BMC Systems Biology

RESEARCH ARTICLE Open Access

Solving the chemical master equation using sliding windows

Verena Wolf*[1], Rushil Goel[2], Maria Mateescu[3] and Thomas A Henzinger[4]

Abstract

Background: The chemical master equation (CME) is a system of ordinary differential equations that describes the evolution of a network of chemical reactions as a stochastic process. Its solution yields the probability density vector of the system at each point in time. Solving the CME numerically is in many cases computationally expensive or even infeasible as the number of reachable states can be very large or infinite. We introduce the sliding window method, which computes an approximate solution of the CME by performing a sequence of local analysis steps. In each step, only a manageable subset of states is considered, representing a "window" into the state space. In subsequent steps, the window follows the direction in which the probability mass moves, until the time period of interest has elapsed. We construct the window based on a deterministic approximation of the future behavior of the system by estimating upper and lower bounds on the populations of the chemical species.

Results: In order to show the effectiveness of our approach, we apply it to several examples previously described in the literature. The experimental results show that the proposed method speeds up the analysis considerably, compared to a global analysis, while still providing high accuracy.

Conclusions: The sliding window method is a novel approach to address the performance problems of numerical algorithms for the solution of the chemical master equation. The method efficiently approximates the probability distributions at the time points of interest for a variety of chemically reacting systems, including systems for which no upper bound on the population sizes of the chemical species is known a priori.

Background

Experimental studies have reported the presence of stochastic mechanisms in cellular processes [1-9] and therefore, during the last decade, *stochasticity* has received much attention in systems biology [10-15]. The investigation of stochastic properties requires that computational models take into consideration the inherent randomness of chemical reactions. Stochastic kinetic approaches may give rise to dynamics that differ significantly from those predicted by deterministic models, because a system might follow very different scenarios with non-zero likelihoods.

Under the assumption that the system is spatially homogeneous and has fixed volume and temperature, at a each point in time the state of a network of biochemical reactions is given by the population vector of the involved chemical species. The temporal evolution of the system can be described by a Markov process [16], which is usually represented as a system of ordinary differential equations (ODEs), called the *chemical master equation* (CME).

The CME can be analyzed by applying numerical solution algorithms or, indirectly, by generating trajectories of the underlying Markov process, which is the basis of Gillespie's *stochastic simulation algorithm* [17,18]. In the former case, the methods are usually based on a matrix description of the Markov process and thus primarily limited by the size of the system. A survey and comparisons of the most established methods for the numerical analysis of discrete-state Markov processes are given by Stewart [19]. These methods compute the probability density vector of the Markov process at a number of time points up to an a priori specified accuracy. If numerical solution algorithms can be applied, almost always they require considerably less computation time than stochastic simulation, which only gives estimations of the measures of interest. This is particularly the case if not only

* Correspondence: wolf@cs.uni-sb.de
[1] Computer Science Department, Saarland University, Saarbrücken, Germany
Full list of author information is available at the end of the article

means and variances of the state variables are estimated with stochastic simulation, but also the probability of certain events. However, for many realistic systems, the number of reachable states is huge or even infinite and, in this case, numerical solution algorithms may not be applicable. This depends mainly on the number of chemical species. In low dimensions (say <10) a direct solution of the CME is possible whereas in high dimensions stochastic simulation is the only choice. In the case of stochastic simulation estimates of the measures of interest can be derived once the number of trajectories is large enough to achieve the desired statistical accuracy. However, the main drawback of simulative solution techniques is that a large number of trajectories is necessary to obtain reliable results. For instance, in order to halve the confidence interval of an estimate, four times more trajectories have to be generated. Consequently, often stochastic simulation is only feasible with a very low level of confidence in the accuracy of the results.

In this paper, we mitigate the performance problems of numerical solution algorithms for the CME. Instead of a global analysis of the state space, we propose the *sliding window method*, which comprises a sequence of analyzes local to the significant parts of the state space. In each step of the sequence, we dynamically choose a time interval and calculate an approximate numerical solution for a manageable subset of the reachable states. In order to identify those states that are relevant during a certain time period, for each chemical species, we estimate an upper and lower bound on the population size. This yields the boundaries of a "window" in which most of the probability mass remains during the time interval of interest. As illustrated in Figure 1, the window "slides" through the state space when the system is analyzed in a stepwise fashion. In each step, the initial conditions are given by a vector of probabilities (whose support is illustrated in light gray), and a matrix is constructed to describe the part of the Markov process where the window (illustrated by the dashed rectangular) is currently located. Then the corresponding ODE is solved using a standard numerical algorithm, and the next vector (illustrated in dark gray) is obtained.

We focus on two specific numerical solution methods, the *uniformization method* and the *Krylov subspace method*. We compare their efficiency when they are used to solve the ODEs that arise during the sliding window iteration. We also compare the sliding window method to the numerical algorithms applied in a global fashion, that is, to all reachable states (not only to the states of the window), for systems of tractable size. We are interested in the probability distribution of the Markov process and not only in means and variances. These probabilities are difficult to estimate accurately with stochastic simulation. Therefore, we compare the solution obtained by the sliding window method only to numerical solution algorithms but not to stochastic simulation.

Recently, *finite state projection algorithms* (FSP algorithms) for the solution of the CME have been proposed [20,21]. They differ from our approach in that they are based solely on the structure of the underlying graph, whereas the sliding window method is based on the stochastic properties of the Markov process. The FSP algorithms start with an initial projection, which is expanded in size if necessary. The direction and the size of the expansion is chosen based on a qualitative analysis of the system in a breadth-first search manner. It is not clear how far the state space has to be explored in order to capture most of the probability mass during the next time step. Thus, if the projection size is too small, the computation has to be repeated with an expanded projection. Moreover, for most models, the location of the main portion of the probability mass follows a certain direction in the state space, whereas the expansion is done in all directions. Therefore, unnecessary calculations are carried out, because the projection contains states that are visited with a small probability. By contrast, in the sliding window approach, we determine the location and direction of the probability mass for the next computation step based on the reaction propensities and the length of the time step. The projection that we obtain is significantly smaller than the projection used in the FSP whereas the

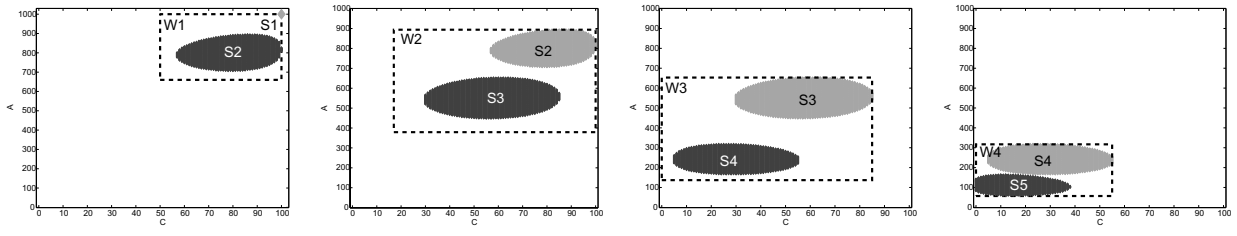

Figure 1 The sliding window method. In each iteration step, the window W_i captures the set S_i of states in which the significant part of the probability mass is initially located (light gray), the set S_{i+1} of states that are reached after a time step (dark gray), as well as the states that are visited in between.

accuracy of our approach is similar to the accuracy of the FSP. In this way we achieve large memory and computational savings, since the time complexity of our window construction is small compared to the calculation of the probability distribution of the window. In our simulations we never had to repeat the computation of the probabilities using a window of larger size.

The Fokker-Planck equation is an approximation of the CME, for which a solution can be obtained efficiently [22,23]. This approximation, however, does not take into account the discrete nature of the system, but changes the underlying model by assuming a continuous state space. Other approaches to approximate the probability distributions defined by the CME are based on sparse grid methods [24], spectral methods [25], or the separation of time scales [26,27]. The latter approach uses a quasi-steady state assumption for a subset of chemical species and calculates the solution of an abstract model of the system. In contrast, we present an algorithm that computes a direct solution of the CME. Our method is also related to tau-leaping techniques [18,28], because they require estimates of the upper and lower bounds on the population sizes of the chemical species, just as our method. The *time leap* must be sufficiently small such that the changes in the population vector do not significantly affect the dynamics of the system. Our method differs from the calculation of the leap in predicting the future dynamics for a dynamically chosen time period. More precisely, we determine the length of the next time step while approximating the future behavior of the process.

Here, we present the sliding window method in more detail and provide an additional comparison between uniformization and Krylov subspace methods for the solution of the window. Moreover, we have improved our implementation of the algorithm and evaluated it on more examples, such as the bistable toggle switch, which is reported in detail.

The remainder of this paper is organized as follows. We first describe the theoretical framework of our modeling approach in the Background Section. In the Results Section we present the sliding window method, and numerical solution approaches for the CME. Experimental results are given at the end of Section Results.

Stochastic model

We model a network of biochemical reactions as a Markov process that is derived from the stochastic chemical reaction kinetics [16,29]. A physical justification of Markovian models for coupled chemical reactions has been provided by Gillespie [17]. We consider a fixed reaction volume with n different chemical species that is spatially homogeneous and in thermal equilibrium. A state of the system is given by a vector $x \in \mathbb{N}_0^n$, where the i-th entry, denoted by x_i describes the number of molecules of type i. We assume that molecules collide randomly and chemical reactions occur at random times. By $R_1, ..., R_k$, we denote the different types of chemical reactions and with each type R_m, $m \in \{1, ..., k\}$, we associate a *propensity function* $\alpha_m : \mathbb{N}_0^n \to \mathbb{R}_{\geq 0}$. The propensity function is of the form

$$\alpha_m(x) = c_m \cdot \prod_{i=1}^{n} \binom{x_i}{l_i}, \tag{1}$$

where $c_m > 0$ is a constant and l_i is the number of molecules of type i that are consumed by a reaction of type R_m. The propensity $\alpha_m(x)$ determines the "speed" of the reaction R_m in x, as explained below. Note that $\prod_{i=1}^{k} \binom{x_i}{l_i}$ equals the number of all distinct combinations of reactants. Besides the propensity function, we associate a *change vector* $\nu^{(m)} \in ?^n$ with R_m that describes the effect of reaction type R_m. If x is the current state and $\alpha_m(x) > 0$ then $x + \nu^{(m)}$ is the state of the system after a reaction of type R_m. Note that $\alpha_m(x) > 0$ implies that $x + \nu^{(m)}$ contains no negative entries.

We denote the initial state of the system by $y \in \mathbb{N}_0^n$ and define $S \subseteq \mathbb{N}_0^n$ as the set of all states reachable from y via an arbitrary number of reactions, that is, S is the smallest subset of $\mathbb{N}_0^n$ such that $y \in S$ and $x' \in S$ iff there exists $m \in \{1, ..., k\}$ and $x \in S$ with $\alpha_m(x) > 0$ and $x + \nu^{(m)} = x'$. Note that S is countable but possibly infinite.

Example 1 *We describe an* enzyme-catalyzed substrate conversion *by the three reactions* $R_1 : E + S \to ES$, $R_2 : ES \to E + S$, $R_3 : ES \to E + P$. *This network involves four chemical species, namely, enzyme (E), substrate (S), complex (ES), and product (P) molecules. The change vectors are* $\nu^{(1)} = (-1, -1, 1, 0)$, $\nu^{(2)} = (1, 1, -1, 0)$, *and* $\nu^{(3)} = (1, 0, -1, 1)$. *For* $(x_1, x_2, x_3, x_4) \in \mathbb{N}_0^4$, *the propensity functions are*

$$\alpha_1(x_1,x_2,x_3,x_4) = c_1 \cdot x_1 \cdot x_2,\ \alpha_2(x_1,x_2,x_3,x_4) = c_2 \cdot x_3,\ \alpha_3(x_1,x_2,x_3,x_4) = c_3 \cdot x_3.$$

As above, the set of states reachable from the initial state $y = (y_1, y_2, y_3, y_4)$ *is finite because of the conservation laws* $y_1 = x_1 + x_3$ *and* $y_2 = x_2 + x_3 + x_4$, *where we assume that* $y_3 = y_4 = 0$.

Example 2 *We consider a gene expression model* [12], *which involves two chemical species, namely, mRNA (M) and protein (P). Transcription of a gene into mRNA is modeled by reaction* R_1 : ? T *M, translation of mRNA into protein by* R_2 : *M* T *M + P, degradation of mRNA by* R_3 : *M* T ?, *and degradation of proteins by* R_4 : *P* T ?. *A state is a pair* (x_M, x_P) $\mathbb{N}_0^2$. *If we assume that initially there are no mRNA molecules and no proteins in the system, i.e.,* $y = (0, 0)$, *then* $S = \mathbb{N}_0^2$ *and thus infinite. The propensity functions are*

$$\alpha_1(x_M,x_P) = c_1,\ \alpha_2(x_M,x_P) = c_2 \cdot x_M,\ \alpha_3(x_M,x_P) = c_3 \cdot x_M,\ \alpha_4(x_M,x_P) = c_4 \cdot \boldsymbol{x}_P.$$

Chemical master equation

We define a time-homogeneous, regular Markov process [30] (CTMC) $(X(t), t \in \mathbb{R}_{\geq 0})$ with state space $S \subseteq \mathbb{N}_0^n$. We assume that the state changes of X are triggered by the chemical reactions. Let y be the initial state of X, which means that $Pr(X(0) = y) = 1$. We assume that the probability of a reaction R_m occurring in the next infinitesimal time interval $[t, t + \tau)$, $\tau > 0$ is given by

$$\Pr(X(t+\tau) = x + v^{(m)} \mid X(t) = x) = \alpha_m(x) \cdot \tau.$$

For $x \in S$ we define the probability that X is in state x at time t by $p^{(t)}(x) = Pr(X(t) = x \mid X(0) = y)$. The *chemical master equation* (CME) describes the behavior of X by the differential equation [29]

$$\frac{\partial p^{(t)}(x)}{\partial t} = \sum_{\substack{m=1 \\ \alpha_m(x-v^{(m)})>0}}^{k} \alpha_m(x - v^{(m)}) \cdot p^{(t)}(x - v^{(m)}) - \sum_{\substack{m=1 \\ \alpha_m(x)>0}}^{k} \alpha_m(x) \cdot p^{(t)}(x).$$

In the sequel, a matrix description of Eq. (2) is more advantageous. It is obtained by defining the *infinitesimal generator matrix* $Q = (Q(x, x'))_{x, x' \in S}$ of the CTMC X by

$$Q(x,x') = \begin{cases} \alpha_m(x) & \text{if } x + v^{(m)} = x', \\ -\sum_{m=1}^{k} \alpha_m(x) & \text{if } x = x', \\ 0 & \text{otherwise,} \end{cases}$$

where we assume a fixed enumeration of the state space. Note that the row sums of the (possibly infinite) matrix Q are zero and $\lambda_x = -Q(x, x)$, the *exit rate* of state x, is the reciprocal value of the average residence time in x.

Let $T^{(0)}$ be equal to the identity matrix I, and, for $\tau > 0$, let $T(\tau)$ be the transition probability matrix for step τ with entries $T(\tau)(x, z) = Pr(X(t + \tau) = z \mid X(t) = x)$. The elements of $T(\tau)$ are differentiable and Q is the derivative of $T(\tau)$ at $\tau = 0$. If Q is given and X is known to be regular, $T(\tau)$ is uniquely determined by the *Kolmogorov backward and forward equations*

$$\frac{\partial T(\tau)}{\partial \tau} = Q \cdot T^{(\tau)}, \frac{\partial T(\tau)}{\partial \tau} = T^{(\tau)} \cdot Q. \tag{3}$$

with the general solution $T(\tau) = e^{Q\tau}$. Let $\boldsymbol{p}^{(t)}$ be the row vector with entries $p^{(t)}(x)$ for $x \in S$. Then the vector form of the CME is

$$\frac{\partial p(t)}{\partial t} = P^{(t)} \cdot Q. \tag{4}$$

If $\sup_{x \in S} \lambda_x < \infty$, Eq. (4) has the general solution

$$\boldsymbol{p}^{(t)} = \boldsymbol{p}^{(0)} e^{Qt}, \tag{5}$$

where the matrix exponential is given by $e^{Qt} = \sum_{i=0}^{\infty} \frac{(Qt)^i}{i!}$.

In the sequel, we will exploit the fact that the set $\{T(\tau) \mid \tau \geq 0\}$ is a *transition semi-group* and satisfies the *Chapman-Kolmogorov equations* [30] $T^{(\tau_1+\tau_2)} = T^{(\tau_1)} \cdot T^{(\tau_2)}$ for all $\tau_1, \tau_2 \geq 0$. Let $t_0, ..., t_r \in \mathbb{R}_{\geq 0}$ be such that $t_0 < \cdots < t_r$. Then,

$$\begin{aligned} \boldsymbol{p}^{(t_t)} &= \boldsymbol{p}^{(t_1)} \cdot T^{(t_2-t_1)} \cdot \ldots \cdot T^{(t_r-t_{r-1})} \\ &\quad \ddots \qquad\qquad \vdots \\ &= \boldsymbol{p}^{(t_{t-1})} \cdot T^{(t_t-t_{r-1})}. \end{aligned} \tag{6}$$

This means that, for $t_0 = 0$ and $t_r = t$, we obtain $\boldsymbol{p}^{(t)}$ by the iterative scheme in Eq. (3) for $t_1 - t_0$, $t_2 - t_1$, ..., $t_r - t_{r-1}$.

If the state space if infinite we can only compute approximations of $\boldsymbol{p}^{(t)}$. But even if Q is finite, several factors can hamper the efficient solution of the matrix exponential in Eq. (5). First of all, the size of the matrix Q might be large because it grows exponentially with the number of state variables. However, usually Q is sparse, as the number of reaction types is small compared to the number of states. But even when Q is sparse often only an approximate solution can be computed efficiently. Adding up a sufficiently large number of terms of the infinite sum $\sum_{i=0}^{\infty} \frac{(Qt)^i}{i!}$ is numerically unstable, as Q contains strictly positive and negative entries, leading to severe round-off errors [31]. Various numerical solution methods exist for systems of first-order linear equations of the form of Eq. (4). However, many of them are not useful as they do not preserve the sparseness of Q. Several surveys and comparisons exist in literature [19,32,33]. Most popular are methods based on uniformization [34,35], approximations in the Krylov subspace [36], or numerical integration [37,38]. We will describe the former two methods in more detail in the section on numerical solution methods.

Results

Sliding window method

The key idea of the algorithm proposed in this paper is to calculate an approximation $\hat{\boldsymbol{p}}^{(t)}$ of $\boldsymbol{p}^{(t)}$ in an iterative fashion, as described in Eq. (6). More precisely, we compute a sequence of approximations $\hat{\boldsymbol{p}}^{(t_1)}, \ldots, \hat{\boldsymbol{p}}^{(t_r)}$ such that for a subset W_j of the state space, $j \in \{1, ..., r\}$, $\hat{p}^{(t_j)}(x) \approx p^{(t_j)}(x)$ for all $x \in W_j$. The sets W_1, ..., W_r are called *windows*, and we assume that W_j contains the states at which (most of) the probability mass is concentrated during the time interval $[t_{j-1}, t_j)$. We discuss the construction of the window W_j later.

Let Q_j be the matrix that refers to W_j, i.e., we define $Q_j(x, x') = Q(x, x')$ if $x, x' \in W_j$, and $Q_j(x, x') = 0$ otherwise. Note that for the simplicity of our presentation we keep a fixed enumeration of S and assume that each Q_j has the same size as Q. However, the implementation of the method considers only the finite submatrix of Q_j that contains entries of states in W_j. For $\tau_j = t_j - t_{j-1}$, we define

$$\hat{\boldsymbol{p}}^{(t_j)} = \hat{\boldsymbol{p}}^{(t_{j-1})} D_j e^{Q_j \tau_j}, \; j \in \{1, 2, \ldots, r\}, \tag{7}$$

where $\hat{\boldsymbol{p}}^{(t_0)} = (\mathbf{1}_y)^T$ and D_j is the diagonal matrix whose main diagonal entries are one for $x \in W_j$ and zero otherwise. The row vector $(\mathbf{1}_y)^T$ is one at position y and zero otherwise.

In the j-th step, the matrix $e^{Q_i \tau_j}$ contains the probabilities to move in τ_j time units within W_j from one state to another. As initial probabilities, Eq. (7) uses the approximations $\hat{p}^{(t_{j-1})}(x)$ for all states $x \in W_j$. The diagonal matrix ensures that the probability mass located in $W_{j-1} \setminus W_j$ is ignored during the computation, that is, only elements of the intersection $W_{j-1} \cap W_j$ can have nonzero entries in the vector $\hat{\boldsymbol{p}}^{(t_{j-1})} D_j$. This is necessary because Q_j does not contain the transition rates of states outside of W_j (these states are absorbing). Intuitively, the vector $\hat{\boldsymbol{p}}^{(t_j)}$ describes the location of the probability mass after moving within W_j.

Although Q_j is not the generator of a CTMC, Eq. (7) has a simple interpretation for all states $x \in W_j$. Let us fix j for the moment, and let the CTMC $\hat{X}$ be identical to X, except that all states $x' \notin W_j$ are absorbing (i.e., once x' is reached, it cannot be left). Let the initial probability distribution of $\hat{X}$ be such that $Pr(\hat{X}(0) = x) = p^{(t_{j-1})}(x)$ for all $x \in W_j$. Then $\hat{p}^{(t_j)}(x) = Pr(\hat{X}(t_j - t_{j-1}) = x)$, for all $x \in W_j$. For all j, the vectors $\hat{\boldsymbol{p}}^{(t_j)}$ are substochastic, and the sum of their entries decreases in each step, i.e,

$$1 = \sum_{x \in S} \hat{p}^{(t_0)}(x) \geq \sum_{x \in S} \hat{p}^{(t_1)}(x) \geq \ldots \geq \sum_{x \in S} \hat{p}^{(t_r)}(x).$$

Probability mass is "lost", because we do not consider the entries $\hat{p}^{(t_{j-1})}(x)$ for $x \in W_{j-1} \setminus W_j$, as we multiply with D_j. In addition, we lose the probability to leave W_j within the next τ_j time units because $e^{Q_i \tau_j}$ is a substochastic matrix. If, for all j, during the time interval $[t_{j-1}, t_j)$ most of the probability mass remains within W_j, then the approximation error $p^{(t_j)}(x) - \hat{p}^{(t_j)}(x)$ is small for all $x \in S$. The probability mass that is lost after j steps due to the approximation is given by

$$\eta_j = 1 - \sum_{x \in W_j} \hat{p}^{(t_j)}(x). \tag{8}$$

Thus, if Eq. (7) is solved exactly, the total approximation error of the sliding window method is η_r. Note that the error in Eq. (8) is the sum of the errors of all components of the vector $p^{(t_j)}$.

Window construction

In each step of the iteration the window W_j must be chosen such that the error η_j is kept small. This is the case if W_j satisfies the following conditions: (a) with a sufficiently high probability $X(t_{j-1}) \in W_j$, (b) the probability of leaving W_j within the time interval $[t_{j-1}, t_j)$ is sufficiently small.

Requirement (a) implies that W_j contains a significant part of the support of $\boldsymbol{p}^{(t_{j-1})}$, that is, a subset $S_j \subseteq S$ such that $1 - \sum_{x \in S_j} p^{(t_{j-1})}(x)$ is small. In the first step we set $S_1 = \{y\}$. For $j > 1$, the window W_j is constructed after $\hat{\boldsymbol{p}}^{(t_{j-1})}$ is calculated. We fix a small $\delta > 0$ and choose $S_j = \{x \mid \hat{p}^{(t_{j-1})}(x) > \delta\}$. If the support of $\hat{\boldsymbol{p}}^{(t_{j-1})}$ is large and distributes almost uniformly, it may be necessary to construct S_j such that $1 - \sum_{x \in S_j} p^{(t_{j-1})}(x)$ is smaller than some fixed threshold. However, our experimental results show that using a fixed threshold yields good results, which makes the additional effort of sorting the support of $\hat{\boldsymbol{p}}^{(t_{j-1})}$ unnecessary in practice. Note that requirement (a) implies that W_j and W_{j-1} intersect. Thus, in each step we "slide" the window in the direction that the probability mass is moving.

The sequel of this section focuses on requirement (b), where it is necessary to predict the future behavior of the process. One possibility to find a set W_j that satisfies the requirements is to carry out stochastic simulation for $t_j - t_{j-1}$ time units with initial states in S_j. This may be costly if we aim at an accurate approximation. Most simulation runs correspond to the average behavior of the system. However, there may be events that are less frequent, but that still have a significant probability. Therefore, we propose an idea that relies on a state-continuous deterministic approximation that, given an initial state $z \in S_j$, estimates the maximal and minimal values each state variable can take during the next $\tau_j = t_j - t_{j-1}$ time units. More precisely, for each dimension $d \in \{1, 2, ..., n\}$, we calculate values $b_d^+(z)$, $b_d^-(z) \in \mathbb{N}$ such that

$$1 - Pr(b_d^-(z) \le X_d(\tau) \le b_d^+(z), \\ t_{j-1} \le \tau \le t_j, 1 \le d \le n \mid X(t_{j-1}) = z)$$

is small, where $X_d(\tau)$ is the d-th component of the random vector $X(\tau)$.

The computation of the extreme values $b_d^+(z)$, $b_d^-(z)$ is carried out for several states z, which are chosen uniformly at random. Our experimental results indicate that the accuracy of our results does not increase when more than 10 states are considered. Let $A_j \subseteq S_j$ be the set of random states. By computing $b_d^+ = \max_{z \in A_j} b_d^+(z)$ and $b_d^- = \min_{z \in A_j} b_d^-(z)$, we obtain estimates for the maximal and minimal values of each state variable during the time interval $[t_{j-1}, t_j)$ under the condition that $X(t_{j-1}) \in S_j$. Window W_j is now constructed as the union of S_j and all states within b_d^- and b_d^+, that is, W_j equals

$$S_j \cup \{(x_1, \ldots, x_n) \in S \mid b_d^- \le x_d \le b_d^+, 1 \le d \le n\}. \tag{9}$$

Table 1: The method *ContDetApprox* (left) and the main procedure *sWindow* (right)

ALGORITHM ContDetApprox (z, τ, b+, b-)

Input: initial state *z*, length τ of time interval, old boundaries $b^+ = (b_1^+, \ldots, b_n^+), b^- = (b_1^-, \ldots, b_n^-)$

Output: new boundaries $b^+ = (b_1^+, \ldots, b_n^+), b^- = (b_1^-, \ldots, b_n^-)$

1 **for** each branch *i* {1, ..., 2*n*} **do**
2 $x^{\langle i \rangle} := z$; *//z is initial state of all branches*
3 **end**
4 *time* := 0;
5 Δ := *step_size*; *//choose length of time steps*
6 **while** *time* < τ **do**
7 **for** each branch *i* {1, ..., 2*n*} **do**
//compare current state variables with boundaries
8 **for** *d* = 1 to *n* **do**
9 **if** $x_d^{\langle i \rangle} > b_d^+$ **then**
10 $b_d^+ := x_d^{\langle i \rangle}$; *//adjust upper bound*
11 **end**
12 **if** $x_d^{\langle i \rangle} < b_d^-$ **then**
13 $b_d^- := x_d^{\langle i \rangle}$; *//adjust lower bound*
14 **end**
15 **end**
16 **for** *m* := 1 to *k* **do**
//choose more/fewer transitions of type R_m
//depending on branch i
17 κ_m := *choose*($a_m(x^{\langle i \rangle}) \cdot \Delta$, *i*);
18 **end**

ALGORITHM sWindow(y, t, ?, δ)

Input: initial state *y*, times $t = (t_0, \ldots, t_r)$, error ? > 0, threshold $\delta > 0$

Output: probability vectors $\boldsymbol{p}_0, \ldots, \boldsymbol{p}_r$

1 $\boldsymbol{p}_0 = (\mathbf{1}_y)^T$; *//start with probability one in y*
2 **for** *j* := 1 to *r* **do**
3 $\tau_j := t_j - t_{j-1}$; *//length of next time step*
//define S_j for construction of W_j
4 $S_j := \{x \mid \boldsymbol{p}_{j-1}(x) > \delta\}$;
5 *numStates* := min(10, *size*(S_j));
//choose numStates random states from S_j
6 A_j := *rand*(S_j, *numStates*);
7 b^+: -∞; b^- : +∞; *//initial boundaries*
8 **for** each *z* in A_j **do**
//run continuous determ. approximation
//on z and update boundaries
9 (b^+, b^-) := *ContDetApprox* (z, τ_j, b^+, b^-);
10 **end**
11 Q_j := *generator* (S_j, b^+, b^-); *//construct Q_j*
//construct diagonal matrix for W_j (cf. Eq. (7))
12 D_j := *diag*(S_j, b^+, b^-);
13 $\boldsymbol{p}_j := Solve(\boldsymbol{p}_{j-1} D_j, Q_j, \tau_j, ?\frac{\tau_j}{t_r - t_0})$; *//solve Eq. (7)*
14 **end**

Table 1: The method *ContDetApprox* (left) and the main procedure *sWindow* (right) (Continued)

19 $x^{\langle i \rangle} := x^{\langle i \rangle} + \sum_{m=1}^{k} v^{(m)} \kappa_m$; //*update state* (*cf. Eq.* (11))

20 **end**

21 *time* := *time* + Δ;

22 **end**

For a fixed state $z \quad A_j$ we exploit the regular structure of the Markov chain for the computation of $b_d^+(z)$ and $b_d^-(z)$. We start in state z and update the state variables one by one. We assume that for a small time interval of length Δ the rate of reaction type R_m remains constant, i.e., is equal to $\alpha_m(z)$. Then the number of R_m-transitions within the next Δ time units is Poisson distributed with parameter $\alpha_m(z)\Delta$. We can approximate this number by the expectation $\alpha_m(z)\Delta$ of the Poisson distribution. Note that the above assumption is warranted since in the case of coupled chemical reactions the propensities $\alpha_m(x)$ are linear or at most quadratic in x, if only elementary reactions are considered, i.e. reactions that correspond to a single mechanistic step and have therefore at most two reactants. In general, reactions may have intermediate products and/or parallel reaction pathways. They can, however, always be decomposed into elementary reactions. As we are interested in an upper and a lower bound, we additionally consider the standard deviation $\sqrt{\alpha_m(z)\Delta}$ of the Poisson distribution. We assume that, if the current state is x, within Δ units of time

- at least $\kappa_m^-(x,\Delta) = \max(0, \alpha_m(x)\Delta - \sqrt{\alpha_m(x)\Delta})$,
- at most $\kappa_m^+(x,\Delta) = \alpha_m(x)\Delta + \sqrt{\alpha_m(x)\Delta}$

transitions of type R_m are taken. Note that if, for instance, $\alpha_m(z)\Delta = 1$, then we have a confidence of 91.97 percent that the real number of reactions lies in the interval

$$\left[\, \alpha_m(x)\Delta - \sqrt{\alpha_m(x)\Delta}, \alpha_m(x)\Delta + \sqrt{\alpha_m(x)\Delta} \,\right]. \quad (10)$$

Let $\kappa_m \quad \{\kappa_m^+, \kappa_m^-\}$ and $l = 0, 1, \ldots$. The iteration

$$\begin{aligned} x^{(0)} &= z, \\ x^{(l+1)} &= x^{(l)} + \sum_{m=1}^{k} v^{(m)} \cdot \kappa_m(x^{(l)}, \Delta) \end{aligned} \quad (11)$$

yields continuous deterministic worst-case approximations of $X(t + \Delta)$, $X(t + 2\Delta)$,... under the condition that $X(t) = z$. For functions α_m that grow extremely fast in the state variables, the iteration may yield bad approximations because it is based on the assumption that the propensities are constant during $[0, \Delta)$.

In the context of biochemical reaction networks, α_m is at most quadratic and therefore the approximation given by Eq. (11) yields adequate results. For a given system, we perform the approximation in Eq. (11) for all possible combinations $\{\kappa_1^+, \kappa_1^-\} \times \ldots \times \{\kappa_k^+, \kappa_k^-\}$. It is possible to skip combinations that treat preferentially transition types leading to opposite directions in the state space, because they will not give a worst-case bound. Consider, for instance, Example (1) with $c_1 = c_2 = c_3 = 1$, $z = (10, 10, 100, 0)$, and $\Delta = 0.01$. If we assume that more reactions of type R_2 and R_3 happen (than on average) and fewer of R_1, we get $\kappa_2^+(z,\Delta) = \kappa_3^+(z,\Delta) = 100 \cdot 0.01 + \sqrt{100 \cdot 0.01} = 2$, and $\kappa_1^-(z,\Delta) = 10 \cdot 10 \cdot 0.01 - \sqrt{10 \cdot 10 \cdot 0.01} = 0$. This means that the number of complex molecules decreases and $x^{(1)} = (14,12, 96, 2)$. We can omit combinations that contain both κ_1^+ and κ_2^+. As R_1 equates R_2 and vice versa, these combinations will not yield good approximations of the extreme values of the state variables. In general, the dependency graph of the reaction network may be helpful to identify those combinations that maximize a

certain population (see also the section on experimental results).

In the sequel, each chosen combination is referred to as a *branch* because, for fixed z, the corresponding iterations lead to different successors $x^{(l+1)}$. Note that for a particular branch, for each $m \in \{1, ..., k\}$ we fix $\kappa_m = \kappa_m^+$, or $\kappa_m = \kappa_m^-$ for all l. The iteration ends after $\lceil \tau_j/\Delta \rceil$ steps (where the length of the last time step is the remainder instead of Δ), and the extreme values $b_d^+(z)$ and $b_d^-(z)$ are given by the minimal and maximal values of the state variables during the iteration. More specifically, $b_d^+(z) = \left\lceil \max_l x_d^{(l)} \right\rceil$ and $b_d^-(z) = \left\lfloor \min_l x_d^{(l)} \right\rfloor$, where $1 \le d \le n$, $x^{(l)} = (x_1^{(l)}, \ldots, x_n^{(l)})$, and $z = x^{(0)}$.

The calculation of $b_d^+(z)$ and $b_d^-(z)$ is described in pseudocode in Table 1 (left column), called *ContDetApprox*. Note that the superscript i refers to the current branch and not to the iteration in Eq. (11) which is carried out in line 19. The number of branches is $2n$ as maximal and minimal values for each dimension are necessary. In line 17, we decide, depending on the current branch i, whether κ_m is set to κ_m^+, or κ_m^-.

Regarding the choice of the time step Δ, we suggest to choose Δ dynamically such that for each m the interval in Eq. (10) covers at least, say, 80% of the probability mass of the corresponding Poisson distribution. Clearly, the accuracy of the method increases in the case of larger intervals covering more probability mass. For our experimental results, we chose Δ such that $\lambda_x \cdot \Delta = 1$ yielded sufficiently accurate results.

Sliding window algorithm

In the right column of Table 1 we describe the main procedure, called *sWindow*, in pseudocode. The for loop in lines 2-14 implements the approximations of $\hat{\boldsymbol{p}}^{(t_1)}, \ldots, \hat{\boldsymbol{p}}^{(t_r)}$ by successively computing vector $\boldsymbol{p}_j$ from $\boldsymbol{p}_{j-1}$. Input ? is a bound for the total approximation error caused by the solutions of the ODEs in line 13. The array t contains the time instances $t_0, ..., t_r$. For our experimental results, we compare two different time stepping mechanisms that are explained below. The parameter δ is the threshold that is used to remove those states in the support of p_{j-1} having a smaller probability than δ. We define S_j as the set of all states x for which $p_{j-1}(x)$ is greater than δ in line 4. Note that for $j = 1$ the set S_1 contains only the initial state y. In line 6, *rand*(S_j, *numStates*) returns a set of *numStates* random elements from S_j that are used to construct the vectors b^+ and b^- in lines 7-10. The rate entries of all states in the window W_j (cf. Eq. (9)) are calculated in line 11, and all remaining entries in Q_j are set to zero. A solution method is invoked in line 13 to calculate p_j from p_{j-1}. This can be, for instance, the uniformization method, an ODE solver or a method based on an approximation in the Krylov subspace. We pass a time step of length τ_j and the corresponding fraction $\frac{\tau_j}{t_r - t_0}$ of the approximation error.

We can calculate the overall loss of probability mass from the output p_r by $\eta_r = 1 - \sum_x p_r(x)$. This value includes both approximation errors of the algorithm: (1) the probability of leaving window W_j during the time interval $[t_{j-1}, t_j)$ and (2) the probability $\sum_{x \notin W_j} \hat{p}^{(t_j)}(x)$ that is lost due to the sliding of the window, obtained by the multiplication with D_j (cf. Eq. (7)).

Note that it is always possible to repeat a computation step in order to increase the obtained accuracy. More precisely, we can determine a larger window by increasing the confidence of the interval in Eq. (10), i.e. by choosing the time step Δ such that for each m the maximal/minimal number of transitions of type R_m lies in the interval with a certain confidence (e.g. with a confidence of 80%). For our experimental results, however, we did not repeat any computation step since we always obtained sufficiently accurate results.

Time intervals

For our experimental results, we compare two different time stepping mechanisms for Algorithm *sWindow* (see Table 1, right). We either choose equidistant time steps $\tau_j = \tau$, for all j, or we determine τ_j during the construction of the window W_j (adaptive time steps). The latter method yields faster running times. Depending on the dynamics of the system, long time steps may cause three problems: (1) the window is large and the size of the matrix Q_j may exceed the working memory capacity, (2) the dynamics of

the system may differ considerably during a long time step and Q_j has bad mathematical properties, (3) the window may contain states that are only significant during a much shorter time interval. If, on the other hand, the time steps are too small then many iterations of the main loop have to be carried out until the algorithm terminates. The windows overlap nearly completely, and even though each step may require little time, the whole procedure can be computationally expensive. One possibility is to fix the size of the windows and choose the time steps accordingly. But this does not necessarily result in short running times of the algorithm either. The reason is that the time complexity of the solution methods does not depend only on the size of the matrix representing the window but also on its mathematical properties.

The problems mentioned above can be circumvented by calculating $\tau_1, ..., \tau_r$ during the construction of the window W_j as follows. We compute the number of the states that are significant at time t_{j-1} and pass it to *ContDetApprox* in line 9 (see Table 1). We run the while loop in Algorithm *ContDetApprox* (see Table 1, left) until (1) the window has at least a certain size and (2) the number of states in the window exceeds twice the number of the states that are significant at time t_{j-1}. The first condition ensures that the window exceeds a certain minimum size of, say, 500 states. The second condition ensures that the new window is just big enough to move the probability mass to a region outside of S_j. More precisely, it ensures that the sets S_1, S_2,...are not overlapping and that subsequent sets are located next to each other (as illustrated in Figure 1). Note that this ensures that the resulting window does not contain many states that are only significant during a much shorter time interval.

On termination of the while-loop, we pass the value of the variable *time* from *ContDetApprox* to *sWindow* and set τ_j to the value of *time*. Obviously, in *sWindow* we add a variable representing the time elapsed so far, and the for loop in line 2 is replaced by a while loop that stops when the time elapsed so far exceeds t. Later, we present experimental results of the sliding window method where we use adaptive time steps in the way described above.

Numerical solution methods

In this section, we present the theoretical basis of two numerical solution algorithms, namely the uniformization method and the Krylov subspace method. We approximate a global solution of the CME (cf. Eq. (5)), as well as the local solutions that are required in line 13 of Algorithm *sWindow* (see also Eq. (7)).

Uniformization

The uniformization method goes back to Jensen [34] and is also referred to as Jensen's method, randomization, or discrete-time conversion. In the performance analysis of computer systems, this method is popular and often preferred over other methods, such as Krylov subspace methods and numerical integration methods [19,39]. Recently, uniformization has also been used for the solution of the CME [40-42].

Global uniformization Let $(X(t),\ t \quad \mathbb{R}_{\geq 0})$ be a CTMC with finite state space S. The basic idea of uniformization is to define a discrete-time Markov chain (DTMC) and a Poisson process. The DTMC is stochastically identical to X, meaning that it has the same transient probability distribution if the number of steps within $[0, t)$ is given, and the Poisson process keeps track of the time as explained below.

Recall that λ_x is the exit rate of state $x \quad S$, and I is the identity matrix. We define a *uniformization rate* λ such that $\lambda \geq \max_{x\ S} \lambda_x$ and construct $P = I + \frac{1}{\lambda} Q$, the transition matrix of the DTMC associated with X. Note that a diagonal entry in P defines the self-loop probability $1 - \lambda_x/\lambda$ of a state x, which is nonzero if and only if $\lambda > \lambda_x$. For $k \geq 1$, the stochastic matrix P^k contains the k-step transition probabilities and, if $\boldsymbol{p}^{(0)}$ is the initial distribution of X, the vector $\boldsymbol{w}^{(k)} = \boldsymbol{p}^{(0)}P^k$ contains the state probabilities after k steps in the DTMC. The number of steps within time interval $[0, t)$ has a Poisson distribution with parameter λt, i.e.,

$$Pr(k \text{ steps until time } t) = e^{-\lambda t} \frac{(\lambda t)^k}{k!}. \tag{12}$$

Now, the solution of the transient state probabilities in Eq. (5) can be rewritten as [19,32,43]

$$\begin{aligned} \boldsymbol{p}^{(t)} &= \boldsymbol{p}^{(0)} \sum_{k=0}^{\infty} e^{-\lambda t} \frac{(\lambda t)^k}{k!} P^k \\ &= \sum_{k=0}^{\infty} e^{-\lambda t} \frac{(\lambda t)^k}{k!} \boldsymbol{w}^{(k)}. \end{aligned} \tag{13}$$

Eq. (13) has nice properties compared to Eq. (5). There are no negative summands involved, as P is a stochastic matrix and $\lambda > 0$. Moreover, $\boldsymbol{w}^{(k)}$ can be computed inductively by

$$\boldsymbol{w}^{(0)} = \boldsymbol{p}^{(0)}, \boldsymbol{w}^{(j)} = \boldsymbol{w}^{(j-1)}P,\ j \in \{1, 2, \ldots\}. \tag{14}$$

If P is sparse, $\boldsymbol{w}^{(k)}$ can be calculated efficiently even if the size of the state space is large. Lower and upper sum-

mation bounds L and U can be obtained such that for each state x the truncation error [44]

$$\begin{aligned} & p^{(t)}(x) - \sum_{k=L}^{U} e^{-\lambda t} \frac{(\lambda t)^k}{k!} w^{(k)}(x) \\ = & \sum_{\substack{0 \le k < L, \\ U < k < \infty}} e^{-\lambda t} \frac{(\lambda t)^k}{k!} w^{(k)}(x) \\ \le & \sum_{\substack{0 \le k < L, \\ U < k < \infty}} e^{-\lambda t} \frac{(\lambda t)^k}{k!} \\ = & 1 - \sum_{k=L}^{U} e^{-\lambda t} \frac{(\lambda t)^k}{k!} < ? \end{aligned} \tag{15}$$

can be a priori bounded by a predefined error tolerance ? > 0. Thus, $\boldsymbol{p}^{(t)}$ can be approximated with arbitrary accuracy by

$$\boldsymbol{p}^{(t)} \approx \sum_{k=L}^{U} e^{-\lambda t} \frac{(\lambda t)^k}{k!} \boldsymbol{w}^{(k)} \tag{16}$$

as long as the required number of summands is not extremely large.

Time complexity and stiffness As λt grows the Poisson distribution flattens, and the left truncation point L in Eq. (16) grows linearly in λt, while the number of significant Poisson probability terms is [44]$O(\sqrt{\lambda t})$. If the vectors $\boldsymbol{w}^{(L)}$, $\boldsymbol{w}^{(L+1)}$, ..., $\boldsymbol{w}^{(U)}$ are computed using U matrix-vector multiplications (cf. Eq. (14)), then the complexity of the uniformization procedure is $O(\nu\lambda t)$ where ν is the number of nonzero elements in P.

All analysis methods (simulation-based or not) encounter serious difficulties if the underlying model is *stiff*. In a stiff model the components of the underlying system act on time scales that differ by several orders of magnitude and this arises in various application domains, especially in systems biology. For a stiff model, the uniformization rate $\lambda \ge \max_{x \in S} \lambda_x$ will correspond to the fastest time scale. By contrast, a significant change of the slow components can be observed only during a period of time that corresponds to the slowest time scale. The uniformization method is then extremely time consuming because of a very large *stiffness index* [45]$t \cdot \max_{x \in S} \lambda_x$.

In the sequel, we show how uniformization can be applied in a local fashion such that stiffness has a less negative effect on the performance of the method. In other words, the sliding window technique enables uniformization to perform well even for stiff systems.

Local uniformization We now combine uniformization and the sliding window method. Assume that S may be infinite, and that we iteratively apply uniformization to solve Eq. (7). More specifically, in line 13 of Algorithm *sWindow* (see Table 1, right), we invoke the uniformization method to approximate

$$\hat{\boldsymbol{p}}^{(t_j)} = \hat{\boldsymbol{p}}^{(t_{j-1})} D_j e^{\tau_j Q_j}.$$

Thus, $P_j = I + \frac{1}{\lambda_j} Q_j$ is a substochastic transition matrix, where $\lambda_j = \max_{x \in W_j} \lambda_x$. By using the same calculation as in Eq. (16), we obtain a substochastic vector

$$\begin{aligned} \hat{\boldsymbol{p}}^{(t_j)} &= \hat{\boldsymbol{p}}^{(t_{j-1})} D_j \sum_{k=L}^{U} e^{-\lambda_j \tau_j} \frac{(\lambda_j \tau_j)^k}{k!} P_j^k \\ &= \sum_{k=L}^{U} e^{-\lambda_j \tau_j} \frac{(\lambda_j \tau_j)^k}{k!} \hat{\boldsymbol{w}}_j^{(k)}, \end{aligned} \tag{17}$$

where L and U are the truncation points depending on $\lambda_j \tau_j$, and $\hat{\boldsymbol{w}}_j^{(k)} = \hat{\boldsymbol{p}}^{(t_{j-1})} D_j P_j^k$. Moreover, as λ_j depends only on W_j, the uniformization rate is usually smaller than the global one, $\sup_{x \in S} \lambda_x$, which means that fewer terms are required in Eq. (17) than in Eq. (16).

The computational complexity of the whole procedure is $O(\sum_{j=1}^{r} \nu_j \lambda_j \tau_j)$, and thus, we save computation time, compared to global uniformization, if $\sum_{j=1}^{r} \nu_j \lambda_j \tau_j \ll \nu \lambda t$, where $\lambda = \sup_{x \in S} \lambda_x$ and ν_j is the number of nonzero elements in P_j.

Krylov subspace

Krylov subspace methods are widely used for large eigenvalue problems, for solving linear equation systems, and also for approximating the product of a matrix exponential and a vector [46,47]. We are interested in the latter approximation and show how it can be used to solve the CME, either in a global fashion or in combination with the sliding window method. Recently, Krylov subspace methods have been applied to the CME by Sidje et al. [21].

Global Krylov subspace method Recall that a global solution of the CME is given by $\boldsymbol{p}^{(t)} = \boldsymbol{p}^{(0)} e^{Qt}$. In the sequel, we describe the approximation of $e^{tA}\boldsymbol{v}$, where A is an $N \times$

N square matrix and $\boldsymbol{v}$ is a column vector of length N. We obtain an approximation of $\boldsymbol{p}^{(t)}$ by choosing $A = (Q)^T$ and $\boldsymbol{v} = (\boldsymbol{p}^{(0)})^T$. Let us first assume that $t = 1$. The main idea is to generate a basis of the m-th Krylov subspace

$$K_m = Span\{\boldsymbol{v}, A\boldsymbol{v}, \ldots, A^{m-1}\boldsymbol{v}\},$$

and to seek an approximate solution for $e^A\boldsymbol{v}$ from this subspace. Let $q^{\min}$ be the nonzero monic polynomial of lowest degree such that $q^{\min}(A)\boldsymbol{v} = 0$. We choose $m \in \mathbb{N}$ such that the degree of $q^{\min}$ is greater or equal to m. In this case, the vectors $\boldsymbol{v}$, $A\boldsymbol{v}$, ..., $A^{m-1}\boldsymbol{v}$ are linearly independent, and for every element $\boldsymbol{x} \in K_m$ there exists a polynomial q of degree at most m - 1 with $\boldsymbol{x} = q(A)\boldsymbol{v}$. Note that in practice we choose $m = 30$ or $m = 20$, because the degree of $q^{\min}$ is usually greater than 30. However, if not, the problem can be solved *exactly* in the d-th Krylov subspace, where d is the degree of $q^{\min}$. Working directly with the basis $\{\boldsymbol{v}, A\boldsymbol{v}, \ldots, A^{m-1}\boldsymbol{v}\}$ is numerically unstable. Therefore, we construct an orthonormal basis $\{\boldsymbol{v}_1, \boldsymbol{v}_2, \ldots, \boldsymbol{v}_m\}$ for K_m by applying Arnoldi's algorithm to $\boldsymbol{v}, A\boldsymbol{v}, \ldots, A^{m-1}\boldsymbol{v}$. Let H_m be the $m \times m$ upper Hessenberg matrix computed by the Arnoldi algorithm and let $h_{m+1,m}$ be the last normalization value. By V_m we denote the $N \times m$ matrix with column vectors $\boldsymbol{v}_1, \boldsymbol{v}_2, \ldots, \boldsymbol{v}_m$. Then

$$\begin{aligned} &(a) \quad AV_m = V_m H_m + h_{m+1,m}\boldsymbol{v}_{m+1}\boldsymbol{e}_m^T, \\ &(b)\ V_m^T A V_m = H_m, \end{aligned} \tag{18}$$

where $\boldsymbol{e}_k$ is a column vector of appropriate size whose k-th component is one and all other components are zero. Intuitively, Eq. (18)(b) states that H_m is the matrix projection of A onto K_m w.r.t. the basis defined by V_m. An approximation of $e^A\boldsymbol{v}$ in K_m expressed using V_m is $e^A\boldsymbol{v} \approx V_m\boldsymbol{y}$, where $\boldsymbol{y}$ is a vector of size m.

We choose

$$\boldsymbol{y} = ||\,\boldsymbol{v}\,||_2\, e^{H_m}\boldsymbol{e}_1, \tag{19}$$

which yields the approximation error [46]

$$||\, e^A\boldsymbol{v} - ||\,\boldsymbol{v}\,||_2\, V_m e^{H_m}\boldsymbol{e}_1\,||_2 \leq 2\,||\,\boldsymbol{v}\,||_2\,\frac{\rho^m e^\rho}{m!}, \tag{20}$$

where $\rho = ||A||_2$ is the spectral norm of A. The approximation in Eq. (19) still involves the computation of the matrix exponential of H_m, but as H_m is of small dimension and has a particular structure (upper Hessenberg), this requires a smaller computational effort. For the matrix exponential of small matrices, methods such as Schur decomposition and Padé approximants may be applied [31].

Assume now that the time instant t is arbitrary, i.e., we want to approximate $e^{tA}\boldsymbol{v}$ for some $t > 0$. In order to control the approximation error, we calculate $e^{tA}\boldsymbol{v}$ stepwise by exploiting that $e^{(\tau_1+\tau_2)A}\boldsymbol{v} = e^{\tau_2 A} \cdot e^{\tau_2 A}\boldsymbol{v}$ for $\tau_1, \tau_2 \geq 0$. For a step size τ, we approximate $e^{\tau A}\boldsymbol{v}$ by $||\boldsymbol{v}||_2\, V_m e^{\tau H_m}\boldsymbol{e}_1$ because the Krylov subspaces associated with A and τA are identical and $V_m^T(\tau A)V_m = \tau H_m$. It follows from Eq. (20) that we have a small error bound if $||A\tau||_2$ is small.

To summarize, the Krylov subspace method approximates $e^{At}\boldsymbol{v}$ by an iteration stepping forward in time with dynamically chosen step sizes $\tau_1, \tau_2, \ldots$. In each iteration step, we compute a vector

$$\boldsymbol{u}_i = ||\,\boldsymbol{u}_{i-1}\,||_2\, V_m^{(i)} e^{\tau_i H_m^{(i)}}\boldsymbol{e}_1,$$

where initially $\boldsymbol{u}_0 = \boldsymbol{v}$. The matrices $V_m^{(i)}$ and $H_m^{(i)}$ result from the i-th execution of Arnoldi's algorithm for the construction of an orthonormal basis of the subspace $Span\{\boldsymbol{u}_{i-1}, A\boldsymbol{u}_{i-1}, \ldots, A^{m-1}\boldsymbol{u}_{i-1}\}$. When the elapsed time equals t, we obtain an approximation of $e^{At}\boldsymbol{v}$.

For the step size of the Krylov subspace method, a popular heuristic is to choose τ_{i+1} depending on an estimate of the error $?_i$ of the previous step. Let $tol > 0$ be an a priori specified tolerance. A common scheme consists of three steps [36]. (1) Define $\tau_i = 0.9\left(\frac{tol}{?_{i-1}}\right)^{1/m}\tau_{i-1}$, (2) compute $\boldsymbol{u}_i$ and the error estimation $?_i$. (3) If $?_i > 1.2\ tol$

	parameters			results				
	name of example	time horizon	δ	sWindow + uniform.	sWindow + Krylov		window construction	
				error	times in sec		perc.	average wind. size
1	Enzyme (pset a)	70	10^{-8}	1.4×10^{-5}	6	5	1%	977
2	Enzyme (pset b)	12	10^{-10}	3.3×10^{-5}	134	98	14%	4777
3	Enzyme (pset c)	5	10^{-10}	3.5×10^{-7}	8	6	37%	5038
4	Gene (pset a)	10^4	10^{-10}	1.6×10^{-5}	103	102	36%	32248
5	Gene (pset b)	10^4	10^{-10}	1.8×10^{-5}	137	123	32%	38282
6	Goutsias' model	300	10^{-11}	7.6×10^{-5}	15943	8412	15%	538815
7	Toggle switch	10^4	10^{-15}	2.7×10^{-5}	31	10	1%	63001

Figure 2 Parameters and results of the sliding window method.

reject $\boldsymbol{u}_i$, replace $?_{i-1}$ with $?_i$, and go to step (1). For our experimental results, we used the Expokit software [48] where the small exponential, $e^{\tau H_m}$, is computed via the irreducible Padé approximation [49].

Local Krylov subspace method Assume now that we use the Krylov subspace method in line 13 of Algorithm *sWindow* (see Table 1, right), to approximate $\hat{\boldsymbol{p}}^{(t_{j-1})} D_j e^{Q_j \tau_j}$ (cf. Eq. (7)). By letting $\boldsymbol{v} = (\boldsymbol{p}^{(t_{j-1})} D_j)^T$, $A = Q_j^T$, and $t = \tau_j$ we can apply the same procedure as in the global case. Note that this yields a nested iteration because the time steps τ_j are usually much bigger than the time steps of the Krylov subspace method. For the Krylov subspace method, using the matrix Q_j instead of Q offers important advantages. The Arnoldi process is faster as Q_j usually contains fewer nonzero entries than Q. As well, the sliding window method is likely to provide matrices with a smaller spectral norm $||Q_j||_2$. This allows for bigger time steps during the Krylov approximation, as can be seen in our experimental results.

Experimental results

We coded both algorithms in Table 1 in C++ and ran experiments on a 3.16 GHz Intel dual-core Linux PC. We discuss experimental results that we obtained for Example 1 and Example 2, as well as Goutsias' model [50] and a bistable toggle switch [51]. Goutsias' model describes the transcription regulation of a repressor protein in bacteriophage λ and involves six different species and ten reactions. The bistable toggle switch is a prototype of a genetic switch with two competing repressor proteins and four reactions. All results are listed in Figure 2.

As explained in detail below, we also implemented the method proposed by Burrage et al. [21] in order to compare it to our algorithm in terms of running time and accuracy. Moreover, for finite examples we compare our method to a global analysis, i.e. where in each step the entire state space is considered. We do not compare our method to Gillespie simulation or approximation methods based on the Fokker-Planck equation. The former method provides only estimates of the probability distribution and becomes infeasible if small probabilities are estimated [52]. The latter type of methods do not take into account the discreteness of the molecule numbers and are known to provide bad approximations in the case of small populations as considered here [53].

Parameters

We fixed the input $? = 10^{-8}$ of Algorithm *sWindow* for all experiments (see Table 1, right). We chose the input δ in a dynamical fashion to ensure that in the j-th step we do not lose more probability than $10^{-5} \cdot \tau_j/(t_r - t_0)$ by restricting to significant states, that is, we decrease δ until after line 4 of Algorithm *sWindow* the set S_j contains at most $10^{-5} \cdot \frac{\tau_j}{t_r - t_0}$ less probability than the former set S_{j-1}. In Figure 2, we list the average value that we used for δ.

In the sequel, we give details about the parameters used for the results that we obtained for Example 1 and Example 2. For the remaining two examples, we list the corresponding chemical reactions and the parameters that we chose for the results in Figure 2.

Enzyme example We tried different parameter sets, referred to as pset a)-c), for Example 1 (see Figure 2). For parameter combination a) we have $c_1 = c_2 = 1$, $c_3 = 0.1$ and start with 1000 enzymes and 100 substrates. In this case the number of reachable states is 5151. For parameter set b) and c) we have $c_1 = c_2 = c_3 = 1$ and start with 100 enzymes and 1000 substrates and 500 enzymes and 500 substrates, which yields 96051 and 125751 reachable

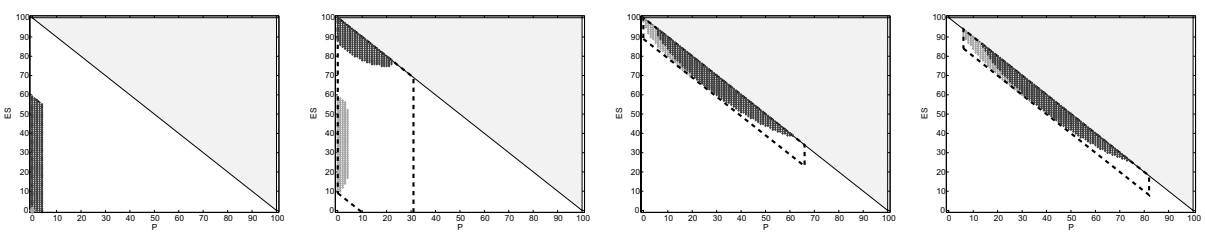

Figure 3 Parallelogram shape. For the enzyme reaction example, the set of reachable states is finite and delimited by the diagonal, which is represented by the line $ES = 100 - P$ if 100 is the initial number of enzyme molecules. For certain parameter sets, the window has a parallelogram shape which corresponds to the direction in which the probability mass is moving.

states, respectively. Each time we choose the time horizon according to the time until most of the probability mass is concentrated in the state in which all substrate molecules are transformed into products. For the time steps τ_j in Algorithm *sWindow*, we apply the condition described above.

We consider four branches for the iteration in Eq. (11) in order to determine upper and lower bounds on the state variables. (1) To obtain an estimate for the maximal number of complex molecules (and a minimum for the enzyme population), we enforce more reactions of type R_1 than on average ($\kappa_1 = \kappa_1^+$), and fewer of types R_2 and R_3 ($\kappa_3 = \kappa_3^-$ and $\kappa_2 = \kappa_2^-$). (2) By considering fewer reactions of type R_1 ($\kappa_1 = \kappa_1^-$), and more of types R_2 and R_3 ($\kappa_3 = \kappa_3^+$ and $\kappa_2 = \kappa_2^+$) the complex population becomes minimal (and the enzyme population maximal). (3) An estimate for the minimal number of type P molecules (and the maximal number of type S molecules) is obtained by enforcing more reactions of type R_2 ($\kappa_2 = \kappa_2^+$), and fewer of types R_1 and R_3 ($\kappa_1 = \kappa_1^-$ and $\kappa_3 = \kappa_3^-$). (4) Finally, more reactions of types R_1 and R_3 ($\kappa_1 = \kappa_1^+$ and $\kappa_3 = \kappa_3^+$), and fewer of type R_2 ($\kappa_2 = \kappa_2^-$) gives a maximal increase of the number of product molecules (and minimizes the number of substrate molecules).

For the enzyme example, if the initial conditions are fixed a state is uniquely determined by at least two entries, say, the population of complex and product molecules. However, a rectangular window shape yields poor results if the expected number of complex molecules is high. The reason is that in this case the probability mass is located on a diagonal (cf. Figure 3). If the set of significant states is captured by a rectangular window it may contain many states that are not significant. This problem can be circumvented by considering bounds for all state variables during the window construction as well as the conservation laws. More precisely, the parallelogram in Figure 3 is constructed by computing for each value $x_4 \in [b_4^-, b_4^+]$ of P upper and lower bounds on ES by

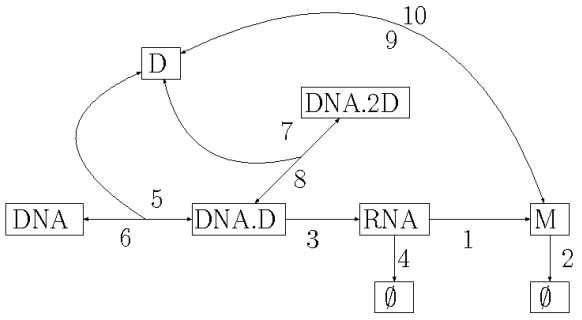

Figure 4 Dependencies between the reactions of Goutsias' model.

$\min\{b_3^+, \gamma_1 - b_1^-, \gamma_2 - x_4 - b_2^-\}$ and $\max\{b_3^-, \gamma_1 - b_1^+, \gamma_2 - x_4 - b_2^+\}$, where $y = (y_1, y_2, 0, 0)$ is the initial population vector and $b^+ = (b_1^+, b_2^+, b_3^+, b_4^+)$ and $b^- = (b_1^-, b_2^-, b_3^-, b_4^-)$ are the upper and lower bounds on the populations of *E*, *S*, *ES*, and *P*.

Note that the parallelogram in Figure 3 was induced by the conservation laws of the system. In general, conservation laws should be taken into account since otherwise the window may be inconsistent with the conservation laws, i.e. it may contain states that are not reachable.

Gene expression example In Figure 2 we present results for Example 2. The difference between parameter set a) and parameter set b), referred to as pset a) and pset b), is that for a) we start with the empty system and for b) we start with 100 mRNA molecules and 1000 proteins. For both variants, we choose rate constants c_1 = 0.5, c_2 = 0.0058, c_3 = 0.0029, c_4 = 0.0001. The time steps that we use are determined by the condition in the section on time intervals. Note that we cannot solve this example using a global method because the number of reachable states is infinite. The column *error* contains the total error η_r (see Eq. (8)) and *times in sec* refers to the running time in seconds. In column *perc.* we list the percentage of the total running time that was spent for the window construction. The column *average wind. size* refers to the average number of states in the window.

For the gene expression example, we use four branches: We maximize the number of mRNA molecules by choosing κ_1^+ and κ_3^- and minimize it with κ_1^- and κ_3^+. Reactions R_2 and R_4 are irrelevant for this species. We maximize the protein population by choosing

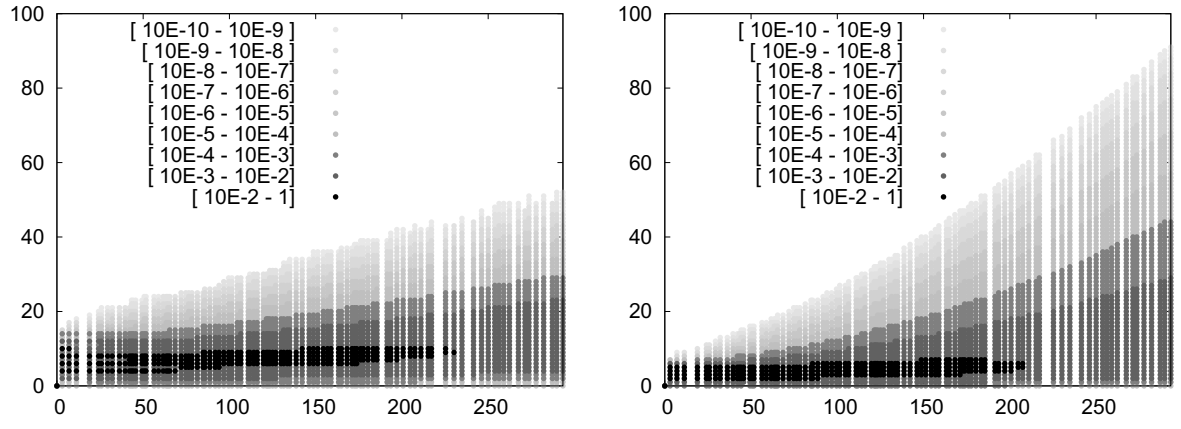

Figure 5 The probability distribution of monomers (left) and dimers (right) during the time interval [0, 300).

$\kappa_1^+, \kappa_2^+, \kappa_3^-$, and κ_4^- and minimize it with $\kappa_1^-, \kappa_2^-, \kappa_3^+$, and κ_4^+.

Goutsias' model The model, referred to as Goutsias' model in Figure 2, is composed by the following chemical reactions [50]:

1: RNA T RNA + M
2: M T ?
3: DNA.D T RNA + DNA.D
4: RNA T ?
5: DNA + D T DNA.D
6: DNA.D T DNA + D
7: DNA.D + D T DNA.2D
8: DNA.2D T DNA.D + D
9: M + M T D
10: D T M + M

We used the same kinetic constants as Goutsias [50] and Sidje et al. [21], as well as the same initial state.

Below, we list the branches for upper bounds on the state variables. Lower bounds are obtained if the opposite combination is considered, respectively. We refer to Figure 4 for an illustration of the dependencies between the reactions that simplifies the choice of the branches. We maximize the RNA population by choosing the combination $\kappa_1^-, \kappa_2^-, \kappa_3^+, \kappa_4^-, \kappa_5^+, \kappa_6^-, \kappa_7^-, \kappa_8^+, \kappa_9^+, \kappa_{10}^-$. We maximize the monomer population by choosing the combination $\kappa_1^+, \kappa_2^-, \kappa_3^+, \kappa_4^-, \kappa_5^+, \kappa_6^-, \kappa_7^-, \kappa_8^+, \kappa_9^-, \kappa_{10}^+$. We maximize the number of dimer molecules by choosing the combination $\kappa_1^+, \kappa_2^-, \kappa_3^+, \kappa_4^-, \kappa_5^+, \kappa_6^-, \kappa_7^-, \kappa_8^+, \kappa_9^+, \kappa_{10}^-$. Note that although dimers are consumed by reaction 5, choosing κ_5^+ maximizes the number of dimers in the system. This is because reaction 5 is necessary to produce monomers and therefore also dimers. We never run out of memory with the sliding window method, but the running times can be huge for a long time horizon. The reason is that the windows are large since the system contains many monomers and dimers at later time instances. For the results in Figure 2 we considered the system till time t = 300, whereas for Sidje et al. [21], the longest time horizon is t = 100. In Figure 5 we plot the distribution of the species M and D.

Bistable toggle switch The toggle switch involves two chemical species A and B and four reactions. Let $x = (x_1, x_2) \quad \mathbb{N}_0^2$. The reactions are ? T A, A T ?, ? T B, B T ? and their propensity functions α_1, ..., α_4 are given by $\alpha_1(x) = c_1/(c_2 + x_2^\beta)$, $\alpha_2(x) = c_3 \cdot x_1$, $\alpha_3(x) = c_4/(c_5 + x_1^\gamma)$, $\alpha_4(x) = c_6 \cdot x_2$. Note that in this example the propensity functions are not of the form described in Eq. 1. For our experimental results, we chose the same parameters as Sjöberg et al. [23], that is, $c_1 = c_4 = 3 \cdot 10^3$, $c_2 = c_5 = 1.1 \cdot 10^4$, $c_3 = c_6 = 0.001$, and $\beta = \gamma = 2$. The initial distribution is a Gaussian distribution $\mathcal{N}(\mu, \sigma^2)$ with $\mu = (133, 133)^T$ and $\sigma = (\sqrt{133}, \sqrt{133})$. We consider the obvious four branches each of which is intended to minimize/maximize one of the two components. The branch minimizing

	global solution				sWindow			
	error	uniform.	Krylov	#states	error	uniform.	Krylov	average wind. size
Enzyme (pset a)	5.0×10^{-9}	44.1 min	4.2 min	5151	1.4×10^{-5}	6 sec	5 sec	977
Enzyme (pset b)	1.5×10^{-7}	6.4 h	2.7 h	96051	3.3×10^{-5}	2.2 min	98 sec	4777
Enzyme (pset c)	–	> 12 h	5.6 h	125751	3.5×10^{-7}	8 sec	6 sec	5038

Figure 6 Sliding window method vs. global analysis for the finite enzyme example.

A for example will have less of the first reaction and more of the second.

Discussion

In this section, we discuss our algorithm w.r.t. accuracy and running time where we consider different solution methods and different time step mechanisms. Moreover, we compare our method with a global analysis.

Accuracy

The column labeled by *error* in Figure 2 shows the total error η_r (cf. Eq. (8)) of the sliding window method plus the uniformization error (which is bounded by ? = 10^{-8}). The error using the Krylov subspace method instead yields the same accuracy because for both, uniformization and the Krylov subspace method, the error bound is specified a priori. For all examples, the total error does not exceed 1×10^{-4}, which means that not more than 0.01 percent of the probability mass is lost during the whole procedure. It would, of course, be possible to add an accuracy check in the while loop of Algorithm *sWindow*, expand the current window if necessary, and recalculate. But as the method consistently returns a small error, this has been omitted.

We also considered relative errors, that is, $(p^{(t_j)}(x) - p^{(t_j)}(x)) \,/\, p^{(t_j)}(x)$ for states $x \quad W_j$ with $p^{(t_j)}(x) > 10^{-5}$. We approximate the value $p^{(t_j)}(x)$ by solving Eq. (13) via global uniformization, where we use truncation error ? = 10^{-8}. Since this is only possible if the state space is finite, we compared relative errors only for the enzyme example. Our calculations show that the relative errors are always smaller than 10^{-4}.

In order to support our considerations in the window construction section, we carried out experiments in which we exclusively chose the average in line 17 of Algorithm *ContDetApprox* (see Table 1, left). More precisely, for the construction of the window we do not consider the deviations in the numbers of reactions but only the average number. In this case, we called the method *ContDetApprox* with input 2τ to make sure that on average the probability mass moves to the center of the window and not too close to the borders. For this configuration, the total error is several orders of magnitude higher, e.g., for parameter set a) of the enzyme example the total error is 0.0224.

Finally, we test the size of the windows constructed in lines 7-10 of Algorithm *sWindow*. We change Algorithm *sWindow* by decreasing the size of the window by 5% between lines 10 and 11. In this case, the total error η_r increases. For instance, η_r = 0.35% for parameter set a) of the enzyme example. These results substantiate that the size and the position of the sliding window is such that the approximation error is small whereas significantly smaller windows result in significantly higher approximation errors.

Running time

For the time complexity analysis, we concentrate on three main issues.

- Sliding window method vs. global analysis: We compare the sliding window method with a global solution in one step, and with another window method, where the size of the window is doubled if necessary.
- Solution method (uniformization vs. Krylov subspace method): In Algorithm *sWindow*, we vary the solution method by exchanging uniformization with the Krylov subspace method.
- Time intervals (equidistant vs. adaptive time steps): We use different methods to determine the length τ_j of the next time step in line 3 of Algorithm *sWindow*.

Sliding window method vs. global analysis

We used the enzyme example to compare the sliding window solution with a global solution (global uniformization and global Krylov subspace method), since it has a

finite state space. Note that all other examples cannot solved using a global method since their state space is infinite. We list the time needed for the computation of $\boldsymbol{p}^{(t_r)}$ (cf. Eq. (3)) with the global method in Figure 6.

Observe that the total error of the global uniformization method is smaller (compare the columns labeled by *error*) since the only error source is the truncation of the infinite sum in Eq. (13). In the column with heading *#states* we list the number of states that are reachable. During the global solution we consider all reachable states at all time whereas in the sliding window method the average number of states considered during a time step is much smaller. This is the main reason why the sliding window method is much faster. Moreover, in the case of uniformization, the rate for global uniformization is the maximum of all exit rates, whereas for local uniformization, we take the maximum over all states in the current window. Note that the global maximum can be huge compared to the local maxima. This explains the bad performance of the global uniformization method. When the Krylov subspace method is used for a global solution, the running times of the global solutions are also higher than the times of the local Krylov subspace method (sliding window method combined with the Krylov subspace method). Again, the reason is that a smaller number of states is considered during the sliding window iteration. Moreover, the matrices Q_j have numerical properties that facilitate the use of bigger, and thus, fewer time steps. The total number of iteration steps used to solve Eq. (6) with the Krylov subspace method and the sliding window method is indeed small when compared to the global Krylov subspace method (on average around 20 times fewer steps).

We now focus on a comparison between our sliding window method and another local method, called *doubling window method*. For the latter, we compute the probability vectors in a similar way as Sidje et al. [21]. We start with an initial window and apply the Krylov algorithm. We do not iterate over the time intervals $[t_{j-1}, t_j)$ but use the step sizes of the Krylov subspace method. After each time step, we remove those parts of the window that will not be used for the remaining calculations. We expand the size of the window if the error exceeds a certain threshold. Since the performance of the method depends heavily on the initial window and the directions in which a window is expanded, we start initially with the same window as the sliding window method and expand always in the directions that are most advantageous for the computation. For this we used information about the direction in which the probability mass is moving that we obtained from experiments with the sliding window method. The expansion of a window is realized by doubling the length of all of its edges.

We applied the doubling window method to the enzyme example and the gene expression. For all parameter sets that we tried, the sliding window method outperforms the doubling window method w.r.t. running time (with an average speed-up factor of 5). The total number of iterations of the Krylov subspace approximation is up to 13 times smaller in the case of the sliding window method compared to the doubling window method (with an average of 6.5). Note that for arbitrary systems the doubling window method cannot be applied without additional knowledge about the system, i.e., it is in general not clear, in which direction the window has to be expanded.

Our results indicate that the sliding window method achieves a significant speed-up compared to global analysis, but also compared to the doubling window method. Moreover, while global analysis is limited to finite-state systems and the doubling window methods requires additional knowledge about the system, our method can be applied to any system where the significant part of the probability mass is located at a tractable subset of states. If the dimension of the system is high, then the significant part of the probability mass may be located at intractably many states and in this case the memory requirements of our algorithm may exceed the available capacity.

Solution method

During the sliding window iteration different solution methods can be applied in line 13 of Algorithm *sWindow*. We concentrate on the uniformization method and on the Krylov subspace method. The running times in Figure 2 (compare the columns labeled by *sWindow + uniformization* with the columns labeled by *sWindow + Krylov*) show that the Krylov subspace method performs better (average speed-up factor of around 1.5). The reason is that the Krylov subspace method is more robust to stiffness than uniformization. For non-stiff systems, uniformization is known to outperform the Krylov subspace method [19,39]. However, since biochemical network models are typically stiff, the Krylov subspace method seems to be particularly well suited in this area.

Time intervals

In order to confirm our considerations in the section on time intervals, we also applied the sliding window method using equidistant time steps. For all examples, using equidistant time steps results in longer computation times compared to using adaptive time steps (with an average speed-up factor of 3.5). A adaptive choice of the time steps has also the advantage that we can control the size of the windows and avoid that the memory requirements of the algorithm exceed the available capacity.

Conclusions

The sliding window method is a novel approach to address the performance problems of numerical algorithms for the solution of the chemical master equation. It replaces a global analysis of the system by a sequence of local analyzes. The method applies to a variety of chemically reacting systems, including systems for which no upper bound on the population sizes of the chemical species is known a priori. The proposed method is compatible with all existing numerical algorithms for solving the CME, and also a combination with other techniques, such as time scale separation [26,27], is possible.

We demonstrated the effectiveness of our method with a number of experiments. The results are promising as even systems with more than two million states with significant probability can be solved in acceptable time. Moreover, for examples that are more complex than those presented here, it is often sufficient to consider only a relatively small part of the state space. The number of molecules in the cell is always finite and, usually, a biochemical system follows only a small number of different trends. Stated differently, it is rarely the case that in biochemical systems a large number of different scenarios have significant likelihoods. Thus, we expect that the sliding window method can be successfully applied to systems with many chemical species and reactions as long as the significant part of the probability mass is always located at a tractable subset of states. In addition, further enhancements are possible, such as a splitting of the windows, which will be particularly useful for multi-stable systems. Moreover, we plan to automate our algorithm in a way that besides the initial conditions and the set of reactions no further input from the user is necessary, such as combinations of reactions that maximize/minimize certain populations.

Authors' contributions

VW and TAH designed the research. VW, RG, MM, and TAH developed the algorithm and the implementation was carried out by RG and MM. VW, MM, and TAH wrote the manuscript, which has been read and approved by all authors.

Acknowledgements

This research has been partially funded by the Swiss National Science Foundation under grant 205321-111840 and by the Cluster of Excellence on Multimodal Computing and Interaction at Saarland University. A preliminary version of this paper appeared in proceedings of the International Conference on Computer Aided Verification [54].

Author Details

[1]Computer Science Department, Saarland University, Saarbrücken, Germany, [2]Department of Computer Science and Engineering, IIT Bombay, Bombay, India, [3]School of Computer and Communication Sciences, EPFL, Lausanne, Switzerland and [4]Institute of Science and Technology, Klosterneuburg, Austria

Received: 1 November 2009 Accepted: 8 April 2010
Published: 8 April 2010

References

1. Elowitz MB, Levine MJ, Siggia ED, Swain PS: **Stochastic Gene Expression in a Single Cell.** *Science* 2002, **297:**1183-1186.
2. Ozbudak EM, Thattai M, Kurtser I, Grossman AD, van Oudenaarden A: **Regulation of Noise in the Expression of a Single Gene.** *Nature Genetics* 2002, **31:**69-73.
3. Blake WJ, Kaern M, Cantor CR, Collins JJ: **Noise in Eukaryotic Gene Expression.** *Nature* 2003, **422:**633-637.
4. Fedoroff N, Fontana W: **Small Numbers of Big Molecules.** *Science* 2002, **297:**1129-1131.
5. Kierzek A, Zaim J, Zielenkiewicz P: **The Effect of Transcription and Translation Initiation Frequencies on the Stochastic Fluctuations in Prokaryotic Gene Expression.** *J Biol Chem* 2001, **276(11):**8165-8172.
6. Paulsson J: **Summing up the noise in gene networks.** *Nature* 2004, **427(6973):**415-418.
7. Swain PS, Elowitz MB, Siggia ED: **Intrinsic and extrinsic contributions to stochasticity in gene expression.** *PNAS, USA* 2002, **99(20):**12795-12800.
8. Turner TE, Schnell S, Burrage K: **Stochastic approaches for modelling in vivo reactions.** *Comp Biol Chem* 2004, **28:**165-178.
9. Wilkinson DJ: *Stochastic Modelling for Systems Biology* Chapman & Hall; 2006.
10. McAdams HH, Arkin A: **It's a noisy business!** *Trends in Genetics* 1999, **15(2):**65-69.
11. Rao C, Wolf D, Arkin A: **Control, exploitation and tolerance of intracellular noise.** *Nature* 2002, **420(6912):**231-237.
12. Thattai M, van Oudenaarden A: **Intrinsic Noise in Gene Regulatory Networks.** *PNAS, USA* 2001, **98(15):**8614-8619.
13. McAdams HH, Arkin A: **Stochastic mechanisms in gene expression.** *PNAS, USA* 1997, **94:**814-819.
14. Arkin A, Ross J, McAdams HH: **Stochastic Kinetic Analysis of Developmental Pathway Bifurcation in Phage λ-Infected Escherichia coli Cells.** *Genetics* 1998, **149:**1633-1648.
15. Srivastava R, You L, Summers J, Yin J: **Stochastic vs. Deterministic Modeling of Intracellular Viral Kinetics.** *J Theor Biol* 2002, **218:**309-321.
16. Gillespie DT: *Markov Processes* Academic Press; 1992.
17. Gillespie DT: **Exact Stochastic Simulation of Coupled Chemical Reactions.** *J Phys Chem* 1977, **81(25):**2340-2361.
18. Gillespie DT: **Approximate Accelerated Stochastic Simulation of Chemically Reacting Systems.** *J Chem Phys* 2001, **115(4):**1716-1732.
19. Stewart WJ: *Introduction to the Numerical Solution of Markov Chains* Princeton University Press; 1995.
20. Munsky B, Khammash M: **The finite state projection algorithm for the solution of the chemical master equation.** *J Chem Phys* 2006, **124:**044144.
21. Burrage K, Hegland M, Macnamara F, Sidje B: **A Krylov-based Finite State Projection algorithm for solving the chemical master equation arising in the discrete modelling of biological systems.** In *Proc of the Markov 150th Anniversary Conference* Edited by: Langville AN, Stewart WJ. Boson Books; 2006:21-38.
22. Sjöberg P: **Numerical Methods for Stochastic Modeling of Genes and Proteins.** In *Phd thesis* Uppsala University, Sweden; 2007.
23. Sjöberg P, Lötstedt P, Elf J: **Fokker-Planck approximation of the master equation in molecular biology.** *Computing and Visualization in Science* 2009, **12:**37-50.
24. Hegland M, Burden C, Santoso L, Macnamara S, Booth H: **A solver for the stochastic master equation applied to gene regulatory networks.** *J Comput Appl Math* 2007, **205:**708-724.
25. Engblom S: **Galerkin spectral method applied to the chemical master equation.** *Comm Comput Phys* 2009, **5:**871-896.
26. Busch H, Sandmann W, Wolf V: **A Numerical Aggregation Algorithm for the Enzyme-Catalyzed Substrate Conversion.** In *Proc of CMSB Volume 4210.* LNCS, Springer; 2006:298-311.
27. Peles S, Munsky B, Khammash M: **Reduction and Solution of the chemical master equation using time scale separation and finite state projection.** *J Chem Phys* 2006, **125:**204104.
28. Cao Y, Gillespie D, Petzold L: **Efficient step size selection for the tau-leaping simulation method.** *J Chem Phys* 2006, **124(4):**044109-11.
29. Kampen NGv: *Stochastic Processes in Physics and Chemistry* 3rd edition. Elsevier; 2007.
30. Çinlar E: *Introduction to Stochastic Processes* Prentice-Hall; 1975.
31. Moler CB, Van Loan CF: **Nineteen Dubious Ways to Compute the Exponential of a Matrix.** *SIAM Review* 1978, **20(4):**801-836.
32. Grassmann WK, (Ed): *Computational Probability* Kluwer Academic Publishers; 2000.

33. Sidje R, Stewart W: **A survey of methods for computing large sparse matrix exponentials arising in Markov chains.** *Markov Chains, Computational Statistics and Data Analysis 29* 1996:345-368.
34. Jensen A: **Markoff chains as an aid in the study of Markoff processes.** *Skandinavisk Aktuarietidskrift* 1953, **36:**87-91.
35. Gross D, Miller D: **The randomization technique as a modeling tool and solution procedure for transient Markov processes.** *Operations Research* 1984, **32(2):**926-944.
36. Philippe B, Sidje R: **Transient solutions of Markov processes by Krylov subspaces.** In *Proc of the 2nd Int Workshop on the Numerical Solution of Markov Chains* Kluwer Academic Publishers; 1995:95-119.
37. Hairer E, Norsett S, Wanner G: *Solving Ordinary Differential Equations I: Nonstiff Problems* Springer; 2008.
38. Hairer E, Wanner G: *Solving Ordinary Differential Equations II* Stiff and Differential-Algebraic Problems Springer; 2004.
39. Reibman A, Trivedi K: **Numerical transient analysis of Markov models.** *Comput Oper Res* 1988, **15:**19-36.
40. Zhang J, Watson LT, Cao Y: **A Modified Uniformization Method for the Solution of the Chemical Master Equation.** *Tech. Rep. TR-07-31, Computer Science, Virginia Tech* 2007.
41. Hellander A: **Efficient computation of transient solutions of the chemical master equation based on uniformization and quasi-Monte Carlo.** *J Chem Phys* 2008, **128(15):**154109.
42. Sidje R, Burrage K, MacNamara S: **Inexact Uniformization Method for Computing Transient Distributions of Markov Chains.** *SIAM J Sci Comput* 2007, **29(6):**2562-2580.
43. de Souza e Silva E, Gail R: **Transient Solutions for Markov Chains.** In *Computational Probability* Edited by: Grassmann WK. Kluwer Academic Publishers; 2000:43-79.
44. Fox BL, Glynn PW: **Computing Poisson probabilities.** *Communications of the ACM* 1988, **31(4):**440-445.
45. Dunkel J, Stahl H: **On the transient analysis of stiff Markov chains.** *Proc of the 3rd IFIP Working Conference on Dependable Computing for Critical Applications* 1993:137-160.
46. Gallopoulos E, Saad Y: **On the parallel solution of parabolic equations.** In *Proc. ACM SIGARCH-89* ACM press; 1989:17-28.
47. Saad Y: **Analysis of some Krylov subspace approximations to the matrix exponential operator.** *SIAM J Numer Anal* 1992, **29:**209-228.
48. Sidje RB: **EXPOKIT: Software Package for Computing Matrix Exponentials.** *ACM Transactions on Mathematical Software* 1998, **24:**130-156.
49. Baker G: *The essentials of Padé approximants* Academic Press, New York; 1975.
50. Goutsias J: **Quasiequilibrium Approximation of Fast Reaction Kinetics in Stochastic Biochemical Systems.** *J Chem Phys* 2005, **122(18):**184102.
51. Gardner T, Cantor C, Collins J: **Construction of a genetic toggle switch in Escherichia coli.** *Nature* 2000, **403:**339-342.
52. Didier F, Henzinger TA, Mateescu M, Wolf V: **Approximation of Event Probabilities in Noisy Cellular Processes.** In *Proc. of CMSB Volume 5688.* LNBI; 2009:173.
53. Ferm L, Lötstedt P, Hellander A: **A Hierarchy of Approximations of the Master Equation Scaled by a Size Parameter.** *Journal of Scientific Computing* 2008, **34:**127-151.
54. Henzinger T, Mateescu M, Wolf V: **Sliding Window Abstraction for Infinite Markov Chains.** In *Proc CAV Volume 5643.* LNCS, Springer; 2009:337-352.

doi: 10.1186/1752-0509-4-42
Cite this article as: Wolf *et al.*, Solving the chemical master equation using sliding windows *BMC Systems Biology* 2010, **4**:42