An efficient and unbiased method for sensitivity analysis of stochastic reaction networks

Ankit Gupta and Mustafa Khammash

We consider the problem of estimating parameter sensitivity for Markovian models of reaction networks. Sensitivity values measure the responsiveness of an output with respect to the model parameters. They help in analysing the network, understanding its robustness properties and identifying the important reactions for a specific output. Sensitivity values are commonly estimated using methods that perform finite-difference computations along with Monte Carlo simulations of the reaction dynamics. These methods are computationally efficient and easy to implement, but they produce a biased estimate which can be unreliable for certain applications. Moreover, the size of the bias is generally unknown and hence the accuracy of these methods cannot be easily determined. There also exist unbiased schemes for sensitivity estimation but these schemes can be computationally infeasible, even for very simple networks. Our goal in this paper is to present a new method for sensitivity estimation, which combines the computational efficiency of finite-difference methods with the accuracy of unbiased schemes. Our method is easy to implement and it relies on an exact representation of parameter sensitivity that we recently proved elsewhere. Through examples, we demonstrate that the proposed method can outperform the existing methods, both biased and unbiased, in many situations.

1. Introduction

Reaction networks are frequently encountered in several scientific disciplines, such as chemistry [1], systems biology [2], epidemiology [3], ecology [4] and pharmacology [5]. It is well known that when the population sizes of the reacting species are small, then a deterministic formulation of the reaction dynamics is inadequate for understanding many important properties of the system [6,7]. Hence, stochastic models are necessary and the most prominent class of such models is where the reaction dynamics is described by a continuous time Markov chain (CTMC). Our paper is concerned with sensitivity analysis of these Markovian models of reaction networks.

Generally, reaction networks involve many kinetic parameters that influence their dynamics. With sensitivity analysis, one can measure the receptiveness of an outcome to small changes in parameter values. These sensitivity values give insights about network design and its robustness properties [8]. They also help in identifying the important reactions for a given output, estimating the model parameters and in fine-tuning the system’s behaviour [9]. The existing literature contains many methods for sensitivity analysis of CTMC models, but all these methods have certain drawbacks. They introduce a bias in the sensitivity estimate [10–13] or the associated estimator can have large variances [14] or the method becomes impractical for large networks [15]. Owing to these reasons, the search for better methods for sensitivity analysis of stochastic reaction networks is still an active research problem.

Consider a stochastic reaction network consisting of $d$ species $S_1, \ldots, S_d$. Under the well-stirred assumption [16], the state of the system at any time is given by a vector in $\mathbb{N}_d^0$, where $\mathbb{N}_d^0$ is the set of non-negative integers.
When the state is \( x = (x_1, \ldots, x_d) \), then the population size corresponding to species \( S_i \) is \( x_i \). The state evolves as the species interact through \( K \) reaction channels. If the state is \( x \), then the \( k \)th reaction fires at rate \( \lambda_k(x) \) and it displaces the state by the stoichiometric vector \( \hat{s}_k \in \mathbb{Z}^d \). Here, \( \lambda_k \) is called the propensity function for the \( k \)th reaction. We can represent the reaction dynamics by a CTMC and the distribution of this process evolves according to the Chemical Master Equation [16] which is quite difficult to solve except in some restrictive cases. Fortunately, the sample paths of this process can be easily simulated using Monte Carlo procedures such as Gillespie’s stochastic simulation algorithm (SSA) and its variants [16,17].

Suppose that the propensity functions of a reaction network depend on a scalar parameter \( \theta \). Hence, when the state is \( x \), the \( k \)th reaction fires at rate \( \lambda_k(x, \theta) \). Let \( (X_\theta(t))_{t>0} \) denote the CTMC representing the reaction dynamics with a known initial state \( x_\theta \). For a function \( f: \mathbb{N}_0^d \rightarrow \mathbb{R} \) and an observation time \( T \geq 0 \), our output of interest is \( f(X_\theta(T)) \). We determine how much the expected value of this output changes with infinitesimal changes in the parameter \( \theta \). In other words, our aim is to compute

\[
S_\theta(f, T) = \frac{\partial}{\partial \theta} \mathbb{E}(f(X_\theta(T))).
\]

(1.1)

Since the mapping \( \theta \mapsto \mathbb{E}(f(X_\theta(T))) \) is generally unknown, it is difficult to evaluate \( S_\theta(f, T) \) directly. Therefore, we need to estimate this quantity using simulations of the process \( X_\theta \). For this purpose, many methods use a finite-difference scheme such as

\[
S_{\delta\theta}(f, T) = \frac{1}{h} \mathbb{E}(f(X_{\theta+k}(T))) - f(X_\theta(T))
\]

(1.2)

for a small \( h \). These methods reduce the variance of the associated estimator by coupling the processes \( X_\theta \) and \( X_{\theta+k} \) in an intelligent way. Three such couplings are: common reaction numbers (CRN) [11], common reaction paths (CRP) [11] and coupled finite differences (CFD) [13]. The two best-performing finite-difference schemes are CRP and CFD, and we shall discuss them in §2.1. It is immediate that a finite-difference scheme produces a biased estimate of the true sensitivity value \( S_\theta(f, T) \). To further compound this problem, the size and even the sign of the bias is unknown in most cases, thereby casting a doubt over the estimated sensitivity values. The magnitude of the bias must be proportional to \( h \), but it can still be significant for small \( h \) (see §4). One can reduce the bias by decreasing \( h \), but as \( h \) gets smaller, the variance of the finite-difference estimator gets larger, making it necessary to generate an extremely large sample to obtain a statistically accurate estimate. In many cases, for small values of \( h \), the computational cost of generating a large sample is so high that finite-difference schemes become inefficient in comparison to the unbiased methods. Unfortunately, there is no way to select an optimum value of \( h \), which is small enough to ensure that the bias is small and simultaneously large enough to ensure that the estimator variance is low. This is the main difficulty in using finite-difference schemes and we shall revisit this point in §4.

To obtain highly accurate estimates of \( S_\theta(f, T) \), we need unbiased methods for sensitivity analysis. The first such scheme is the likelihood-ratio method which relies on the Girsanov measure transformation [14,18]. Hence, we refer to it as the Girsanov method in this paper. In this method, the sensitivity value is expressed as

\[
S_\theta(f, T) = \mathbb{E}(s_\theta(f, T)),
\]

(1.3)

where \( s_\theta(f, t) \) is a random variable whose realizations can be easily obtained by simulating the CTMC \( X_\theta \) in the time interval \([0, T]\). The Girsanov method is easily implementable, and its unbiasedness guarantees convergence to the right value \( S_\theta(f, T) \) as the sample size tends to infinity. However in many situations, the estimator associated with this method has a very high variance, making it extremely inefficient [11–13,15]. To remedy this problem, we proposed another unbiased scheme in [15] based on a different sensitivity formula of the form (1.3), which is derived using the random time-change representation of Kurtz (see ch. 7 in [19]). Unfortunately in this formula, the realizations of \( s_\theta(f, t) \) cannot be easily computed from the paths of the process \( X_\theta \) as it involves several expectations of functions of the underlying CTMC at various times and various initial states. If all these expectations are estimated serially using independent paths, then the problem becomes computationally intractable. Hence, we devised a scheme in [15], called the Auxiliary Path Algorithm (APA), to estimate all these expectations in parallel using a fixed number of auxiliary paths. The implementation of APA is quite difficult and the memory requirements are high because one needs to store the paths and dynamically maintain a large table to estimate the relevant quantities. These reasons make APA impractical for large networks and also for large observation times \( T \). However, in spite of these difficulties, we showed in [15] that APA can be far more efficient than the Girsanov method in certain situations.

The above discussion suggests that all the existing methods for sensitivity analysis, both biased and unbiased, have certain drawbacks. This motivated us to develop a new method for sensitivity estimation which is unbiased, easy to implement, has low memory requirements and is more versatile than the existing unbiased schemes. We use the main result in [15] to construct another random variable \( \tilde{s}_\theta(f, T) \) such that (1.3) holds.

In §4, we consider many examples to compare the performance of PPA with the Girsanov method, CFD and CRP. We find that PPA is usually far more efficient than the Girsanov method. Perhaps surprisingly, in many situations PPA can also outperform the finite-difference schemes (CRP and CRN) if we impose the restriction that the bias is small. This makes PPA an attractive method for sensitivity analysis because one can efficiently obtain sensitivity estimates, without having to worry about the (unknown) bias caused by finite-difference approximations.

This paper is organized as follows. In §2, we provide the relevant mathematical background and discuss the existing schemes for sensitivity analysis in greater detail. We also present our main result which shows that for a suitably defined random variable \( \tilde{s}_\theta(f, T) \), relation (1.3) is satisfied. In §3, we describe PPA which is a simple method to obtain realizations of \( \tilde{s}_\theta(f, T) \). In §4,
2. Preliminaries

Recall the description of the reaction network from the previous section and suppose that the propensity functions depend on a real-valued parameter $\theta$. We can model the reaction dynamics by a CTMC whose generator $\Lambda_\theta$ is given by

$$\Lambda_\theta f(x) = \sum_{k=1}^{\infty} \lambda_k(x, \theta) \Delta f_k(x),$$

where $\Delta f_k(x) = f(x + \Delta x) - f(x)$. Under some mild conditions on the propensity functions (see Condition 2.1 in [15]), a CTMC $(X(t))_{t \geq 0}$ with generator $\Lambda_\theta$ and any initial state $x_0$ exists uniquely. The random time-change representation of Kurtz (see ch. 7 in [19]) shows that this CTMC can be expressed as

$$X(t) = x_0 + \sum_{k=1}^{\infty} Y_k \left( \int_{0}^{t} \lambda_k(X(s), \theta) ds \right) \xi_k,$$

where $\{Y_k : k = 1, \ldots, K\}$ is a family of independent unit rate Poisson processes.

In this paper, we are interested in computing the sensitivity value $S_d(f, T)$ defined by (1.1). Assume that we can express the sensitivity value as (1.3) for some random variable $S_d(f, T)$. If we are able to generate $N$ independent samples $S^{(1)}_d(f, T), \ldots, S^{(N)}_d(f, T)$ from the distribution of $S_d(f, T)$, then $S_d(f, T)$ can be estimated as

$$\hat{S}_d(f, T) = \frac{1}{N} \sum_{i=1}^{N} S^{(i)}_d(f, T),$$

which is a random variable with mean $\mu_N$ and variance $\sigma^2_N$, given by

$$\mu_N = E\left( \frac{1}{N} \sum_{i=1}^{N} S^{(i)}_d(f, T) \right) = E(S_d(f, T))$$

and

$$\sigma^2_N = Var\left( \frac{1}{N} \sum_{i=1}^{N} S^{(i)}_d(f, T) \right) = \frac{1}{N} Var(S_d(f, T)).$$

Owing to the central limit theorem, for large values of $N$, the distribution of $\hat{S}_d(f, T)$ is approximately normal with mean $\mu_N$ and variance $\sigma^2_N$. Hence, for any interval $[a, b] \subset \mathbb{R}$, the probability $P(\hat{S}_d(f, T) \in [a, b])$ can be approximated as

$$P(\hat{S}_d(f, T) \in [a, b]) \approx \Phi\left( \frac{b - \mu_N}{\sigma_N} \right) - \Phi\left( \frac{a - \mu_N}{\sigma_N} \right),$$

where $\Phi(\cdot)$ is the cumulative distribution function for the standard normal distribution and $\sigma_N$ is the standard deviation. Generally, $\mu_N$ and $\sigma_N$ are unknown but they can be estimated from the sample as

$$\mu_N \approx \hat{S}_d(f, T) \quad \text{and} \quad \sigma_N \approx \sqrt{\frac{1}{N(N-1)} \sum_{i=1}^{N} (S^{(i)}_d(f, T) - \hat{S}_d(f, T))^2}.$$ 

Note that the mean $\mu_N$ is just the one-point estimate for the sensitivity value, while the standard deviation $\sigma_N$ measures the statistical spread around $\mu_N$. The standard deviation $\sigma_N$ can also be seen as the estimation error. From (2.3), it is immediate that $\sigma_N$ is directly proportional to $Var(S_d(f, T))$ but inversely proportional to $N$. Hence, if $Var(S_d(f, T))$ is high then a larger number of samples is needed to achieve a certain statistical accuracy.

In finite-difference schemes, instead of $S_d(f, T)$ one estimates a finite difference $S_{d_h}(f, T)$ of the form (1.2) which can be written as $S_{d_h}(f, T) = E(S_{d_h}(f, T))$ for

$$S_{d_h}(f, T) = \frac{f(X_{d_h}(T)) - f(X_d(T))}{h}.$$

Here, the Markov processes $X_d$ and $X_{d_h}$ are defined on the same probability space and they have generators $\Lambda_d$ and $\Lambda_{d_h}$, respectively. In this case, the estimator mean $\mu_{d_h}$ and variance $\sigma^2_{d_h}$ are given by (2.3) with $S_d(f, T)$ replaced by $S_{d_h}(f, T)$.

2.1. Finite-difference schemes

The main idea behind the various finite-difference schemes is that by coupling the processes $X_d$ and $X_{d_h}$ one can increase the covariance between $f(X_{d_h}(T))$ and $f(X_d(T))$, thereby reducing the variance of $S_{d_h}(f, T)$ and making the estimation procedure more efficient. As mentioned before, three such couplings suggested in the existing literature are: CRN [11], CRP [11] and CFD [13]. Among these, we only consider the two best-performing schemes, CRP and CFD, in this paper.

In CRP, the processes $X_d$ and $X_{d_h}$ are coupled by their random time-change representations according to

$$X_d(t) = x_0 + \sum_{k=1}^{K} Y_k \left( \int_{0}^{t} \lambda_k(X(s), \theta) ds \right) \xi_k$$

and

$$X_{d_h}(t) = x_0 + \sum_{k=1}^{K} Y_k \left( \int_{0}^{t} \lambda_k(X_{d_h}(s), \theta + h) ds \right) \xi_k,$$

where $\{Y_k : k = 1, \ldots, K\}$ is a family of independent unit rate Poisson processes. Note that these Poisson processes are the same for both processes $X_d$ and $X_{d_h}$, indicating that their reaction paths are the same.

In CFD, the processes $X_d$ and $X_{d_h}$ are coupled by their random time-change representations as

$$X_d(t) = x_0 + \sum_{k=1}^{K} Y_k \left( \int_{0}^{t} \lambda_k(X_d(s), \theta) ds \right) \xi_k + \sum_{k=1}^{K} Y_k^{(1)} \left( \int_{0}^{t} \lambda_k(X_d(s), \theta + h) ds \right) \xi_k$$

and

$$X_{d_h}(t) = x_0 + \sum_{k=1}^{K} Y_k \left( \int_{0}^{t} \lambda_k(X_{d_h}(s), \theta) ds \right) \xi_k + \sum_{k=1}^{K} Y_k \left( \int_{0}^{t} \lambda_k(X_{d_h}(s), \theta + h) ds \right) \xi_k,$$

where $\lambda_k(X_d(s), \theta) = \lambda_k(X_{d_h}(s), \theta)$ and $\lambda_k(X_d(s), \theta + h) = \lambda_k(X_{d_h}(s), \theta + h)$ for $k = 1, \ldots, K$. Note that this is again a family of independent unit rate Poisson processes. Under this coupling, the processes $X_d$ and $X_{d_h}$ have the same state until the first time the counting process
corresponding to $Y_k^{(1)}$ or $Y_k^{(2)}$ fires for some $k$. The probability that this separation time will come before $T$ is proportional to $h$, which is usually small. This suggests that the processes $X_0$ and $X_{(a+h)}$ are strongly coupled.

Note that both CRP and CFD are estimating the same quantity $S_{\theta h}(f, T)$ (1.2) and hence they both suffer from the same bias $|S_{\theta h}(f, T) - S_0(f, T)|$. The only difference between them is in the coupling of the processes $X_0$ and $X_{(a+h)}$, which alters the variance of $S_{\theta h}(f, T)$. For a given example, the method with a lower variance is more efficient as it requires a smaller sample to achieve the desired statistical accuracy.

For a finite-difference scheme of the form (1.1), the bias $|S_{\theta h}(f, T) - S_0(f, T)|$ is of order $h$. However, replacing $\theta$ by $(\theta - h)/2$, we obtain a central finite-difference scheme which has a bias of order $h^2$. Such a centring does not cause any additional difficulties in simulating the coupled processes for CFD or CRN, but it may lead to a bias which is substantially smaller. Therefore, for the purpose of comparison, we work with the central versions of CFD and CRN in this paper. In particular, we simulate processes $X_{(\theta - h/2)}$ and $X_{(a+h/2)}$ according to the appropriate coupling mechanism and estimate sensitivity as $S_{\theta h}(f, T) = E(s_{\theta h}(f, T))$, where

$$s_{\theta h}(f, T) = \frac{(f(X_{(\theta - h/2)}(T)) - f(X_{(a+h/2)}(T)))}{h}.$$ (2.6)

One can show that the variance $\text{Var}(s_{\theta h}(f, T))$ of the random variable $s_{\theta h}(f, T)$ is proportional to $1/h$. Hence, by making $h$ smaller, we may reduce the bias but we will have to pay a computational cost by generating a large number of samples for the required estimation. This trade-off between bias and efficiency is the major drawback of finite-difference schemes as one generally does not know the right value of $h$ for which $s_{\theta h}(f, T)$ is close enough to $S_0(f, T)$, while the variance of $s_{\theta h}(f, T)$ is not too large.

### 2.2. Unbiased schemes

Unbiased schemes are desirable because one does not have to worry about the bias, and the accuracy of the estimate can be improved by simply increasing the number of samples. The Girsanov method is the first such unbiased scheme [14,18] for sensitivity estimation. Recall the random time-change representation (2.1) of the process $(X_0(t))_{t \geq 0}$. In the Girsanov method, we express $S_0(f, T)$ as (1.3) where

$$s_0(f, T) = f(X_0(T)) \sum_{k=1}^{K} \left( \int_0^T \frac{1}{\lambda_k(X_0(t), \theta)} \frac{\partial \lambda_k(X_0(t), \theta)}{\partial \theta} R_k(dt) \right),$$ (2.7)

and $(R_k(t))_{t \geq 0}$ is the counting process given by

$$R_k(t) = Y_k \left( \int_0^t \lambda_k(X_0(s), \theta) ds \right).$$

In (2.7), $R_k(dt) = 1$ if and only if the $k$th reaction fires at time $t$. For any other $t$, $R_k(dt) = 0$. Hence, if the $k$th reaction fires at times $t_1^{(k)}, \ldots, t_{n_k}^{(k)}$ in the time interval $[0, T]$, then

$$\int_0^T \frac{1}{\lambda_k(X_0(t), \theta)} \frac{\partial \lambda_k(X_0(t), \theta)}{\partial \theta} R_k(dt) = \sum_{j=1}^{n_k} \left( \int_0^{t_j^{(k)}} \frac{1}{\lambda_k(X_0(t), \theta)} \frac{\partial \lambda_k(X_0(t), \theta)}{\partial \theta} dt \right).$$

The Girsanov method is simple to implement because realizations of $s_0(f, T)$ can be easily generated by simulating the paths of the process $X_0$ until time $T$. However, as mentioned before, the variance of $s_0(f, T)$ can be very high [11–13], which is a serious problem as it implies that a large number of samples are needed to produce statistically accurate estimates. As simulations of the process $X_0$ can be quite cumbersome, generating a large sample can take an enormous amount of time.

Now consider the common situation where we have mass-action kinetics and $\theta$ is the rate constant of reaction $k_0$. Then $\lambda_k$ has the form $\lambda_k(x, \theta) = \theta k_0 \chi_k(x)$ and for every $k \neq k_0$ $\lambda_k$ does not depend on $\theta$. In this case, the formula (2.7) for $s_0(f, T)$ simplifies to

$$s_0(f, T) = \frac{1}{\theta} f(X_0(T)) \left( N_{k_0}^0(T) - \int_0^T \lambda_0(X_0(t), \theta) dt \right),$$ (2.8)

where $N_{k_0}^0(T)$ is the number of times reaction $k_0$ fires in the time interval $[0, T]$. Clearly, the Girsanov method cannot estimate the sensitivity value at $\theta = 0$ even though $S_0(f, T)$ (1.1) is well defined. This is a big disadvantage as the sensitivity value at $\theta = 0$ is useful for understanding network design as it informs us whether the presence or absence of reaction $k_0$ influences the output or not. Unfortunately, the problem with the Girsanov method is not just limited to $\theta = 0$. Even for $\theta$ close to 0, the variance of $s_0(f, T)$ can be very high, rendering this method practically infeasible [15]. This is again a serious drawback as reaction rate constants with small values are frequently encountered in systems biology and other areas.

These issues with the Girsanov method severely restrict its use and also highlight the need for new unbiased schemes for sensitivity analysis. In our recent paper [15], we provide a new unbiased scheme based on another sensitivity formula of the form (1.3). To motivate this formula, we first discuss the problem of computing parameter sensitivity in the deterministic setting.

Consider the deterministic model for the parameter-dependent reaction network with reaction flux $\lambda_k(x, \theta)$ [20] and propensity vector $\zeta_k$ for reaction $k$. If $x_0(t) \in [0, \infty]^d$ denotes the vector of species concentrations at time $t$, then $(x_0(t))_{t \geq 0}$ is the solution of the following system of ordinary differential equations:

$$\frac{dx}{dt} = \sum_{k=1}^{K} \lambda_k(x, \theta) \zeta_k,$$ (2.9)

with initial condition $x(0) = x_0$. Pick an observation time $T > 0$ and a differentiable output function $f: \mathbb{R}^d \to \mathbb{R}$, and suppose we are interested in computing the sensitivity value

$$\frac{\partial f(x_0(T))}{\partial \theta}.$$ 

Using (2.9) we can write

$$f(x_0(T)) = f(x_0) + \sum_{k=1}^{K} \left( \int_0^T \lambda_k(x(t), \theta) (\nabla f(x(t))) \cdot \zeta_k dt \right),$$

where $\nabla f(x)$ denotes the gradient of the map $x \mapsto f(x)$ and $\cdot$ denotes the standard inner product on $\mathbb{R}^d$. Assuming
differentiability of \( \lambda_1 \)'s we get

\[
\frac{\partial f(x(t))}{\partial \theta} = \sum_{k=1}^{K} \left( \int_0^T \left( \frac{\partial \lambda_k(X(t), \theta)}{\partial \theta} \right) \nabla f(x(t)) \cdot \xi \right) dt + \int_0^T \nabla \lambda_k(x(t), \theta) \cdot \nabla f(x(t)) \cdot \xi dt + \int_0^T \nabla \lambda_k(x(t), \theta) \cdot \nabla f(x(t)) \cdot \eta dt,
\]

(2.10)

where \( \eta \) can be easily computed by solving the ordinary differential equation obtained by differentiating (2.9) with respect to \( \theta \). Hence, in the deterministic setting, computation of the parameter sensitivity is relatively straightforward.

Relation (2.10) helps us in viewing parameter sensitivity as the sum of two parts: sensitivity of the reaction fluxes \( \lambda_k \) and the sensitivity of the states \( x(t) \). In [15], we provide such a decomposition for parameter sensitivity in the stochastic setting. However, unlike the deterministic scenario, measuring the second contribution is computationally very challenging. In order to present this result, we need to define certain quantities. Let \( (X(t))_{t \geq 0} \) be a CTMC with generator \( A_0 \). For any \( f: \mathbb{R}_0^+ \rightarrow \mathbb{R}, t \geq 0, k = 1, \ldots, K \) and \( x \in \mathbb{R}_0^+ \) define

\[
\Psi_d(f, x, t) = \mathbb{E}(f(X(t)) | X(0) = x)
\]

and

\[
D_d(x, f, t) = \Psi_d(x + \xi, f, t) - \Psi_d(x, f, t).
\]

(2.11)

Define \( \lambda_0(x, \theta) = \sum_{k=1}^{K} \lambda_k(x, \theta) \) and let \( \sigma_0, \sigma_1, \sigma_2 \) denote the successive jump times of the process \( X_\theta \). Theorem 2.3 in [15] shows that \( S_d(f, T) \) satisfies (1.3) with

\[
s_d(f, T) = \sum_{k=1}^{K} \left( \int_0^T \frac{\partial \lambda_k(X(t), \theta)}{\partial \theta} \Delta \lambda_k(X(t), \theta) dt \right) + \sum_{i=0}^{\infty} \frac{\partial \lambda_k(X(\sigma), \theta)}{\partial \theta} R_d(X(\sigma), f, t - \alpha, k),
\]

(2.12)

where

\[
R_d(x, f, t, k) = \int_0^T (D_d(x, f, s, k) - \Delta \lambda_k(x(s), \xi)) e^{\lambda_0(x(s), \theta) - \lambda_0(x(s), \theta)} ds.
\]

(2.13)

This result was proved by coupling the processes \( X_\theta \) and \( X_{\theta+h} \) as in CFD (see §2.1) and computing the limit of the finite difference \( S_d(f, T) \) (see (1.2)) as \( h \to 0 \). Similar to the deterministic case, this result shows that parameter sensitivity \( S_d(f, T) \) can be seen as the sum of two parts: the sensitivity of the reaction functions \( \lambda_0 \) and the sensitivity of the states \( X_\theta \) of the Markov process at the jump times \( \sigma \). Computing the latter contribution is difficult because it involves the function \( R_d \) which generally does not have an explicit formula. To overcome this problem, one needs to estimate all the quantities of the form \( R_d(x, f, t, k) \) that appear in (2.12). However, the number of these quantities is proportional to the number of jumps of the process before time \( T \), which can be quite high even for small networks. If we estimate each of these quantities serially, as and when they appear, using a collection of independently generated paths of the process \( X_\theta \) then the problem becomes computationally intractable for most examples of interest. In [15], we devised a scheme, called the APA, to estimate all these quantities in parallel by generating a fixed number of auxiliary paths in addition to the main path. APA stores information about all the required quantities in a big Hashtable and tracks the states visited by the auxiliary paths to estimate those quantities. Owing to all the necessary book-keeping, APA is hard to implement and it also has high memory requirements. In fact, the space and time complexity of APA scales linearly with the number of jumps in the time interval \( [0, T] \) and this makes APA practically unusable for large networks or for large values of observation time \( T \). Moreover, APA only works well if the stochastic dynamics visits the same states again and again, which may not be generally true.

The above discussion suggests that if we can modify the expression for \( s_d(f, T) \) in such a way that only a small fraction of unknown quantities (of the form \( R_d(x, f, t, k) \)) require estimation, then it can lead to an efficient method for sensitivity estimation. We exploit this idea in this paper. By adding extra randomness to the random variable \( s_d(f, T) \), we construct another random variable \( s_d(f, T) \), which has the same expectation as \( s_d(f, T) \)

\[
\mathbb{E}(s_d(f, T)) = \mathbb{E}(s_d(f, T)),
\]

(2.14)

even though its distribution may be different. We then show that realizations of the random variable \( s_d(f, T) \) can be easily obtained through a simple procedure. This gives us a new method for unbiased parameter sensitivity estimation for stochastic reaction networks.

2.3. Construction of \( s_d(f, T) \)

We now construct the random variable \( s_d(f, T) \) introduced in the previous section. Let \( (X_\theta(t))_{t \geq 0} \) be the CTMC with generator \( A_0 \) and initial state \( x_0 \). Let \( \sigma_0, \sigma_1, \ldots \) denote the successive jump times of this process. The total number of jumps until time \( T \) is given by

\[
\eta = \max \{ i \geq 0 : \sigma_i < T \}.
\]

(2.15)

For any \( T > 0 \), we have \( \eta \geq 1 \) because \( \sigma_0 = 0 \). If the process \( X_\theta \) reaches a state \( s \) for which \( \lambda_0(s, \theta) = 0 \), then \( x \) is an absorbing state and the process stays in \( s \) forever. From (2.15), it is immediate that for any non-negative integer \( i < \eta, X_\theta(\sigma_i) \) cannot be an absorbing state. Let \( \alpha \) denote the probability of the state \( X_\theta(\sigma_i) \) absorbing

\[
\alpha = \begin{cases} 1 & \text{if } \lambda_0(X_\theta(\sigma)), \theta = 0 \\ 0 & \text{otherwise} \end{cases}.
\]

(2.16)

For each \( i = 0, \ldots, \eta - 1 \), let \( \gamma_i \) be an independent exponentially distributed random variable with rate \( \lambda_0(X_\theta(\sigma)), \theta \) and define

\[
\Gamma_i = \begin{cases} 1 & \text{if } \gamma_i < (T - \sigma_i) \\ 0 & \text{otherwise} \end{cases}.
\]

For each \( i = 0, \ldots, \eta - 1 \) and \( k = 1, \ldots, K \), let \( \beta_{ki} \) be given by

\[
\beta_{ki} = \text{Sign}(\frac{\partial \lambda_k(X_\theta(\sigma), \theta)}{\partial \theta}),
\]

where

\[
\text{Sign}(x) = \begin{cases} 1 & \text{if } x > 0 \\ -1 & \text{if } x < 0 \\ 0 & \text{if } x = 0 \end{cases}.
\]

Fix a normalizing constant \( c > 0 \). The choice of \( c \) and its role will be explained later in the section. If \( \beta_{ki} \neq 0 \) and \( \Gamma_i = 1 \), then let \( \rho_{ki}^T \) be an independent \( \mathbb{N}_0 \)-valued random variable whose distribution is Poisson with parameter

\[
\frac{c}{\lambda_0(X_\theta(\sigma), \theta)} \frac{\partial \lambda_k(X_\theta(\sigma), \theta)}{\partial \theta}.
\]

(2.16)

Here, the denominator \( \lambda_0(X_\theta(\sigma), \theta) \) is non-zero because for \( i \leq \eta - 1 \) the state \( X_\theta(\sigma) \) cannot be absorbing. On the
where \( D_k \) and \( (Z_k(t))_{k \geq 0} \) are two processes which are coupled by the following random time-change representations:

\[
Z_1(t) = (X_0 + \xi_1) + \sum_{k=1}^{K} \int_0^t \lambda_k(Z_k(s), \theta) \, ds \xi_k
\]

and

\[
Z_2(t) = X_0 + \sum_{k=1}^{K} \int_0^t \lambda_k(Z_k(s), \theta) \, ds \xi_k
\]

where \( \{\xi_k, \xi_1^{(k)}, \xi_2^{(k)} : k = 1, \ldots, K\} \) is an independent family of unit rate Poisson processes. This coupling is similar to the coupling used by CFD (see §2.1). Note that \( (Z_k(t))_{k \geq 0} \) and \( (Z_0(t))_{k \geq 0} \) are Markov processes with generator \( \mathcal{P}_0 \) and initial states \( \{X_0 + \xi_1\} \) and \( X_0 \), respectively. Therefore,

\[
\mathbb{E}(D_k | T - \sigma_r - \gamma_r, X_0, \lambda_k) = \mathbb{E}_0(X_0 + \xi_k, f, T - \sigma_r - \gamma_r)
\]

\[
= D_0(X_0, f, T - \sigma_r - \gamma_k, k)
\]

where \( D_0 \) is defined by (2.11). In other words, given \( T - \sigma_r - \gamma_r = t \) and \( X_0 = x \), the mean of the random variable \( D_k \) is just \( D_0(\cdot, f, t, k) \). The above coupling between \( (Z_k(t))_{k \geq 0} \) and \( (Z_0(t))_{k \geq 0} \) makes them strongly correlated, thereby lowering the variance of the difference \( D_k - f(Z_k(t)) - f(Z_0(t)) \). This strong correlation is evident from the fact that if \( Z_k(s) = Z_0(s) \) for some \( s \geq 0 \) then \( Z_k(u) = Z_0(u) \) for all \( u \geq s \). Finally, if the last state \( X_k(X_0, \lambda_k, \theta) \) is absorbing (that is, \( \alpha = 1 \)) and \( \lambda_k(X_0, \lambda_k, \theta) \) is non-zero, then we define another random variable \( \bar{l}_k \) as

\[
\bar{l}_k = \int_0^T f(Z(s)) \, ds,
\]

where \( (Z(t))_{k \geq 0} \) is an independent Markov process with generator \( \mathcal{P}_0 \) and initial state \( X_0 + \xi_k \). Note that

\[
\mathbb{E}(\bar{l}_k | X_0, T - \sigma_r) = \int_0^T \mathbb{P}_0(X_0, f, s) \, ds.
\]

We are now ready to provide an expression for \( s_0(f, T) \). Let

\[
\Delta_i = \begin{cases} 
(\sigma_{i+1} - \sigma_i) & \text{for } i = 0, \ldots, \eta - 1, \\
(\eta - \sigma_\eta) & \text{for } i = \eta.
\end{cases}
\]

and define

\[
\hat{s}_0(f, T) = \sum_{k=1}^{K} \sum_{i=0}^{\eta - 1} \int_0^{\Delta_i} \frac{\partial \lambda_k(X_0(\sigma_i, \theta))}{\partial \theta} \Delta_i \, f(X_0(\sigma_i))
\]

\[
+ \int_0^{\eta - \sigma_\eta} \frac{\partial \lambda_k(X_0(\sigma_i, \theta))}{\partial \theta} \Delta_i \, f(X_0(\sigma_i))
\]

\[
+ \frac{\partial \lambda_k(X_0(\sigma_\eta, \theta))}{\partial \theta} (\bar{l}_k - (T - \sigma_\eta)(f(X_0(\sigma_\eta))))
\]

By a simple conditioning argument, we prove in the electronic supplementary material, §1, that relation (2.14) holds.

At first glance, it seems that obtaining realizations of \( \hat{s}_0(f, T) \) suffers from the same difficulties as obtaining realizations of \( s_0(f, T) \), because at each jump time \( \sigma_r < T \), one needs to compute an unknown quantity \( D_i \) which requires simulation of three new paths of the process \( X_0 \). However, note that \( D_0 \) is only needed if the Poisson random variable \( \rho_0^{(0)} \) is strictly positive. Therefore, if we can effectively control the number of positive \( \rho_0^{(0)} \)’s, then we can efficiently generate realizations of \( \hat{s}_0(f, T) \). This can be achieved using the positive parameter \( c \) (see (2.16)) as we later explain. The construction of \( \hat{s}_0(f, T) \) outlined above also provides a recipe for obtaining its realizations. Hence, we have a method for estimating the parameter sensitivity \( s_0(f, T) \). We call this method the PPA because at each jump time \( \sigma_r \), the crucial decision of whether new paths of the process \( X_0 \) are needed (for \( D_0 \)) is based on the value of a Poisson random variable \( \rho_0^{(0)} \). We describe PPA in greater detail in §3.

We now discuss how the positive parameter \( c \) can be selected. Let \( n_{eq} \) denote the total number of positive \( \rho_0^{(0)} \)’s that appear in (2.21). This is the number of \( D_0 \)’s that are required to obtain a realization of \( s_0(f, T) \). It is immediate that \( n_{eq} \) is bounded above by \( n_{tot} = \sum_{i=1}^{K} \sum_{j=0}^{\eta - 1} \rho_0^{(j)} \), which is a Poisson random variable with parameter \( c n_{tot} \), where

\[
R_{tot} = \sum_{i=1}^{K} \sum_{j=0}^{\eta - 1} \frac{1}{k_0(X_0(\sigma_j, \theta))} \left| \frac{\partial \lambda_k(X_0(\sigma_j, \theta))}{\partial \theta} \right|.
\]

By picking a small \( c > 0 \), we can ensure that \( \mathbb{E}(R_{tot}) = c \mathbb{E}(R_{tot}) \) is small, which would also guarantee that \( n_{tot} \) and \( n_{eq} \) are small. Specifically, we choose a small positive integer \( M_0 \) (like 10, for instance) and set

\[
c = \frac{M_0}{\mathbb{E}(R_{tot})}.
\]

where \( \mathbb{E}(R_{tot}) \) is estimated using \( N_0 \) simulations of the process \( X_0 \) in the time interval \([0, T]\). The choice of \( N_0 \) is not critical and typically a small value (like 100, for example) is sufficient to provide a decent estimate of \( \mathbb{E}(R_{tot}) \). The role of parameter \( M_0 \) is also not very important in determining the efficiency of PPA. If \( M_0 \) increases, then \( \eta_{eq} \) increases as well, and the computational cost of generating each realization of \( s_0(f, T) \) becomes higher. However, as \( \eta_{eq} \) increases, the variance of \( \hat{s}_0(f, T) \) decreases and hence fewer realizations of \( \hat{s}_0(f, T) \) are required to achieve the desired statistical accuracy. These two effects usually offset each other and the overall efficiency of PPA remains relatively unaffected. Note that PPA provides unbiased estimates for the sensitivity values, regardless of the choice of \( N_0 \) and \( M_0 \).
3. The Poisson path algorithm

We now describe PPA which produces realizations of the random variable \( \hat{S}_d(f, T) \) (defined by (2.21)) for some observation time \( T \) and some output function \( f \). Since \( S_d(f, T) = E(\hat{S}_d(f, T)) \), we can estimate \( S_d(f, T) \) by generating \( N \) realizations \( s_d^{(1)}(f, T), \ldots, s_d^{(N)}(f, T) \) of the random variable \( \hat{S}_d(f, T) \) and then computing their empirical mean (2.2).

Let \( x_0 \) denote the initial state of the process \( X_0 \) and assume that the function \( rand() \) returns independent samples from the uniform distribution on \([0,1]\). We simulate the paths of our CTMC using Gillespie’s SSA [16]. When the state is \( x \), the next time increment (\( \Delta t \)) and reaction index (\( k \)) is given by the function \( SSA(x) \) (see the electronic supplementary material, algorithm 1 in §2). We first fix a normalization parameter \( c \) according to (2.23), by estimating \( E(R_{\text{tot}}) \) using \( N_0 \) simulations of the CTMC (see the electronic supplementary material, algorithm 2 in §2).

Once \( c \) is chosen, a single realization of \( \hat{S}_d(f, T) \) can be computed using \( \text{GENERATESAMPLE}(x_0, T, c) \) (algorithm 1). This method simulates the process \( X_0 \) according to SSA and at each state \( x \) and jump time \( t \), the following happens:

- If \( x \) is an absorbing state (\( \lambda_0(x, \theta) = 0 \)), then \( t \) is the last jump time before \( T \) (where \( t = t_\sigma \)). For each \( k = 1, \ldots, K \) such that \( \partial \lambda_0(x, \theta)/\partial \theta \neq 0 \), the quantity \( \hat{\mu}_k \) (see (2.19)) is evaluated using \( \text{EVALUATEINTERG}(x + \xi_0, T - t) \) (see the electronic supplementary material, algorithm 3 in §2) and then used to update the sample value according to (2.21).

- If \( x \) is not an absorbing state, then \( t = t_\sigma \) for some jump time \( t_\sigma \) with \( \gamma < \sigma \). An exponential random variable \( \gamma \) (where \( \gamma = \gamma_1 \) in (2.21)) is generated, and for each \( k = 1, \ldots, K \), \( \partial \lambda_0(x, \theta)/\partial \theta \neq 0 \), the Poisson random variable \( n \) (where \( n = \gamma_1 \) in (2.21)) is also generated. If \( \gamma < (T - t) \) and \( n > 0 \), then the quantity \( \hat{\mu}_k \) (see (2.17)) is evaluated using \( \text{EVALUATECOUPLEDDIFFERENCE}(x, x + \xi_0, T - t - \gamma) \) (see the electronic supplementary material, algorithm 5 in §2) and then used to update the sample value according to (2.21). A Poisson random variable with parameter \( r \) is generated using the function \( \text{GENERATEPOISSON}(r) \) (see the electronic supplementary material, algorithm 4 in §2).

4. Numerical examples

In this section, we present many examples to compare the performance of PPA with the Girsanov method which is unbiased, and the two best-performing finite-difference schemes, CFD and CRP, which are of course biased. To compare the performance of different methods, we compare the time they need to produce a statistically accurate estimate of the true sensitivity value \( S_d(f, T) \) given by (1.1). Our first task is to find a way to judge that an estimate of \( S_d(f, T) \) is statistically accurate.

A common way to assess statistical accuracy of an estimator (biased or unbiased) is by computing its mean-squared error (m.s.e.), which is the sum of its variance and its bias (squared). However m.s.e. is not well-suited for our situation. To see this, consider two sensitivity estimators whose means are \( \mu_1 = 2 \) and \( \mu_2 = 1.1 \), and standard deviations are \( \sigma_1 = 0.1 \) and \( \sigma_2 = 1 \). Assuming that the true sensitivity value is \( s_0 = 1 \) and the distribution of the estimators is approximately normal, the two estimators have the same m.s.e., which is equal to \( (\mu_1 - s_0)^2 + \sigma_1^2 = (\mu_2 - s_0)^2 + \sigma_2^2 = 1.01 \). However, the second estimator is clearly more accurate than the first one, because it is far more likely to yield a sensitivity estimate which is ‘close’ to the actual value \( s_0 \). This explains how m.s.e. can be misleading and therefore we now describe another method to compare the statistical accuracy of estimators.

Algorithm 1. Generates one realization of \( S_d(f, T) \) according to (2.21).

```
1: function GENERATESAMPLE(x0, T, c)
2:     Set x = x_0, t = 0 and s = 0
3:     while t < T do
4:         Calculate (\Delta t, k) = SSA(x)
5:         Update \Delta t \leftarrow \min(\Delta t, T - t)
6:         if \lambda_0(x, \theta) > 0 then
7:             Set \gamma = - log(rand()) / \lambda_0(x, \theta)
8:             end if
9:         for k = 1 to K do
10:            Set r = \frac{\partial \lambda_0(x, \theta)}{\partial \theta} and \beta = \text{Sign}\left(\frac{\partial \lambda_0(x, \theta)}{\partial \theta}\right)
11:            if r > 0 then
12:               if \lambda_0(x, \theta) = 0 then
13:                  Update s \leftarrow s + \left(\frac{\partial \lambda_0(x, \theta)}{\partial \theta}\right)
14:               else
15:                  Set n = \text{GENERATEPOISSON}(r / \lambda_0(x, \theta))
16:                  if \gamma < (T - t) then
17:                     Update s \leftarrow s + \left(\frac{\partial \lambda_0(x, \theta)}{\partial \theta}\right)(f(x + \xi_k) - f(x)) \left(\Delta t - \frac{1}{\lambda_0(x, \theta)}\right)
18:                  if n > 0 then
19:                     Update s \leftarrow s + \left(\frac{\beta n}{\theta}\right)
20:                  end if
21:               end else
22:            end if
23:         end for
24:     end while
25: return s
```

Suppose that a method estimates parameter sensitivity using $N$ samples whose mean is $\mu_N$ and standard deviation is $\sigma_N$ (see §2). Assume that the true value of $S(f, T)$ is $s_0$. Let $a = s_0 - 0.05|s_0|$ and $b = s_0 + 0.05|s_0|$, where $| \cdot |$ denotes the absolute value function. Then $[a, b]$ is the 5% interval around the true value $s_0$. Under the central limit approximation, the estimator $\hat{S}(f, T)$ can be seen as a normally distributed random variable with mean $\mu_N$ and variance $\sigma_N^2$. Hence, using (2.4) we can calculate the probability

$$p = P(\hat{S}(f, T) \in [a, b])$$

that the estimator lies within the 5% interval around $s_0$. Henceforth, we refer to $p$ as the confidence level for the estimator. The statistical accuracy of an estimate can be judged from the value of $p$: the higher the value of $p$, the more accurate is the estimate. As we calculate $p$ using the 5% interval around $s_0$, we ensure that the statistical precision is high or low depending on whether the sensitivity value is small or large.

The standard deviation $\sigma_N$ is inversely proportional to the number of samples $N$ (see §2), which shows that $\sigma_N \to 0$ as $N \to \infty$. For an unbiased scheme (Girsanov and PPA), we have $\mu_N \to s_0$ as $N \to \infty$, due to the law of large numbers and therefore (2.4) implies that the confidence level $p$ can be made as close to 1 as desired by picking a large sample size $N$. However, the same is not true for finite-difference schemes such as CFD and CRP. In such schemes, $\mu_N \to S_{\partial f}(f, T)$ as $N \to \infty$, where $S_{\partial f}(f, T)$ is generally different from $S(f, T) = s_0$ because of the bias. If this bias is large, then $S_{\partial f}(f, T)$ may lie outside the 5% interval $[a, b]$ around $s_0$, and hence $p$ is close to 0 for large $N$. Therefore, if high confidence levels are desired with finite-difference schemes, then a small $h$ must be chosen to ensure that the bias $[S_{\partial f}(f, T) - S(f, T)]$ is low. However, the variance $\sigma_N^2$ of the associated estimator scales as $1/h$ (see §2.1), implying that if $h$ is too small then an extremely large sample size is required to achieve high confidence levels, and this can impose a heavy computational burden.

In this paper, we compare the efficiency of different methods by measuring the CPU time $t$ that is needed to produce a sample with least possible size, such that the corresponding estimate has a certain threshold confidence level. This threshold confidence level is close to 1 (either 0.95 or 0.99) and hence the produced estimate is statistically quite accurate. Previous discussion suggests that for finite-difference schemes, there is a trade-off between the statistical accuracy and the computational burden. This trade-off is illustrated in figure 1 which depicts four cases that correspond to four values of $h$ arranged in decreasing order. We describe these cases below.

**Case 1.** Here, $h$ is large and so the computational burden is low. However, due to the large bias, the estimator distribution (normal curve in figure 1) puts very little mass on the 5% interval around the true sensitivity value. Therefore, the confidence level $p$ is low and the estimate is unacceptable.

**Case 2.** Now $h$ is smaller, the computational burden is higher and the bias is lower. However, the estimator distribution still does not put enough mass on the 5% interval and so the value of $p$ is not high enough for the estimate to be acceptable.

**Case 3.** Now $h$ is even smaller and so the computational burden is higher, but fortunately the bias is small enough to ensure that the estimator distribution puts nearly all its mass on the 5% interval. Therefore, the confidence level $p$ is close to 1 and the estimate is acceptable.
Case 4. Here, $h$ is smallest and so the computational burden is the highest among all the cases. However in comparison to Case 3, the additional computational burden is unnecessary because there is only a slight improvement in the bias and the confidence level.

To use a finite-difference scheme efficiently, we need to know the largest value of $h$ for which an estimate with a high confidence level can be produced (Case 3 in figure 1). However, in general, it is impossible to determine this value of $h$. A blind choice of $h$ is likely to correspond to Cases 1 or 4 in figure 1. To make matters worse, since the true sensitivity value $S_h(f, T)$ is usually unknown, one cannot measure the statistical accuracy without re-estimating $S_h(f, T)$ using an unbiased method. Most practitioners who use finite-difference schemes assume that the statistical accuracy is high if $h$ is small. However, this is not true in general (see example 4.1). Therefore, if high accuracy is desired, then an unbiased method should be preferred over finite-difference schemes.

As it is difficult to determine the optimum value of $h$, we adopt the following strategy to gauge the efficiency of a finite-difference scheme. We find the largest value of $h$ in the sequence 0.1, 0.01, 0.001, . . . , for which the threshold confidence level $p$ is achievable for a large enough sample size. For performance comparison, we only use the CPU time corresponding to this value of $h$ and we disregard all the CPU time wasted in testing other values of $h$. We have intentionally made our comparison strategy more lenient towards the finite-difference schemes, in order to compensate for the arbitrariness in choosing the ‘test’ values of $h$.

To compute the confidence level $p$, we require the exact sensitivity value $S_h(f, T)$. In some examples, $S_h(f, T)$ can be analytically evaluated and the computation of $p$ is straightforward. In other examples, where $S_h(f, T)$ cannot be explicitly computed, we use PPA to produce an estimate with a very low sample variance and since PPA is unbiased, this estimate would be close to the true sensitivity value $S_h(f, T)$ and hence it can be used in the calculation of $p$.

We now discuss the examples. In all the examples, the propensity functions $A_i$ follow mass-action kinetics unless stated otherwise. Our method PPA depends on two parameters $N_0$ and $M_0$ (see §2.3) and in this paper we always use $N_0 = 100$ and $M_0 = 10$. In each example, we provide a bar chart with the CPU times required by different methods to produce an estimate with the desired confidence level $p = 0.95$ or $p = 0.99$. This facilitates an easy performance comparison between various methods. The exact values of the CPU times, sample size $N_0$ estimator mean, estimator standard derivation $\sigma_{\hat{p}}$ and $h$ (for finite-difference schemes) are provided in the electronic supplementary material, §3.

**Example 4.1. (Single-species birth–death model).** Our first example is a simple birth–death model in which a single species $S$ is created and destroyed as

\[ \theta_1 \rightarrow S \xrightarrow{\theta_2} \emptyset. \]

Let $\theta_1 = \theta_2 = 0.1$ and assume that the sensitive parameter is $\theta = \theta_2$. Suppose that $X(0) = 0$ and let $(X(t))_{t \geq 0}$ be the CTMC representing the reaction dynamics. Hence the population of $S$ at time $t$ is $X(t) \in N_0$. For $f(x) = x$, we wish to estimate

\[ S_h(f, T) = \frac{\partial}{\partial \theta} \mathbb{E}(f(X(T))) \bigg| \frac{\partial}{\partial \theta} f(X(T)), \]

for $T = 20$ and $100$. As the propensity functions of this network are affine, we can compute $S_h(f, T)$ exactly (see the electronic supplementary material, §3) and use this value in computing the confidence level $p$ of an estimate. We estimate $S_h(f, T)$ with all the methods with two threshold confidence levels ($p = 0.95$ and $p = 0.99$) and plot the corresponding CPU times in figure 2. For CFD and CRP, the desired confidence level was achieved in all the cases for $h = 0.01$.

From figure 2, it is immediate that PPA generally performs better than the other methods. To measure this gain in performance more quantitatively, we compute the average speed-up factor of PPA with respect to other methods. For each threshold confidence level, this factor is calculated by simply taking the ratio of the aggregate CPU times required by a certain method and PPA to perform all the sensitivity estimations. In this example, this aggregate involves the CPU times for $T = 20$ and $T = 100$. These average speed-up factors are presented in table 1. Note that in this example, PPA is significantly faster than CRP but only slightly faster than CFD and the Girsanov method.

Now we use this example to demonstrate the pitfalls of choosing a ‘wrong’ value of $h$ for finite-difference schemes. We know from figure 2 that for $T = 100$, the desired confidence level of 0.99 was achieved with $h = 0.01$. This ‘right’ value of $h$ corresponds to Case 3 in figure 1. If we select a higher value of $h$, such as $h = 0.1$ or $h = 0.05$, then with 10 000 samples we see that the achieved confidence level $p$ is much lower than the desired value 0.99 (table 2), indicating that the estimate is not sufficiently accurate. Note that $h = 0.1$ and $h = 0.05$ correspond to Cases 1 and 2 in figure 1. Observe that for $h = 0.1$, the estimate produced by finite-difference schemes ($\mu_{\hat{p}}$) is more than 30% off from the exact sensitivity value of $-9.995$, which shows that the bias can be large for $h$ that appears small. On the other hand, if we select $h$ which is really small, such as $h = 0.001$, then we do produce an estimate with the desired confidence level of 0.99 (see table 2), but the required sample size is very large and hence the computational burden is much higher than the ‘right’ value of $h$, which is $h = 0.01$. The value $h = 0.001$ corresponds to Case 4 in figure 1.

**Example 4.2. (Gene expression network).** Our second example considers the model for gene transcription given in [2]. It has three species, Gene ($G$), mRNA ($M$) and protein ($P$), and there are four reactions given by

\[ C \xrightarrow{\theta_1} G + M \quad M \xrightarrow{\theta_2} P \quad M \xrightarrow{\theta_3} \emptyset \quad P \xrightarrow{\theta_4} \emptyset. \]

The rate of translation of a gene into mRNA is $\theta_1$, while the rate of transcription of mRNA into protein is $\theta_2$. The degradation of mRNA and protein molecules occurs at rates $\theta_3$ and $\theta_4$, respectively. Typically, $\theta_3 >> \theta_4$ implying that a protein molecule lives much longer than a mRNA molecule. In accordance with the values given in [2] for lacA gene in bacterium E. coli, we set $\theta_1 = 0.6 \text{ min}^{-1}$, $\theta_2 = 1.7329 \text{ min}^{-1}$ and $\theta_3 = 0.3466 \text{ min}^{-1}$. Our sensitive parameter is $\theta = \theta_4$.

Let $(X(t))_{t \geq 0}$ be the CTMC representing the dynamics with initial state $(X_0(0), X_2(0)) = (0, 0)$. For any time $t$, $(X(t) = (X_1(t), X_2(t)))$, where $X_1(t)$ and $X_2(t)$ are the number of mRNA and protein molecules, respectively. Define $f : N_0 \rightarrow \mathbb{R}$ by $f(x_1, x_2) = x_2$. We would like to estimate the logarithmic
Figure 2. Efficiency comparison for birth–death model with two threshold confidence levels: $p = 0.95$ (a) and $p = 0.99$ (b) and two observation times: $T = 20$ and $T = 100$. The exact CPU time is shown at the top of each bar. Note that efficiency is inversely related to the CPU time. For finite-difference schemes, $h = 0.01$ was used in all the cases.

Table 1. Birth–death model: average speed-up factors of PPA with respect to other methods.

<table>
<thead>
<tr>
<th>$p$</th>
<th>Giranov</th>
<th>CFD</th>
<th>CRP</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.95</td>
<td>2.16</td>
<td>1.75</td>
<td>8.59</td>
</tr>
<tr>
<td>0.99</td>
<td>1.5</td>
<td>1.22</td>
<td>7.25</td>
</tr>
</tbody>
</table>

The CPU times required by all the methods for the two threshold confidence levels ($p = 0.95$ and $p = 0.99$) are presented in figure 3. For CFD and CRP, the desired confidence level was achieved in all cases with $h = 0.1$. We compute the average speed-up factor for PPA with respect to other methods, by aggregating CPU times for the two values of $\theta$ and $T$. These average speed-up factors are presented in table 3 and they show that PPA can be far more efficient than the Girsanov method, but its performance is comparable to CFD and CRP. Furthermore, the results in figure 3 indicate that unlike PPA, the performance of the Girsanov method deteriorates drastically as $\theta$ gets smaller: on average PPA is 76 times faster for $\theta = 0.0693$ min$^{-1}$ and 2413 times faster for $\theta = 0.0023$ min$^{-1}$. This deterioration in the performance of the Girsanov method is consistent with the results in [15]. Even though PPA is only slightly faster than CFD and CRP in this example, its unbiasedness guarantees that one does not require any additional checks to verify its accuracy. This gives PPA a distinct advantage over finite-difference schemes.

Example 4.3. (Circadian clock network). We now consider the network of a circadian clock studied by Vilar et al. [21]. It is a large network with nine species $S_1, \ldots, S_9$ and 16 reactions. These reactions along with the rate constants are given in table 4. Let $(X(t))_{t \geq 0}$ be the CTMC corresponding to the dynamics with initial state $X(0) = (1, 0, 0, 0, 0, 1, 0, 0, 0, 0)$. For $T = 5$ and $f(x) = x_i$, we wish to estimate

$$S_\theta(f, T) = \frac{\partial}{\partial \theta} E(f(X(T))) = \frac{\partial}{\partial \theta} E(X_i(T)),$$

for $\theta = \theta_i$ and $i = 5, 6, 8, 12$ and 14.

Owing to the presence of many bimolecular interactions, exact values of $S_\theta(f, T)$ are difficult to compute analytically, but they are required in computing the confidence levels of estimates. To solve this problem, we use PPA to produce sensitivity estimates with very low variance, and then use these approximate values in computing confidence levels.
approximate values are given in the electronic supplementary material, §3.

The CPU times needed by all the methods for two threshold confidence levels (\( p = 0.95 \) and \( p = 0.99 \)) are shown in figure 4. As in the previous example, for CFD and CRP, the desired confidence level was achieved in all the cases with maximum allowed \( h \) value (\( h = 0.1 \)). Results in figure 4 indicate that PPA has much better performance than the Girsanov method. In fact in many cases, the Girsanov method could not produce an estimate with the desired confidence level even with the maximum sample size of \( 10^7 \). These values are marked with asterisks in figure 4 and the confidence level they correspond to is shown in parentheses.

If we compare the performance of PPA with finite-difference schemes, then we find that PPA is slower for all parameters except \( u = u_{14} \), in which case PPA is far more efficient than either CFD or CRP. The overall efficiency of PPA can be judged from its average speed-up factor with respect to other methods. These average speed-up factors are displayed in table 5 and they are calculated by aggregating CPU times for the five values of \( u \). As in many cases the Girsanov method did not yield an estimate with the desired confidence level even with the maximum sample size, the actual speed-up factor of PPA with respect to this method may be much higher than what is shown in table 5.

Example 4.4. (Genetic toggle switch). As our last example, we look at a simple network with nonlinear propensity functions. Consider the network of a genetic toggle switch proposed by Gardner et al. [22]. This network has two species \( U \) and \( V \)
that interact through the following four reactions:

\[ \emptyset \xrightarrow{\lambda_1} \text{U}, \quad \text{U} \xrightarrow{\lambda_2} \emptyset, \quad \emptyset \xrightarrow{\lambda_3} \text{V} \quad \text{and} \quad \text{V} \xrightarrow{\lambda_4} \emptyset, \]

where the propensity functions \( \lambda_i \) are given by

\[ \lambda_1(x_1, x_2) = \frac{\alpha_1}{1 + x_2}, \quad \lambda_2(x_1, x_2) = x_1, \quad \lambda_3(x_1, x_2) = x_1, \quad \lambda_4(x_1, x_2) = x_2. \]

In the above expressions, \( x_1 \) and \( x_2 \) denote the number of molecules of \( \text{U} \) and \( \text{V} \), respectively. We set \( \alpha_1 = 50, \alpha_2 = 16, \beta = 2.5 \) and \( \gamma = 1 \). Let \( (X(t))_{t \geq 0} \) be the CTMC representing the dynamics with initial state \( (X_1(0), X_2(0)) = (0, 0) \). For \( T = 10 \) and \( f(x) = x_1 \), our goal is to estimate

\[ S_{\theta}(f, T) = \frac{\partial}{\partial \theta} \mathbb{E}(f(X(T))) = \frac{\partial}{\partial \theta} \mathbb{E}(X_1(T)), \]

for \( \theta = \alpha_1, \alpha_2, \beta \) and \( \gamma \). In other words, we would like to measure the sensitivity of the mean of the number of \( \text{U} \) molecules at time \( T = 10 \), with respect to all the model parameters.

As the propensity functions \( \lambda_1 \) and \( \lambda_3 \) are nonlinear, it is difficult to compute \( S_{\theta}(f, T) \) exactly. As before, we obtain close approximations of \( S_{\theta}(f, T) \) using PPA (values given in the electronic supplementary material, §3), and then use these values in computing confidence levels of estimates. The CPU times needed by all the methods for two threshold confidence levels \( (p = 0.95 \) and \( p = 0.99 \) are presented in figure 5. As in the

Table 4. Circadian clock network: list of reactions.

<table>
<thead>
<tr>
<th>no.</th>
<th>reaction</th>
<th>rate constant</th>
<th>no.</th>
<th>reaction</th>
<th>rate constant</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>( S_6 + S_2 \rightarrow S_7 )</td>
<td>( \theta_1 = 1 )</td>
<td>9</td>
<td>( S_2 \rightarrow \emptyset )</td>
<td>( \theta_9 = 100 )</td>
</tr>
<tr>
<td>2</td>
<td>( S_7 \rightarrow S_6 + S_2 )</td>
<td>( \theta_2 = 50 )</td>
<td>10</td>
<td>( S_8 \rightarrow S_6 + S_3 )</td>
<td>( \theta_{10} = 0.01 )</td>
</tr>
<tr>
<td>3</td>
<td>( S_6 + S_2 \rightarrow S_8 )</td>
<td>( \theta_3 = 50 )</td>
<td>11</td>
<td>( S_9 \rightarrow S_6 + S_3 )</td>
<td>( \theta_{11} = 50 )</td>
</tr>
<tr>
<td>4</td>
<td>( S_9 \rightarrow S_6 + S_2 )</td>
<td>( \theta_4 = 500 )</td>
<td>12</td>
<td>( S_7 \rightarrow \emptyset )</td>
<td>( \theta_{12} = 0.5 )</td>
</tr>
<tr>
<td>5</td>
<td>( S_7 \rightarrow S_6 + S_1 )</td>
<td>( \theta_5 = 10 )</td>
<td>13</td>
<td>( S_7 \rightarrow S_6 + S_4 )</td>
<td>( \theta_{13} = 5 )</td>
</tr>
<tr>
<td>6</td>
<td>( S_6 \rightarrow S_6 + S_1 )</td>
<td>( \theta_6 = 50 )</td>
<td>14</td>
<td>( S_6 \rightarrow \emptyset )</td>
<td>( \theta_{14} = 0.2 )</td>
</tr>
<tr>
<td>7</td>
<td>( S_1 \rightarrow \emptyset )</td>
<td>( \theta_7 = 1 )</td>
<td>15</td>
<td>( S_6 \rightarrow S_6 + S_3 )</td>
<td>( \theta_{15} = 20 )</td>
</tr>
<tr>
<td>8</td>
<td>( S_1 \rightarrow S_6 + S_2 )</td>
<td>( \theta_8 = 1 )</td>
<td>16</td>
<td>( S_6 \rightarrow S_6 )</td>
<td>( \theta_{16} = 1 )</td>
</tr>
</tbody>
</table>

Table 5. Circadian clock network: average speed-up factors of PPA with respect to other methods.

<table>
<thead>
<tr>
<th>( p )</th>
<th>Girsanov</th>
<th>CFD</th>
<th>CRP</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.95</td>
<td>355.08</td>
<td>12.76</td>
<td>15.73</td>
</tr>
<tr>
<td>0.99</td>
<td>189.82</td>
<td>13.44</td>
<td>15.78</td>
</tr>
</tbody>
</table>

Figure 4. Efficiency comparison for circadian clock network with two threshold confidence levels: \( p = 0.95 \) (a) and \( p = 0.99 \) (b) and observation time \( T = 5 \). The values marked with asterisks correspond to cases where the desired confidence level was not reached even with the maximum sample size of \( 10^5 \). The final confidence level reached is shown in parentheses. For finite-difference schemes, \( h = 0.1 \) was used in all the cases.
previous two examples, both CFD and CRP yield an estimate with the desired confidence level with \( h = 0.1 \) in all the cases.

From figure 5, it can be easily seen that PPA is the best-performing method. The average speed-up factors for PPA with respect to other methods are given in table 6. They are calculated by aggregating CPU times for the four values of \( \theta \). Note that in this example, the unbiased schemes perform much better than the finite-difference schemes, possibly due to nonlinear parameter-dependence of the propensity functions.

The above examples illustrate that in producing an estimate with a specified statistical accuracy, PPA can be much more efficient than other methods, both biased and unbiased. The finite-difference schemes may outperform PPA in some cases but they can also be much slower in other cases. As it is difficult to independently verify the statistical accuracy of finite-difference schemes, PPA is an attractive alternative to these schemes, as its unbiasedness presents a theoretical guarantee for statistical accuracy.

5. Conclusion and future work

We provide a new unbiased method for estimating parameter sensitivity of stochastic reaction networks. Using a result from our recent paper [15], we construct a random variable whose expectation is the required sensitivity value. We then present a simple procedure, called the PPA, to compute the realizations of this random variable. This gives us a way to generate samples for estimating the parameter sensitivity. Our method can be viewed as an improved version of the APA, that we presented in [15]. Unlike APA, the proposed method is easy to implement, has low memory requirements and it works well for large networks and large observation times.

Through examples, we compare the performance of PPA with other methods for sensitivity estimation, both biased and unbiased. Our results indicate that PPA easily outperforms the unbiased Girsanov method. Moreover in many cases, it can even be faster than the best-performing finite-difference schemes (CRP and CFD) in producing a statistically accurate estimate. This makes PPA an appealing method for sensitivity estimation because it is computationally efficient and one does not have to tolerate a bias of an unknown size that is introduced by finite-difference approximations.

In our method, we simulate the paths of the underlying Markov process using Gillespie’s SSA [16]. However, SSA is very meticulous in the sense that it simulates the occurrence of each and every reaction in the dynamics. This can be very cumbersome for large networks with many reactions. To resolve this problem, a class of \( \tau \)-leaping methods have been developed [23,24]. These methods are approximate in nature, but they significantly reduce the computational effort that is required to simulate the reaction dynamics. In future, we would like to develop a version of PPA that uses a \( \tau \)-leaping method instead of SSA and still produces an accurate estimate for the sensitivity values. Such a method would greatly simplify the sensitivity analysis of large networks. Recently, hybrid approaches that combine several existing sensitivity estimation methods in a multi-level setting have
been introduced in [25]. Integrating PPA into this hybrid framework may further enhance its efficiency. This will be investigated in a future work.

Funding statement. We acknowledge funding from Human Frontier Science Programme (RGP0061/2011).

References


