Abstract. In this work, we extend the hybrid Chernoff tau-leap method to the multilevel Monte Carlo (MLMC) setting. Inspired by the work of Anderson and Higham on the tau-leap MLMC method with uniform time steps, we develop a novel algorithm that is able to couple two hybrid Chernoff tau-leap paths at different levels. Using dual-weighted residual expansion techniques, we also develop a new way to estimate the variance of the difference between two consecutive levels. This is crucial because the computational work required to stabilize the coefficient of variation of the sample variance estimator of the difference between two consecutive levels is often unaffordable for the deepest levels of the MLMC hierarchy. Our algorithm enforces the total error to be below a prescribed tolerance, $TOL$, with high probability. This is achieved with nearly optimal computational work. Indeed, the computational complexity of our method is of order $O(TOL^{-2})$, the same as with an exact method, but with a smaller constant. Our numerical examples show substantial gains with respect to the previous single-level approach and the Stochastic Simulation Algorithm.

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

1. Introduction. This work, inspired by [1], extends the hybrid Chernoff tau-leap method [19] to the multilevel Monte Carlo setting [9]. Consider a non-homogeneous Poisson process, $X$, taking values in the lattice of non-negative integers, $\mathbb{Z}^d_+$. We want to estimate the expected value of a given observable $g : \mathbb{R}^d \rightarrow \mathbb{R}$ of $X$, at a final time, $T$, i.e., $E[g(X(T))]$. For example, in a chemical reaction in thermal equilibrium, the $i$-th component of $X$, $X_i(t)$, could describe the number of particles of species $i$ present at time $t$. In the systems modeled here, different species undergo reactions at random times by changing the number of particles in at least one of the species. The probability of a single reaction to happen in a small time interval is modeled by a non-negative propensity function that depends on the current state of the system. We present a formal description of the problem in Section 1.1.

Pathwise realizations of such pure jump processes (see, e.g., [7]) can be simulated exactly using the Stochastic Simulation Algorithm (SSA), introduced by Gillespie in [10], or the Modified Next Reaction Method (MNRM) introduced by Anderson in [3]. Although these algorithms generate exact realizations for the Markov process, $X$, they are computationally feasible only for relatively low propensities.

For that reason, Gillespie proposed in [11] the tau-leap method to approximate the SSA by evolving the process with fixed time steps and by keeping the propensity fixed within each time step. In fact, the tau-leap method can be seen as a forward Euler method for a stochastic differential equation driven by Poisson random measures (see, e.g., [16]).

A drawback of the tau-leap method is that the simulated process may take negative values, which is an undesirable consequence of the approximation and not a qualitative feature of the original process. For this purpose, we proposed in [19] a Chernoff-type bound that controls the probability of reaching negative values by...
adjusting the time steps. Also, to avoid extremely small time steps, we proposed to switch adaptively between the tau-leap and an exact method, creating a hybrid tau-leap/exact method that combines the strengths of both methods.

More specifically, let $\bar{x}$ be the state of the approximate process at time $t$, and let $\delta \in (0, 1)$ be given. The main idea is to compute a time step, $\tau = \tau(\delta, \bar{x})$, such that the probability that the approximate process reaches an unphysical negative value in $[t, t+\tau]$ is less than $\delta$. This allows us to control the probability that a whole hybrid path exits the lattice, $\mathbb{Z}^d_+$. In turn, this quantity leads to the definition of the global exit error, which is a global error component along with the time discretization error and the statistical error (see Section 3.2 for details).

The multilevel Monte Carlo idea goes back at least to [13, 12]. In that setting, the main goal was to solve high-dimensional parameter-dependent integral equations and to conduct corresponding complexity analyses. Later, in [9], Giles presented multigrid ideas that were used to reduce the computational work when estimating an expected value using Monte Carlo path simulations of a certain quantity of interest of a stochastic differential equation. At almost the same time, [20] introduced a multilevel approach to control variates. Control variates are a widespread variance reduction technique with the main goal of increasing the precision of an estimator or reducing the computational effort. The main idea is as follows: to reduce the variance of the standard Monte Carlo estimator of $E[X]$,

$$\hat{\mu}_1 := \frac{1}{M} \sum_{m=1}^{M} X(\omega_m),$$

we consider another unbiased estimator of $E[X]$,

$$\hat{\mu}_2 := \frac{1}{M} \sum_{m=1}^{M} (X(\omega_m) - (Y(\omega_m) - E[Y])),$$

where $Y$ is a random variable correlated with $X$ with known mean, $E[Y]$. The variable $Y$ is called a control variate. Since $\text{Var} [\hat{\mu}_2] = \text{Var} [\hat{\mu}_1] + \text{Var} [Y] - 2 \text{Cov} [X, Y]$, whenever $\text{Cov} [X, Y] > \text{Var} [Y] / 2$, we have that $\text{Var} [\hat{\mu}_2] \leq \text{Var} [\hat{\mu}_1]$. If we assume that the computational work of generating the pair $(X(\omega), Y(\omega))$ is less than twice the computational work of generating $X(\omega)$, it is straightforward to conclude that $\hat{\mu}_2$ is preferred when $\rho^2_{X,Y} > 1/2$, where $\rho_{X,Y}$ is the correlation coefficient of the pair $(X, Y)$. We observe that $\hat{\mu}_2$ can be written as

$$\hat{\mu}_2 = E[Y] + \frac{1}{M} \sum_{m=1}^{M} (X - Y)(\omega_m).$$

In the case where $E[Y]$ is unknown and $Y$ is computationally less expensive than $X$ to sample from, it is natural to estimate $E[Y]$ using Monte Carlo sampling to yield a two-level Monte Carlo estimator of $E[X]$ based on the control variate, $Y$, i.e.,

$$\tilde{\mu}_2 := \frac{1}{M_0} \sum_{m_0=1}^{M_0} Y(\omega_{m_0}) + \frac{1}{M_1} \sum_{m_1=1}^{M_1} (X - Y)(\omega_{m_1}).$$

See Section 1.6 for details about the definition of levels in our context.

In this work, we apply the Giles multilevel control variates idea to the hybrid Chernoff tau-leap approach to reduce the computational cost, which is measured as
the amount of time needed for computing an estimate of $E[g(X(T))]$, within $TOL$, with a given level of confidence. We show that our hybrid MLMC method has the same computational complexity of the pure SSA, i.e., order $O(TOL^{-2})$. From this perspective, our method can be seen as a variance reduction for the SSA since our MLMC method does not change the complexity, it just reduces the constant. We note in passing that in the work [2], the authors show that the computational complexity for the pure tau-leap case has order $O(TOL^{-2}(\log(TOL))^2)$.

The global error arising from our hybrid tau-leap MLMC method can naturally be decomposed into three components: the global exit error, the time discretization error and the statistical error. This global error should be less than a prescribed tolerance, $TOL$, with probability larger than a certain confidence level. The global exit error is controlled by the one-step exit probability bound, $\delta$ [19]. The time discretization error, inherent to the tau-leap method, is controlled through the size of the mesh, $h$ [14]. At this point, it is crucial to stress that, by controlling the exit probability of the set of hybrid paths, we are indirectly turning this event into a rare event. We observe that direct sampling of exit paths is not an affordable way to estimate the probability of such an event.

Motivated by results of Collier et al. [6] (see appendix A, Theorem 1), we approximate the statistical error by a Gaussian random variable with zero mean. Its variance is a linear combination of the variance at the coarsest level and variances of the difference of two consecutive levels, which we sometimes call strong errors. In Section 3.3, we develop a novel dual-weighted residual expansion that allows us to estimate those strong errors. We control the statistical error through the number of coupled hybrid paths, $(M_\ell)_{\ell=0}^L$, simulated at each level.

1.1. A Class of Markovian Pure Jump Processes. To describe the class of Markovian pure jump process, $X : [0,T] \times \Omega \to \mathbb{Z}_+^d$, that we use in this work, we consider a 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 want to study the evolution of the state vector, $X(t) = (X_1(t), \ldots, X_d(t)) \in \mathbb{Z}_+^d$, modelled as a continuous-time, discrete-space Markov chain starting at some state, $X(0) \in \mathbb{Z}_+^d$. 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

$$x \rightarrow x + \nu_j.$$

The probability that reaction $j$ will occur during the small interval $(t, t+dt)$ is then assumed to be

$$P(X(t+dt) = x + \nu_j | X(t) = x) = a_j(x) dt + o(dt),$$

with a given non-negative polynomial propensity function, $a_j : \mathbb{R}^d \rightarrow \mathbb{R}$. We set $a_j(x) = 0$ for those $x$ such that $x + \nu_j \notin \mathbb{Z}_+^d$. A process, $X$, that satisfies (1.1), is a continuous-time, discrete-space Markov chain that admits the following random time change representation [7]:

$$X(t) = X(0) + \sum_{j=1}^{J} \nu_j Y_j \left( \int_0^t a_j(X(s)) \, ds \right),$$
where $Y_j : \mathbb{R}_+ \times \Omega \rightarrow \mathbb{Z}_+$ are independent unit-rate Poisson processes. Hence, $X$ is a non-homogeneous Poisson process.

In [14], the authors assume that there exists a vector, $w \in \mathbb{R}^d_+$, such that $(w, \nu_j) \leq 0$, for any reaction $\nu_j$. Therefore, every reaction, $\nu_j$, must have at least one negative component. This means that the species can be either transformed into other species or be consumed during the reaction. As a consequence, the space of states is contained in a simplex with vertices in the coordinate axis. This assumption excludes, for instance, birth processes. In our numerical examples, we allow the set of possible states of the system to be infinite, but we explicitly avoid cases in which one or more species grows exponentially fast or blows up in the time interval $[0,T]$.

**Remark 1.1.** In this setting, the solution of the following system of ordinary differential equations

$$
\begin{cases}
    \dot{x}(t) &= \nu a(x(t)), \quad t \in \mathbb{R}_+ \\
    x(0) &= x_0 \in \mathbb{R}_+.
\end{cases}
$$

is called Mean Field, where $\nu$ is the matrix with columns $\nu_j$ and $a(x)$ is the column vector of propensities. In Section 4, we use the Mean Field path for scaling and pre-processing constants associated with the computational work of the SSA and Chernoff tau-leap steps.

**1.2. Description of the Modified Next Reaction Method.** The MNRM, introduced in [3], based on the Next Reaction Method (NRM) [8], 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. For each reaction, $j$, the internal time is defined as $R_j(t) = \int_0^t a_j(X(s))ds$. There are $J+1$ time frames in the system, the absolute one, $t$, and one for each Poisson process, $Y_j$. 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. The NRM and MNRM make use of the internal times to reduce the number of simulated random variables by half. In the following, we describe the MRNM and then we present its implementation in Algorithm 1.

Given $t$, we have the propensity $a_j = a_j(X(t))$ and the internal time $R_j = R_j(t)$. Now, let $\Delta R_j$ be the remaining time for the reaction, $j$, to fire, assuming that $a_j$ stays constant over the interval $[t, t+\Delta R_j]$. Then, $t+\Delta R_j$ is the time when the next reaction, $j$, occurs. The next internal time at which the reaction, $j$, fires is then given by $R_j + a_j \Delta R_j$. When simulating the next step, the first reaction that fires occurs after $\Delta = \min_j \Delta R_j$. We then update the state of the system according to that reaction, add $\Delta$ to the global time, $t$, and then update the internal times by adding $a_j \Delta$ to each $R_j$. We are left to determine the value of $\Delta R_j$, i.e., the amount of time until the Poisson process, $Y_j$, fires, taking into account that $a_j$ remains constant until the first reaction occurs. Denote by $R_j$ the first firing time of $Y_j$ that is strictly larger than $R_j$, i.e., $P_j := \min\{s > R_j : Y_j(s) > Y_j(R_j)\}$ and finally $\Delta R_j = \frac{1}{\nu_j}(P_j - R_j)$.

Among the advantages already mentioned, we can easily modify Algorithm 1 to generate paths in the cases where the rate functions depend on time and also when there are reactions delayed in time. Finally, it is possible to simulate correlated exact/tau-leap paths using this algorithm as well as nested tau-leap/tau-leap paths.
Algorithm 1 The Modified Next Reaction Method. Inputs: the initial state, $X(0)$, the next grid point, $T_0$, the propensity functions, $(a_j)_{j=1}^J$, the stoichiometric vectors, $(\nu_j)_{j=1}^J$. Outputs: the history of system states, $(X(t_k))_{k=0}^K$. Here, we denote $S \equiv (S_j)_{j=1}^J$, $P \equiv (P_j)_{j=1}^J$, and $R \equiv (R_j)_{j=1}^J$.

1: $k \leftarrow 0$, $t_k \leftarrow 0$, $X(t_k) \leftarrow X(0)$ and $R \leftarrow 0$
2: Generate $J$ independent, uniform$(0, 1)$ random numbers, $r_j$
3: $P \leftarrow (\log(1/r_j))_{j=1}^J$
4: while $t_k < T_0$ do
5: $S \leftarrow (a_j(X(t_k)))_{j=1}^J$
6: $(\Delta R_j)_{j=1}^J \leftarrow ((P_j - R_j)/S_j)_{j=1}^J$
7: $\mu \leftarrow \arg\min_j \{\Delta R_j\}$
8: $\Delta \leftarrow \min_j \{\Delta R_j\}$
9: $t_{k+1} \leftarrow t_k + \Delta$
10: $X(t_{k+1}) \leftarrow X(t_k) + \nu_\mu$
11: $R \leftarrow R + S\Delta$
12: $r \leftarrow \text{uniform}(0, 1)$
13: $P_\mu \leftarrow P_\mu + \log(1/r)$
14: $k \leftarrow k+1$
15: end while

In [1], this technique is used to develop a uniform step, unbiased multilevel Monte Carlo (MLMC) algorithm. In Section 2.2, we use this feature for coupling two exact paths.

1.3. The Tau-Leap Approximation. In this section, we define $\tilde{X}$, the tau-leap approximation of the process, $X$, which follows from applying the forward Euler approximation to the integral term in the following random time change representation of $X$:

$$X(t + \tau) = X(t) + \sum_{j=1}^J \nu_j Y_j \left( \int_t^{t+\tau} a_j(X(s)) \, ds \right).$$

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, $\tilde{X}$, starts from $\tilde{X}(0)$ at time 0, and given that $X(t) = \bar{x}$ and a time step $\tau > 0$, we have that $\tilde{X}$ at time $t + \tau$ is generated by

$$\tilde{X}(t + \tau) = \bar{x} + \sum_{j=1}^J \nu_j P_j (a_j(\bar{x})\tau),$$

where $\{P_j(\lambda_j)\}_{j=1}^J$ are independent Poisson distributed random variables with parameter $\lambda_j$, 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., [16]).

In the limit, when $\tau$ tends to zero, the tau-leap method gives the same solution as the exact methods [17]. 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 $s \in (t, t+\tau)$. 

5
We observe that the computational work of a tau-leap step involves the generation of \( J \) independent Poisson random variables. This is in contrast to the computational work of an exact step, which only involves the work of generating two uniform random variables, in the case of the SSA, and only one in the case of MNRM.

### 1.4. The Chernoff-Based Pre-Leap Check

In [19], 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, \( \delta > 0 \). We now briefly summarize the main idea. Consider the following pre-leap check problem: find the largest possible time steps, \( \tau_i \geq 0 \), 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. The solution to that problem can be achieved by solving \( d \) auxiliary problems, one for each \( x \)-coordinate, \( i = 1, 2, \ldots, d \) as follows. Find the largest possible \( \tau_i \geq 0 \), such that

\[
(1.3) \quad P \left( \bar{X}_i(t) + \sum_{j=1}^{J} \nu_{ji} \mathcal{P}_j \left( a_j \left( \bar{X}(t) \right) \right) \tau_i < 0 \mid \bar{X}(t) \right) \leq \delta_i,
\]

where \( \delta_i = \delta/d \), and \( \nu_{ji} \) is the \( i \)-th coordinate of the \( j \)-th reaction channel, \( \nu_j \). Finally, we let \( \tau := \min \{ \tau_i : i = 1, 2, \ldots, d \} \). In order to find the largest time steps, \( \tau_i \), let \( Q_i(t, \tau_i) := \sum_{j=1}^{J} (-\nu_{ji}) \mathcal{P}_j \left( a_j \left( \bar{X}(t) \right) \right) \tau_i \). Then, for all \( s > 0 \), we have the Chernoff bound:

\[
P \left( Q_i(t, \tau_i) > \bar{X}_i(t) \mid \bar{X}(t) \right) \leq \inf_{s > 0} \exp \left( -s \bar{X}_i(t) + \tau_i \sum_{j=1}^{J} a_j(\bar{X}(t))(e^{-sv_j} - 1) \right).
\]

Expressing \( \tau_i \) as a function of \( s \), we write

\[
\tau_i(s) = \frac{\log(\delta_i) + s \bar{X}_i(t)}{-a_0(\bar{X}(t)) + \sum_{j=1}^{J} a_j(\bar{X}(t)) e^{-sv_j}} =: \frac{R_i(s)}{D_i(s)},
\]

where

\[
a_0(\bar{X}(t)) := \sum_{j=1}^{J} a_j(\bar{X}(t)).
\]

We want to maximize \( \tau_i \) while satisfying condition (1.3). Let \( \tau_i^* \) be this maximum.

We then have the following possibilities: If \( \nu_{ji} \geq 0 \), for all \( j \), then naturally \( \tau_i^* = +\infty \); otherwise, we have the following three cases:

1. \( D_i(s_i) > 0 \). In this case, \( \tau_i(s_i) = 0 \) and \( D_i(s) \) is positive and increasing as \( s \geq s_i \). Therefore, \( \tau_i(s) \) is equal to the ratio of two positive increasing functions. The numerator, \( R_i(s) \), is a linear function and the denominator, \( D_i(s) \), grows exponentially fast. Then, there exist an upper bound, \( \tau_i^* \), and a unique number, \( \hat{s}_i \), which satisfy \( \tau_i(\hat{s}_i) = \tau_i^* \). We developed an algorithm in [19] for approximating \( \hat{s}_i \), using the relation \( \tau_i^*(\hat{s}_i) = 0 \).

2. If \( D_i(s_i) < 0 \), then \( \tau_i^* = +\infty \).

3. If \( D_i(s_i) = 0 \), then \( \tau_i^* = \bar{X}_i(t)/D_i(s_i) \).

Here \( s_i := -\log(\delta_i)/\bar{X}_i(t) \).
1.5. The Hybrid Algorithm for Single-Path Generation. In this section, we briefly summarize our previous work on hybrid paths, presented in [19].

The main idea behind the hybrid algorithm is the following. A path generated by an exact method (like SSA or MNRM) never exits the lattice, \( \mathbb{Z}^d_+ \), although the computational cost may be unaffordable due to many small inter-arrival times typically occurring when the process is “far” from the boundary. A tau-leap path, which may be cheaper than an exact one, could leave the lattice at any step. The probability of this event depends on the size of the next time step and the current state of the approximate process, \( \bar{X}(t) \). This one-step exit probability could be large, especially when the approximate process is “close” to the boundary. We developed in [19] a Chernoff-type of bound to control the mentioned one-step exit probability. Even more, by construction, the probability that one hybrid path exits the lattice, \( \mathbb{Z}^d_+ \), can be estimated by

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

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 \( N_{TL}(\bar{\omega}) \equiv N_{TL} \) 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 exact/Chernoff tau-leap path, we first developed a one-step switching rule that, given the current state of the approximate process, \( \bar{X}(t) \), adaptively determines whether to use an exact or an approximated method for the next step. This decision is based on the relative computational cost of taking an exact step (MNRM) versus the cost of taking a Chernoff tau-leap step. We show the switching rule in Algorithm 2. To compare the mentioned computational costs, we define \( K_1 \) as the ratio between the cost of computing \( \tau_{Ch} \) and the cost of computing one step using the MNRM method, and \( K_2 = K_2(\bar{X}(t), \delta) \) is defined as the cost of taking a Chernoff tau-leap step, divided by the cost of taking a MNRM step plus the cost of computing \( \tau_{Ch} \). For further details on the switching rule, we refer to [19].
1.6. The Multilevel Monte Carlo Setting. In this subsection, we briefly summarize the control variates idea developed by Giles in [9]. Let \( \{X_\ell(t)\}_{\ell \in [0,T]} \) be a hybrid Chernoff tau-leap process with a time mesh of size \( \Delta t_\ell \) and a one-step exit probability bound \( \delta \). We can simulate paths of \( \{X_\ell(t)\}_{\ell \in [0,T]} \) by using Algorithm 4 in [19]. Let \( g_\ell := g(X_\ell(T)) \).

Consider a hierarchy of nested meshes of the time interval \([0,T]\), indexed by \( \ell = 0, 1, \ldots, L \). Let \( \Delta t_0 \) be the size of the coarsest time mesh that corresponds to the level \( \ell = 0 \). 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.

Assume that we are interested in estimating \( E[g_L] \), and we are able to simulate correlated pairs, \( (g_\ell, g_{\ell-1}) \) for \( \ell = 1, \ldots, L \). Then, the following unbiased Monte Carlo estimator of \( E[g_L] \) uses \( g_{L-1} \) as a control variate:

\[
\hat{\mu}_L := \frac{1}{M_L} \sum_{m_L=1}^{M_L} (g_L(\omega_{m_L}) - (g_{L-1}(\omega_{m_L}) - E[g_{L-1}]))
\]

\[
= E[g_{L-1}] + \frac{1}{M_L} \sum_{m_L=1}^{M_L} (g_L - g_{L-1})(\omega_{m_L}).
\]

Applying this idea recursively, and taking into account the following telescopic decomposition: \( E[g_L] = E[g_0] + \sum_{\ell=1}^{L} E[g_\ell - g_{\ell-1}] \), we arrive at the multilevel Monte Carlo estimator of \( E[g_L] \)

\[
(1.4) \quad \hat{\mu}_L := \frac{1}{M_0} \sum_{m_0=1}^{M_0} g_0(\omega_{m_0}) + \sum_{\ell=1}^{L} \frac{1}{M_\ell} \sum_{m_\ell=1}^{M_\ell} (g_\ell - g_{\ell-1})(\omega_{m_\ell}).
\]

We have that \( \hat{\mu}_L \) is unbiased, since \( E[\hat{\mu}_L] = E[g_L] \). The variance of \( \hat{\mu}_L \) is given by \( \text{Var}[\hat{\mu}_L] = \frac{\text{Var}[g_0]}{M_0} + \sum_{\ell=1}^{L} \frac{\text{Var}[g_\ell - g_{\ell-1}]}{M_\ell} \). Here we are assuming independence among the batches between levels. For highly correlated pairs \((g_\ell, g_{\ell-1})\), we can expect, for the same computational work, that \( \text{Var}[\hat{\mu}_L] \) is much less than the variance of the standard Monte Carlo estimator of \( E[g_L] \).

1.7. Outline of this Work. In Section 2, we first show the main idea for coupling two tau-leap paths, that is a construction by Kurtz [15] for coupling two Poisson random variables. Then, inspired by the ideas of Anderson and Higham in [1], we propose an algorithm for coupling two hybrid Chernoff tau-leap paths (see [19]). 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 novel hybrid MLMC estimator. Next, we introduce a global error decomposition, and finally, we develop a technique to estimate the variance of the difference of two consecutive levels based on dual weighted residuals. This estimate results from the analysis of the local error caused by taking two tau-leap coupled paths associated with two consecutive uniform time meshes. Next, in Section 4, we show how to control the three error components of the global error and how to obtain the parameters needed for computing the hybrid MLMC estimator in order to achieve a given tolerance, with nearly optimal computational work. We also show that the computational complexity of our method is of order \( O(TOL^{-2}) \). In Section 5, the numerical examples illustrate the advantages of the hybrid MLMC method over the single-level approach presented in [19] and to the SSA. Section 6 presents the conclusions and suggestions for future work.
2. Generating Coupled Hybrid Paths. In this section, we present an algorithm that generates coupled hybrid Chernoff tau-leap paths, which is an essential ingredient for the multilevel Monte Carlo estimator. We first show how to couple two Poisson random variables and then we explain how we make use of the two algorithms presented in [1] (as Algorithms 2 and 3) and two additional algorithms we developed to create an algorithm that generates coupled hybrid paths.

2.1. Coupling Two Poisson Random Variables. We motivate our coupling algorithm (Algorithm 3) by first describing how to couple two Poisson random variables. In our context, ‘coupling’ means that we want to induce a correlation between them that is as strong as possible. This construction was first proposed by Kurtz in [15]. Suppose that we want to couple $P_1(\lambda_1)$ and $P_2(\lambda_2)$, two Poisson random variables, with rates $\lambda_1$ and $\lambda_2$, respectively. Consider the following decompositions,

$$P_1(\lambda_1) := P^*(\lambda_1 \land \lambda_2) + Q_1(\lambda_1 - \lambda_1 \land \lambda_2)$$
$$P_2(\lambda_2) := P^*(\lambda_1 \land \lambda_2) + Q_2(\lambda_2 - \lambda_1 \land \lambda_2),$$

where $P^*(\lambda_1 \land \lambda_2)$, $Q_1(\lambda_1 - \lambda_1 \land \lambda_2)$ and $Q_2(\lambda_2 - \lambda_1 \land \lambda_2)$ are three independent Poisson random variables. Here, $\lambda_1 \land \lambda_2 := \min\{\lambda_1, \lambda_2\}$. Observe that at least one of the following vanishes: $Q_1(\lambda_1 - \lambda_1 \land \lambda_2)$ and $Q_2(\lambda_2 - \lambda_1 \land \lambda_2)$. This is because at least one of the rates is zero. Algorithm 3 implements these ideas. Finally, note that, by construction, we have

$$\text{Var}[P_1(\lambda_1) - P_2(\lambda_2)] = \text{Var}[Q_1(\lambda_1 - \lambda_1 \land \lambda_2) - Q_2(\lambda_2 - \lambda_1 \land \lambda_2)] = |\lambda_1 - \lambda_2|.$$

However, if instead we consider making $P_1(\lambda_1)$ and $P_2(\lambda_2)$ independent, then

$$\text{Var}[P_1(\lambda_1) - P_2(\lambda_2)] = \lambda_1 + \lambda_2,$$

which may be a large value even when $\lambda_1$ and $\lambda_2$ are close.

2.2. Coupling Two Hybrid Paths. In this section, we describe how to generate two coupled hybrid Chernoff tau-leap paths, $\bar{X}$ and $\bar{\bar{X}}$, 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 $\bar{\bar{X}}(t)$. Based on this knowledge, we have to determine a method for each level. This method can be either the MNRM or the tau-leap one, determining four possible combinations leading to four algorithms, B1, B2, B3 and B4, that we use as building blocks. Table 2.1 summarizes them.

| Algorithm | at coarser mesh | at fine mesh |
|-----------|----------------|-------------|
| B1        | (part of Algorithm 3) | TL          | TL          |
| B2        | (Algorithm 5)       | TL          | MNRM        |
| B3        | "                | MNRM        | TL          |
| B4        | "                | MNRM        | MNRM        |

Table 2.1

*Building blocks for simulating two coupled hybrid Chernoff tau-leap paths. Algorithms B1 and B2 are presented as Algorithms 2 and 3 in [1]. Algorithm B3 can be directly obtained from Algorithm B2. Algorithm B4 is also based on Algorithm B2, but in order to produce MNRM steps, we update the propensities at the coarse level at the beginning of each time interval defined by the fine level.*
We note that the only case in which we use a Poisson random variates generator for the tau-leap method is in Algorithm B1. For Algorithms B2 and B3, the Poisson random variables are simulated by adding independent exponential random variables with the same rate, $\lambda$, until exceeding a given time final time $T$. The rate, $\lambda$, is obtained by freezing the propensity functions, $a$, at time $t$. More specifically, the Poisson random variates are obtained by using the MNRM repeatedly without updating the intensity.

We now briefly describe the Chernoff hybrid coupling algorithm, i.e., Algorithm 3. Given the current time, $t$, and the current state of the process at the coarse level, $\bar{X}(t)$, and the fine level, $\bar{\bar{X}}(t)$, this algorithm determines the next time point at which we run the algorithm (called time “horizon”). In order to fix ideas, let us assume that, based on $\bar{X}(t)$, the one-step switching rule, i.e., Algorithm 2, chooses the tau-leap method at the coarse level, with the corresponding Chernoff step size, $\bar{\tau}$. As we mentioned, this $\bar{\tau}$ is the largest step size such that the probability that the process, in the next time step, takes a value outside $\mathbb{Z}_d^+$, is less than $\delta$. This step size plus the current time, $t$, cannot be greater than the final time, $T$, and also cannot be greater than the next time discretization grid point in the coarse grid, because the discretization error must be controlled. Taking the minimum of all those values, we obtain the next time horizon at the coarse grid, $\bar{\bar{H}}$. Note that, if the chosen method is MNRM instead of tau-leap, we do not need to take into account the grid, and the next time horizon will be the minimum between the next reaction time and the final time, $T$.

We now explain algorithm B1 (TL-TL). Assume that tau-leap is chosen at the coarse and at the fine level, thus obtaining two time horizons, one for the coarse level, $\bar{H}$, and another for the fine level, $\bar{\bar{H}}$. In this case, the global time horizon will be $H := \min\{\bar{H}, \bar{\bar{H}}\}$. Since the chosen method in both grid levels is tau-leap, we need to freeze the propensities at the beginning of the corresponding intervals. In the coarse case, during the interval $[t, \bar{H})$ (the propensities are equal to $a(\bar{X}(t))=\bar{a}$), and in the fine case during the interval $[t, \bar{\bar{H}})$ (the propensities are equal to $a(\bar{\bar{X}}(t))=\bar{\bar{a}}$). Suppose that $\bar{H} < \bar{\bar{H}}$. Then, we couple two Poisson random variables at time $t=H$, using the idea described in Section 2.1. When time reaches $\bar{H}$, the decision of which method to use (and the corresponding step size) at the coarse level must be made again. Note that the propensities of the process at the fine grid will be kept frozen until $\bar{\bar{H}}$. The case when $\bar{\bar{H}} < \bar{H}$ is analogous to the one we described, but the decisions on the method and step size are made at the finer level, when time reaches $\bar{\bar{H}}$. It can also be possible that $\bar{H} = \bar{\bar{H}}$. In that case, the decision of which method to use (and the corresponding step size) must be made at the coarse and at the fine level.

In the case of algorithm B2 (TL-MNRM), we assume that tau-leap is chosen at the coarse level, and MNRM at the fine level, obtaining two time horizons, one for the coarse level, $\bar{H}$, and another for the fine level, $\bar{\bar{H}}$. The only difference in how we determine the time horizons between algorithm B1 and B2 is that the time discretization grid points at the fine grid are not taken into account to determine $\bar{H}$. Algorithm B2 is then applied until the simulation reaches $H := \min\{\bar{H}, \bar{\bar{H}}\}$. Suppose that $\bar{\bar{H}} < \bar{H}$. In this case, the process $\bar{X}$ could take more than one step to reach $\bar{\bar{H}}$. At each step, the propensity functions $a(\bar{X}())$ are computed, but not the propensities for the coarse level, because in that case the tau-leap method is used. Note that the decision on which algorithm to use (B2 or another) is not made at those steps, but only when time reaches $\bar{\bar{H}}$. When time reaches $\bar{\bar{H}}$, the decision of which method to use (and the corresponding step size) at the fine level must be made again. In this
case, the propensities at the coarse grid will be kept frozen until \( \bar{H} \). The reasoning for the cases \( \bar{H} > \bar{H} \) and \( \bar{H} = \bar{H} \) are similar as before.

The other two cases, that is, B3 and B4, are the same as B2. The only difference resides is when to update the propensity values, \( \bar{a} \) and \( \tilde{a} \). See Algorithm 3 for more details. It is clear in the preceding paragraphs, that the decision on which algorithm should be used for a certain time interval is made only at the horizon points.

3. Multilevel Monte Carlo Estimator and Global Error Decomposition.
In this section, we present the multilevel Monte Carlo estimator. We first show the estimator and its properties and then we 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. We give upper bounds for each one of the three components.

3.1. The MLMC Estimator. In this section, we discuss and implement a variation of the multilevel Monte Carlo estimator (1.4) for the hybrid Chernoff tau-leap case. The main ingredient of this section is Algorithm 3, which generates coupled hybrid paths at levels \( \ell - 1 \) and \( \ell \). Let us now introduce some notation. Let \( A_L \) be the event in which the \( \bar{X}_L \)-path arrived at the final time, \( T \), without exiting the state space of \( X \). Let \( 1_{A_L} \) be the indicator function of an arbitrary set, \( A \). Finally, \( g(L) := g(\bar{X}_L(T)) \) was defined in Section 1.6.

Consider the following telescopic decomposition:

\[
E[g_L1_{A_L}] = E[g_01_{A_0}] + \sum_{\ell=1}^{L} E[g_\ell 1_{A_\ell} - g_{\ell-1}1_{A_{\ell-1}}],
\]

which motivates the definition of our MLMC estimator of \( E[g(X(T))] \),

\[
\mathcal{M}_L := \frac{1}{M_0} \sum_{m_0=1}^{M_0} g_01_{A_0}(\omega_{m_0}) + \sum_{\ell=1}^{L} \frac{1}{M_\ell} \sum_{m_\ell=1}^{M_\ell} [g_\ell 1_{A_\ell} - g_{\ell-1}1_{A_{\ell-1}}](\omega_{m_\ell}).
\]

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 hybrid method and the Monte Carlo statistical error, \( \mathcal{E}_{S,L} \). Next, we show how to model and control the global error, \( \mathcal{E}_L \), giving upper bounds for each one of the three components. We define the computational global error, \( \mathcal{E}_L \), as

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

Now, consider the following decomposition of \( \mathcal{E}_L \):

\[
E[g(X(T))] - \mathcal{M}_L = E[g(X(T))(1_{A_L} + 1_{A_L^c})] + E[g_L1_{A_L}] - \mathcal{M}_L
\]

\[
= E[g(X(T))(1_{A_L} + 1_{A_L^c})] + E[(g(X(T)) - g_L)1_{A_L}] + E[g_L1_{A_L}] - \mathcal{M}_L
\]

\[
= E_{E,L} + E_{I,L} + E_{S,L}.
\]

We show in [19] that by choosing adequately the one-step exit probability bound, \( \delta \), the exit error, \( \mathcal{E}_{E,L} \), satisfies \( |\mathcal{E}_{E,L}| \leq |E[g(X(T))]| P(A_L^c) \leq TOL^2 \). An efficient procedure for accurately estimating \( \mathcal{E}_{I,L} \) in the context of the tau-leap method is
described in [14]. We adapt this method in Algorithm 9 for estimating the weak error in the hybrid context. A brief description follows. For each hybrid path, \( \tilde{X}(t_{n}, \omega) \), we define the sequence of dual weights \( (\varphi_{n, \ell}^{N}(\omega))_{n=1}^{N} \) backwards as follows:

\[
(3.2) \quad \varphi_{N(\omega), \ell} := \nabla g(\tilde{X}(t_{N(\omega)}, \ell, \omega)) \quad \varphi_{n, \ell} := \left( \mathbf{I}d + \Delta_{t_{n, \ell}} \mathbf{J}_{a}^{T}(\tilde{X}(t_{n, \ell}, \omega)) \mu^{T} \right) \varphi_{n+1, \ell}, \quad n = N(\omega) - 1, \ldots, 1,
\]

where \( \Delta_{t_{n, \ell}} := t_{n+1, \ell} - t_{n, \ell} \), \( \nabla \) is the gradient operator and \( \mathbf{J}_{a}(\tilde{X}(t_{n, \ell}, \omega)) \equiv [\partial_{i} a_{j} (\tilde{X}(t_{n, \ell}, \omega))]_{j,i} \) is the Jacobian matrix of the propensity function, \( a_{j} \), for \( j = 1 \ldots J \) and \( i = 1 \ldots d \). According to this method, \( \mathcal{E}_{I,L}^{2} \) is approximated by \( \mathcal{A}(\mathcal{E}_{I,L}^{2}; \cdot) \), where

\[
\mathcal{E}_{I,L}^{2} := \sum_{n=1}^{N(\omega)} \left( \frac{\Delta_{t_{n,L}}}{2} \varphi_{n,L} \mathbf{1}_{TL}(n) \sum_{j=1}^{J} \nu_{j}^{2} \Delta a_{j,n} \right) (\omega),
\]

\( \mathcal{A}(X; M) := \frac{1}{M} \sum_{m=1}^{M} X(\omega_{m}) \), and, \( \mathcal{S}^{2}(X; M) := \mathcal{A}(X^{2}; M) - \mathcal{A}(X; M)^{2} \) denote the sample mean and the sample variance of the random variable, \( X \), respectively. Here, \( \Delta a_{j,n}(\omega) := a_{j}(\tilde{X}(t_{n+1, \ell}, \omega)) - a_{j}(\tilde{X}_{L}(t_{n, \ell}, \omega)) \). \( \mathbf{1}_{TL}(n) = 1 \) if and only if, at time \( t_{n, \ell} \), the tau-leap method was used, and we denote by \( \mathbf{I}d \) the \( d \times d \) identity matrix.

The variance of the statistical error, \( \mathcal{E}_{S,L}^{2} \), is given by \( \sum_{\ell=0}^{L} \frac{\nu_{\ell}^{2}}{M_{\ell}} \), where \( \nu_{0} := \text{Var}[g_{0} \mathbf{1}_{A_{0}}] \) and \( \nu_{\ell} := \text{Var}[g_{\ell} \mathbf{1}_{A_{\ell}} - g_{\ell-1} \mathbf{1}_{A_{\ell-1}}] \), \( \ell \geq 1 \). In the next subsection, we show how to estimate \( \mathcal{E}_{\ell}^{2} \).

3.3. Dual-weighted Residual Estimation of \( \mathcal{E}_{\ell}^{2} \). Here, we derive the formula (3.3) for estimating the variance, \( \mathcal{E}_{\ell}^{2}, \ell \geq 1 \). It is based on dual-weighted local errors arising from two consecutive tau-leap approximations of the process, \( X \). For \( \ell \geq 1 \), it estimates \( \mathcal{E}_{\ell}^{2} \) with much smaller statistical error than the standard Monte Carlo estimator.

Let us introduce some notation:

\[
\begin{align*}
\bar{f}_{j,n} & := (\varphi_{n+1} \cdot \nu_{j}), \\
\bar{\mu}_{j,n} & := \frac{\Delta_{t_{n}}}{2} \sum_{i} (\nabla a_{j}(x_{n}) \cdot \nu_{i}) a_{i}(x_{n}), \\
\bar{\mu}_{2,n} & := \frac{\Delta_{t_{n}}}{2} \sum_{i} |(\nabla a_{j}(x_{n}) \cdot \nu_{i})| a_{i}(x_{n}), \\
\sigma_{j,n}^{2} & := \frac{\Delta_{t_{n}}}{2} \sum_{i} (\nabla a_{j}(x_{n}) \cdot \nu_{i})^{2} a_{i}(x_{n}), \\
m_{j,n} & := \min\{\bar{\mu}_{j,n}, \sqrt{\bar{\mu}_{j,n}^{2} + \sigma_{j,n}^{2}}\}, \\
q_{j,n} & := \frac{\bar{\mu}_{j,n}}{\sigma_{j,n}}, \\
p_{j,n} & := \Phi(-q_{j,n}), \\
\bar{\mu}_{j,n} & := \mu_{j,n}(1 - 2p_{j,n}), \\
\bar{\sigma}_{j,n} & := \sqrt{\frac{2}{\pi}} \sigma_{j,n} \exp(-q_{j,n}^{2}/2).
\end{align*}
\]
Here, \( \Phi(x) \) is the cumulative distribution function of a standard Gaussian random variable. We define our dual-weighted estimator of \( \mathcal{V}_\ell \) as

\[
\hat{\mathcal{V}}_\ell := S^2 \left( \sum_n 1_{TL}(n) \frac{\Delta t_n}{2} \sum_j f_{j,n} \mu_{j,n}; M_\ell \right) + A \left( \sum_n 1_{TL}(n) \frac{(\Delta t_n)^3}{8} \sum_{j,j'} f_{j,n} f_{j',n} \sum_i (\nabla a_j(x_n) \cdot \nu_i)(\nabla a_{j'}(x_n) \cdot \nu_i) a_i(x_n); M_\ell \right) + A \left( \sum_n 1_{TL}(n) \frac{\Delta t_n}{2} \sum_j f_{j,n}^2 (1 G_n (\tilde{\mu}_{j,n} + \tilde{\sigma}_{j,n}) + 1 G_m m_{j,n}); M_\ell \right),
\]

where \( 1 G_n = 1 \) if and only if, \( a_j(x_n) \frac{\Delta t_n}{2} > c \) for all \( j \in \{1, \ldots, J\} \), where \( c \) is a positive user-defined constant.

First, notice that \( \mathcal{V}_\ell \) could be a very small positive number. In fact, in our numerical experiments, we observe that the standard Monte Carlo sample estimation of this quantity turns out to be computationally unfeasible due to the huge number of simulations required to stabilize its coefficient of variation. For this reason, we initially consider the following dual-weighted approximations:

\[
\mathbb{E} \left[ g_{\ell} - g_{\ell-1} \right] \approx \mathbb{E} \left[ \sum_n \varphi_{n+1,\ell-1} \cdot e_{n+1,\ell-1} \right],
\]

\[
\text{Var} \left[ g_{\ell} - g_{\ell-1} \right] \approx \text{Var} \left[ \sum_n \varphi_{n+1,\ell-1} \cdot e_{n+1,\ell-1} \right],
\]

where \( (\varphi_{n+1,\ell-1})_{n=0}^{N(\tilde{\omega})} \), defined in (3.2), is a sequence of dual weights computed backwards from a simulated path, \( (\tilde{X}_\ell(t_n,\tilde{\omega}))_{n=1}^{N(\tilde{\omega})} \), and the sequence of local errors, \( (e_{n+1,\ell-1})_{n=0}^{N(\tilde{\omega})} \), is the subject of the next subsection.

At this moment, it is convenient to recall the tower properties of the conditional expectation and the conditional variance: given a random variable, \( X \), and a sigma algebra, \( \mathcal{F} \), defined over the same probability space, we have

\[
\mathbb{E} [X] = \mathbb{E} [\mathbb{E} [X | \mathcal{F}] ],
\]

\[
\text{Var} [X] = \text{Var} [\mathbb{E} [X | \mathcal{F}] ] + \mathbb{E} [\text{Var} [X | \mathcal{F}] ].
\]

Hereafter, we fix \( \ell \) and, for the sake of brevity, omit it as subindex.

Applying (3.5) to \( \sum_n \varphi_{n+1} \cdot e_{n+1} \) and conditioning on \( \mathcal{F} \), we obtain

\[
\text{Var} \left[ \sum_n \varphi_{n+1} \cdot e_{n+1} \right] = \text{Var} \left[ \mathbb{E} \left[ \sum_n \varphi_{n+1} \cdot e_{n+1} | \mathcal{F} \right] \right] + \mathbb{E} \left[ \text{Var} \left[ \sum_n \varphi_{n+1} \cdot e_{n+1} | \mathcal{F} \right] \right] = \text{Var} \left[ \sum_n \mathbb{E} \left[ \varphi_{n+1} \cdot e_{n+1} | \mathcal{F} \right] \right] + \mathbb{E} \left[ \sum_n \text{Var} \left[ \varphi_{n+1} \cdot e_{n+1} | \mathcal{F} \right] \right].
\]

The main idea is to generate \( M_\ell \) Monte Carlo paths, \( (\tilde{X}_\ell(t_n;\tilde{\omega}))_{n=1}^{N(\tilde{\omega})} \), and to estimate
Var $[\sum_n \varphi_{n+1} \cdot e_{n+1}]$ using

$$
(3.6)
\hat{\nu}_t := S^2 \left( \sum_n \mathbb{E} \left[ \varphi_{n+1} \cdot e_{n+1} \mid \mathcal{F} \right] (\omega); M_t \right) + A \left( \sum_n \text{Var} \left[ \varphi_{n+1} \cdot e_{n+1} \mid \mathcal{F} \right] (\omega); M_t \right).
$$

To avoid nested Monte Carlo calculations, in what follows we develop exact and approximate formulas for computing $\mathbb{E} \left[ \varphi_{n+1} \cdot e_{n+1} \mid \mathcal{F} \right]$ and $\text{Var} \left[ \varphi_{n+1} \cdot e_{n+1} \mid \mathcal{F} \right]$. To derive those formulas, we consider a sigma algebra $\mathcal{F}$, such that $(\varphi_n(\omega))_{n=1}^{N(\omega)}$, conditioned on $\mathcal{F}$, is deterministic, i.e., $(\varphi_n(\omega))_{n=1}^{N(\omega)}$ is measurable with respect to $\mathcal{F}$. In this way, the only randomness in $\mathbb{E} \left[ \varphi_{n+1} \cdot e_{n+1} \mid \mathcal{F} \right]$ and $\text{Var} \left[ \varphi_{n+1} \cdot e_{n+1} \mid \mathcal{F} \right]$ comes from the local errors, $(e_n)_{n=1}^{N(\omega)}$.

**Forward Local Error Approximation.** The main result of this section is the representation (3.11) that is used to derive $\hat{\nu}_t$, given by (3.3).

For simplicity of analysis, we make two assumptions: i) the time mesh associated with the level, $\ell$, is obtained by halving the intervals of the level $\ell-1$; ii) we perform the tau-leap at both levels without considering the Chernoff bounds.

Let $\tilde{X}$ and $\bar{X}$ be two tau-leap approximations of $X$ based on two consecutive grid levels, for instance, $\tilde{X} := X_{\ell-1}$, and $\bar{X} := X_{\ell}$. Consider two consecutive time-mesh points for $\tilde{X}$, $(t_n, t_{n+1})$, and three consecutive time-mesh points for $\bar{X}$, $(t_n, (t_n+t_{n+1})/2, t_{n+1})$. Let $\tilde{X}$ and $\bar{X}$ start from $x_n$ at time $t_n$. The first step for coupling them is to define

$$
(3.7)
\tilde{X}_{n+1} := x_n + \sum_j \nu_j \mathcal{Y}_{j,n}(a_j(x_n) \Delta t_n),
$$

$$
(3.8)
\bar{X}_{n+1} := x_n + \sum_j \nu_j \mathcal{Q}_{j,n}(a_j(x_n) \frac{\Delta t_n}{2}),
$$

$$
\bar{X}_{n+1} := Z_{n+1} + \sum_j \nu_j \mathcal{R}_{j,n}(a_j(Z_{n+1}) \frac{\Delta t_n}{2}),
$$

where $\{\mathcal{Y}_{j,n}\}_{j=1}^{\tilde{J}} \cup \{\mathcal{Q}_{j,n}\}_{j=1}^{\bar{J}} \cup \{\mathcal{R}_{j,n}\}_{j=1}^{\bar{J}}$ are independent Poisson random variables.

To couple the $\tilde{X}$ and $\bar{X}$ processes, we first decompose $\mathcal{Y}_{j,n}(a_j(x_n) \Delta t_n)$ as the sum of two Poisson independent random variables, $Q_{j,n}(a_j(x_n) \frac{\Delta t_n}{2}) + Q'_{j,n}(a_j(x_n) \frac{\Delta t_n}{2})$. As a consequence, $\tilde{X}$ and $\bar{X}$ coincide in the closed interval $[t_n, (t_n+t_{n+1})/2]$. By applying this decomposition in (3.7), we obtain

$$
(3.9)
\tilde{X}_{n+1} = x_n + \sum_j \nu_j Q_{j,n}(a_j(x_n) \frac{\Delta t_n}{2}) + \sum_j \nu_j Q'_{j,n}(a_j(x_n) \frac{\Delta t_n}{2}),
$$

$$
\bar{X}_{n+1} = x_n + \sum_j \nu_j Q_{j,n}(a_j(x_n) \frac{\Delta t_n}{2}) + \sum_j \nu_j R_{j,n}(a_j(Z_{n+1}) \frac{\Delta t_n}{2}).
$$

The second step for coupling $\tilde{X}$ and $\bar{X}$, according to [1], is as follows: let $m_j := \min\{a_j(x_n), a_j(Z_{n+1})\}$, $c_j := a_j(x_n) - m_j$ and $f_j := a_j(Z_{n+1}) - m_j$. Notice that for each $j$, either $c_j$ or $f_j$ is zero (or both).
Now, consider the following decompositions:

\begin{align}
Q'_{j,n}(a_j(x_n)\frac{\Delta t_n}{2}) &= P'_{j,n}(m_j \frac{\Delta t_n}{2}) + P''_{j,n}(c_j \frac{\Delta t_n}{2}), \\
R_{j,n}(a_j(x_n)\frac{\Delta t_n}{2}) &= P'_{j,n}(m_j \frac{\Delta t_n}{2}) + R'_j(n(f_j \frac{\Delta t_n}{2}),
\end{align}

where \( P', P'' \) and \( R' \) are independent Poisson random variables.

By substituting (3.10) into (3.9), we can write the local error, \( e_{n+1} \), as

\begin{equation}
e_{n+1} := \bar{X}_{n+1} - \bar{X}_{n+1} = \sum_j \nu_j \left( R'_{j,n}(f_j - P'_{j,n}(c_j \frac{\Delta t_n}{2})) - P''_{j,n}(-\Delta a_{j,n} \frac{\Delta t_n}{2}) 1\{\Delta a_{j,n} > 0\} \right),
\end{equation}

where \( \Delta a_{j,n} := a_j(Z_{n+1}) - a_j(x_n) \), and \( Z_{n+1} \) has been defined in (3.8). Note that in (3.11) not only are \( R'_{j,n} \) and \( P'_{j,n} \) random variables, but \( \Delta a_{j,n} \) is also random because it depends on the random variables \( (Q'_{j,n})_{j=1}^J \). Also note that all the mentioned random variables are independent.

**Conditional Local Error Representation.** In this section, we derive a local error representation that takes into account the fact that the dual is computed backwards and the distribution of the local errors that is relevant to our calculations is therefore not exactly the one given by (3.11), but the distribution given by (3.12).

Consider the sequence \( (\bar{X}_{n})_{n=0}^N \) defined in (3.7). For fixed \( n \), define \( \mathcal{F}_n \) as the sigma algebra

\[ \mathcal{F}_n := \sigma \left( (\mathcal{Y}_{j,k}(a_j(x_k)\Delta t_k))_{j=1,\ldots,J, k=1,\ldots,n} \right). \]

That is, this is the information we obtain by observing the randomness used to generate \( \bar{X}_{n+1} \) from \( x_0 \). Motivated by dual-weighted expansions (3.4), we want to express the local error representation (3.11) conditional on \( \mathcal{F} := \mathcal{F}_N \).

At this point, it is convenient to remember a key result for building Poissonian bridges. If \( X_1 \) and \( X_2 \) are two independent Poisson random variables with parameters \( \lambda_1 \) and \( \lambda_2 \), respectively, we have that \( X_1 \mid X_1 + X_2 = k \) is a binomial random variable with parameters \( k \) and \( \lambda_1/ (\lambda_1 + \lambda_2) \).

Applying this observation to the decomposition \( \mathcal{Y}_{j,n}(a_j(x_n)\Delta t_n) = Q_{j,n}(a_j(x_n)\frac{\Delta t_n}{2}) + Q'_{j,n}(a_j(x_n)\frac{\Delta t_n}{2}) \), we conclude that the conditional distribution of \( Q_{j,n}(a_j(x_n)\frac{\Delta t_n}{2}) \) given \( \mathcal{Y}_n \), i.e., \( Q_{j,n}(a_j(x_n)\frac{\Delta t_n}{2}) \mid \mathcal{Y}_n \), is binomial with parameters \( \mathcal{Y}_{j,n} \) and \( 1/2 \). Define now the sigma algebra \( \mathcal{G}_n := \sigma \left( (Q_{j,n}(a_j(x_n)\frac{\Delta t_n}{2}) \mid \mathcal{Y}_n)_{j=1}^J \right) \).

Applying the same argument to \( P'' \), defined in (3.10), we conclude that

\[ P''_{j,n} \mid \mathcal{F}_n, \mathcal{G}_n \sim \text{binomial} \left( \mathcal{Y}_{j,n} - Q_{j,n}, \frac{c_j}{a_j(x_n)} \right). \]

From the definition of \( Z_{n+1} = x_n + \sum_j \nu_j Q_{j,n} \) in (3.8), we conclude that

\[ R'_{j,n} \mid \mathcal{G}_n \sim \text{Poisson} \left( (a_j(Z_{n+1}) - m_j) \frac{\Delta t_n}{2} \right). \]
Notice that, by construction, $\mathcal{P}_n^{n} \mid \{\mathcal{F}_n, \mathcal{G}_n\}$ and $\mathcal{P}_n^{n} \mid \mathcal{G}_n$ are independent random variables. Since $c_j = -\Delta a_{j,n}1_{\{\Delta a_{j,n} < 0\}}$ and $a_j(Z_{n+1}) - m_j = \Delta a_{j,n}1_{\{\Delta a_{j,n} \geq 0\}}$, we can express the conditional local error as

\begin{equation}
(3.12)
\begin{aligned}
& e_{n+1} \mid \{\mathcal{F}_n, \mathcal{G}_n\} = \\
& \sum_j \nu_j \left( R_{j,n}^{n} \left( a_{j,n} \frac{\Delta t_n}{2} \right) \right) 1_{\{\Delta a_{j,n} \geq 0\}} - \mathcal{P}_n^{n} \left( \frac{y_{j,n} - Q_{j,n}}{a_j(x_n)} \right) 1_{\{\Delta a_{j,n} < 0\}} \\
\end{aligned}
\end{equation}

in a distribution sense. For instance, we can easily compute the expectation of $e_{n+1} \mid \{\mathcal{F}_n, \mathcal{G}_n\}$ as follows:

\begin{equation}

\begin{aligned}
& E[e_{n+1} \mid \{\mathcal{F}_n, \mathcal{G}_n\}] = \\
& \sum_j \nu_j \Delta a_{j,n} \left( \frac{\Delta t_n}{2} \right) 1_{\{\Delta a_{j,n} \geq 0\}} + \frac{y_{j,n} - Q_{j,n}}{a_j(x_n)} 1_{\{\Delta a_{j,n} < 0\}} \\
\end{aligned}
\end{equation}

Taking into account that the joint distribution of $(Q_{j,n})_{j=1}^{N(\omega)} \mid \mathcal{F}_n$ is given by

\begin{equation}
P \left( (Q_{j,n})_{j=1}^{N(\omega)} \mid \mathcal{F}_n \right) = 2^{-N(\omega)} \prod_{j=1}^{N(\omega)} \frac{y_{j,n} - Q_{j,n}}{q_{j,n}(y_{j,n} - Q_{j,n})}, \quad 0 \leq q_{j,n} \leq y_{j,n},
\end{equation}

we can compute exactly the expected value and the variance of $\varphi_{n+1} \cdot e_{n+1} \mid \mathcal{F}_n$, for any given deterministic vector, $v_{n+1}$. Notice that given $\mathcal{F}$, the backward sequence $(\hat{X}_n)_{n=0}^{N(\omega)}$ is deterministic, and the sequence $(\varphi_n)_{n=1}^{N(\omega)} \mid \mathcal{F}$ is therefore also a deterministic sequence of vectors. We can thus compute

\begin{equation}
(3.13)
\begin{aligned}
& E \left[ \sum_n \varphi_{n+1} \cdot e_{n+1} \mid \mathcal{F} \right] \text{ and } Var \left[ \sum_n \varphi_{n+1} \cdot e_{n+1} \mid \mathcal{F} \right],
\end{aligned}
\end{equation}

exactly and proceed as stated at the beginning of this section. However, trying to develop computable expressions from (3.12) has two main disadvantages: i) it may lead to very computationally demanding procedures, especially for systems with many reaction channels or in regimes with high activity; ii) it may be affected by the variance associated with the randomness in $\mathcal{F}_n$ and $\mathcal{G}_n$.

**Deriving a Formula for $\hat{V}_t$.** In this section, we derive the formula (3.3). Our goal is to find computable approximations of (3.13), where now, the underlying sigma algebra, $\mathcal{F}$, is just the information gathered by observing the coarse path, $\bar{X}$. This means that our formula should not depend explicitly on the knowledge of the random variables $\{\mathcal{F}_n, \mathcal{G}_n\}$. At this point, it is important to recall the comments in Section 3.3, that is, the sequence $(\varphi_n(\bar{\omega}))_{n=1}^{N(\bar{\omega})}$ is measurable with respect $\mathcal{F}$. This implies that, for all $n$, $\varphi_{n+1}$ is independent of $\mathcal{G}_n$. Hereafter, for notational convenience, we omit writing explicitly the conditioning on $\mathcal{F}$ in our formulae.

It turns out that, the leading order terms of the conditional moments obtained from (3.12) are essentially the same of those computed from (3.11). We will then derive (3.3) from (3.11). Using the notation from Section 3.3, we have that

\begin{equation}
(\varphi_{n+1} \cdot e_{n+1}) = \sum_j f_{j,n} \left( R_{j,n}^{n} \left( a_{j,n} \frac{\Delta t_n}{2} \right) \right) 1_{\{\Delta a_{j,n} \geq 0\}} - \mathcal{P}_n^{n} \left( -a_{j,n} \frac{\Delta t_n}{2} \right) 1_{\{\Delta a_{j,n} < 0\}}.
\end{equation}
By the tower property, we obtain
\[
E[(\varphi_{n+1} \cdot e_{n+1})] = E[E[(\varphi_{n+1} \cdot e_{n+1}) | G_n]] = \frac{\Delta t_n}{2} \sum_j f_{j,n} E[\Delta a_{j,n}].
\]

Now let us consider the first-order Taylor expansion:
\[
\Delta a_{j,n} := a_j(x_n + \sum_i \nu_i Q_{i,n}(a_i(x_n) \Delta t_n/2)) - a_j(x_n)
\]
\[
\approx (\nabla a_j(x_n) \cdot \sum_i \nu_i Q_{i,n}(a_i(x_n) \Delta t_n/2))
\]
\[
= \sum_i (\nabla a_j(x_n) \cdot \nu_i) Q_{i,n}(a_i(x_n) \Delta t_n/2).
\]

Since \(Q_{i,n}(a_i(x_n)\Delta t_n/2) \sim \text{Poisson}(a_i(x_n)\Delta t_n/2)\), we have that \(E[\Delta a_{j,n}] = \mu_{j,n}\) and \(\text{Var}[\Delta a_{j,n}] = \sigma_{j,n}^2\). Thus,
\[
E[(\varphi_{n+1} \cdot e_{n+1})] \approx \frac{\Delta t_n}{2} \sum_j f_{j,n} \mu_{j,n}.
\]

Now, we use again the tower property for the variance,
\[
\text{Var}[(\varphi_{n+1} \cdot e_{n+1})] = \text{Var}[E[(\varphi_{n+1} \cdot e_{n+1}) | G_n]] + E[\text{Var}[(\varphi_{n+1} \cdot e_{n+1}) | G_n]].
\]
It is immediate to obtain
\[
\text{Var}[E[(\varphi_{n+1} \cdot e_{n+1}) | G_n]] \approx \frac{(\Delta t_n)^3}{8} \sum_{j,j'} f_{j,n} f_{j',n} \sum_i (\nabla a_j(x_n) \cdot \nu_i)(\nabla a_{j'}(x_n) \cdot \nu_i) a_i(x_n),
\]
\[
E[\text{Var}[(\varphi_{n+1} \cdot e_{n+1}) | G_n]] \approx \frac{\Delta t_n}{2} \sum_j f_{j,n}^2 E[\text{sgn}(\Delta a_{j,n})].
\]

Let us consider the case where \(a_i(x_n)\Delta t_n/2\) is large enough for all \(i\). It is well known, that a Poisson random variable, \(Q(\lambda)\), is well approximated by a Gaussian random variable, \(N(\lambda, \lambda)\), for moderate values of \(\lambda\), say \(\lambda > 10\). Since \(Q_{i,n}(a_i(x_n)\Delta t_n/2) \sim \text{Poisson}(a_i(x_n)\Delta t_n/2)\), we have that, when \(a_i(x_n)\Delta t_n/2\) is large enough for all \(i\), \(\Delta a_{j,n} \approx N(\mu_{j,n}, \sigma_{j,n}^2)\). Consider a Gaussian random variable \(Z\) with parameters \(\mu\) and \(\sigma^2 > 0\), then,
\[
(3.14) \quad E[(\mu + \sigma Z)1_{(\mu + \sigma Z > 0)}] = \mu P(\mu + \sigma Z > 0) + \frac{\sigma}{\sqrt{2\pi}} \int_{-\mu/\sigma}^{+\infty} z \exp (-z^2/2) dz
\]
\[
= \mu(1 - \Phi(-\mu/\sigma)) + \frac{\sigma}{\sqrt{2\pi}} \exp (-(-\mu/\sigma)^2/2).
\]

From (3.14), it is immediate to get
\[
(3.15) \quad E[\Delta a_{j,n}1_{(\Delta a_{j,n} > 0)}] \approx \mu_{j,n} (1 - p_{j,n}) + \frac{\sigma_{j,n}}{\sqrt{2\pi}} \exp \left(-\frac{q_{j,n}^2}{2}\right),
\]
\[
E[\Delta a_{j,n}1_{(\Delta a_{j,n} < 0)}] \approx \mu_{j,n} p_{j,n} - \frac{\sigma_{j,n}}{\sqrt{2\pi}} \exp \left(-\frac{q_{j,n}^2}{2}\right).
\]
By subtracting the expressions in (3.15), we obtain

\begin{equation}
E \left[ \text{Var} \left[ (\varphi_{n+1} \cdot e_{n+1}) \mid G_{n} \right] \right] \approx \frac{\Delta t_{n}}{2} \sum_{j} f_{j,n}^{2} \left( \hat{\mu}_{j,n} + \hat{\sigma}_{j,n} \right).
\end{equation}

Let us consider now the case where \( a_{i}(x_{n})\Delta t_{n}/2 \) is close to zero for some \( i \). We can bound the expression \( E \left[ |\Delta a_{i,j,n}| \right] \) by \( E \left[ |\Delta a_{i,j,n}| \right] \) and also \( \sqrt{E \left[ (\Delta a_{j,n})^{2} \right]} \). It is easy to see that \( E \left[ |\Delta a_{j,n}| \right] \leq \hat{\mu}_{j,n} \). Regarding \( E \left[ (\Delta a_{j,n})^{2} \right] \), it can be approximated by

\[
E \left[ \sum_{i,i'} (\nabla a_{j}(x_{n}) \cdot \nu_{i}) (\nabla a_{j}(x_{n}) \cdot \nu_{i'}) \mathbb{Q}_{i} \mathbb{Q}_{i'} \right] = \sum_{i,i'} (\nabla a_{j}(x_{n}) \cdot \nu_{i}) (\nabla a_{j}(x_{n}) \cdot \nu_{i'}) E \left[ \mathbb{Q}_{i} \mathbb{Q}_{i'} \right].
\]

Since

\begin{equation}
E \left[ \mathbb{Q}_{i} \mathbb{Q}_{i'} \right] = \frac{(\Delta t_{n})^{2}}{4} a_{i}(x_{n})a_{i'}(x_{n})1_{i \neq i'} + \left( a_{i}(x_{n}) \frac{\Delta t_{n}}{2} + \left( a_{i}(x_{n}) \frac{\Delta t_{n}}{2} \right)^{2} \right) 1_{i = i'},
\end{equation}

we can rearrange terms and approximate \( E \left[ (\Delta a_{j,n})^{2} \right] \) by \( \mu_{j,n}^{2} + \sigma_{j,n}^{2} \).

We conclude that \( E \left[ |\Delta a_{i,j,n}| \right] \) can be bounded by \( m_{j,n} \), which has been defined as \( \min \{ \mu_{j,n}, \sqrt{\mu_{j,n}^{2} + \sigma_{j,n}^{2}} \} \).

REMARK 3.1. Formula (3.3) can be considered as an initial, relatively successful attempt to estimate \( V_{\ell} \), but there is still room for improvement. The main problem is the lack of sharp concentration inequalities for linear combinations of Poisson random variables. With the numerical examples, we show that the efficiency index of the formula is acceptable for our estimation purposes.

REMARK 3.2. We are assuming that only tau-leap steps are taken, but in our hybrid algorithms, some steps can be exact, and, hence, do not contribute to the local error. For that reason, we include the indicator function of the tau-leap step, \( 1_{TL} \), into the estimator, \( \hat{V}_{\ell} \).

REMARK 3.3. The dual-weighted residual approach makes the estimation of \( V_{\ell} \) feasible. It avoids the following high-kurtosis problem: the difference \( g_{\ell} - g_{\ell-1} \) is concentrated near zero but there is a positive probability of observing relatively high values of this quantity. This phenomenon makes the direct estimation of \( V_{\ell} \) difficult, especially for large values of \( \ell \). In our numerical experiments, we found that the variance of \( \mathcal{V}_{\ell} \) is much smaller than the variance of \( g_{\ell} - g_{\ell-1} \) estimated by a standard Monte Carlo estimator for the same number of simulated coupled hybrid paths. Note that \( \mathcal{V}_{\ell} \) can be computed using only single-level hybrid paths at level \( \ell-1 \).

4. Estimation Procedure. In this section, we present a procedure that estimates \( E[\varphi(X(T))] \) within a given prescribed relative tolerance, \( TOL > 0 \), with high probability. The process contains three phases:

**Phase I** Calibration of virtual machine-dependent quantities.

**Phase II** Solution of the work optimization problem: we obtain the total number of levels, \( L \), and the sequences \( (\delta_{l})_{l=0}^{L} \) and \( (M_{l})_{l=0}^{L} \), \( i.e., \) the one-step exit probability bounds and the required number of simulations at each level. We recall that in Section 1.6, we defined \( h_{\ell} := h_{0}R^{-\ell} \), where \( R > 1 \) is a given integer constant. For that reason, to define the whole sequence of meshes, \( (h_{\ell})_{\ell=0}^{L} \), we just need to define the size of the coarsest mesh, \( h_{0} \).

**Phase III** Estimation of \( E[\varphi(X(T))] \).
4.1. Phase I. In this section, we describe the estimation of several constants: $C_1$, $C_2$, $C_3$ and $K_1$, and functions: $C_P$ and $K_2$, that allow us to model the expected computational work (or just work), measured in terms of runtime of hybrid paths, see definitions (4.1) and (4.2). Those quantities are virtual machine dependent; that is, they are dependent on the computer system used for running the simulations and also on the implementation language. Those quantities are also off-line estimated; that is, we need to estimate them only once for each virtual machine on which we want to run the hybrid method.

Constants $C_1$, $C_2$, and $C_3$ reflect the average execution times of each logical path of Algorithm 2. We have that $C_1$ and $C_2$ reflect the work associated with the two different types of steps in the MNRM. Constant $C_3$ reflects the work needed for computing the Chernoff tau-leap size, $\tau_{CH}$. Finally, when we perform a tau-leap step, we have the work needed for simulating Poisson random variates, which is modeled by the function $C_P$ [19]. This function has two constants that are also virtual machine dependent.

The constant, $K_1$, and the function, $K_2 \equiv K_2(x, \delta)$, defined through $C_1$, $C_2$, and $C_3$, were introduced in Section 1.5.

4.2. Phase II. In this section, we set and solve the work optimization problem. Our objective function is the expected total work of the MLMC estimator, $M_L$, defined in (3.1), i.e.,

$$\sum_{\ell=0}^{L} \psi_{\ell}M_{\ell},$$

where $L$ is the maximum level (deepest level), $\psi_0$ is the expected work of a single-level path at level 0, and $\psi_{\ell}$, for $\ell \geq 1$, is the expected computational work of two coupled paths at levels $\ell-1$ and $\ell$. Finally, $M_0$ is the number of single-level paths at level 0, and $M_{\ell}$, for $\ell \geq 1$, is the number of coupled paths at levels $\ell-1$ and $\ell$.

Let us now describe in detail the quantities, $(\psi_{\ell})_{\ell=0}^{L}$. For $\ell=0$, Algorithm 12 generates a single hybrid path. The building block of a single hybrid path is Algorithm 2, which adaptively determines whether to use an MNRM step or a tau-leap one. According to this algorithm, there are two ways of taking an MNRM step, depending on the logical conditions, $K_1/a_0(x)>T_0-t$ and $K_2/a_0(x)>(\tau_{CH}$. Given one particular hybrid path, let $N_{K1}(h_0, \delta_0)$ be the number of MNRM steps such that $K_1/a_0(x)>T_0-t$ is true, and let $N_{K2}(h_0, \delta_0)$ be the number of MNRM steps such that $K_1/a_0(x)>T_0-t$ is false and $K_2/a_0(x)>(\tau_{CH}$ is true. When a Chernoff tau-leap step is taken, we have constant work, $C_3$, and variable work computed with the aid of $C_P$. Then, the expected work of a single hybrid path, at level $\ell = 0$, is

$$\psi_0 := C_1 E[N_{K1}(h_0, \delta_0)] + C_2 E[N_{K2}(h_0, \delta_0)] + C_3 E[N_{TL}(h_0, \delta_0)]$$

$$+ \sum_{j=1}^{J} E \left[ \int_{[0,T]} C_P(a_j(X_0(s))\tau_{CH}(X_0(s), \delta_0))1_{TL}(X_0(s))ds \right],$$

where $h_0$ is the size of the time mesh at level 0 and $\delta_0$ is the exit probability bound at level 0. Therefore, the expected work at level 0 is $\Psi_0M_0$, where $M_0$ is the total number of single hybrid paths.

For $\ell \geq 1$, we use the Algorithm 3 to generate $M_{\ell}$-coupled paths that couple the $\ell-1$ and $\ell$ levels. Given two coupled paths, let $N_{K1}(h_{\ell-1}, \delta_{\ell-1})$ and $N_{K1}(h_\ell, \delta_\ell)$ be the number of exact steps for level $\ell-1$ (coarse mesh) and $\ell$ (fine mesh), respectively,
with associated work $C_1$. We define $N_{K2}(h_{\ell-1}, \delta_{\ell-1})$ and $N_{K2}(h_{\ell}, \delta_{\ell})$ analogously. Then, the expected work of a pair of coupled hybrid paths at levels $\ell$ and $\ell-1$ is

\begin{equation}
\psi_{\ell} := C_1 E \left[ N_{K1}^{(c)}(\ell) \right] + C_2 E \left[ N_{K2}^{(c)}(\ell) \right] + C_3 E \left[ N_{TL}^{(c)}(\ell) \right] \\
+ \sum_{j=1}^{J} \int_{[0,T]} \mathcal{C}_P(a_j(\bar{X}_\ell(s))\tau_{Ch}(\bar{X}_\ell(s), \delta_{\ell})) 1_{TL}(\bar{X}_\ell(s))ds \\
+ \sum_{j=1}^{J} \int_{[0,T]} \mathcal{C}_P(a_j(\bar{X}_{\ell-1}(s))\tau_{Ch}(\bar{X}_{\ell-1}(s), \delta_{\ell-1})) 1_{TL}(\bar{X}_{\ell-1}(s))ds,
\end{equation}

where

$N_{K1}^{(c)}(\ell) := N_{K1}(h_{\ell}, \delta_{\ell}) + N_{K1}(h_{\ell-1}, \delta_{\ell-1})$

$N_{K2}^{(c)}(\ell) := N_{K2}(h_{\ell}, \delta_{\ell}) + N_{K2}(h_{\ell-1}, \delta_{\ell-1})$

$N_{TL}^{(c)}(\ell) := N_{TL}(h_{\ell}, \delta_{\ell}) + N_{TL}(h_{\ell-1}, \delta_{\ell-1})$

Now, recalling the definitions of the error decomposition, given at the beginning of Section 3.2, we have all the elements to formulate the work optimization problem. Given a relative tolerance, $TOL>0$, we solve

\begin{equation}
\begin{cases}
\min_{(h_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}
\end{equation}

It is natural to consider the following family of auxiliary problems indexed on $L \geq 1$, where we assume for now that the double sequence $(h_{\ell}, \delta_{\ell})_{\ell=0}^{L}$ is known:

\begin{equation}
\begin{cases}
\min_{(M_{\ell} \geq 1)_{\ell=0}^{L}} \sum_{\ell=0}^{L} \psi_{\ell} M_{\ell} \\
\text{s.t.} \\
\mathcal{E}_{I,L} + C_A \sqrt{\sum_{\ell=0}^{L} \frac{\psi_{\ell}}{M_{\ell}}} \leq TOL - TOL^2,
\end{cases}
\end{equation}

where we have $C_A \geq 2$ to guarantee an asymptotic confidence level of at least 95%.

Let us assume for now that we know $\psi_{\ell}$, $V_{\ell}$ and $\mathcal{E}_{I,\ell}$, for $\ell = 0, 1, \ldots, L$. Let $L_0$ be the smallest value of $L$ such that $\mathcal{E}_{I,L} < TOL - TOL^2$. This value exists and it is finite since the discretization error, $\mathcal{E}_{I,L}$ tends to zero, as $L$ goes to infinity. For each $L \geq L_0$, define $w_{L} := \sum_{\ell=0}^{L} \psi_{\ell} M_{\ell}^*$, where the sequence $(M_{\ell}^*)_{\ell=0}^{L}$ is the solution of the problem (4.4). It is worth mentioning that $(M_{\ell}^*)_{\ell=0}^{L}$ is quickly obtained as the solution of the following Karush-Kuhn-Tucker problem (see, e.g., [18]):

\begin{equation}
\begin{cases}
\min_{(M_{\ell} \geq 1)_{\ell=0}^{L}} \sum_{\ell=0}^{L} \psi_{\ell} M_{\ell} \\
\text{s.t.} \\
\sum_{\ell=0}^{L} \frac{\psi_{\ell}}{M_{\ell}} \leq R
\end{cases}
\end{equation}

We do not develop here all the calculations, but a pseudo code is given in Algorithm 11.

Let us now analyze two extreme cases: i) for $L$ such that $\mathcal{E}_{I,L}$ is less but very close to $TOL - TOL^2$, we have that $\sum_{\ell=0}^{L} V_{\ell}/M_{\ell}^*$ is a very small number. As a consequence, we obtain large values of $M_{\ell}^*$ and, hence, a large value of $w_{L}$. By adding one more
level, i.e., \(L \leftarrow L + 1\), we expect a larger gap between \(E_{L,L}\) and \(TOL_{0}\); that means that we expect a larger value of \(\sum_{\ell=0}^{L} V_{\ell}/M_{\ell}^{*}\) that may lead to smaller values of \(M_{\ell}^{*}\). We observe that, in spite of adding one more term to \(w_{L}\), this leads to a smaller value of \(w_{L}\). ii) At the other extreme, a large value of \(L\) is associated with large values of \(\psi_{L}\) and therefore with large values of \(w_{L}\).

This informal ‘extreme case analysis’ has been confirmed by our numerical experiments (see, for example, Figures 5.2 and 5.7 (lower-right)) which allow us to conjecture that the sequence \((w_{L})_{L=0}^{+\infty}\) is a convex function of \(L\) and, hence, that it has a unique optimal value achieved at a certain \(L^{*}\). A pseudo algorithm to find \(L^{*}\) could be to start computing \(w_{L_{0}}\) and \(w_{L_{0} + 1}\). If \(w_{L_{0} + 1} \geq w_{L_{0}}\), we accept \(L^{*} = L_{0}\); otherwise, we proceed to computing the next term of the sequence, \((w_{L})_{L=0}^{+\infty}\). If, for some \(p\), we have \(w_{L_{0} + p} \geq w_{L_{p}}\), we accept \(L^{*} = L_{p}\). Of course, we can stop even if \(w_{L_{p+1}} < w_{L_{p}}\), but the difference \(|w_{L_{p+1}} - w_{L_{p}}|\) is sufficiently small. In this last case, we accept \(L^{*} = L_{p+1}\).

### 4.2.1. Computational Complexity.

At this point, we have all the necessary elements to establish a key point of this work, the computational complexity of the multilevel hybrid Chernoff tau-leap method.

Let us now analyze the optimal amount of work at level \(L\), \(w_{L}\), as a function of the given relative tolerance, \(TOL\). For simplicity, let us assume that \(M_{\ell}^{*} > 1, \ell = 0, \ldots, L\). In this case, the optimal number of samples at level \(\ell\) is given by

\[
M_{\ell}^{*} = \left(\frac{C_{A}}{\theta}\right)^{2} TOL^{-2} \sqrt{V_{\ell}/\psi_{\ell}} \sum_{\ell=0}^{L} \sqrt{V_{\ell} \psi_{\ell}},
\]

for some \(\theta \in (0, 1)\). In fact, \(\theta\) is the proportion of the tolerance, \(TOL\), that our cost optimization algorithm selects for the statistical error, \(E_{S,L}\). In our numerical experiments \(\theta\) is always larger than 0.5 (see Figures 5.3 and 5.8).

By substituting \(M_{\ell}^{*}\) into the total work formula, \(w_{L}\), we conclude that the optimal work is given by

\[
w_{L}^{*}(TOL) = \left(\frac{C_{A}}{\theta}\right)^{2} \left( \sum_{\ell=0}^{L} \sqrt{V_{\ell} \psi_{\ell}} \right)^{2} TOL^{-2}.
\]

Let us consider the sum \(\sum_{\ell=0}^{L} \sqrt{V_{\ell} \psi_{\ell}}\). The expected computational work per path at level \(\ell\), \(\psi_{\ell}\), is bounded by a \(K \psi_{MNRM}\), i.e., a multiple of the expected computational work of the MNRM (see Section 1.2). In our numerical experiments, we observe that \(K\) is around three. Therefore, \(\sum_{\ell=0}^{L} \sqrt{V_{\ell} \psi_{\ell}} \leq \sqrt{K} \psi_{MNRM} \sum_{\ell=0}^{L} \sqrt{V_{\ell}}\). Now observe that for sufficiently large \(\ell\), say \(\ell \geq \ell_{0}\), we have that \(V_{\ell} = O(h^{q})\), with \(q > 2\). This is because as \(\ell\) goes to \(+\infty\), the hybrid paths become exact. Then, the series \(\sum_{\ell=\ell_{0}+1}^{\ell} V_{\ell}^{q/2}\) converges. We conclude that \(\sup_{L} \{\sum_{\ell=0}^{L} \sqrt{V_{\ell} \psi_{\ell}}\}\) is bounded and, therefore, the expected computational complexity of the multilevel hybrid Chernoff tau-leap method is \(w_{L}^{*}(TOL) = O(TOL^{-2})\).

### 4.2.2. Some Comments on the Algorithms for Phase II.

In Algorithm 7, we propose an iterative method to obtain an approximate solution to the problem (4.3). Notice that we are assuming that there are at least two levels in the multilevel hierarchy, i.e., \(L \geq 1\).

To solve the problem (4.3), we bound the global exit error, \(E_{E,L}\), by \(TOL^{2}\). More specifically, we choose \(\delta_{L}\) to be sufficiently small such that

\[
|A \langle g_{L}; \cdot \rangle | \delta_{L} |A(N_{\text{TL}}(h_{L}, \delta_{L}); \cdot \rangle < TOL^{2},
\]
At this point, it is crucial to observe that, condition (4.6), when $\ell< L$, artificially enforces a dependence on $TOL$ that may result in very small values of $\delta\ell$. This could affect the expected number of exact steps and tau-leap steps at level $\ell$. In the appendix of [19], we proved that, when $\delta\ell$ tends to zero, the expected values of the number of tau-leap steps at level $\ell$ goes to zero, and therefore our hybrid MLMC strategy would converge to the SSA method without the desired reduction in the computational work. In order to avoid the dependence of $(\delta\ell)_{\ell=0}^{L-1}$ on $TOL$, we adopt a different strategy based on the following decomposition:

\[
\mathcal{V}_\ell = \text{Var} \left[ g_\ell 1_{A_\ell} - g_{\ell-1} 1_{A_{\ell-1}} \right] = \text{Var} \left[ g_\ell - g_{\ell-1} \mid A_\ell \cap A_{\ell-1} \right] P (A_\ell \cap A_{\ell-1}) \\
+ \text{Var} \left[ g_\ell \mid A_\ell \cap A^c_{\ell-1} \right] P (A_\ell \cap A^c_{\ell-1}) \\
+ \text{Var} \left[ g_{\ell-1} \mid A^c_\ell \cap A_{\ell-1} \right] P (A^c_\ell \cap A_{\ell-1}).
\]

We impose that the first term of the right-hand side dominates the other two. This is because the conditional variances appearing in the last two terms are of order $O(1)$, while the conditional variance appearing in the first term is of order $O(h\ell)$, and we make our computations with approximations of $\mathcal{V}_\ell$ assuming that $P (A_\ell \cap A_{\ell-1})$ is close to one. We proceed as follows: first, we approximate $P (A_\ell \cap A_{\ell-1})$ by $P (A_\ell) P (A_{\ell-1})$, then, we consider $1 - \delta\ell A (N_{TL}(h\ell, \delta\ell); \cdot)$ as an approximate upper bound for $P (A_\ell)$ when $\delta\ell A (N_{TL}(h\ell, \delta\ell); \cdot) \ll 1$. Those considerations lead us to impose

\[
(4.7) \quad \text{Var} \left[ g_\ell - g_{\ell-1} \mid A_\ell \cap A_{\ell-1} \right] (1-\delta\ell A (N_{TL}(h\ell, \delta\ell); \cdot)) (1-\delta_{\ell-1} A (N_{TL}(h\ell-1, \delta_{\ell-1}); \cdot)) > \\
\text{Var} \left[ g_\ell \mid A_\ell \cap A^c_{\ell-1} \right] \delta_{\ell-1} A (N_{TL}(h\ell-1, \delta_{\ell-1}); \cdot) + \text{Var} \left[ g_{\ell-1} \mid A^c_\ell \cap A_{\ell-1} \right] \delta\ell A (N_{TL}(h\ell, \delta\ell); \cdot).
\]

To avoid simultaneous refinements on $\delta\ell$ and $\delta_{\ell-1}$, based on (4.7), we impose to $\delta\ell$ the following condition:

\[
\hat{\mathcal{V}}_\ell (1-\delta\ell A (N_{TL}(h\ell, \delta\ell); \cdot))^2 > 2 S^2 (g; \cdot) \delta\ell A (N_{TL}(h\ell, \delta\ell); \cdot).
\]

Remark 4.1 (Multilevel estimators used in Algorithm 7). Although in algorithm 7 we show that the estimations of $E[g(X(T))]$ and $\text{Var} [g(X(T))]$ are computed using the information of the last level only, in fact we are computing them using a multilevel estimator. We omit the details in the algorithm for the sake of simplicity. In the case of $E[g(X(T))]$, we use the standard multilevel estimator, and, for the case of $\text{Var} [g(X(T))]$, we use the following telescopic decomposition:

\[
\text{Var} [g(\hat{X}_l(T))] = \text{Var} [g(\hat{X}_0(T))] + \sum_{\ell=1}^{l} (\text{Var} [g(\hat{X}_\ell(T))] - \text{Var} [g(\hat{X}_{\ell-1}(T))]),
\]

where $l > 1$ is a fixed level. Using the usual variance estimators for each level, we obtain an unbiased multilevel estimator of the variance of $g(\hat{X})$. We refer to [5] for details.

Remark 4.2 (Coupled paths exiting the lattice, $\mathbb{Z}_d^d$). Algorithm 3 could compute four types of paths. It could happen that no approximate process (the coarse one, $X_{\ell-1}$, or the fine one, $X_\ell$) exits the lattice, which is the most common case. It could also happen that one of the approximate processes exits the lattice. And finally, both approximate processes could exit the lattice. The first case is the most common one
and no further explanation is required. We now explain the case that one of process
exists the lattice. Suppose the coarse one exited the lattice. In that case, until the fine
process reaches time \(T\) or exits the lattice, we still simulate the coupled process by
simulating only the fine path using the single-level hybrid algorithm, presented in [19].
If the fine path reaches \(T\), we have that \(1_{A_{t-1}} = 0\), and \(1_{A_t} = 1\). Vice versa, if the
fine process exits and the coarse one reaches \(T\), we have \(1_{A_{t-1}} = 1\), and \(1_{A_t} = 0\).

Remark 4.3 (Coupling with an exact path). Algorithm 7 uses a computational-
cost-based stopping criteria. That is, the algorithm stops refining the time mesh when
the estimated total computational cost of the multilevel estimator, \(\hat{W}_{ML} = \sum_{l=0}^L \hat{\psi}_l M_l\),
at level \(l\), is greater than the corresponding cost for level \(l-1\), and only when condition
\(\hat{E}_l < TOL - TOL^2\) is already satisfied. In that case, \(L^* = l-1\). The latter condition is
required for obtaining a solution of the optimization problem (4.5). In our experi-
ments, we observed that, the cost of two hybrid coupled paths, \(ψ_\ell\), may be greater than
the cost of “hybrid-exact” coupled paths, that is, a hybrid path at level \(l-1\) coupled
with an exact path at level \(l\). That kind of path, used only at the last level, leads to
the following unbiased multilevel estimator:

\[
\hat{M}_L := \frac{1}{M_0} \sum_{m_0=1}^{M_0} g_0 1_{A_0}(\omega_{m_0}) + \sum_{\ell=1}^{L-1} \frac{1}{M_\ell} \sum_{m_\ell=1}^{M_\ell} [g_\ell 1_{A_\ell} - g_{\ell-1} 1_{A_{\ell-1}}](\omega_{m_\ell}) + \frac{1}{M_L} \sum_{m_L=1}^{M_L} [g(X(T)) - g_{L-1} 1_{A_{L-1}}](\omega_{m_L}).
\]

Therefore, it is possible to add another stopping criteria to Algorithm 7 related to
the comparison between the estimated cost of two hybrid coupled paths and the cost of
hybrid-exact coupled paths. Please note that the condition \(δ_1 A(N_{TLL}; \cdot) A(g_{\ell}; \cdot) \leq TOL^2\)
trivially holds because \(A(N_{TLL}; \cdot)\) is zero. In the two examples analyzed, there are no
significant computational gains in the estimation phase by using that stopping rule
and its corresponding estimator. This unbiased estimator is inspired in the work of
Anderson and Higham, [1].

4.3. Phase III. From Phase II, we found that, to compute our multilevel Monte
Carlo estimator, \(M_L\), for a given tolerance, we have to run \(M_0^*\) single hybrid paths
with parameters \((h_0, \delta_0)\) and \(M_{\ell}^*\) coupled hybrid paths with parameters \((h_{\ell-1}, h_\ell, \delta_\ell)\)
for \(\ell = 1, 2, \ldots, L^*\). But, we will follow a slightly different strategy: we run half of
the required simulations and use them to update our estimations of the sequences
\((\hat{E}_\ell)_{\ell=0}^{L^*}, (\hat{\psi}_\ell)_{\ell=0}^{L^*}\), and \((\hat{\psi}_\ell)_{\ell=0}^{L^*}\). Then, we solve problem 4.4 again and re-calculate
the values of \(M_{\ell}^*\), for all \(\ell\). We proceed iteratively until convergence. In this way,
we take advantage of the information generated by new simulated paths and update
the estimations of the sequences of weak errors, costs, and variances, obtaining more
control over the total work of the method.

5. Numerical Examples. In this section, we present two examples to illus-
trate the performance of our proposed method, and we compare the results with
the single-level approach given in [19]. For benchmarking purposes, we use Gille-
spie’s Stochastic Simulation Algorithm (SSA) instead of the Modified Next Reaction
Method (MNRM), because the former is widely used in the literature.

5.1. A Simple Decay Model. The classical radioactive decay model provides
a simple and important example for the application of our method. This model has
only one species and one reaction,

\[ X \xrightarrow{c} \emptyset. \]

Its stoichiometric matrix, \( \nu \in \mathbb{R} \), and the propensity function, \( a : \mathbb{Z}_+ \to \mathbb{R} \), are given by

\[ \nu = -1 \quad \text{and} \quad a(X) = cX. \]

Here, we choose \( c = 1 \) and \( g(x) = x \). In this particularly simple example, we have that \( E[g(X(T))|X(t) = X_0] = X_0 \exp(-c(T-t)) \). Consider the initial condition \( X_0 = 10^5 \), and the final time, \( T = 0.5 \). In this case, the process starts relatively far from the boundary, i.e., it is a tau-leap dominated setting.

We now analyze an ensemble of 5 independent runs of the calibration algorithm (Algorithm 7), using different relative tolerances. In Figure 5.1, we show, in the left panel, the total predicted work (runtime) for the single-level hybrid method, for the multilevel hybrid method and for the SSA method, versus the estimated error bound. The multilevel method is preferred over the SSA and the single-level hybrid method for all the tolerances. We also show the estimated asymptotic work of the multilevel method. In the right panel, we show, for different tolerances, the actual work (runtime), using a 12 core Intel GLNXA64 architecture and MATLAB version R2012b.

In Table 5.1, we summarize an ensemble run of the calibration algorithm, where \( W_{ML} \) is the average actual computational work of the multilevel estimator (the sum of all the seconds taken to compute the estimation) and \( W_{SSA} \) is the corresponding average actual work of the SSA. We compare those values with the corresponding estimations, \( \hat{W}_{ML} \), and \( \hat{W}_{SSA} \). The difference between the estimated work and the actual one gets smaller as we refine the tolerance.
work, $\hat{\psi}_\ell$, increases as we refine the mesh. Observe that it increases more slowly than linear. This is because the work needed for generating Poisson random variables becomes smaller as we refine the time mesh. In the lower right panel, we show the total computational work, only in the cases in which $\hat{E}_{I,\ell} < TOL - TOL^2$.

Fig. 5.2. Upper left: estimated weak error, $\hat{E}_{I,\ell}$, as a function of the time mesh size, $h$. Upper right: estimated variance of the difference between two consecutive levels, $\hat{V}_\ell$, 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 4, it is well above 0.5 for all the tolerances. Right: $\sqrt{\hat{V}_\ell \hat{\psi}_\ell}$, as a function of $\ell$, 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.

25
In Figure 5.4, we show the main outputs of Algorithm 7, \( \delta_\ell \) and \( M_\ell \) for \( \ell = 0, \ldots, L^* \), for the smallest considered tolerance. In this example, \( L^* \) is 12 or 13, depending on the run of Algorithm 7. We observe that the number of realizations decreases slower than linear, from levels 1 to \( L^* - 1 \), until it reaches \( M_{L^*} = 1 \).

In the simulations, we observed that, as we refine \( TOL \), the optimal number of levels approximately increases logarithmically, which is a desirable feature. We fit the model \( L^* = a + b \log(TOL^{-1}) \), obtaining \( b = -2.28 \) and \( a = -8.43 \).

![Graphs showing \( \delta_\ell \) and \( M_\ell \) for different tolerances](image)

**Figure 5.4.** One-step exit probability bound, \( \delta_\ell \), and \( M_\ell \) for \( \ell = 0, 1, \ldots, L^* \), for the smallest tolerance.

| \( TOL \)    | \( L^* \)    | \( \frac{W_{ML}}{W_{SSA}} \) | \( \frac{W_{ML}}{W_{SSA}} \) |
|------------|-------------|----------------|----------------|
| 3.13e-03   | 4.8 ± 0.392 | 0.17 ± 0.036   | 0.12 ± 0.027   |
| 1.56e-03   | 6.0         | 0.09 ± 0.016   | 0.07 ± 0.011   |
| 7.81e-04   | 8.0         | 0.07 ± 0.009   | 0.05 ± 0.006   |
| 3.91e-04   | 9.0         | 0.07 ± 0.011   | 0.05 ± 0.007   |
| 1.95e-04   | 11.0        | 0.08 ± 0.010   | 0.06 ± 0.008   |
| 9.77e-05   | 12.2 ± 0.392| 0.08 ± 0.008   | 0.07 ± 0.007   |

**Table 5.1**

Details for the ensemble run of Algorithm 7, for the simple decay model with \( X_0 = 10^5 \) and \( T = 0.5 \). As an example, the second row of the table tells us that, for a tolerance \( TOL = 1.56 \cdot 10^{-3} \), 6 levels are needed on average. The predicted work of the multilevel hybrid method is, on average, 9% of the predicted work of the SSA method, whereas the actual one is 7%. Confidence intervals at 95% are also provided.

In Figure 5.5, we show the performance of the formula (3.3), implemented in Algorithm 10, used to estimate the strong error, \( V_\ell \). The quotient of \( V_\ell \) over a standard Monte Carlo estimate of \( V_\ell \) is almost 1 for the first ten levels. At levels 11 and 12, we obtain 0.99 and 0.91, respectively. Both quantities are estimated using a coefficient of variation less than 5%, but there is a remarkable difference in terms of computational work in favor of our dual-weighted estimator. In the same figure, we also 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 except the smallest one.

In the simulations, we observed that, as we refine \( TOL \), the optimal number of levels approximately increases logarithmically, which is a desirable feature. We fit the model \( L^* = a + b \log(TOL^{-1}) \), obtaining \( b = -2.28 \) and \( a = -8.43 \).
5.2. Gene Transcription and Translation [4]. This model has five reactions,

\[
\emptyset \xrightarrow{c_1} R, \quad R \xrightarrow{c_2} R + P \\
2P \xrightarrow{c_3} D, \quad R \xrightarrow{c_4} \emptyset \\
P \xrightarrow{c_5} \emptyset
\]

described respectively by the stoichiometric matrix and the propensity function

\[
\nu = \begin{pmatrix}
1 & 0 & 0 \\
0 & 1 & 0 \\
0 & -2 & 1 \\
-1 & 0 & 0 \\
0 & -1 & 0
\end{pmatrix}
\quad \text{and} \quad a(X) = \begin{pmatrix}
c_1 \\
c_2 R \\
c_3 P(P-1) \\
c_4 R \\
c_5 P
\end{pmatrix},
\]

where \( X(t) = (R(t), P(t), D(t)) \), and \( c_1=25, \ c_2=10^3, \ c_3=0.001, \ c_4=0.1, \) and \( c_5=1 \).

In the simulations, the initial condition is \( (0,0,0) \) and the final time is \( T=1 \). The observable is given by \( g(X) = D \). We observe that the abundance of the mRNA species, represented by \( R \), is close to zero for \( t \in [0,T] \). However, as we point out in [19], the reduced abundance of one of the species is not enough to ensure that the SSA method should be used.

We now analyze an ensemble of 5 independent runs of the calibration algorithm (Algorithm 7), using different relative tolerances. In Figure 5.6, we show, in the left panel, the total predicted work (runtime) for the single-level hybrid method, for the multilevel hybrid method and for the SSA method, versus the estimated error bound. We also show the estimated asymptotic work of the multilevel method. Again, the multilevel hybrid method beats the others and we remark that the computational work of the multilevel is of order \( O(TOL^{-2}) \).

In Figure 5.7, we can observe how the estimated weak error decrease linearly for the coarser time meshes, but, as we continue refining the time mesh, quickly decreases towards zero. In the case of the estimated variance, \( \hat{V}_t \), it decreases faster than linear, and also quickly decreases towards zero afterwards. This is a consequence of the transition from a hybrid regime to a pure exact one. The estimated total path work, \( \hat{\psi}_t \), increases sublinearly as we refine the mesh. Note that \( \hat{\psi}_t \) reaches a maximum,
Fig. 5.6. Left: Predicted work (runtime) versus the estimated error bound for the gene transcription and translation model. The hybrid method is preferred over the SSA for the first three tolerances only. The multilevel hybrid method is preferred over the SSA and the single-level method for all the tolerances. Right: Actual work (runtime) versus the estimated error bound.

which corresponds to a SSA-dominant regime. In the lower right panel we show the total computational work, only in the cases in which \( \hat{E}_{I,\ell} < TO\!L - TOL^2 \).

Fig. 5.7. Upper left: estimated weak error, \( \hat{E}_{I,\ell} \), as a function of the time mesh size, \( h \). Upper right: estimated variance of the difference between two consecutive levels, \( \hat{V}_\ell \), as a function of \( h \). Lower left: estimated path work, \( \hat{\psi}_\ell \), as a function of \( h \). Lower right: estimated total computational work, \( \sum_{\ell=0}^{L^*} \hat{\psi}_\ell M_\ell \), as a function of the level, \( L \).

In Figure 5.9, we show the main outputs of Algorithm 7, \( \delta_\ell \) and \( M_\ell \) for \( \ell = 0, ..., L^* \), for the smallest tolerance. We observe that the number of realizations de-
Fig. 5.8. Left: Percentage of the statistical error over the total error. As we mentioned in Section 4, it is well above 0.5 for all the tolerances. Right: $\sqrt{V_\ell \psi_\ell}$, as a function of $\ell$, for the smallest tolerance, which decreases as the level increases. Observe that the contribution of level 0 is almost equal to the sum of the other levels.

creases slower than linear, from levels 1 to 12.

Fig. 5.9. One-step exit probability bound, $\delta_\ell$, and $M_\ell$ for $\ell=0,1,...,L^*$, for the smallest tolerance, for the gene transcription and translation model.

| TOL   | $L^*$ | $\frac{\hat{W}_{ML}}{W_{SSA}}$ | $\frac{W_{ML}}{W_{SSA}}$ |
|-------|-------|--------------------------------|--------------------------|
| 1.00e-01 | 3.6 ±0.480 | 0.13 ±0.020 | 0.09 ±0.015 |
| 5.00e-02 | 5.2 ±0.392 | 0.13 ±0.012 | 0.08 ±0.010 |
| 2.50e-02 | 7.0 | 0.11 ±0.018 | 0.09 ±0.009 |
| 1.25e-02 | 8.6 ±0.480 | 0.11 ±0.009 | 0.08 ±0.003 |
| 6.25e-03 | 10.0 | 0.11 ±0.004 | 0.08 ±0.006 |
| 3.13e-03 | 11.2 ±0.392 | 0.12 ±0.012 | 0.09 ±0.009 |

Table 5.2
Details for the ensemble run of Algorithm 7 for the gene transcription and translation model.

In Figure 5.10, we see that our dual-weighted estimator of the strong error, $V_\ell$, gives essentially the same results as the standard Monte Carlo estimator, but with much less computational work. In this case, an accurately empirical estimate of $V_\ell$
took almost 48 hours, but the computation of $\hat{V}_7$ just took some minutes. In the same figure we also show TOL versus the actual computational error. It can be seen that the prescribed tolerance is achieved, except for the second smallest tolerance, with the required confidence of 95%, since $C_A=1.96$.

![Graph showing variances and TOL vs. Total error](image)

Fig. 5.10. Left: performance of the formula (3.3) as a strong error estimate. Right: 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 closely with the expected confidence of 95%.

In the simulations, we observed that, as we refine TOL, the optimal number of levels approximately increases logarithmically, which is a desirable feature. We fit the model $L^* = a + b \log(TOL^{-1})$, obtaining $b=-2.27$ and $a=-1.25$.

**Remark 5.1.** The savings in computational work when generating Poisson random variables, heavily depend on MATLAB’s performance capabilities. For example, we do not generate the random variates in batches, as in [1], and that could have an impact on the results. In fact, we should expect better results from our method if we implement our algorithms in more performance oriented languages or if we sample Poisson random variables in batches.

6. **Conclusions.** In this work, we developed a multilevel Monte Carlo version for the single-level hybrid Chernoff tau-leap algorithm presented in [19]. We showed that the computational complexity of this method is of order $O(TOL^{-2})$, and therefore, it can be seen as a variance reduction of the SSA method, which has the same complexity. This represents an important advantage of the hybrid tau-leap with respect to the pure tau-leap in the multilevel context. In our numerical examples, we obtained substantial gains with respect to both the SSA and the single-level hybrid Chernoff tau-leap. The present approach, as the one in [19], also provides an approximation of $E[g(X(T))]$ with a prescribed accuracy and a confidence level, at nearly optimal computational work. For reaching this optimality, we derived novel formulas based on dual-weighted residual estimations for computing the variance of the difference of the observables between two consecutive levels in coupled hybrid paths. Future extensions may involve better hybridization techniques as well as implicit and higher order versions of the hybrid MLMC.

**Acknowledgments.** The authors are members of the KAUST SRI Center for Uncertainty Quantification in the Computer, Electrical and Mathematical Sciences and Engineering Division at King Abdullah University of Science and Technology (KAUST).
REFERENCES

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

[2] D. Anderson, D. Higham, and Y. Sun. Complexity of multilevel monte carlo tau-leaping. *arXiv:1310.2676v1*, 2013.

[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] D. F. Anderson and M. Koyama. Weak error analysis of numerical methods for stochastic models of population processes. *SIAM Multiscale Modeling and Simulation*, 10(4):1493–1524, 2012.

[5] C. Bierig and A. Chernov. Convergence analysis of multilevel variance estimators in multilevel monte carlo methods and application for random obstacle problems. Preprint 1309, Institute for Numerical Simulation, University of Bonn, 2013. Submitted.

[6] N. Collier, A.-L. Haji-Ali, F. Nobile, E. von Schwerin, and R. Tempone. A continuation multilevel monte carlo algorithm. *Mathematics Institute of Computational Science and Engineering, Technical report Nr. 10-2014*, EPFL, 2014.

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

[8] 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.

[9] M. Giles. Multi-level Monte Carlo path simulation. *Operations Research*, 53(3):607–617, 2008.

[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] S. Heinrich. Monte carlo complexity of global solution of integral equations. *Journal of Complexity*, 14(2):151–175, 1998.

[13] S. Heinrich. Multilevel monte carlo methods. In *Large-Scale Scientific Computing*, volume 2179 of Lecture Notes in Computer Science, pages 58–67. Springer Berlin Heidelberg, 2001.

[14] J. Karlsson and R. Tempone. Towards automatic global error control: Computable weak error expansion for the tau-leap method. *Monte Carlo Methods and Applications*, 17(3):233–278, March 2011.

[15] T. G. Kurtz. Representation and approximation of counting processes. *Adv. Filtering Optimal Stochastic Control 42*, 1982.

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

[17] T. Li. Analysis of explicit tau-leaping schemes for simulating chemically reacting systems. *Multiscale Modeling and Simulation*, 6(2):417–436, 2007.

[18] D. G. Luenberger and Y. Ye. *Linear and Nonlinear Programming (International Series in Operations Research and Management Science)*. Springer, 2010.

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

[20] A. Speight. A multilevel approach to control variates. *Journal of Computational Finance*, 12:1–25, 2009.
Algorithm 3 Coupled hybrid path. Inputs: the initial state, \( X(0) \), the final time, \( T \), the propensity functions, \( a=(a_j)_{j=1}^l \), the stoichiometric vectors, \( \nu=(\nu_j)_{j=1}^l \), two one-step exit probability bounds; one for the coarse level, \( \delta \), and another for the fine level, \( \tilde{\delta} \), and two time meshes, one coarse \( (t_k)_{k=0}^K \), such that \( t_K=T \) and a finer one, \( (s_l)_{l=0}^{K'} \), such that \( s_0=t_0 \), \( s_{K'}=t_K \), and \( (t_k)_{k=0}^K \subset (s_l)_{l=0}^{K'} \). 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 \( (\check{X}(s_l))_{l=0}^{K'} \subset \mathbb{Z}_+^d \), such that \( \check{X}(t_K) \in \mathbb{Z}_+^d \) or \( \check{X}(s_{K'}) \in \mathbb{Z}_+^d \). If \( t_K < T \), both paths exited 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 \( \check{X}(t_k) \) and \( \check{X}(s_l) \) from the current state variables \( \bar{X} \) and \( \check{X} \), respectively, and the counting of the number of steps.

1: \( t \leftarrow 0 \), \( \bar{X} \leftarrow X(0) \), \( \check{X} \leftarrow X(0) \)
2: \( \bar{t} \leftarrow \) next grid point in the coarse grid larger than \( t \)
3: \((\bar{H},\bar{m},\bar{a}) \leftarrow \text{Algorithm 4 with } (\bar{X},\bar{t},\bar{t},T,\bar{\delta},\bar{a})\)
4: \( \check{t} \leftarrow \) next grid point in the fine grid larger than \( t \)
5: \((\check{H},\check{m},\check{a}) \leftarrow \text{Algorithm 4 with } (\check{X},\check{t},\check{t},T,\check{\delta},\check{a})\)
6: \textbf{while } \( t < T \) \textbf{ do}
7: \quad \( H \leftarrow \min\{\bar{H},\check{H}\} \)
8: \quad \textbf{if } \bar{m} = \text{TL} \text{ and } \check{m} = \text{TL} \text{ then}
9: \quad \quad S \leftarrow \text{Algorithm 6 with } (\bar{a},\check{a})
10: \quad \quad \Lambda \leftarrow P(S,(H-t)) \text{ (generate Poisson random variates)}
11: \quad \quad \bar{X} \leftarrow \bar{X} + (\Lambda_1+\Lambda_2)\nu
12: \quad \quad \check{X} \leftarrow \check{X} + (\Lambda_1+\Lambda_3)\nu
13: \quad \quad t \leftarrow H
14: \quad \textbf{else}
15: \quad \quad \text{Initialize internal clocks } R,P \text{ if needed (see Algorithm 1)}
16: \quad \quad \textbf{while } \( t < H \) \textbf{ do}
17: \quad \quad \quad \textbf{if } \bar{m} = \text{MNRM} \text{ then}
18: \quad \quad \quad \quad \bar{a} \leftarrow a(\bar{X})
19: \quad \quad \quad \textbf{end if}
20: \quad \quad \quad \textbf{if } \check{m} = \text{MNRM} \text{ then}
21: \quad \quad \quad \quad \check{a} \leftarrow a(\check{X})
22: \quad \quad \quad \textbf{end if}
23: \quad \quad \quad S \leftarrow \text{Algorithm 6 with } (\bar{a},\check{a})
24: \quad \quad \quad (t,\bar{X},\check{X},R,P) \leftarrow \text{Algorithm 5 with } (t,H,\bar{X},\check{X},R,P,S)
25: \quad \quad \textbf{end while}
26: \quad \textbf{end if}
27: \quad \textbf{if } t < T \text{ then}
28: \quad \quad \textbf{if } H \leq \bar{H} \text{ then}
29: \quad \quad \quad \bar{t} \leftarrow \text{next grid point in the coarse grid larger than } t
30: \quad \quad \quad (\bar{H},\bar{m},\bar{a}) \leftarrow \text{Algorithm 4 with } (\bar{X},\bar{t},\bar{t},T,\bar{\delta},\bar{a})
31: \quad \quad \textbf{end if}
32: \quad \quad \textbf{if } H \geq \check{H} \text{ then}
33: \quad \quad \quad \check{t} \leftarrow \text{next grid point in the fine grid larger than } t
34: \quad \quad \quad (\check{H},\check{m},\check{a}) \leftarrow \text{Algorithm 4 with } (\check{X},\check{t},\check{t},T,\check{\delta},\check{a})
35: \quad \quad \textbf{end if}
36: \quad \textbf{end if}
37: \textbf{end while}
Algorithm 4 Compute next time horizon. Inputs: the current state, $X$, 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 selected method $m$, current propensity values $\tilde{a}$.

1: $\tilde{a} \leftarrow a(X)$
2: $(m, \tilde{\tau}) \leftarrow$ Algorithm 2 with $(X, t, \tilde{a}, \tilde{\delta}, \tilde{t})$
3: if $\tilde{m} = \text{TL}$ then
4: $H \leftarrow \min\{\tilde{t}, t+\tilde{\tau}, T\}$
5: else
6: $H \leftarrow \min\{t+\tilde{\tau}, T\}$
7: end if

Algorithm 5 Auxiliary function used in Algorithm 3. Inputs: the current time, $t$, the current time horizon, $H$, the current system state at coarser level, $\bar{X}$, and finer level, $\hat{X}$, the internal clocks $R_i, P_i, i=1, 2, 3$, and the values, $S_i, i=1, 2, 3$ (see Section 1.2 for more information on these values). Outputs: updated time, $t$, updated system states, $\bar{X}, \hat{X}$, and updated internal clocks $R_i, P_i, i=1, 2, 3$.

1: $\Delta t_i \leftarrow (P_i - R_i)/S_i$, for $i=1, 2, 3$
2: $\Delta \leftarrow \min_i \{\Delta t_i\}$
3: $\mu \leftarrow \arg\min_i \{\Delta t_i\}$
4: if $t + \Delta > H$ then
5: $R \leftarrow R + S \cdot (H - t)$
6: $t \leftarrow H$
7: else
8: update $\bar{X}$ and $\hat{X}$
9: $R \leftarrow R + S \Delta$
10: $r \leftarrow \text{uniform}(0, 1)$
11: $P_\mu \leftarrow P_\mu + \log(1/r)$
12: $t \leftarrow t + \Delta$
13: end if

Algorithm 6 Auxiliary function used in Algorithm 3. Inputs: the propensity values at the coarse and fine grid, $\bar{a}, \hat{a}$. Output: $S_i, i=1, 2, 3$.

1: $S_1 \leftarrow \min(\bar{a}, \hat{a})$
2: $S_2 \leftarrow \tilde{a} - S_1$
3: $S_3 \leftarrow \tilde{a} - S_1$
Algorithm 7 Multilevel calibration and error estimation. Inputs: same as Algorithm 3 plus the observable, \( g \), and the prescribed tolerance, \( TOL > 0 \). Outputs: \( (\hat{M}_I)_{t=0}^L \), \( (\hat{\delta}_I)_{t=0}^L \), \( (t_n, n)_{n=0}^{N_T} \), \( (s_l)_{l=0}^{K'} \), the estimated computational work of the multilevel estimator, \( \hat{W}_{ML} \), and the estimated computational work of the SSA method, \( \hat{W}_{SSA} \). We denote by \( g_t \equiv g(X_t(T; \bar{\omega})) \), and \( g_{t+1} - g_t \equiv g(X_{t+1}(T; \bar{\omega})) - g(X_t(T; \bar{\omega})) \). Here, \( C^* \) is the unitary cost of a pure SSA step, and \( c \) is the factor of refinement of \( \delta \) (in our experiments \( c = 10 \)). See also Remark 4.1 regarding the estimators of \( \text{Var}[g(X(T))] \) and \( E[g(X(T))] \), and Remark 4.3.

1: \( t \leftarrow 0 \), \( \hat{\delta}_I \leftarrow 0.01 \), \( \hat{W}_{ML}^{(0)} = \infty \)
2: Set initial meshes \( (t_k)_{k=0}^{K} \) and \( (s_l)_{l=0}^{K'} \)
3: \( \text{fin-delta} \leftarrow \text{false} \)
4: while \( \text{not fin-delta} \) do
5: \( (\hat{\psi}_t, S^2(g_t; :), \hat{A}(\{g_t, \bar{N}_{SSA}, \bar{E}_I, N_{TL,T}^{(t)}; :\}; )) \leftarrow \text{Algorithm 12} \)
6: if \( \hat{V}_t(1 - \delta_{t+1} A(N_{TL,T}^{(t)}; :)) \geq 2 S^2(g_t; : \delta_{t+1} A(N_{TL,T}^{(t)}; :)) \) and \( \delta_{t+1} A(N_{TL,T}^{(t)}; :)< 0.1 \) then
7: \( \text{fin-delta} \leftarrow \text{true} \)
8: Refine \( \delta_{t} \) by a factor of \( c \)
9: end if
10: end while
11: \( \hat{\delta}_{t+1} \leftarrow \hat{\delta}_{t} \)
12: \( \text{fin} \leftarrow \text{false} \)
13: while \( \text{not fin} \) do
14: \( \text{fin-delta} \leftarrow \text{false} \)
15: while \( \text{not fin-delta} \) do
16: \( (\hat{\psi}_{t+1}, \hat{V}_{t+1}, \hat{A}(\{g_{t+1}, N_{SSA}, \bar{E}_I, N_{TL,T}^{(t+1)}; :\}; ), S^2(g_{t+1}; :)) \leftarrow \text{Algorithm 8} \)
17: if \( \hat{V}_{t+1}(1 - \delta_{t+1} A(N_{TL,T}^{(t+1)}; :)) \geq 2 S^2(g_{t+1}; : \delta_{t+1} A(N_{TL,T}^{(t+1)}; :)) \)
18: \( \text{fin-delta} \leftarrow \text{true} \)
19: \( \delta_{t} \leftarrow \delta_{t+1} \)
20: else
21: Refine \( \delta_{t+1} \) by a factor of \( c \)
22: end if
23: end while
24: \( M_{SSA} \leftarrow C^2 S^2(g_{t+1}; :)/TOL^2 \)
25: \( \hat{W}_{SSA} \leftarrow C^* M_{SSA} A(N_{SSA}; :)) \)
26: if \( \hat{E}_I < TOL \cdot TOL^2 \) then
27: \( (\hat{M}_I)_{t=0}^{t+l} \leftarrow \text{Algorithm 11} \) with \( ((\hat{\psi}_t)_{t=0}^{t+l}, (\hat{V}_t)_{t=0}^{t+l}, TOL, \hat{E}_I) \)
28: \( \hat{W}_{ML} \leftarrow \sum_{t=0}^{t+l} \hat{\psi}_t M_t \)
29: end if
30: \( \hat{W}_{ML} \leftarrow \infty \)
31: end if
32: if \( (\hat{W}_{ML}) > \hat{W}_{ML} \) or \( \hat{E}_I > TOL \cdot TOL^2 \) and \( A(N_{TL,T}^{(t+1)}; :)) > 0 \) then
33: \( l \leftarrow l + 1 \)
34: \( (\hat{M}_I)_{t=0}^{t+l} \leftarrow \hat{W}_{ML} \)
35: Refine meshes \( (t_k)_{k=0}^{K} \) and \( (s_l)_{l=0}^{K'} \)
36: else
37: \( \text{fin} \leftarrow \delta_{t+1} A(N_{TL,T}^{(t+1)}; :)) \cdot A(g_{t+1}; :)) \leq TOL^2 \)
38: if \( \text{not fin} \) then
39: \( \delta_{t+1} \leftarrow C^2 \cdot TOL^2/(A(g_{t+1}; :)) \cdot A(N_{TL,T}^{(t+1)}; :)) \)
40: while \( \text{not fin do} \)
41: \( \text{fin} \leftarrow \delta_{t+1} A(N_{TL,T}^{(t+1)}; :)) \cdot A(g_{t+1}; :)) \leq TOL^2 \)
42: if \( \text{not fin} \) then
43: Refine \( \delta_{t+1} \) by a factor of \( c \)
44: end if
45: end if
46: end while
47: end if
48: end if
49: end while
For the sake of simplicity, we omit the arguments of the algorithms when there is no process, \( \bar{X} \in \mathbb{R} \), an estimate of \( \text{Var}[g(\bar{X}(T))−g(X(T))] \), \( \hat{V} \), an estimate of \( \text{E}[g(X(T))] \), \( \hat{A}(g(\bar{X}(T); :)) \), an estimate of the expected number of steps needed by the SSA method, \( \hat{A}(N_{SSA}; :)) \), an estimate of \( \text{E}[\mathcal{E}_I] \), \( \hat{A}(\mathcal{E}_I; :)) \), an estimate of the expected number of tau-leap steps taken at the fine level, \( \hat{A}(N_{\text{TL}}; :)) \), and an estimate of \( \text{Var}[g(X(T))] \), \( S^2(g(\bar{X}(T); :)) \). Here, \( \bar{X}(t) \) refers to the approximated process using a finer grid than the approximated process, \( \bar{X}(t) \). Moreover, \( (\bar{X}(t), \hat{X}(t)) \) are two coupled paths. Here, \( 1_{\text{TL}}(k) = 1 \) if and only if the decision at time \( t_k \) was tau-leap. Set appropriate values for \( M_0 \) and \( CV_0 \).

For the sake of simplicity, we omit the arguments of the algorithms when there is no risk of confusion. See also Remark 4.1 regarding the estimators of \( \text{Var}[g(X(T))] \) and \( \text{E}[g(X(T))] \).

\begin{algorithm}
\caption{Auxiliary function for Algorithm 7.} \label{Alg:Auxiliary}
\textbf{Inputs:} same as Algorithm 3.
\textbf{Outputs:} the estimated runtime of the coupled path, \( \hat{\psi} \), an estimate of \( \text{Var}[g(\bar{X}(T))−g(X(T))] \), \( \hat{V} \), an estimate of \( \text{E}[g(X(T))] \), \( \hat{A}(g(\bar{X}(T); :)) \), an estimate of the expected number of steps needed by the SSA method, \( \hat{A}(N_{SSA}; :)) \), an estimate of \( \text{E}[\mathcal{E}_I] \), \( \hat{A}(\mathcal{E}_I; :)) \), an estimate of the expected number of tau-leap steps taken at the fine level, \( \hat{A}(N_{\text{TL}}; :)) \), and an estimate of \( \text{Var}[g(X(T))] \), \( S^2(g(\bar{X}(T); :)) \).
\end{algorithm}

\begin{algorithmic}[1]
\STATE \( M \leftarrow M_0 \), \( cv \leftarrow \infty \), \( M_f \leftarrow 0 \)
\WHILE {\( cv > CV_0 \)}
\FOR {\( m \leftarrow 1 \) to \( M \)}
\STATE Generate two coupled paths: \( (\bar{X}(s_l; \bar{\omega}_m))_{l=0}^{K}, (\bar{X}(s_l; \bar{\omega}_m))_{l=0}^{K} \), \( \leftarrow \) Algorithm 3
\STATE \textbf{if} the path does not exit \( \mathbb{Z}_+^2 \) \textbf{then}
\STATE \quad \( M_f \leftarrow M_f + 1 \)
\STATE \quad \( (S_v(\bar{\omega}_m), S_v(\bar{\omega}_m)) \leftarrow \) Algorithm 10 with \( (\bar{X}(t_k; \bar{\omega}_m))_{K=0}^{K} \)
\STATE \quad \( \mathcal{E}_f(\bar{\omega}_m) \leftarrow \) Algorithm 9 with \( (\bar{X}(s_l; \bar{\omega}_m))_{K=0}^{K} \)
\STATE \quad \( \hat{\psi} \leftarrow C_1 \mathcal{A}(N_{\text{K1}}^{(c)}; M_f) + C_2 \mathcal{A}(N_{\text{K2}}^{(c)}; M_f) + C_3 \mathcal{A}(N_{\text{TL}}^{(c)}; M_f) + \mathcal{A}(C_{\text{PA}}; M_f) \)
\STATE \quad \( M \leftarrow 2M \)
\STATE \textbf{end if}
\STATE \( \hat{\psi} \leftarrow C_1 \mathcal{A}(N_{\text{K1}}^{(c)}; M_f) + C_2 \mathcal{A}(N_{\text{K2}}^{(c)}; M_f) + C_3 \mathcal{A}(N_{\text{TL}}^{(c)}; M_f) + \mathcal{A}(C_{\text{PA}}; M_f) \)
\RETURN \( \hat{\psi}, \hat{\psi}, \mathcal{A}(N_{SSA}; \mathcal{E}_I, N_{\text{TL}}); M_f), S^2(g(\bar{X}(T); :)) \)
\ENDWHILE
\end{algorithmic}
Algorithm 9 Compute the discretization error of a given approximated path. Inputs: $(X(t_k))_{k=0}^K$. Here, $1_{TL}(k) = 1$ if and only if the decision at time $t_k$ was tau-leap, and $Id$ is the $d \times d$ identity matrix. Output: $\mathcal{E}_I$. Notes: $x_k \equiv \bar{X}(t_k)$.

1: $\mathcal{E}_I \leftarrow 0$
2: Compute $\varphi_K \leftarrow \nabla g(x_k)$
3: for $k \leftarrow K-1$ to 1 do
4: \quad $\Delta t_k \leftarrow t_k+1 - t_k$
5: \quad Compute $\mathbb{J}_a = [\partial_j a_j(x_k)]_{j,i}$
6: \quad $\varphi_k \leftarrow (Id + \Delta t_k \mathbb{J}_a^T \nu^T) \varphi_{k+1}$
7: \quad $\Delta a_k \leftarrow a(x_{k+1}) - a(x_k)$
8: \quad $\mathcal{E}_I \leftarrow \mathcal{E}_I + \frac{\Delta t_k}{2} (\Delta a_k 1_{TL}(k) \nu^T) \varphi_k$
9: end for
10: return $\mathcal{E}_I$

Algorithm 10 Compute $S_c \equiv S_c(\bar{\omega})$ and $S_v \equiv S_v(\bar{\omega})$ defined in (3.6). Inputs: $(\bar{X}(t_k))_{k=0}^K$ and a positive constant $c$. Outputs: $S_c$ and $S_v$. Notes: if $a$ is a vector, then, $\text{diag}(a)$ is a diagonal matrix with main diagonal $a$. Here, $1_{TL}(k) = 1$ if and only if the decision at time $t_k$ was tau-leap, $Id$ is the $d \times d$ identity matrix, $x_k \equiv \bar{X}(t_k)$, and $\Phi(x)$ is the cumulative distribution function of a Gaussian random variable.

1: $S_c \leftarrow 0$
2: $S_v \leftarrow 0$
3: Compute $\varphi_K \leftarrow \nabla g(x_k)$
4: for $k \leftarrow K-1$ to 1 do
5: \quad $\Delta t_k \leftarrow t_k+1 - t_k$
6: \quad Compute $\mathbb{J}_a = [\partial_j a_j(x_k)]_{j,i}$
7: \quad $\varphi_k \leftarrow (Id + \Delta t_k \mathbb{J}_a^T \nu^T) \varphi_{k+1}$
8: \quad $\nu_{\varphi} \leftarrow \nu^T \varphi_k$
9: \quad $\nu_a \leftarrow (\mathbb{J}_a \nu)^T$
10: \quad $\mu_j \leftarrow \frac{\Delta t_k}{2} \sum_i (\nabla a_j(x_k) \cdot \nu_i) a_i(x_k)$
11: \quad $\bar{\mu}_j \leftarrow \frac{\Delta t_k}{2} \sum_i (|\nabla a_j(x_k)\cdot \nu_i|) a_i(x_k)$
12: \quad $\sigma_j^2 \leftarrow \frac{\Delta t_k}{2} \sum_i (\nabla a_j(x_k) \cdot \nu_i)^2 a_i(x_k)$
13: \quad $S_c \leftarrow S_c + 1_{TL}(k) \frac{\Delta t_k}{2} \nu_{\varphi}$
14: \quad aux1 \leftarrow \frac{(\Delta t_k)^2}{8} (\nu_a \nu_{\varphi})^T \text{diag}(a) (\nu_a \nu_{\varphi})$
15: aux2 \leftarrow \frac{\Delta t_k}{2} \sum_j (\varphi_k \cdot \nu_j)^2 1_{a_j(x_k) > c} \left( \mu_j (1 - 2 \Phi(-\frac{\mu_j}{\sigma_j})) + \sqrt{\frac{2}{\pi}} \sigma_j \exp(-\frac{1}{2} (\frac{\mu_j}{\sigma_j})^2) \right)$
16: aux3 \leftarrow \frac{\Delta t_k}{2} \sum_j (\varphi_k \cdot \nu_j)^2 1_{a_j(x_k) < c} \min \left\{ \bar{\mu}_j, \sqrt{\frac{\mu_j^2 + \sigma_j^2}{2}} \right\}$
17: $S_v \leftarrow S_v + 1_{TL}(k) (\text{aux1} + \text{aux2} + \text{aux3})$
18: end for
19: return $(S_c, S_v)$
Algorithm 11 Solve the optimization problem (4.5) using a greedy scheme. Inputs: the estimations of the coupled path cost for all the levels, \((\hat{\psi}_\ell)_{\ell=0}^L\), the estimation of the variance of the quantity of interest at level 0, \(\hat{\mathcal{V}}_0\), the estimations of the differences of the quantity of interest for all the coupled levels, \((\hat{\mathcal{V}}_\ell)_{\ell=1}^L\), the prescribed tolerance, \(TOL\), and the weak error estimation for level \(L\), \(\mathcal{E}_L\). Output: the number of realizations needed for each level, \((M)_{\ell=0}^L\).

Define \(q_k := \frac{\sum_{\ell=0}^{L-k} \sqrt{\hat{\psi}_\ell \hat{\mathcal{V}}_\ell}}{RHS - \sum_{\ell=L-k+1}^L \hat{\mathcal{V}}_\ell}\)

1: \(RHS \leftarrow ((TOL - TOL^2 - \mathcal{E}_L)/C_A)^2\)
2: \(\text{fin} \leftarrow \text{false}\)
3: \(k \leftarrow 0\)
4: **while not** \(\text{fin} \text{ and } k \leq L\) **do**
5: **if** \(\hat{\psi}_{L-k} - q_k^2 \hat{\mathcal{V}}_{L-k} < 0\) **then**
6: \(\text{fin} \leftarrow \text{true}\)
7: \((M)_{\ell=0}^{L-k} \leftarrow q_k \sqrt{\hat{\mathcal{V}}_L/\hat{\psi}_L}\)
8: **else**
9: \(M_{L-k} \leftarrow 1\)
10: \(k \leftarrow k + 1\)
11: **end if**
12: **end while**
13: **return** \((M)_{\ell=0}^L\)
Algorithm 12 Auxiliary function for Algorithm 7. Inputs: same as Algorithm 3. Outputs: the estimated runtime of the hybrid path at level 0, \( \hat{\psi}_0 \), an estimate of Var \( \left[g(\bar{X}(T))\right] \), \( S^2 (g(\bar{X}(T)); \cdot) \), an estimate of E \( \left[g(\bar{X}(T))\right] \), \( A (g(\bar{X}(T)); \cdot) \), an estimate of E \( \left[E_I\right] \), \( A (E_I; \cdot) \), an estimate of the expected number of steps needed by the SSA method, \( A (N_{SSA}^*; \cdot) \) and \( A (N_{TL}; \cdot) \). Here, \( 1_{TL}(k) = 1 \) if and only if the decision at time \( t_k \) was tau-leap. Notes: the values \( C_1, C_2 \) and \( C_3 \) are described in Section 4. Set appropriate values for \( M_0 \) and \( CV_0 \). For the sake of simplicity, we omit the arguments of the algorithms when there is no risk of confusion.

1: \( M \leftarrow M_0, \ cv \leftarrow \infty, M_f \leftarrow 0 \)
2: while \( cv > CV_0 \) do
3:   for \( m \leftarrow 1 \) to \( M \) do
4:     \( ((\bar{X}(t_k))_{k=0}^K, N_{TL}, N_{SSA,K1}, N_{SSA,K2}) \leftarrow \text{generate one hybrid path (see [19])} \)
5:     if the path does not exit \( \mathbb{Z}_+^d \) then
6:       \( M_f \leftarrow M_f + 1 \)
7:       Compute \( g(\bar{X}(t_k; \tilde{\omega}_m)) \)
8:     \( E_I \leftarrow \text{Algorithm 9 with } (\bar{X}(t_k))_{k=0}^K \)
9:     \( (S_e(\tilde{\omega}_m), S_v(\tilde{\omega}_m)) \leftarrow \text{Algorithm 10 with } (\bar{X}(t_k))_{k=0}^K \)
10:    \( \tilde{C}_Poi(\tilde{\omega}_m) \leftarrow \sum_{j=1}^J \sum_{k=0}^K C_P(a_j(\bar{X}(t_k))(t_{k+1} - t_k))1_{TL}(k) \)
11:   end if
12: end for
13: \( \psi_0 \leftarrow \text{Var } [g(\bar{X}(T)) - g(\bar{X}(T))] \), \( \text{Var } [g(\bar{X}(T))] \) and \( E[I] \), respectively.
14: \( cv \leftarrow \max \{cv_V, cv_g, cv_{E_I}\} \)
15: \( \hat{\psi}_0 \leftarrow C_1 A (N_{SSA,K1}; M_f) + C_2 A (N_{SSA,K2}; M_f) + C_3 A (N_{TL}; M_f) + A (C_{Poi}; M_f) \)
16: \( M \leftarrow 2M \)
17: end while
18: return \( (\hat{\psi}_0, S^2 (g(\bar{X}(T)); M_f), A ((g(\bar{X}(T)); E_I, N_{SSA^*}, N_{TL}); M_f)) \)