Fast Adaptive Uniformization of the Chemical Master Equation

(Invited Paper)

Frédéric Didier*, Thomas A. Henzinger*‡, Maria Mateescu*, and Verena Wolf†
*School of Computer and Communication Sciences, EPFL, Lausanne. Switzerland
†Department of Computer Science, Saarland University, Saarbrücken, Germany
‡IST Austria (Institute of Science and Technology Austria)

***Abstract*—Within systems biology there is an increasing interest in the stochastic behavior of biochemical reaction networks. An appropriate stochastic description is provided by the chemical master equation, which represents a continuous-time Markov chain (CTMC).**

Standard Uniformization (SU) is an efficient method for the transient analysis of CTMCs. For systems with very different time scales, such as biochemical reaction networks, SU is computationally expensive. In these cases, a variant of SU, called adaptive uniformization (AU), is known to reduce the large number of iterations needed by SU. The additional difficulty of AU is that it requires the solution of a birth process.

In this paper we present an on-the-fly variant of AU, where we improve the original algorithm for AU at the cost of a small approximation error. By means of several examples, we show that our approach is particularly well-suited for biochemical reaction networks.

***Keywords*-Markov chains, transient analysis, uniformization, biochemical reactions, chemical master equation**

I. Introduction

Molecular noise, which arises from the randomness of the discrete events in the cell, significantly influences fundamental biological processes such as gene expression [28], [8], [43], decisions of the cell fate [1], [26], and circadian oscillations [13], [2]. During the last decade, stochastic models with discrete state spaces have seen growing interest because they provide appropriate descriptions of systems that are subject to molecular noise [29], [36], [10], [33].

Given a network of biochemical reactions, the theory of stochastic chemical kinetics allows to derive semantics in terms of a *continuous-time Markov chain* (CTMC). Its state space consists of population vectors of size n where n is the number of different molecule types, called chemical species, that are involved in the reactions. In other words, if $x = (x_1, \ldots, x_n)$ is the current state of the system, the entry x_j is the number of molecules of type j. The evolution of the CTMC is given by a system of linear ordinary differential equations, known as the *chemical master equation* (CME). A single equation in the CME describes the time-derivative of the probability of a certain state at all times $t \geq 0$. Thus, the solution of the CME is the probability distribution over all states of the CTMC at a particular time t, that is, the transient state probabilities at time t. The solution of the CME is then used to derive measures of interest such as the distribution of switching delays [28], the distribution of the time of DNA replication initiation at different origins [32], or the distribution of gene expression products [45]. Moreover, many parameter estimation methods require the computation of the posterior distribution because means and variances do not provide enough information to calibrate parameters [21].

Analytic solutions of the CME are only possible for CTMCs with a simple structure. For CTMCs where the number of reachable states is of manageable size, numerical solution algorithms such as numerical integration methods can be applied. For systems arising in applications, however, the number of reachable states is large or even infinite, which renders the analysis of the CTMC difficult. Therefore, statistical estimation procedures such as Monte Carlo simulation are widely used to circumvent the problem of state space explosion. Recent work, however, indicates that numerical approximation methods for the CME can be used to compute the transient state probabilities more accurately and, in particular, with shorter running times [6]. Especially if the probabilities of interest are small, numerical approximations turn out to be superior to Monte Carlo simulation.

In this paper we focus on uniformization methods for CTMCs that represent biochemical reaction networks. Standard uniformization (SU) is widely used for the computation of transient state probabilities of CTMCs that arise in other application domains such as the performance analysis of computer systems [42], [37]. A generalization of it, called adaptive unifomization (AU), splits the given CTMC into a discrete-time Markov chain (DTMC) and a birth process [44], whereas SU splits the CTMC into a DTMC and a Poisson process. For a small time horizon t, experimental results indicate that AU performs better than SU [7]. It was noted, however, that the obtained savings depend on the particular application problem [5].

Here, we present a new variant of AU, which is particularly efficient for CTMCs that represent chemical reaction networks. Our modifications to the original AU algorithm include

- an accelerated solution of the DTMC by avoiding the construction of a transition matrix and by using

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.

sophisticated data structures,
- the consideration of fewer states by dynamically neglecting states with insignificant probability,
- an accelerated solution of the birth process by using a technique similar to the sliding window method [22].

Besides the theory behind our method, we discuss the details of our implementation. To validate our approach, we present experimental results using two example networks from biology. Our largest example has 10 reactions and 6 chemical species.

Related Work: Various attempts have been made to numerically approximate the solution of the CME. Some rely on continuous approximations of the CME using the Fokker-Planck equation [40], [41]. These methods are appropriate if the molecular populations are large enough that the discrete nature of the system can be ignored.

Other approaches are based on a splitting of the time horizon into several time intervals [22], [3], [47], [31]. Then for each interval, they construct submodels of the CTMC that represent the system's behavior during the interval.

The probability distributions defined by the CME can also be approximated using sparse grid methods [19], spectral methods [9], or the separation of time scales [4], [34]. 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.

Recently, uniformization has been used in the context of biochemical reaction networks. Hellander [20] combines SU with Monte Carlo simulation. Sidje et al. also consider SU and, similar to our approach, they neglect states in the DTMC that have insignificant probability [38]. Zhang et al. apply the external uniformization method to biochemical reaction networks [46]. As opposed to the approaches mentioned above we consider AU and modify both, the solution approach for the DTMC as well as the solution approach for the associated birth process.

II. Mathematical Background

A. Transition Class Models

Consider a dynamical system with a finite number of discrete state variables such as the number of instances of some chemical species in a reaction volume. Assume that these variables change at discrete points in time. A *transition class* provides a rule for these changes and a function for the calculation of the state-dependent *transition rate* at which a state change occurs. Let S be a countable set of states.

A *transition class* C is a triple (G, u, α) such that (i) the guard $G \subset S$ is a subset of S, (ii) $u : G \to S$ is an injective update function with $u(x) \neq x$ for all $x \in G$, (iii) $\alpha : G \to \mathbb{R}_{>0}$ is a rate function. A *transition class model* (TCM) $M = (y, \{C_1, \ldots, C_m\})$ consists of an initial state $y \in S$ and a finite set of transition classes $C_1, \ldots, C_m$.

The set G contains all states x in which a transition of type C is possible and $u(x)$ is the target state of the transition. The probability of the C-transition depends on the transition rate $\alpha(x)$ in the way explained below.

In practice, we can usually express G by a finite number of constraints on the state variables, and u and α by elementary arithmetic functions. Thus, a TCM provides a finite description of a (possibly infinite-state) system.

Biochemical Reaction Networks: Transition class models can be used to model networks of biochemical reactions. Consider a fixed reaction volume with n different chemical species that is spatially homogeneous and in thermal equilibrium. Then, the state space of the system is given by $S = \mathbb{N}_0^n$. We assume that molecules collide randomly and that collisions may lead to chemical reactions. For a given set of chemical reaction types, we construct a TCM such that each transition class corresponds to a reaction type. The guard indicates whether the reaction type is possible in a state or not, that is, whether enough reactant molecules are available. The update function subtracts for each species the required numbers of reactant molecules and adds the number of produced molecules. The associated rate function determines the probability of a reaction in the way described in Section II-B.

Example 1: We consider a *crystallization example* that involves the chemical species A, A_2, B, and D. The two possible reaction types R_1 and R_2 are given by

$$R_1 : 2A \to A_2, \quad R_2 : A + B \to D.$$

Let $S = \mathbb{N}^4$ and $(x_1, x_2, x_3, x_4) \in S$. The transition class that corresponds to R_1 is defined as $C_1 = (G_1, u_1, \alpha_1)$, where $G_1 = \{(x_1, x_2, x_3, x_4) \in S \mid x_1 \geq 2\}$ (there must be at least two molecules of type A), and $u_1(x_1, x_2, x_3, x_4) = (x_1 - 2, x_2 + 1, x_3, x_4)$. The rate function is $\alpha_1(x_1, x_2, x_3, x_4) = c_1 \binom{x_1}{2} = c_1 x_1(x_1 - 1)/2$, where $c_1 > 0$ is a constant. For R_2 we define $C_2 = (G_2, u_2, \alpha_2)$ with $G_2 = \{(x_1, x_2, x_3, x_4) \in S \mid x_1, x_3 > 0\}$ (there must be at least one A and one B molecule), and $u_2(x_1, x_2, x_3, x_4) = (x_1 - 1, x_2, x_3 - 1, x_4 + 1)$. Let $c_2 > 0$ be a constant and define $\alpha_2(x_1, x_2, x_3, x_4) = c_2 x_1 x_3$.

Note that for a given initial state $y = (y_1, y_2, y_3, y_4)$ the set of reachable states is given by

$$\{(x_1, x_2, x_3, x_4) \in S \mid y_1 = x_1 + 2 \cdot x_2 + x_4, y_3 = x_3 + x_4\},$$

where we assume for simplicity that $y_2 = y_4 = 0$.

B. Chemical Master Equation

A transition class model $M = (y, \{C_1, \ldots, C_m\})$ represents a time-homogeneous, discrete-state Markov process (CTMC, for short) $(X(t))_{t \geq 0}$ with state space S.

The j-th entry of the random vector $X(t) = (X_1(t), \ldots, X_n(t))$ represents the value of the j-th state variable. Let $C_i = (G_i, u_i, \alpha_i)$, $1 \leq i \leq m$, and assume that at time $t \geq 0$ the process is in state $x \in G_i$.

The probability of a transition of type C_i occurring in the next infinitesimal time interval $[t, t+\tau)$, $\tau > 0$ is given by

$$Pr(X(t+\tau) = u_i(x) \mid X(t) = x) = \alpha_i(x) \cdot \tau.$$

Since y is the initial state of M we have $Pr(X(0) = y) = 1$, and for $x \in S$ we define the probability that X is in state x at time t by $p^{(t)}(x) = Pr\left(X(t) = x \mid X(0) = y\right)$. Recall that u_i is injective. To simplify our presentation, we define the set H_i as the set of all states x for which $u_i^{-1}(x)$ is defined, that is, that can be reached by a transition of type C_i. The *chemical master equation* (CME) describes the behavior of X by the differential equation [25]

$$\begin{aligned} \frac{\partial p^{(t)}(x)}{\partial t} &= \sum_{i:x\in H_i} \alpha_i(u_i^{-1}(x)) \cdot p^{(t)}(u_i^{-1}(x)) - \\ &- \sum_{i:x\in G_i} \alpha_i(x) \cdot p^{(t)}(x). \end{aligned} \tag{1}$$

Note that there exist pathological cases in which X is not uniquely defined by M [23]. These cases are, however, not relevant for the application area that we consider here. We therefore assume that M is such that it uniquely defines a Markov process X.

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

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

where we assume a fixed enumeration of the state space. 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 $P^{(0)}$ be equal to the identity matrix I, and for $t > 0$ we define $P^{(t)}$ as the matrix with entries $P^{(t)}(x,x') = Pr\big(X(h+t) = x' \mid X(h) = x\big)$. Note that $P^{(t)}$ is a stochastic matrix that does not depend on the time instant $h \geq 0$. Then the *Kolmogorov backward and forward equations* relate $P^{(t)}$ and Q by

$$\frac{\partial P^{(t)}}{\partial t} = QP^{(t)}, \quad \frac{\partial P^{(t)}}{\partial t} = P^{(t)}Q. \tag{2}$$

Let $p^{(t)}$ be the row vector with entries $p^{(t)}(x)$ for $x \in S$. We refer to the entries as *transient state probabilities*. The CME (see Eq. (1)) is obtained from Eq. (2) by multiplying both sides with $p^{(0)}$. A general solution of the CME is given by $p^{(t)} = p^{(0)}e^{Qt}$ and if Q is finite, from the definition of the matrix exponential

$$p^{(t)} = p^{(0)}e^{Qt} = p^{(0)} \sum_{k=0}^{\infty} \frac{(Qt)^k}{k!}. \tag{3}$$

An analytic solution for the function $p^{(t)}$ can however only be derived for special cases, such as in the case of a birth-death structure. If the underlying graph of the CTMC is acyclic, a closed-form expression for $p^{(t)}(x)$ can be calculated using the recursive scheme of the ACE algorithm [27]. In general, finding the state probabilities as a symbolic function of t is not possible. If Q is finite and the number of nonzero entries is of manageable size, an approximate numerical solution can be computed. Adding up a sufficiently large number of terms of the infinite sum in Eq. (3) is numerically unstable, as Q contains strictly positive and negative entries, leading to severe round-off errors [30]. Numerically stable methods are based on uniformization [24], [16] or approximations in the Krylov subspace [35]. Also numerical integration methods such as Runge-Kutta methods have been successfully used to compute $p^{(t)}$. Several surveys and comparisons exist in literature [15], [42], [39]. For realistic systems, however, upper bounds on the state variables of the system are often not known and even if upper bounds are known, the size of the (truncated) state space is still too large for an efficient solution using standard approaches.

III. Uniformization

A. Standard Uniformization

Assume that the exit rates of X are bounded, i.e.

$$\lambda := \sup_{x\in S} \lambda_x < \infty.$$

In general, CME models of biochemical reaction networks do not fulfill this restriction, because the rate functions α_i grow linearly in the number of reactant combinations which approach infinity if no upper bounds on the species' populations are known. We will see later that the method can be modified such that this restriction is no longer necessary.

Recall that I is the identity matrix. We define a Poisson process $(N(t))_{t\geq 0}$, the so-called *clock*. It follows a Poisson distribution, which means that $P(N(t) = k) = e^{-\lambda t}(\lambda t)^k/k! =: \psi_{\lambda t}(k)$ for $k \in \{0, 1, \ldots\}$. Furthermore, we define a discrete-time Markov chain (DTMC) $(Y(k))_{k\in\mathbb{N}_0}$, called *subordinated* chain, that is independent of N. Assume that Y has the same initial distribution as X and the (one-step) probability matrix $P = I + \frac{1}{\lambda}Q$. Then it can be shown that the CTMC $(Y(N(t)))_{t\geq 0}$ has the same transient state probabilities as X. Since

$$\begin{aligned} Pr(Y(N(t)) = x) &= \sum_{k=0}^{\infty} Pr(Y(k) = x, N(t) = k) \\ &= \sum_{k=0}^{\infty} Pr(Y(k) = x) \cdot Pr(N(t) = k) \end{aligned}$$

the transient state probabilities of X can then be computed as

$$p^{(t)} = \sum_{k=0}^{\infty} p^{(0)}P^k \cdot \psi_{\lambda t}(k) = \sum_{k=0}^{\infty} w^{(k)} \cdot \psi_{\lambda t}(k) \tag{4}$$

For $k \geq 1$, the stochastic matrix P^k contains the k-step transition probabilities of Y and, the vector $w^{(k)} = p^{(0)}P^k$ contains the state probabilities in Y after k steps.

Eq. 4 has nice properties compared to Eq. 3. The matrix P is stochastic and therefore all summands are positive. If the state space is finite, this leads to an iterative algorithm to compute $p^{(t)}$. It preserves the sparseness of P since we do not have to multiply P with P^{k-1} but only with $w^{(k-1)}$. Moreover, lower and upper summation bounds L and R can be obtained such that for each state x the truncation error

$$\begin{aligned} & p^{(t)}(x) - \textstyle\sum_{k=L}^{R} \psi_{\lambda t}(k) \cdot w^{(k)}(x) \\ = \; & \textstyle\sum_{k<L, k>R} \psi_{\lambda t}(k) \cdot w^{(k)}(x) \\ \leq \; & \textstyle\sum_{k<L, k>R} \psi_{\lambda t}(k) \\ = \; & 1 - \textstyle\sum_{k=L}^{R} \psi_{\lambda t}(k) < \epsilon \end{aligned} \tag{5}$$

can be a-priori bounded by a predefined error tolerance $\epsilon > 0$ [12]. Thus, $p^{(t)}$ can be approximated with accuracy ϵ by

$$p^{(t)} \approx \sum_{k=L}^{R} \psi_{\lambda t}(k) \cdot w^{(k)} \tag{6}$$

as long as the required number of summands is not extremely large. If the state space S is infinite, we can calculate L and R such that the inequality above holds and construct $\hat{P}$ such that it contains only the entries of states reachable in Y after at most R steps.

The SU algorithm described above has several drawbacks. It is inefficient if the system is *stiff*, i.e., if the reaction rates differ by several orders of magnitude. In this case, the time scale $[0, t)$ usually corresponds to the slow reactions whereas the minimal average residence time $1/\lambda$ has the time scale of the fast reactions. Thus, λt is large. But as λt grows the Poisson distribution flattens, and the left truncation point L in Eq. 6 grows linearly in λt, while the number of significant Poisson probability terms is $O(\sqrt{\lambda t})$ [12]. If the vectors $w^{(L)}, w^{(L+1)}, \ldots, w^{(R)}$ are computed using R matrix-vector multiplications, then the complexity of SU is $O(\nu \lambda t)$ where ν is the number of nonzero elements in P.

Another drawback is that even if the state space is truncated, the matrix P ($\hat{P}$ for infinite state space, respectively) may contain a large number of nonzero entries. Moreover, if the reaction rates are unbounded, it may be the case that no truncation point can be calculated [47].

B. Adaptive Uniformization

Adaptive uniformization overcomes the drawback related to stiffness mentioned above by replacing the Poisson process $(N(t))_{t\geq 0}$ with a *birth process* [11] $(B(t))_{t\geq 0}$. Intuitively, the clock B runs a slower speed than N and has fewer jumps within the time interval $[0, t)$. Therefore AU requires fewer terms in the truncated sum in Eq. (6), the downside being that the birth process is more expensive to solve than the Poisson process.

Consider a CTMC $(X(t))_{t\geq 0}$ with state space S, initial state y and generator matrix Q. For $k = 0, 1, \ldots$ let Q_k be such that for all $x, x' \in S$, $Q_k(x, x') = Q(x, x')$ if $x \in S_k$ and $Q_k(x, x') = 0$ otherwise. We inductively define a sequence $S_0, S_1, \ldots$ of subsets of S, a sequence of row vectors $w^{(0)}, w^{(1)}, \ldots$, and a sequence of numbers $\lambda_0, \lambda_1, \ldots$. Let $S_0 = \{y\}$ and $w^{(0)} = p^{(0)}$. For $k = 0, 1, \ldots$, we define

$$w^{(k+1)} = w^{(k)} \cdot P_k$$

and

$$S_{k+1} = \{x \in S \mid w^{(k+1)}(x) > 0\}, \tag{7}$$

where $\lambda_k = \max_{x \in S_k} \lambda_x$ and $P_k = I + \frac{1}{\lambda_k} Q_k$ is a stochastic matrix. The birth process B is then uniquely determined by the time-independent transition probabilities

$$Pr(B(t+\tau) = k+1 \mid B(t) = k) = \lambda_k \cdot \tau,$$

where $[t, t+\tau)$ is an infinitesimal time interval. Let $(Y(k))_{k\in\mathbb{N}_0}$ be the DTMC with step-dependent one-step transition probability matrices $P_0, P_1, \ldots$ and initial state y, that is, $Pr(Y(0) = y) = 1$ and

$$Pr(Y(k+1) = x' \mid Y(k) = x) = P_k(x, x')$$

for $k \in \mathbb{N}_0$. Note that $w^{(0)}$ is the initial distribution of Y and $w^{(k)}$ contains the state probabilities of Y after k steps. Van Moorsel showed that the CTMC $(Y(B(t)))_{t\geq 0}$ has the same transient state probabilities as $(X(t))_{t\geq 0}$ provided that B does not explode[1] [44]. Thus, the transient state probabilities of X can be written as

$$p^{(t)} = \sum_{k=0}^{\infty} w^{(k)} \cdot Pr(B(t) = k). \tag{8}$$

Similar to Eq. (6), we can derive truncation points for the sum above from the probability distribution of $B(t)$, that is, for $\epsilon > 0$, we choose truncation points L and R such that

$$\sum_{k=L}^{R} Pr(B(t) = k) \geq 1 - \epsilon.$$

Since the sets S_k are constructed during the iteration, the values $\lambda_1, \lambda_2, \ldots$ are not known a-priori, and nor are the truncation points L and R. We can, however, set $L = 0$ and add up summands $w^{(k)} \cdot Pr(B(t) = k)$ until the entries of the current approximation of $p^{(t)}$ sum up to at least $1 - \epsilon$.

If $\sup_{x\in S} \lambda_x = \lambda < \infty$, we can compare Eq. (8) and Eq. (4). We observe that $\lambda_k \leq \lambda$ for all k. Hence, for any infinitesimal time interval $[h, h+\tau)$,

$$\begin{aligned} & Pr(B(h+\tau) = k+1 \mid B(h) = k) = \lambda_k \cdot \tau \\ \leq \; & Pr(N(h+\tau) = k+1 \mid N(h) = k) = \lambda \cdot \tau. \end{aligned}$$

[1]The process B is said to explode iff the sum of the average residence times in the visited states converges, i.e., $\sum_{k\geq 0} \frac{1}{\lambda_k} < \infty$.

This means that during the interval $[0,t)$, the Poisson process N has at least as many jumps as B. This implies that the truncation of the sum in Eq. (8) w.r.t. a given accuracy ϵ may yield a smaller right truncation point compared to the truncation in Eq. (6). Hence, fewer matrix-vector multiplications have to be carried out. If the computational complexity of the algorithm is dominated by the computation of the vectors $w^{(k)}$, AU outperforms SU. For a large time horizon t, however, the right truncation point for AU often approaches that of SU, i.e., after a certain number ℓ of steps, $\lambda_k = \lambda$ for all $k \geq \ell$. Then AU becomes less efficient than SU.

Further drawbacks of adaptive uniformization are the fact that the computation of the values $Pr(B(t) = k)$ is more costly than the computation of the values $Pr(N(t) = k)$. The reason is that closed form expressions for the solution of a birth process in general do not give rise to numerically stable algorithms. Moreover, as mentioned above for AU the truncation points cannot be calculated a-priori since the construction of B is part of the iterative computation of the vectors $w^{(1)}, w^{(2)}, \ldots$.

C. Fast Adaptive Uniformization

We derive a fast variant of adaptive uniformization by modifying the vectors $w^{(0)}, w^{(1)}, \ldots$ such that all entries smaller than a small positive δ in the result $w^{(k+1)}$ of the matrix-vector multiplication $w^{(k)} \cdot P_k$ are set to zero. All remaining definitions are identical to those in the previous section. The modification of $w^{(k)}$ introduces an additional approximation error, since in each step probability mass is lost within $w^{(k)}$, i.e.

$$1 \geq \sum\nolimits_{x \in S} w^{(0)}(x) \geq \sum\nolimits_{x \in S} w^{(1)}(x) \geq \ldots.$$

It has, however, several important advantages that we list below.

- *Smaller Right Truncation Point:* The sets $S_0, S_1, \ldots$ may contain fewer states because the definition of S_k depends on $w^{(k)}$. Thus, $\lambda_0, \lambda_1, \ldots$ may be smaller since they are the maximal exit rates over fewer states. In this case, the birth process B has smaller transition probabilities than the birth process used in the original AU algorithm. In the case of smaller transition probabilities, less probability mass moves rightwards within $[0,t)$ and thus the right truncation point is smaller.
- *Smaller Vectors:* Each vector-matrix multiplication $w^{(k)} \cdot P_k$ requires less computational effort since $w^{(k)}$ contains fewer nonzero entries.
- *Non-explosive Birth Process:* For infinite Markov chains that are ergodic, the threshold δ also ensures that the limit of the sequence $S_0, S_1, \ldots$ will remain finite since only finitely many states can have a probability greater δ. Therefore, the sequence $\lambda_0, \lambda_1, \ldots$ will be bounded even if $\sup_{x \in S} \lambda_x = \infty$. Thus, B does not explode.

For the computation of the probabilities $Pr(B(t) = k)$, we use the following strategy. The generator matrix representing the birth process B is a simple infinite matrix Q_B with entries are $Q_B(k,k) = -\lambda_k$, $Q_B(k,k+1) = \lambda_k$, for $k \in \{0,1\ldots\}$, and zero elsewhere. We use standard uniformization to solve B and derive a subordinated DTMC Y_B with transition probability matrix $P_B = I + \frac{1}{\mu} Q_B$, where $\mu \geq \sup_{k \geq 0} \lambda_k$. Thus,

$$\begin{aligned} Pr(B(t) = k) &= \textstyle\sum_{l=0}^{\infty} Pr(Y_B(l) = k) \cdot \psi_{\mu t}(l), \\ &\approx \textstyle\sum_{l=L'}^{R'} Pr(Y_B(l) = k) \cdot \psi_{\mu t}(l) \end{aligned} \tag{9}$$

where $Pr(Y_B(0) = 0) = 1$. The matrix P_B inherits the simple structure of Q_B. The entry $P_B(k,k+1)$ equals

$$Pr(Y_B(l) = k+1 \mid Y_B(l-1) = k) = \tfrac{\lambda_k}{\mu} =: a_k$$

and the diagonal entry $P_B(k,k) = 1 - a_k$. Let $w_B^{(l)}(k) = Pr(Y_B(l) = k)$. Then

$$w_B^{(l)}(k) = a_{k-1} \cdot w_B^{(l-1)}(k-1) + (1 - a_k) \cdot w_B^{(l-1)}(k), \tag{10}$$

that is, after l steps, the birth process is in state k if after $l-1$ steps it is in the state $k-1$ and takes a transition to state k, or after $l-1$ steps it is in state k and takes the self-loop. In order to compute $w_B^{(l)}(k)$, we only need the transition rates $\lambda_0, \ldots, \lambda_k$ but not $\lambda_{k+1}, \lambda_{k+2}, \ldots$. It is important to point out that for the birth process we can afford the large number of iterations that are necessary during SU. The reason is that the simple structure of Y_B permits a fast computation of the values $w_B^{(l)}(k)$. Moreover, similar to our strategy for the solution of Y, we set entries in $w_B^{(l)}$ to zero if they drop below the threshold δ. This introduces an additional approximation error for $Pr(Y_B(l) = k)$, but results in a significant speed-up.

If we combine Eq. (9) and the solution of the DTMC Y, we obtain an approximation $\hat{p}^{(t)}$ for $p^{(t)}$, that is,

$$\begin{aligned} p^{(t)} &\approx \sum_{k=0}^{R} w^{(k)} \cdot Pr(B(t) = k) \\ &\approx \sum_{k=0}^{R} w^{(k)} \cdot \sum_{l=L'}^{R'} w_B^{(l)}(k) \cdot \psi_{\mu t}(l) =: \hat{p}^{(t)}. \end{aligned} \tag{11}$$

The outer sum is only truncated on the right and the truncation point R is found during the AU-iteration. For the inner sum, we can compute L' and R' a-priori as mentioned above for SU. Due to the simple structure of B, however, it is possible to derive closer truncation points. Instead of deriving L' and R' only from the inequality

$$\sum_{l=L'}^{R'} \psi_{\mu t}(l) > 1 - \epsilon,$$

we choose dynamical truncation points L'_k and R'_k depending on the probabilities $w_B^{(l)}(k)$. More precisely, we choose $[L'_k, R'_k]$ to be the smallest interval that includes all integers l for which $w_B^{(l)}(k) > \delta$ (compare Section IV-D).

Input	TCM $(y, \{C_1, \ldots, C_m\})$, time horizon t, threshold δ, max exit rate μ
Output	Approximation $\hat{p}$
Global Variables	State space $\hat{S}$, transition cl. $\{C_1, \ldots, C_m\}$, trunc. points $\hat{L}', \hat{R}'$, trans. prob. b, vect. *col*

```
1  Ŝ := {y}; y.dtmc := 1; //initialize Ŝ and y
2  L̂' := 0, R̂' := 0; b := 1; //initialize birth proc. vars.
3  k := 0;
4  do
5     λ̂ := collect(δ);
6     coeff := birthProcessState(k, λ̂, μ, t);
7     for each x ∈ Ŝ do
8        p̂(x) := p̂(x) + coeff * x.dtmc;
9     end for
10    propagate(λ̂);
11    k := k + 1;
12 until 1 − Σ_x p̂(x) achieves desired accuracy
13 return p̂.
```

Alg. 1. The main loop of the fast adaptive uniformization algorithm.

Both truncations in Eq. 11 lead to an underapproximation of the true value. The same holds for the error introduced by neglecting states in Y and Y_B whose entries in $w^{(k)}$ and $w_B^{(l)}$ drop below a certain threshold, respectively. Thus, the approximation $\hat{p}^{(t)}$ is an underapproximation of $p^{(t)}$ and the total error is given by $1-\sum_{x\in S}\hat{p}^{(t)}(x)$. In our experimental results, we report the total error using different values for δ. In the case that an a-priori specified error bound has to be met it is possible to repeat steps of the iteration if the total error exceeds the specified bound.

IV. Algorithm

A. Main Loop

The main loop of the algorithm is presented in Alg. 1. It approximates $p^{(t)}$, but if needed, intermediate results can be obtained by splitting the interval $[0, t]$ into several intervals. Each step of the while loop in line 4-12 represents a step in the subordinated DTMC Y, i.e. one summand of the outer sum in Eq. 11. In the k-th step the global state space $\hat{S}$ contains all states x for which $w^{(k)}(x)$ is at least δ, i.e., $\hat{S}$ is equal to the set S_k defined in Section III-C. The method *collect* computes the current maximal exit rate $\hat{\lambda}$ of all states in $\hat{S}$ (that is, $\hat{\lambda}$ is equal to λ_k as defined in Section III-C). The values *coeff* are approximations of the probabilities $Pr(B(t) = k)$, which are computed using input $\hat{\lambda}$.

In line 8 we compute for each $x \in \hat{S}$ the approximate value for $w^{(k)}(x) \cdot Pr(B(t) = k)$ and this new term of the summation of Eq. 11 is added to the approximation $\hat{p}$ of $p^{(t)}$. In line 10, we call *propagate* in order to compute an approximation of $w^{(k+1)}$ (see Section IV-C).

Field	Type	Description
dtmc	real	probability $w^{(k)}(x)$
acc	real	variable in which all propagated probabilities are added
er	real	exit rate λ_x of the state

Table I
Associated data structure of state x.

The global variable $\hat{S}$ is used by both propagate and collect methods, the transition classes C_i are used by the propagate method, while the rest of the global variables are used by the method that computes the birth process as it needs to keep the values of these variables from one call to another.

Note that an a-priori specified error bound for the total error cannot be guaranteed since as k becomes larger, $\sum_x \hat{p}(x)$ might not approach one. In our implementation, we choose the threshold δ several orders of magnitudes smaller than the desired accuracy (e.g. $\delta = 10^{-10}$) and stop the iteration if $1 - \sum_x \hat{p}(x)$ is small enough (e.g. 10^{-5}). It is, however, also possible to bound the number of iterations by considering the sequence of the transition rates $\hat{\lambda}$ in a similar way as Hahn et al. [17].

B. State Space

We use a dynamical data structure for the state space $\hat{S}$, where we associate with each state $x \in \hat{S}$ the fields listed in Table I. The field *dtmc* holds the approximate value $w^{(k)}(x)$ of the probability $P(Y(k) = x)$. Within *acc* the state accumulates the incoming probability for the next value of the field *dtmc* (see Section IV-C). Finally, *er* holds the exit rate λ_x of state x. In the following, we refer to the fields associated to x as $x.dtmc$, $x.acc$, and $x.er$. Note that each time a new state x is added to $\hat{S}$ the fields $x.dtmc$ and $x.acc$ are initialized with zero, and the field $x.er$ is initialized to λ_x. We update $\hat{S}$ in Alg. 1 by calling the methods *collect* and *propagate*. The former method removes all states x for which $w^{(k+1)}(x)$ is less than δ (line 5-7 in Alg. 3). The latter method adds all states in $S_{k+1} \setminus S_k$ (line 5-6 in Alg. 2).

C. Solution of the DTMC

The vector $w_{(k+1)}$ is obtained from $w^{(k)} \cdot P_k$ by setting to 0 entries whose value is below a threshold δ. The above matrix-vector product is computed in two phases, the propagate phase and the collect phase that push and pull probability mass without explicitly constructing the matrix P_k. The propagate method (presented in Alg. 2) iterates over all states $x \in \hat{S}$. For all transition classes $C = (G, u, \alpha)$ that are possible in x (i.e. $x \in G$), we compute the successor $x' = u(x)$ and the probability x' receives from x in the DTMC Y (line 4 and 7). If x' is not already part of the current state space then a new node is created and x' is

Method	propagate
Input	birth process transition rate $\hat{\lambda}$,

```
1  for all x ∈ Ŝ do
2    for all C = (G, u, α) ∈ {C_1, ..., C_m} do
3      if x ∈ G then
4        tmp := x.dtmc * α(x)/λ̂
5        x' := u(x);
6        if (x' ∉ Ŝ) then Ŝ := Ŝ ∪ {x'}
7        x'.acc := x'.acc + tmp;
8      end if;
9    end for;
10 end for;
11 x.dtmc := x.dtmc * (1 − x.er/λ̂)
```

Alg. 2. Propagate phase.

Method	collect
Output	transition rate $\hat{\lambda}$ of birth process

```
1  λ̂ := 0;
2  for all x ∈ Ŝ do
3    x.dtmc := x.dtmc + x.acc
4    x.acc := 0
5    if x.dtmc < δ then
6      Ŝ := Ŝ \ {x}
7    end if
8    if x.er > λ̂ then λ̂ := x.er;
9  end for;
10 return λ̂
```

Alg. 3. Collect phase.

added (line 6). Note that we use the field *acc* for this and that several predecessors may add probability to the field *acc* during the propagate phase. This additional field is needed because the old value of x'*.dtmc* cannot be altered before being used at its own turn by the loop at line 1. Recall that the self-loop probability of a state in Y is $1 - a_k = 1 - \frac{\lambda_x}{\lambda_k}$. In the last line of Alg. 2 we store in the field x*.dtmc* the probability that x receives from its self-loop. Note that the final result $w^{(k+1)}$ is calculated in method *collect* where all incoming probability mass of a state x is added up (see line 3).

D. Solution of the Birth Process

In order to compute the values $Pr(B(t) = k)$, we need to sum up the vectors $w_B^{(l)}(k)$ weighted with the Poisson probabilities $\psi_{\mu t}(l)$ (compare Eq. 9). As mentioned before, this requires an input μ such that $\mu \geq \max_{k \geq 0} \lambda_k$. To find a suitable candidate for μ, we use a heuristic that is based on the sliding window method [22]. The idea is to perform a cheap continuous over-approximation of the system's behavior during the interval $[0, t]$.

In Fig. 4 we illustrate the computation of the values $w_B^{(l)}(k)$. Assume that the values $w_B^{(l)}(k)$ are arranged in a matrix. In our algorithm we multiply (parts of) the k-th column with the Poisson probabilities $\psi_{\mu t}(l)$ in order to obtain an approximation of $Pr(B(t) = k)$. The variable *col* in Alg. 5 refers to the k-th column of which *col*$[l]$ is the l-th entry. Since we know a_{k-1} from the previous call of Alg. 5 as well as the $(k-1)$-th column of the matrix, we can compute column k as described in Eq. 10. Note that the self-loop probability $1 - a_k$ of Y_B is stored as global variable b.

For each call k of the method, the global variables L' and R' are updated on lines 9 and 11 to include the significant states of the DTMC Y_B. Before line 11 is executed, the values of L' and R' are equal to L'_k and R'_k as introduced in Section III-C.

Note that in line 6 the value of p is equal to $w_B^{(l)}(k)$ and, thus, $col[l]$ is equal to $w_B^{(l)}(k)$. This can be seen as follows: If $l = L'$ and $k = 0$ we have $L' = 0$ and $w_B^{(0)}(0) = 1 = p$. Note that in terms of the matrix $(w_B^{(l)}(k))_{l,k}$ all remaining entries in row $l = 0$ must be zero. Then, for $k = 0$ and $l > L'$, we have $a = 0$ and $b = 1 - \frac{\lambda_0}{\mu}$. In this case $col[l] = b^l$, which is the probability to stay in state 0 for l steps.

For $k > 0$, $l = L'$ is the first index for which $w_B^{(l)}(k-1) > \delta$. Thus, for all indices $l < L'$ we assume that $w_B^{(l)}(k-1) = 0$. Since, for $l < L'$, the entries $w_B^{(l)}(k)$ iteratively receive only probability from the entries in column $k-1$ until row $l-1$, we have $w_B^{(l-1)}(k) = 0$ and thus, by Eq. (10), $w_B^{(L')}(k) = 0$. But $p = 0$ in line 1 for $k > 0$ and thus, $w_B^{(L')}(k) = p$.

For $k > 0$ and $l > L'_k$, we know by induction that $col[l]$=$w_B^{(l)}(k-1)$ in line 5 and that $col[l] = p = w_B^{(l)}(k)$ in line 6. Thus, in line 7, p is set to $w_B^{(l+1)}(k) = a_{k-1} \cdot w_B^{(l)}(k-1) + (1 - a_k) \cdot w_B^{(l)}(k)$ (compare Eq. (10)).

V. Case Study

We implemented fast adaptive uniformization in C++ and run experiments on a 3.16 GHz Intel Linux PC with 6 GB of RAM. For our case study, we consider two examples for which we present our results in Table II.

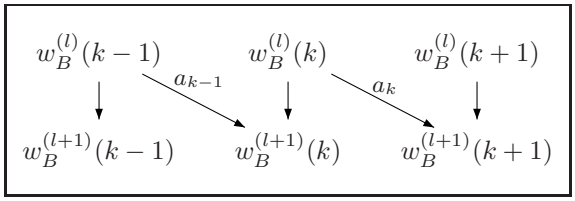

Figure 4. Computational scheme for the solution of Y_B.

Method	birthProcessState
Input	$\hat{\lambda}, k, \mu, t$.
Output	*coeff*

1 **if** $k = 0$ **then** $p := 1$ **else** $p := 0$;
2 $a := 1 - b$; $b := 1 - \frac{\hat{\lambda}}{\mu}$;
3 *coeff* $:= 0$;
4 **for** $l \in \{L', \ldots, R'\}$ **do**
5 *tmp* $:= a * col[l]$; // *store* $a_{k-1} \cdot w_B^{(l)}(k-1)$
6 $col[l] := p$; // *get* $w_B^{(l)}(k)$ *from prev. step*
7 $p :=$ *tmp* $+ b * col[l]$; // *add* $(1 - a_k) \cdot w_B^{(l)}(k)$
8 *coeff* $:=$ *coeff* $+ \psi_{\lambda t}(l) * col[l]$;
9 **if** $(l = R'$ and $col[R'] > \delta)$ $R' := R' + 1$;
10 **end for**;
11 **while** $(col[L'] \leq \delta)$ **do** $L' := L' + 1$; **end while**;
12 **return** *coeff*;

Alg. 5. On-the-fly birth process computation.

The first example is the crystallization example that we introduced in Section II-A. We chose rate constants $c_1 = c_2 = 10^{-7}$ and initial state $y = (10^6, 0, 10, 0)$ [18]. The time horizon for the crystallization example is $t = 100$. The second example is a model for the transcription regulation of a repressor protein in bacteriophage λ [14]. This protein is responsible for maintaining lysogeny of the λ virus in E. coli [1].

Phage λ Model: The Phage λ model involves 6 different species and 10 reactions. Thus, a state is a vector $x = (x_1, x_2, x_3, x_4, x_5, x_6) \in \mathbb{N}_0^6$. Note that infinitely many states are reachable in the corresponding CTMC. The transition classes $C_i = (G_i, u_i, \alpha_i)$, $1 \leq i \leq 10$ are given as follows [14].

- Production of proteins: $G_1 = \{x \in \mathbb{N}_0^6 \mid x_3 > 0\}$, $u_1(x) = (x_1 + 1, x_2, x_3, x_4, x_5, x_6)$, $\alpha_1(x) = c_1 x_3$.
- Degradation of proteins: $G_2 = \{x \in \mathbb{N}_0^6 \mid x_1 > 0\}$, $u_2(x) = (x_1 - 1, x_2, x_3, x_4, x_5, x_6)$, $\alpha_2(x) = c_2 x_1$.
- Production of mRNA: $G_3 = \{x \in \mathbb{N}_0^6 \mid x_5 > 0\}$, $u_3(x) = (x_1, x_2, x_3 + 1, x_4, x_5, x_6)$, $\alpha_3(x) = c_3 x_5$.
- Degradation of mRNA: $G_4 = \{x \in \mathbb{N}_0^6 \mid x_3 > 0\}$, $u_4(x) = (x_1, x_2, x_3 - 1, x_4, x_5, x_6)$, $\alpha_4(x) = c_4 x_3$.
- First dimer binding at operator site: $G_5 = \{x \in \mathbb{N}_0^6 \mid x_2, x_4 > 0\}$, $u_5(x) = (x_1, x_2 - 1, x_3, x_4 - 1, x_5 + 1, x_6)$, $\alpha_5(x) = c_5 x_2 x_4$.
- First dimer unbinding: $G_6 = \{x \in \mathbb{N}_0^6 \mid x_5 > 0\}$, $u_6(x) = (x_1, x_2 + 1, x_3, x_4 + 1, x_5 - 1, x_6)$, $\alpha_6(x) = c_6 x_5$.
- Second dimer binding at operator site: $G_7 = \{x \in \mathbb{N}_0^6 \mid x_2, x_5 > 0\}$, $u_7(x) = (x_1, x_2 - 1, x_3, x_4, x_5 - 1, x_6 + 1)$, $\alpha_7(x) = c_7 x_2 x_5$.
- Second dimer unbinding: $G_8 = \{x \in \mathbb{N}_0^6 \mid x_6 > 0\}$, $u_8(x) = (x_1, x_2 + 1, x_3, x_4, x_5 + 1, x_6 - 1)$, $\alpha_8(x) = c_8 x_6$.
- Dimerization: $G_9 = \{x \in \mathbb{N}_0^6 \mid x_1 > 1\}$, $u_9(x) = (x_1 - 2, x_2 + 1, x_3, x_4, x_5, x_6)$, $\alpha_9(x) = c_9 x_1(x_1 - 1)/2$.
- Dissociation into monomers: $G_{10} = \{x \in \mathbb{N}_0^6 \mid x_2 > 0\}$, $u_{10}(x) = (x_1 + 2, x_2 - 1, x_3, x_4, x_5, x_6)$, $\alpha_{10}(x) = c_{10} x_2$.

For $c_1, \ldots, c_{10}$, we choose $c_1 = 0.043$, $c_2 = 0.0007$, $c_3 = 0.0715$, $c_4 = 0.0039$, $c_5 = 1.992647 \times 10^{-2}$, $c_6 = 0.4791$, $c_7 = 1.992647 \times 10^{-4}$, $c_8 = 8.765 \times 10^{-12}$, $c_9 = 8.30269 \times 10^{-2}$, and $c_{10} = 0.5$ (see [14], [3]). The initial state of the system is given by $y = (2, 6, 0, 2, 0, 0)$ and the time horizon is $t = 300$.

Experimental Results: In Table II, we compare the running times for the phage λ model and the simple crystallization using standard uniformization and fast adaptive uniformization. Note that, as for fast AU, we modified SU such that probabilities $w^{(k)}(x) < \delta$ in the subordinated DTMC Y are neglected. Without this modification the number of nonzero entries in the vectors $w^{(0)}, w^{(1)}, \ldots$ becomes intractably large. The last row of Table II indicates the results of the original AU/SU algorithm, because in this case the threshold is $\delta = 0$. Each row in Table II corresponds to a different choice for δ. The right half of the table lists the results of the crystallization and the left half those of the phage λ example. The entry TO indicates that the running time exceeded 30 minutes (crystallization) and 60 minutes (phage λ).

We also considered fast AU without the dynamic truncation points discussed in Section IV-D. Since for this variant the running times are above the time out, we do not give details.

In Table II, we list in column **ex. time** the running times of fast AU and SU for the two examples. Column **states** refers to the total number of states that were considered during the iteration, i.e. $|\cup_k S_k|$. In the case of time out we give the number of states at the point of time out.

In the case of time out of the SU method, we list in column $\mathbf{k/R}$ the fraction of the number of iterations that were performed before time out. This fraction is given w.r.t. the total number of iterations that have to be performed to achieve an accuracy R of $\epsilon = 10^{-7}$ for the Poisson distribution (compare Eq. 5). Note that the total error (see column **tot. err.**) is higher since we remove entries in the vectors $w^{(k)}$ that are smaller than δ. Moreover, initially iterations are cheaper since the number of nonzero entries in the vectors $w^{(k)}$ is small, so the fraction of the number of iterations that were completed within the time out interval is only a rough estimate of the progress of the computation. The same cannot be done in case of a time out of the AU method as the truncation point R is not known a-priori in this case.

Our results show that for the crystallization example the speed up factor of fast AU compared to SU is large. Moreover, the total error is similar for all choices of δ

	Fast AU phage λ example			**SU** phage λ example			**Fast AU** crystallization example			**SU** crystallization example		
thres. δ	**ex. time**	**states**	**tot.err.**	**ex. time**	**states**	**tot.err.** or $\mathbf{k/R}$	**ex. time**	**states**	**tot.err.**	**ex. time**	**states**	$\mathbf{k/R}$
10^{-11}	$479s$	$2{\times}10^5$	$7{\times}10^{-3}$	$2020s$	$1.1{\times}10^5$	$5{\times}10^{-2}$	$56s$	$4.7{\times}10^6$	$1{\times}10^{-5}$	TO	$\geq 4.22{\times}10^6$	0.96
10^{-12}	$864s$	$3{\times}10^5$	$1{\times}10^{-3}$	TO	$>1.8{\times}10^5$	0.95	$58s$	$4.8{\times}10^6$	$4{\times}10^{-6}$	TO	$\geq 4.37{\times}10^6$	0.71
10^{-13}	$1412s$	$4{\times}10^5$	$1{\times}10^{-4}$	TO	$\geq 2.0{\times}10^5$	0.82	$60s$	$4.8{\times}10^6$	$2{\times}10^{-6}$	TO	$\geq 4.42{\times}10^6$	0.58
10^{-14}	$2229s$	$6{\times}10^5$	$3{\times}10^{-5}$	TO	$\geq 2.2{\times}10^5$	0.73	$64s$	$4.9{\times}10^6$	$2{\times}10^{-6}$	TO	$\geq 4.43{\times}10^6$	0.52
10^{-15}	$3270s$	$7{\times}10^5$	$1{\times}10^{-5}$	TO	$\geq 2.4{\times}10^5$	0.66	$66s$	$4.9{\times}10^6$	$2{\times}10^{-6}$	TO	$\geq 4.44{\times}10^6$	0.48
0	TO	$\geq 4{\times}10^6$		TO	$\geq 4.2{\times}10^6$	0.007	TO	$\geq 1.6{\times}10^7$		TO	$\geq 7.1{\times}10^5$	0.01

Table II
EXPERIMENTAL RESULTS FOR THE TWO EXAMPLE SYSTEMS.

whereas for the phage λ example the total error decreases for smaller δ. We observed that the number of iterations of fast AU was always significantly smaller than the number of iterations of SU, e.g. in the case of the phage lambda model and $\delta = 10^{-11}$, we had 302313 iterations for SU but only 19990 for AU.

VI. CONCLUSION

Numerical solution methods are often too expensive or inapplicable for the large or infinite state spaces that arise from stochastic population models in systems biology. This is why Monte-Carlo simulation methods, which sample the likely behavior of a system but avoid an exhaustive exploration of the state space, have been the main computational tool in this area.

We studied an intermediate algorithmic idea, which avoids constructing the entire state space yet approximates the numerical solution of the chemical master equation. Specifically, after a conversion to discrete time, our computation neglects states that have insignificant probability. For standard uniformization, this localization technique has been called the "sliding-window" method [22]. In this paper we show that the technique is even better suited for *adaptive* uniformization, where localization can be used not only to reduce the number of states that are processed at each step of the computation. It also permits larger time steps in the associated birth process, which results in fewer iterations. Indeed, only localization unleashes the full potential of adaptive uniformization, which is therefore often superior to standard uniformization.

In future work, we plan to experimentally compare our fast adaptive uniformization to other approximation techniques for the chemical master equation.

REFERENCES

[1] A. Arkin, J. Ross, and H. H. McAdams. Stochastic kinetic analysis of developmental pathway bifurcation in phage λ-infected E. coli cells. *Genetics*, 149:1633–1648, 1998.

[2] N. Barkai and S. Leibler. Biological rhythms: Circadian clocks limited by noise. *Nature*, 403:267–268, 2000.

[3] K. Burrage, M. Hegland, F. Macnamara, and B. Sidje. A Krylov-based finite state projection algorithm for solving the chemical master equation arising in the discrete modelling of biological systems. In A. N. Langville and W. J. Stewart, editors, *Proceedings of the Markov 150th Anniversary Conference*, pages 21–38. Boson Books, 2006.

[4] H. Busch, W. Sandmann, and V. Wolf. A numerical aggregation algorithm for the enzyme-catalyzed substrate conversion. In *Proc, of CMSB*, volume 4210 of *LNCS*, pages 298–311. Springer, 2006.

[5] E. de Souza e Silva and R. Gail. Transient solutions for Markov chains. In *Computational Probability*, chapter 3, pages 43–79. Kluwer Academic Publishers, 2000.

[6] F. Didier, T. A. Henzinger, M. Mateescu, and V. Wolf. Approximation of event probabilities in noisy cellular processes. In *Proc. of CMSB*, 2009. To appear.

[7] J. D. Diener and W. H. Sanders. Empirical comparison of uniformization methods for continuous-time Markov chains. In *Computations with Markov Chains*, pages 547–570. Kluwer Academic Publishers, Boston, 1995.

[8] M. B. Elowitz, M. J. Levine, E. D. Siggia, and P. S. Swain. Stochastic gene expression in a single cell. *Science*, 297:1183–1186, 2002.

[9] S. Engblom. Galerkin spectral method applied to the chemical master equation. *Comm. Comput. Phys.*, 5:871–896, 2009.

[10] N. Fedoroff and W. Fontana. Small numbers of big molecules. *Science*, 297:1129–1131, 2002.

[11] W. Feller. *An Introduction to Probability Theory and Its Applications, Volume 1*. Wiley, January 1968.

[12] B. L. Fox and P. W. Glynn. Computing Poisson probabilities. *Communications of the ACM*, 31(4):440–445, 1988.

[13] D. Gonze, J. Halloy, and A. Goldbeter. Robustness of circadian rhythms with respect to molecular noise. *PNAS, USA*, 99(2):673–678, 2002.

[14] J. Goutsias. Quasiequilibrium approximation of fast reaction kinetics in stochastic biochemical systems. *J. Chem. Phys.*, 122(18):184102, 2005.

[15] W. K. Grassmann, editor. *Computational Probability*. Kluwer Academic Publishers, 2000.

[16] D. Gross and D. Miller. The randomization technique as a modeling tool and solution procedure for transient Markov processes. *Operations Research*, 32(2):926–944, 1984.

[17] E. M. Hahn, H. Hermanns, B. Wachter, and L. Zhang. INFAMY: An infinite-state Markov model checker. In *Proc. CAV*, LNCS. Springer, 2009. To appear.

[18] E. L. Haseltine and J. B. Rawlings. Approximate simulation of coupled fast and slow reactions for chemical kinetics. *J. Chem. Phys.*, 117:6959–6969, 2002.

[19] M. Hegland, C. Burden, L. Santoso, S. Macnamara, and H. Booth. A solver for the stochastic master equation applied to gene regulatory networks. *J. Comput. Appl. Math.*, 205:708–724, 2007.

[20] A. Hellander. Efficient computation of transient solutions of the chemical master equation based on uniformization and quasi-Monte carlo. *J. Chem. Phys.*, 128(15):154109, 2008.

[21] D. A. Henderson, R. J. Boys, C. J. Proctor, and D. J. Wilkinson. Linking systems biology models to data: a stochastic kinetic model of p53 oscillations. In A. O'Hagan and M. West, editors, *Handbook of Applied Bayesian Analysis*. Oxford University Press, 2009.

[22] T. Henzinger, M. Mateescu, and V. Wolf. Sliding window abstraction for infinite Markov chains. In *Proc. CAV*, volume 5643 of *LNCS*, pages 337–352. Springer, 2009.

[23] T. A. Henzinger, B. Jobstmann, and V. Wolf. Formalisms for specifying Markovian population models. In *Proc. LIX Colloquium Reachability Problems*, volume 5797 of *LNCS*. Springer, 2009.

[24] A. Jensen. Markoff chains as an aid in the study of Markoff processes. *Skandinavisk Aktuarietidskrift*, 36:87–91, 1953.

[25] N. G. v. Kampen. *Stochastic Processes in Physics and Chemistry.* Elsevier, 3rd edition, 2007.

[26] H. Maamar, A. Raj, and D. Dubnau. Noise in gene expression determines cell fate in *Bacillus subtilis*. *Science*, 317(5837):526 – 529, 2007.

[27] R. A. Marie, A. L. Reibman, and K. S. Trivedi. Transient analysis of acyclic Markov chains. *Perform. Eval.*, 7(3):175–194, 1987.

[28] H. H. McAdams and A. Arkin. Stochastic mechanisms in gene expression. *PNAS, USA*, 94:814–819, 1997.

[29] H. H. McAdams and A. Arkin. It's a noisy business! *Trends in Genetics*, 15(2):65–69, 1999.

[30] C. B. Moler and C. F. Van Loan. Nineteen dubious ways to compute the exponential of a matrix. *SIAM Review*, 20(4):801–836, 1978.

[31] B. Munsky and M. Khammash. The finite state projection algorithm for the solution of the chemical master equation. *J. Chem. Phys.*, 124:044144, 2006.

[32] P. Patel, B. Arcangioli, S. Baker, A. Bensimon, and N. Rhind. DNA replication origins fire stochastically in fission yeast. *Mol. Biol. Cell*, 17:308–316, 2006.

[33] J. Paulsson. Summing up the noise in gene networks. *Nature*, 427(6973):415–418, 2004.

[34] S. Peles, B. Munsky, and M. Khammash. Reduction and solution of the chemical master equation using time scale separation and finite state projection. *J. Chem. Phys.*, 125:204104, 2006.

[35] B. Philippe and R. Sidje. Transient solutions of Markov processes by Krylov subspaces. In *Proc. of the 2nd International Workshop on the Numerical Solution of Markov Chains*, pages 95–119. Kluwer Academic Publishers, 1995.

[36] C. Rao, D. Wolf, and A. Arkin. Control, exploitation and tolerance of intracellular noise. *Nature*, 420(6912):231–237, 2002.

[37] A. Reibman and K. Trivedi. Numerical transient analysis of Markov models. *Comput. Oper. Res.*, 15(1):19–36, 1988.

[38] R. Sidje, K. Burrage, and S. MacNamara. Inexact uniformization method for computing transient distributions of Markov chains. *SIAM J. Sci. Comput.*, 29(6):2562–2580, 2007.

[39] R. Sidje and W. Stewart. A survey of methods for computing large sparse matrix exponentials arising in Markov chains. In *Markov Chains, Computational Statistics and Data Analysis 29*, pages 345–368, 1996.

[40] P. Sjöberg. *Numerical Methods for Stochastic Modeling of Genes and Proteins*. Phd thesis, Uppsala University, Sweden, 2007.

[41] P. Sjöberg, P. Löstedt, and J. Elf. Fokker-Planck approximation of the master equation in molecular biology. *CComputing and Visualization in Science*, 12:37–50, 2009.

[42] W. J. Stewart. *Introduction to the Numerical Solution of Markov Chains.* Princeton University Press, 1995.

[43] M. Thattai and A. van Oudenaarden. Intrinsic noise in gene regulatory networks. *PNAS, USA*, 98(15):8614–8619, July 2001.

[44] A. van Moorsel and W. Sanders. Adaptive uniformization. *ORSA Communications in Statistics: Stochastic Models*, 10(3):619–648, 1994.

[45] A. Warmflash and A. Dinner. Signatures of combinatorial regulation in intrinsic biological noise. *PNAS*, 105(45):17262–17267, 2008.

[46] J. Zhang, L. T. Watson, and Y. Cao. A modified uniformization method for the solution of the chemical master equation. Technical Report TR-07-31, Computer Science, Virginia Tech., 2007.

[47] L. Zhang, H. Hermanns, E. M. Hahn, and B. Wachter. Time-bounded model checking of infinite-state continuous-time Markov chains. In *ACSD*, 2008. China.