta\hat{\hat{X}} + \Delta\bar{\bar{X}}$
  $t \leftarrow H$
  if $t < T$ **then**
    if $\bar{H} \leq \bar{\bar{H}}$ **then**
      $\bar{t} \leftarrow$ next grid point in $(t_i)_{i=0}^N$ larger than t
      $(\bar{H}, \bar{\mathcal{R}}_{\mathrm{TL}}, \bar{\mathcal{R}}_{\mathrm{MNRM}}, \bar{a}) \leftarrow$ Algorithm 4 with $(\bar{X},t,\bar{t},T,\bar{\delta})$
    end if
    if $\bar{H} \geq \bar{\bar{H}}$ **then**
      $\bar{\bar{t}} \leftarrow$ next grid point in $(s_j)_{j=0}^{N'}$ larger than t
      $(\bar{\bar{H}}, \bar{\bar{\mathcal{R}}}_{\mathrm{TL}}, \bar{\bar{\mathcal{R}}}_{\mathrm{MNRM}}, \bar{\bar{a}}) \leftarrow$ Algorithm 4 with $(\bar{\bar{X}},t,\bar{\bar{t}},T,\bar{\bar{\delta}})$
    end if
  end if
end while

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, $\tilde{\mathcal{R}}_{\rm TL}$, the set of reaction channels to which MNRM should be applied, $\tilde{\mathcal{R}}_{\rm MNRM}$, and current propensity values $\tilde{a}$.

$\tilde{a} \leftarrow a(\tilde{X})$
$(\tilde{\mathcal{R}}_{\rm TL}, \tilde{\mathcal{R}}_{\rm MNRM}) \leftarrow$ Algorithm 1 with $(\bar{X}, t, (a_j(\bar{X}))_{j=1}^J, \tilde{\delta}, \tilde{t}, \kappa)$
if $\tilde{\mathcal{R}}_{\rm TL} \neq \emptyset$ **then**
  $\tilde{H} \leftarrow \min\{\tilde{t}, t{+}\tau(\tilde{\mathcal{R}}_{\rm TL}), T\}$
else
  $\tilde{H} \leftarrow \min\{t{+}\tau(\tilde{\mathcal{R}}_{\rm TL}), T\}$
end if
return $(\tilde{H}, \tilde{\mathcal{R}}_{\rm TL}, \tilde{\mathcal{R}}_{\rm MNRM}, \tilde{a})$

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

$t_r \leftarrow t$
$\Delta\hat{\bar{X}} \leftarrow 0$; $\Delta\hat{X} \leftarrow 0$
while $t_r < H$ **do**
  $\bar{\tau}_r \leftarrow$ Compute the Chernoff tau-leap step size using $(\bar{X}{+}\Delta\hat{X}, \bar{a}_{j\in B_1}, H, \bar{\delta})$
  $\bar{\bar{\tau}}_r \leftarrow$ Compute the Chernoff tau-leap step size using $(\bar{\bar{X}}{+}\Delta\hat{\bar{X}}, \bar{\bar{a}}_{j\in B_1}, H, \bar{\bar{\delta}})$
  $H_r \leftarrow \min\{H, t_r{+}\bar{\tau}_r, t_r{+}\bar{\bar{\tau}}_r\}$
  $A_1 \leftarrow \min(\bar{a}_{j\in B_1}, \bar{\bar{a}}_{j\in B_1})$
  $A_2 \leftarrow \bar{a}_{j\in B_1} - A_1$
  $A_3 \leftarrow \bar{\bar{a}}_{j\in B_1} - A_1$
  $\Lambda \leftarrow \mathcal{P}(A{\cdot}(H_r{-}t_r))$
  $\Delta\hat{\bar{X}} \leftarrow \Delta\hat{\bar{X}} + (\Lambda_1{+}\Lambda_2)\nu_{j\in B_1}$
  $\Delta\hat{X} \leftarrow \Delta\hat{X} + (\Lambda_1{+}\Lambda_3)\nu_{j\in B_1}$
  $t_r \leftarrow H_r$
end while

Algorithm 6 The auxiliary function used in algorithm 3. Inputs: current time, t, current time horizon, $\bar{\bar{T}}$, current system state at coarser level and 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, $i{=}1,2,3$.

$\Delta t_i \leftarrow (P_i - R_i)/A_i$, for $i = 1, 2, 3$
$\Delta \leftarrow \min_i\{\Delta t_i\}$
$\mu \leftarrow \operatorname{argmin}_i\{\Delta t_i\}$
if $t + \Delta > \bar{\bar{T}}$ **then**
  $R \leftarrow R + A{\cdot}(\bar{\bar{T}}{-}t)$
  $t \leftarrow \bar{\bar{T}}$
else
  update $\bar{X}$ and $\bar{\bar{X}}$ using $\nu_{j\in B}$
  $R \leftarrow R + A\Delta$
  $r \leftarrow \text{uniform}(0, 1)$
  $P_\mu \leftarrow P_\mu + \log(1/r)$
  $t \leftarrow t + \Delta$
end if
return $(t, \bar{X}, \bar{\bar{X}}, R, P)$