An adaptive solution to the chemical master equation using quantized tensor trains with sliding windows

Physical Biology

PAPER

An adaptive solution to the chemical master equation using quantized tensor trains with sliding windows

RECEIVED
2 March 2020

REVISED
25 June 2020

ACCEPTED FOR PUBLICATION
1 July 2020

PUBLISHED
29 October 2020

Trang Dinh* and **Roger B Sidje**

Department of Mathematics, University of Alabama, Tuscaloosa, United States of America
* Author to whom any correspondence should be addressed.

E-mail: tndinh@crimson.ua.edu and roger.b.sidje@ua.edu

Keywords: chemical master equation, sliding windows, adaptive finite state project, tensor train decomposition

Abstract

To cope with an extremely large or even infinite state space when solving the chemical master equation in biological problems, a potent strategy is to restrict to a finite state projection (FSP) and represent the transition matrix and probability vector in quantized tensor train (QTT) format, leading to savings in storage while retaining accuracy. In an earlier adaptive FSP–QTT algorithm, the multidimensional state space was downsized and kept in the form of a hyper rectangle that was updated when needed by selectively doubling some of its side dimensions. However, this could result in a much larger state space than necessary, with the effect of hampering both the execution time and stepping scheme. In this work, we improve the algorithm by enabling sliding windows that can dynamically slide, shrink or expand, with updates driven by a number of stochastic simulation algorithm trajectories. The ensuing state space is a considerably reduced hyper rectangle containing only the most probable states at each time step. Three numerical experiments of varying difficulty are performed to compare our approach with the original adaptive FSP–QTT algorithm.

1. Introduction

It is well known that networks with complex interacting components arise in a variety of disciplines [14], and that a master equation can help describe the joint probability function of the components over time. Of interest to us in this work is the chemical master equation (CME) used to model the randomness of biochemical processes in systems biology [13]. Upon setting the CME as a system of ordinary differential equations (ODE), its solution is the time-dependent probability mass function of the interacting species over time.

While our focus is on solving the CME directly, we refer the interested reader to the recent tutorial review of Schnoerr *et al* [31] that discussed a variety of other methods, for instance the chemical Langevin equation and moment closure approximations, just to name a few. We also note that closed-form solutions can usually be carried out only for extremely simple cases, such as monomolecular reaction systems [18].

One of the main disadvantages of solving the CME numerically is the so-called 'curse of dimensionality' that grows worse with the number of species and/or their population counts. Gillepsie's stochastic simulation algorithm (SSA) [12] is a roundabout way to deal with the CME, but it is time-consuming when numerous runs are needed to output an acceptable approximation. Munsky and Khamash's finite state projection (FSP) [24] is an alternate approach that truncates the state space and then seeks to directly approximate the probability function there. But using the FSP in its tradtional form to solve the CME is still time-consuming because the size of the transition matrix can dramatically blow up due to the aforementioned 'curse of dimensionality'. Addressing this challenge has motivated recent tensor developments that leveraged the observation of Hegland and Garcke [15] that the transition rate matrix can be written as a sum of Kronecker products of sparse matrices by having the state space in the form of a hyper rectangle and a lexicographic ordering of its multi-index. Building on this, Kazaev *et al* [20] showed that it can be stored at low memory cost using a decomposition in tensor train (TT) or, more particularly, quantized tensor train (QTT) [7, 20, 28].

It is remarkable that prior to these developments, Wolf [39] had earlier outlined a tensor representation based on stochastic automata networks,

which Neubrander [27] recently showed to be essentially equivalent to [15]. Even with tensors, solving the CME can still raise practical difficulties if not implemented with care. To understand why, recall that the so-called TT-ranks play an important role in the storage efficiency of the TT-format. But the TT-ranks tend to increase when performing certain operations with the TT-format, especially during the matrix-vector products that arise in many solution techniques of the CME. To overcome this situation, the authors in [20] used a QTT-structured hp-discontinuous Galerkin (hp-DG) discretization in time [21]. Another approach introduced in [7] used an implicit Crank–Nicolson integrator with the alternating minimal energy (AMEn) algorithm [8] for solving the linear equations in tensor formats. In these earlier studies, the reduced state space is fixed and treated as an input of the algorithm. As noted in [24], the reduced state space must be large enough for the solution to be accurate. Unfortunately, if the state space is much larger than necessary, there is an undue overhead in runtime. The more recent work in [35] pioneered a criteria based on the current probability vector to determine whether the current hyper rectangle is appropriate. If the criteria is not satisfied, the state space is increased by doubling some sides of the hyper rectangle and the probability vector is padded with zeroes to be compatible with the new hyper rectangle. However, the initial state space in the hyper rectangle format was also an input, along with the initial state of the system, and there was no suggested programmatic strategy to choose an ideal initial hyper rectangle.

Here, we advance the work in [35]. Instead of using a hyper rectangle with all lower bounds permanently anchored at 0 and that may contain a lot of implausible states, we incorporate sliding windows [40]. The initial window is automatically generated to contain a tight neighborhood of the initial state. From there, at each time step, the window is updated in two stages: first it is reduced to only keep states with meaningful probabilities, and then it is expanded using the SSA to predict states that are likely to be visited within the next time step. As in [35], the probability vector is then approximated by a combination of the inexact uniformization method [32] with the AMEn solver [8]. Similarly to the original FSP algorithm, we also check the L^1-norm during integration. Components of a good approximation should sum close to one, and so if the gap exceeds a tolerance, the state space is expanded by extra SSA runs to get a more reliable solution [33]. In summary, our study has built on previous works [25, 33, 40] that seek to track the support drift and we have explored new directions with tensors that have led to improved findings that we share here. The tighter tracking of our approach reduces the runtimes, and its coupling with the QTT format enables much larger problems. This opens an avenue for investigating other problems such as the reaction–diffusion master equation where the state space is very large and where the conventional FSP is challenged [9].

The organization of the paper is as follows. Section 2 reviews the CME and FSP as well as the TT format and some of its related operations. It also summarizes the original FSP–QTT algorithm in [35], where an adaptive tensor formulation of the FSP on hyper rectangles was discussed. Section 3 takes from there to outline our improvements, including details on how we make sliding windows work with the QTT-format. Section 4 shows the performance of our method on some biological examples. Section 5 finally gives some concluding remarks.

2. Preliminaries on the CME, FSP, and tensors

2.1. CME and FSP

Consider a biochemical process of N chemical species interacting via M reactions. The state of the system at any time t is a vector $\mathbf{x} = (x_1, \ldots, x_N)^T$, where x_i is a population count of the ith species, $1 \leqslant i \leqslant N$. Let $P(\mathbf{x}, t)$ be the probability that the system is at state $\mathbf{x}$ at time t. The CME [13] states that

$$\frac{\mathrm{d}P(\mathbf{x},t)}{\mathrm{d}t} = \sum_{k=1}^{M} \alpha_k(\mathbf{x}-\boldsymbol{\nu}_k)\,P(\mathbf{x}-\boldsymbol{\nu}_k,t) - \sum_{k=1}^{M} \alpha_k(\mathbf{x})\,P(\mathbf{x},t), \tag{1}$$

where, for the kth reaction, $1 \leqslant k \leqslant M$, the $\boldsymbol{\nu}_k$ and α_k are, respectively, the stoichiometric vector and propensity function.

For many biological problems, the size of CME can be infinite or too large to be handled directly. Restricting the state space to a finite set $\mathbf{X}$ of cardinality n through the FSP [2, 4, 24], we can rewrite (1) as a reduced set of linear ODEs

$$\dot{\mathbf{p}}(t) = \mathbf{A}\cdot\mathbf{p}(t), \quad t \in [0, t_{\mathrm{f}}], \tag{2}$$

where the transition rate matrix $\mathbf{A} = [a_{ij}] \in \mathbb{R}^{n\times n}$ is defined as

$$a_{ij} = \begin{cases} -\sum_{k=1}^{M} \alpha_k(\mathbf{x}_j), & \text{if } i = j, \\ \alpha_k(\mathbf{x}_j), & \text{if } \mathbf{x}_i = \mathbf{x}_j + \boldsymbol{\nu}_k, \\ 0, & \text{otherwise.} \end{cases}$$

Assuming that the initial state of the system at time $t = 0$ is known, the probability vector of the system at the time point t_{f} is given by

$$\mathbf{p}(t_{\mathrm{f}}) = \exp(t_{\mathrm{f}}\mathbf{A})\,\mathbf{p}(0),$$

where the exponential matrix is defined as

$$\exp(t_f\mathbf{A}) = \sum_{m=0}^{\infty} \frac{(t_f\mathbf{A})^m}{m!}.$$

The CME matrix $\mathbf{A}$ is sparse under any order relation imposed on the state space, and it is natural to use the lexicographic order on the state space $\mathbf{X}$ as a subset of $\mathbb{N}$, the set of nonnegative natural numbers. An important finding is that if we impose the lexicographic order on a state space $\mathbf{X}$ in the form of a window of lower bound $(lb^{(1)}, \ldots, lb^{(N)})$ and upper bound $(ub^{(1)}, \ldots, ub^{(N)})$, i.e., $\mathbf{X}$ consists of states $\mathbf{x} = (x_1, \ldots, x_N)^T$ such that $lb^{(s)} \leqslant x_s \leqslant ub^{(s)}, \quad \forall\, s = 1, \ldots, N$, then the transition rate matrix can be decomposed into a sum of Kronecker products [7, 15, 20, 39],

$$\mathbf{A} = \sum_{k=1}^{M} \left(\otimes_{s=1}^{N} \mathbf{S}_k^{(s)} - \mathbf{I} \right) \mathbf{M}_k. \tag{3}$$

Each term in the sum corresponds to a chemical reaction; $\mathbf{S}_k^{(s)}$ is a 'shift-diagonal' matrix corresponding to the change in species s when reaction k happens; $\mathbf{I}$ is the identity matrix; and $\mathbf{M}_k$ is the diagonal matrix that stores the values of the propensity function α_k. Precisely, let $I_s = ub^{(s)} - lb^{(s)} + 1$ be the window size for species s, $n = I_1 \cdot \ldots \cdot I_N$ be the total number of states in $\mathbf{X}$, and denote $\boldsymbol{\nu}_k = \left(\nu_k^{(1)}, \ldots, \nu_k^{(N)}\right)^T$. The identity matrix $\mathbf{I}$ is then of order n, $\mathbf{S}_k^{(s)} \in \mathbb{R}^{I_s \times I_s}$ is given by

$$\mathbf{S}_k^{(s)} = \begin{cases} \begin{pmatrix} 0 & \cdots & 1 & & \\ & \ddots & & \ddots & \\ & & \ddots & & 1 \\ & & & \ddots & \vdots \\ & & & & 0 \end{pmatrix} & \text{shifted up } \left|\nu_k^{(s)}\right| \text{ rows} \quad \text{if } \nu_k^{(s)} < 0, \\ \begin{pmatrix} 0 & & & & \\ \vdots & \ddots & & & \\ 1 & & \ddots & & \\ & \ddots & & \ddots & \\ & & 1 & \cdots & 0 \end{pmatrix} & \text{shifted down } \nu_k^{(s)} \text{ rows} \quad \text{if } \nu_k^{(s)} \geqslant 0, \end{cases}$$

and

$$\mathbf{M}_k = \operatorname{diag}\left(\alpha_k(\mathbf{x}_1), \alpha_k(\mathbf{x}_2), \ldots, \alpha_k(\mathbf{x}_n)\right), \tag{4}$$

where $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n$ are states of $\mathbf{X}$ in the increasing lexicographic order.

Similar to [35], we assume that the propensity functions are separable, i.e.,

$$\alpha_k(\mathbf{x}) = \alpha_k^{(1)}(x_1) \cdot \ldots \cdot \alpha_k^{(N)}(x_N), \quad k = 1, \ldots, M, \tag{5}$$

where $\alpha_k^{(\cdot)} \geqslant 0$ are scalar functions. This condition is satisfied by standard mass-action kinetics but not Hill functions with multiple species in the denominator.

As a result of (5), formula (4) can be rewritten as

$$\mathbf{M}_k = \otimes_{s=1}^{N} \operatorname{diag} \alpha_k^{(s)}, \tag{6}$$

where

$$\operatorname{diag} \alpha_k^{(s)} = \operatorname{diag}\left(\alpha_k^{(s)}\left(lb^{\{s\}}\right), \alpha_k^{(s)}\left(lb^{\{s\}}+1\right), \ldots, \alpha_k^{(s)}\left(ub^{\{s\}}\right)\right), s = 1, \ldots, N.$$

The lower bounds in [35] are $lb^{(s)} = 0$ and $ub^{(s)} = I_s - 1$. Therefore,

$$\operatorname{diag} \alpha_k^{(s)} = \operatorname{diag}\left(\alpha_k^{(s)}(0), \alpha_k^{(s)}(1), \ldots, \alpha_k^{(s)}(I_s - 1)\right), \quad s = 1, \ldots, N.$$

Under these assumptions, the transition matrix can be represented by simple matrices of small sizes,

$$\mathbf{A} = \sum_{k=1}^{M} \left(\otimes_{s=1}^{N} \mathbf{S}_k^{(s)} - \mathbf{I} \right) \left(\otimes_{s=1}^{N} \operatorname{diag} \alpha_k^{(s)} \right). \tag{7}$$

We now outline how both the transition operator $\mathbf{A}$ and the probability distribution $\mathbf{p}$ can be kept as multidimensional arrays (or tensors) that allow us to use tensor decompositions [7, 15, 20, 21] to resolve storage issues.

2.2. CME in tensor train format

We begin by recapping the terminology, notation and some tensor operations that are necessary to understand how to recast the CME in a more general TT format. Recall that a Nth-order tensor $\mathcal{X} \in \mathbb{R}^{I_1 \times \cdots \times I_N}$ is a N-way array of real numbers $\mathcal{X}(i_1, \ldots, i_N)$, with $i_k \in \{1, \ldots, I_k\}$. We call $\mathcal{X}$ a tensor of *mode sizes* $I_1, \ldots, I_N$. An explicit storage for this tensor would require $\prod_{k=1}^{N} I_k = O(I^N)$ entries, where $I = \max\{I_1, \ldots, I_N\}$. Because real world problems often require the order N to be high, it becomes impractical to implement tensors in their standard representation. This has motivated decomposition techniques such as the TT [28], where $\mathcal{X}$ is decomposed as

$$\mathcal{X}(i_1, \ldots, i_N) = \mathbf{X}_1(i_1) \cdot \ldots \cdot \mathbf{X}_N(i_N), \qquad (8)$$

in which each $\mathbf{X}_k(i_k)$ is a $r_{k-1} \times r_k$ matrix, $k = 1, \ldots, N$, with the end conditions $r_0 = 1 = r_N$.

The quantities r_k and the arrays $\mathbf{X}_k$ are called, respectively, the TT *ranks*, and TT *cores* of $\mathcal{X}$. The $\mathbf{X}_k \in \mathbb{R}^{r_{k-1} \times I_k \times r_k}$ and $\mathbf{X}_k(\alpha, i_k, \beta)$ references the (α, β)-entry of the matrix $\mathbf{X}_k(i_k)$. The TT format only stores the TT cores at a total cost of $O(r_{\mathrm{TT}}^2 \cdot N \cdot I)$ entries, where $r_{\mathrm{TT}} = \max\{r_1, \ldots, r_{N-1}\}$. Hence, provided that r_{TT} is small, storing $\mathcal{X}$ in TT format only scales linearly with N rather than exponentially as the expanded form does. As hinted in our introduction, the storage efficiency of the TT format is therefore tied to the TT ranks and illustrates the need to maintain small ranks during tensor operations. The dramatic savings in storage unleash the ultimate potential to tackle problems that were beyond reach, with the trade-off that accessing an array entry now requires an algorithmic computation (8) instead of the simpler conventional table lookup.

A popular variant to the TT format is the QTT format where, assuming $I_s = 2^{L_s}$, $L_s \in \mathbb{N}$, $s = 1, \ldots, N$, the tensor $\mathcal{X} \in \mathbb{R}^{I_1 \times \cdots \times I_N}$ is reshaped into a tensor $\mathcal{Y}$ of order $L = L_1 + \cdots + L_N$ and size $\underbrace{2 \times \cdots \times 2}_{L \text{ times}}$ such that

$$\mathcal{Y}\left(\ell_1^{(1)}, \ldots, \ell_{L_1}^{(1)}, \ell_1^{(2)}, \ldots, \ell_{L_2}^{(2)}, \ldots, \ell_1^{(N)}, \ldots, \ell_{L_N}^{(N)}\right) = \mathcal{X}(i_1, i_2, \ldots, i_N),$$

in which $\ell_k^{(s)} - 1 \in \{0, 1\}$ are obtained from the binary representation of $i_s - 1$, i.e.,

$$i_s - 1 = \left(\ell_1^{(s)} - 1\right) + \cdots + 2^{L_s - 1}\left(\ell_{L_s}^{(s)} - 1\right).$$

The TT format of $\mathcal{Y}$ is called the QTT *format* of $\mathcal{X}$ and its ranks are called QTT *ranks*. Denoting r_{QTT} the maximum of the QTT ranks, the storage complexity becomes $O(r_{\mathrm{QTT}}^2 \cdot N \cdot \log_2 I)$, so that it scales logarithmic in the mode size I if r_{QTT} is relatively small. This reduces the storage cost even further from the TT format.

Now, to recast the CME in tensor train format, the transition matrix $\mathbf{A}$ is represented in the TT matrix format, where it is indexed by rows $(i_1, \ldots, i_N)$ and columns $(j_1, \ldots, j_N)$, and satisfies

$$\mathbf{A}\left((i_1, \ldots, i_N), (j_1, \ldots, j_N)\right) = \mathbf{A}_1(i_1, j_1) \cdot \ldots \cdot \mathbf{A}_N(i_N, j_N), \qquad (9)$$

where each $\mathbf{A}_k(i_k, j_k)$ is a $r_{k-1} \times r_k$ matrix, along with $r_0 = 1 = r_N$.

2.3. Adaptive FSP–QTT on hyper rectangles

In the adaptive integration procedure for solving the CME in [35], the authors construct the approximate solution through time points $t_0 = 0 < t_1 < \cdots < t_K := t_{\mathrm{f}}$ using the recurrence

$$\mathbf{p}(t_{k+1}) \approx \mathbf{p}_{k+1} := \exp\left(\tau_k \mathbf{A}_{\mathbf{H}_k}\right) \mathbf{p}_k, \quad t_{k+1} = t_k + \tau_k, \qquad (10)$$

where τ_k is the step-size, $\mathbf{H}_k$ is a state space in the form of a hyper rectangle,

$$\mathbf{H}_k = \left[0 : I_1^k - 1\right] \times \cdots \times \left[0 : I_N^k - 1\right],$$

where $I_i^k = 2^{L_i^k}$, $L_i^k \in \mathbb{N}$, for $i = 1, \ldots, N$, and $\mathbf{A}_{\mathbf{H}_k}$ is the truncated transition matrix on $\mathbf{H}_k$. The initial hyper rectangle $\mathbf{H}_0$ is given as an input of the algorithm which contains the initial state, and the initial distribution $\mathbf{p}_0$ is defined on $\mathbf{H}_0$ with 1 at the initial state and 0 elsewhere.

At each time point t, the authors use the inexact uniformization method [32],

$$\exp(\tau \mathbf{A})\mathbf{v} \approx \mathrm{e}^{-\theta\tau}\left(\mathbf{v} + \theta\tau\mathbf{P}\mathbf{v} + \cdots + \frac{(\theta\tau)^m}{m!}\mathbf{P}^m\mathbf{v}\right),$$
$$\mathbf{P} := \frac{1}{\theta}\mathbf{A} + \mathbf{I}, \qquad (11)$$

where $\mathbf{I}$ is the identity matrix, and $\theta \geqslant \max\left(-\operatorname{diag}(\mathbf{A})\right)$. With the integration domain $[t, t_{\mathrm{f}}]$, the step size τ is chosen to be $\tau := \frac{t_{\mathrm{f}} - t}{n_{\mathrm{step}}}$, with $n_{\mathrm{step}} = \lceil \theta(t_{\mathrm{f}} - t)/\sigma \rceil$, and the safety parameter σ is chosen to prevent underflow in the factor $\mathrm{e}^{-\theta\tau}$.

Finally, the vectors $\mathbf{f}_j = \mathbf{P}^j\mathbf{v}$, $j = 0, \ldots, m$, in the right-hand side of (11) are arranged to be the solution of the linear equation,

$$\begin{pmatrix} \mathbf{I} & & & \\ -\mathbf{P} & \mathbf{I} & & \\ & \ddots & \ddots & \\ & & -\mathbf{P} & \mathbf{I} \end{pmatrix} \begin{pmatrix} \mathbf{f}_0 \\ \mathbf{f}_1 \\ \vdots \\ \mathbf{f}_m \end{pmatrix} = \begin{pmatrix} \mathbf{v} \\ \mathbf{0} \\ \vdots \\ \mathbf{0} \end{pmatrix},$$

which was solved in tensor form by the AMEn method [8]. It is clear that the overhead of using tensors and the AMEn machinery may result in slower runtime if applied to simpler problems suited for cheaper conventional methods.

The integration scheme (10) undertakes all these calculations in each time interval. To decide whether to accept a step and/or construct the next hyper rectangle, [35] then uses a FSP-like criteria based on estimating the FSP truncation error by augmenting the

hyper rectangle with an artificial absorbing boundary aimed at collecting probability leaks. If the leak on a side of the current hyper rectangle was less than a given tolerance, that side remained the same. Otherwise, that side length was doubled.

The intent of estimating the FSP truncation error through an artificial absorbing boundary in [35] was to avoid computing the norm over potentially immense hyper rectangles. Because of the many approximation levels in solving the CME, having a robust error estimate is a strength. We were able to revert to the classic FSP criteria in our new approach. Indeed, rather than having hyper rectangles that repeatedly increase, and this only by doubling, our new approach has the ability to slide, shrink, and tightly expand the hyper rectangles. While this prevents using the simple mechanism explained in [35] to reuse the QTT data, our experiments with recomputing the data and having robust error estimates via our considerably smaller hyper rectangles will show a faster overall runtime.

3. Adaptive FSP–QTT with sliding windows

We now describe the particularities of our approach in greater detail. Our state space at each time point t_k is of the form

$$\mathbf{H}_k = \left[lb_k^{(1)} : ub_k^{(1)}\right] \times \cdots \times \left[lb_k^{(N)} : ub_k^{(N)}\right],$$

with integer bounds $0 \leqslant lb_k^{(s)} \leqslant ub_k^{(s)}$, $s = 1, \ldots, N$. Given $\mathbf{p}_k$ at t_k, the next $\mathbf{p}_{k+1}$ is computed from $\mathbf{p}_k$ and the transition matrix is generated by the window $\mathbf{H}_k$. We then only have the TT format of $\mathbf{p}_{k+1}$ mapped to $\mathbf{H}_k$, and the first question to tackle is how to efficiently remap the TT format of $\mathbf{p}_{k+1}$ to a new window $\mathbf{H}_{k+1}$, without the assumptions that its side lengths are 0-based or that they only change by doubling, as was the case in [35].

The problem is precisely formulated as follows. Let a probability vector $\mathbf{p}_{k+1}$ on $\mathbf{H}_k$ be characterized by a TT tensor

$$\mathbf{v} \in \mathbb{R}^{I_1 \times \cdots \times I_N}, \quad I_s = ub_k^{(s)} - lb_k^{(s)} + 1, \quad s = 1, \ldots, N,$$

by identifying, for each state $\mathbf{x} = (x_1, \ldots, x_N)$ in $\mathbf{H}_k$,

$$\mathbf{p}_{k+1}(\mathbf{x}) = \mathbf{v}(i_1, \ldots, i_N), \quad x_s = i_s + lb_k^{(s)} - 1, \quad s = 1, \ldots, N. \tag{12}$$

Consider now another window of states,

$$\mathbf{H}_{k+1} = \left[lb_{k+1}^{(1)} : ub_{k+1}^{(1)}\right] \times \cdots \times \left[lb_{k+1}^{(N)} : ub_{k+1}^{(N)}\right].$$

The goal is to find a TT tensor

$$\mathbf{w} \in \mathbb{R}^{J_1 \times \cdots \times J_N}, \quad J_s = ub_{k+1}^{(s)} - lb_{k+1}^{(s)} + 1, \quad s = 1, \ldots, N,$$

so that the distribution $\hat{\mathbf{p}}_{k+1}$ defined for each state $\mathbf{x} = (x_1, \ldots, x_N)$ in $\mathbf{H}_{k+1}$ by

$$\hat{\mathbf{p}}_{k+1}(\mathbf{x}) = \mathbf{w}(j_1, \ldots, j_N), \quad x_s = j_s + lb_{k+1}^{(s)} - 1, \quad s = 1, \ldots, N, \tag{13}$$

satisfies the following property

$$\hat{\mathbf{p}}_{k+1}(\mathbf{x}) = \begin{cases} \mathbf{p}_{k+1}(\mathbf{x}), & \mathbf{x} \in \mathbf{H}_{k+1} \bigcap \mathbf{H}_k, \\ 0, & \text{otherwise.} \end{cases} \tag{14}$$

Note that for a state $\mathbf{x} = (x_1, \ldots, x_N)$, we have

$$x_s = \begin{cases} i_s + lb_k^{(s)} - 1 & \text{if } \mathbf{x} \in \mathbf{H}_k \\ j_s + lb_{k+1}^{(s)} - 1 & \text{if } \mathbf{x} \in \mathbf{H}_{k+1} \end{cases}, \quad s = 1, \ldots, N.$$

Hence we have the following connection between the two parameterizations of a state $\mathbf{x} \in \mathbf{H}_{k+1} \bigcap \mathbf{H}_k$,

$$i_s = j_s + lb_{k+1}^{(s)} - lb_k^{(s)}, \quad s = 1, \ldots, N. \tag{15}$$

Moreover, if a state $\mathbf{x}$ is found to be in $\mathbf{H}_{k+1}$, we can check that $\mathbf{x}$ was also in the previous $\mathbf{H}_k$ if and only if

$$1 \leqslant j_s + lb_{k+1}^{(s)} - lb_k^{(s)} \leqslant I_s, \quad \forall\, s = 1, \ldots, N. \tag{16}$$

The following gives a constructive method on how to go from the characterization $\mathbf{v}$ that represents $\mathbf{p}_{k+1}$ on $\mathbf{H}_k$ to a characterization $\mathbf{w}$ that represents $\hat{\mathbf{p}}_{k+1}$ on $\mathbf{H}_{k+1}$.

Proposition 3.1. *Let $G_s^{(\mathbf{v})} \in \mathbb{R}^{r_{s-1} \times I_s \times r_s}$, $s = 1, \ldots, N$ be the known TT cores of $\mathbf{v} \in \mathbb{R}^{I_1 \times \cdots \times I_N}$ corresponding to $\mathbf{p}_{k+1}$. If we define the TT tensor $\mathbf{w} \in \mathbb{R}^{J_1 \times \cdots \times J_N}$ with cores $G_s^{(\mathbf{w})} \in \mathbb{R}^{r_{s-1} \times J_s \times r_s}$ in terms of the cores of $\mathbf{v}$ as*

$$G_s^{(\mathbf{w})}(\alpha_{s-1}, j_s, \alpha_s) = \begin{cases} G_s^{(\mathbf{v})}\left(\alpha_{s-1}, j_s + lb_{k+1}^{(s)} - lb_k^{(s)}, \alpha_s\right), & 1 \leqslant j_s + lb_{k+1}^{(s)} - lb_k^{(s)} \leqslant I_s, \\ 0, & \text{otherwise,} \end{cases} \tag{17}$$

then the distribution $\hat{\mathbf{p}}_{k+1}$ characterized by $\mathbf{w}$ satisfies (14).

Proof. By definition of the TT cores, we have

$$\mathbf{p}_{k+1}(\mathbf{x}) = \mathbf{v}(i_1, \ldots, i_N) = G_1^{(\mathbf{v})}(i_1) \cdot \ldots \cdot G_N^{(\mathbf{v})}(i_N),$$

Algorithm 1. Update of probability with sliding windows.

Input: TT-tensor $\mathbf{v} \in \mathbb{R}^{I_1 \times \cdots \times I_N}$, $\mathbf{H}_k$, $\mathbf{H}_{k+1}$.
Output: TT-tensor $\mathbf{w} \in \mathbb{R}^{J_1 \times \cdots \times J_N}$ that satisfies (17).
1: Set the dimension, mode sizes and ranks of $\mathbf{w}$:
$\mathbf{w}.d = N$, $\mathbf{w}.n = (J_1, \ldots, J_N)$ and $\mathbf{w}.r = \mathbf{v}.r = (r_0, r_1, \ldots, r_N)$.
2: Set the markers of $\mathbf{w}$ by $ps_1 = 1$, and $ps_{k+1} = (ps_k + r_{k-1}) \cdot J_k \cdot r_k$, for $k = 2, \ldots, N$
3: Set the core for $\mathbf{w}$ as a vector of length $ps_{N+1} - 1$.
4: **for** $s = 1 : N$ **do**
  5: **for** $j_s = 1 : J_s$ **do**
    6: **if** [j_s satisfies (16)] **then**
      $G_s^{(\mathbf{w})}(:, j_s, :) = G_s^{(\mathbf{v})}\left(:, j_s + lb_{k+1}^{(s)} - lb_k^{(s)}, :\right)$.
    else
      $G_s^{(\mathbf{w})}(:, j_s, :) = 0$.
    end
  end
end

Table 1. Parameters used in the solver for algorithm 3.

Parameters	Default values in our solver	Meaning
ϵ_{FSP}	10^{-4}	Tolerance for FSP
ϵ_{unif}	10^{-4}	Tolerance for uniformization
Tol_{AMEn}	10^{-4}	Tolerance for the AMEn solver
$\text{Tol}_{\text{window-reduction}}$	10^{-15}	Tolerance to reduce window size

Table 2. Breakdown of the runtimes of the main subtasks in the original FSP–QTT algorithm and our FSP–QTT algorithm with sliding windows in the Michaelis–Menten example.

	FSP–QTT with sliding windows (s)	Original FSP–QTT (s)
State space reduction	1.6 (0.30%)	N/A
State space expansion via SSA	2.8 (0.54%)	N/A
AMEn solver	516.3 (96.49%)	1305 (95.22%)

Table 3. Breakdown of the runtimes of the main subtasks in the original FSP–QTT algorithm and our FSP–QTT algorithm with sliding windows in the gene toggle example.

	FSP–QTT with sliding windows (s)	Original FSP–QTT (s)
State space reduction	10 (2.37%)	N/A
State space expansion via SSA	59 (13.73%)	N/A
AMEn solver	297 (69.59%)	313 (93.74%)

Table 4. Breakdown of the runtimes of the main subtasks in the original FSP–QTT algorithm and our FSP–QTT algorithm with sliding windows in the p53 example.

	FSP–QTT with sliding windows (s)	Original FSP–QTT (s)
State space reduction	4.5 (2.48%)	N/A
State space expansion via SSA	13 (7.10%)	N/A
AMEn solver	89.2 (48.89%)	194 (76.46%)

for a state $\mathbf{x} \in \mathbf{H}_k$ and multi-index $(i_1, \ldots, i_N) \in [1 : I_1] \times \cdots \times [1 : I_N]$ that satisfies (12).

Let $\mathbf{x} \in \mathbf{H}_{k+1}$ be the state that is parameterized with respect to $\mathbf{H}_{k+1}$ by the multi-index $(j_1, \ldots, j_N) \in [1 : J_1] \times \cdots \times [1 : J_N]$. Property (16) shows that $\mathbf{x} \in \mathbf{H}_{k+1} \bigcap \mathbf{H}_k$ if and only if $1 \leqslant j_s + lb_{k+1}^{(s)} - lb_k^{(s)} \leqslant I_s$, for all $s = 1, \ldots, N$. Hence, assuming $\mathbf{x} \in \mathbf{H}_{k+1} \bigcap \mathbf{H}_k$, and using definition (17) with (15),

$$\begin{aligned}\hat{\mathbf{p}}_{k+1}(\mathbf{x}) = \mathbf{w}(j_1, \ldots, j_N) &= G_1^{(\mathbf{w})}(j_1) \cdot \ldots \cdot G_N^{(\mathbf{w})}(j_N) \\ &= G_1^{(\mathbf{v})}\left(j_1 + lb_{k+1}^{(1)} - lb_k^{(1)}\right) \cdot \ldots \cdot G_N^{(\mathbf{v})} \\ &\quad \times \left(j_N + lb_{k+1}^{(N)} - lb_k^{(N)}\right) \\ &= G_1^{(\mathbf{v})}(i_1) \cdot \ldots \cdot G_N^{(\mathbf{v})}(i_N) = \mathbf{p}_{k+1}(\mathbf{x}).\end{aligned}$$

For the case $\mathbf{x} \in \mathbf{H}_{k+1} \backslash \mathbf{H}_k$, we have $\hat{\mathbf{p}}_{k+1}(\mathbf{x}) = \mathbf{w}(j_1, \ldots, j_N) = 0$ since there exists $s \in \{1, \ldots, N\}$ such that $G_s^{(\mathbf{w})}(j_s)$ is identically zero. □

Algorithm 1 is used to update the characterization of the probability tensor. In the pseudo-code, the dimension and mode sizes of $\mathbf{w}$ are set with the

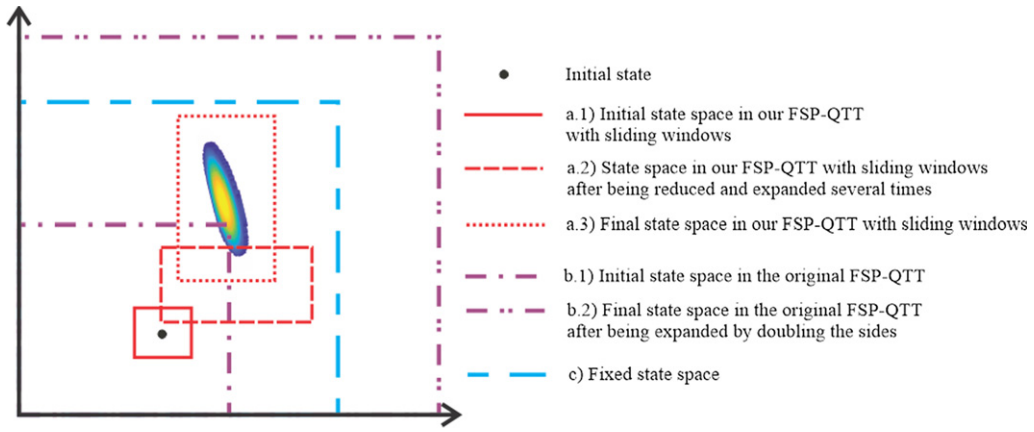

Figure 1. A representation of the state space expansion between three algorithms—(a) adaptive sliding state space in our FSP–QTT with sliding windows. (b) Adaptive doubling of state space in the original FSP–QTT algorithm [35]. (c) Fixed state space that is used in Kazeev *et al* [20] and Dolgov and Khoromskij [7].

Algorithm 2. Adaptive FSP–QTT with sliding windows. All matrices and vectors are implicitly understood as being in the QTT format. All sizes of the window are in the exponential base two.

Input: Number of species N, number of reactions M, stoichiometry vectors $\boldsymbol{\nu}_j$ and separable propensities $\alpha_j(\mathbf{x})$.
The initial state $\mathbf{x}_0$ and final time t_f.
Error tolerances: ϵ_{FSP} for FSP truncation, ϵ_{unif} for uniformization, Tol_{AMEn} for AMEn, $\text{Tol}_{\text{window-reduction}}$ to reduce the windows.
Output: Approximate solution $\mathbf{p}_k$ to the CME at final time t_f. (K is the number of steps.)
1: Initialize window $\mathbf{H}_0$ that contains the initial state by using traditional SSA procedure with a given time step τ_0.
2: Form the initial probability $\mathbf{p}_0$ on $\mathbf{H}_0$.
3: Generate $\mathbf{A}$ corresponding to $\mathbf{H}_0$.
4: Set $t := 0, k := 0$.
5: **while** $t < t_f$ **do**
  6: Compute the tentative $\mathbf{p}_{k+1} = \exp(\tau_k \mathbf{A}_{\mathbf{H}_k})\mathbf{p}_k$.
  7: **if** ($\|\mathbf{p}_{k+1}\|_1$ does not exceed a current tolerance) **then**
    8: Use the SSA on the current window to expand it.
    9: Form the new $\mathbf{A}_{\mathbf{H}_k}$ and update $\mathbf{p}_k$ on the new window.
    10: Return to step 6.
  else
    11: Reduce the current window by removing states with low probabilities.
    12: Expand the reduced state space via SSA runs with boundaries states.
    13: Update $\mathbf{p}_{k+1}$ on the new window.
    14: $t := t + \tau_k$.
    15: $k := k + 1$.
  end
end

Table 5. Runtimes for the three numerical experiments under 10 000 SSA trajectories, the original FSP–QTT algorithm and the FSP–QTT with sliding windows.

	SSA (s)	FSP–QTT with sliding windows (s)	Original FSP–QTT (s)
Michaelis–Menten	558	535	1369
Gene toggle	21252	426	333
p53	3053	182	253

notation

$$\mathbf{w}.d = N \quad \text{and} \quad \mathbf{w}.n = (J_1, \ldots, J_N)$$

and by the proposition, the ranks of $\mathbf{w}$ are those of $\mathbf{v}$, which are set with the notation $\mathbf{w}.r = (r_0, r_1, \ldots, r_N)$. The algorithm also has to determine pointers, set with the notation $\mathbf{w}.ps = (ps_1, ps_2, \ldots, ps_{N+1})$, by letting

$$ps_1 = 1,\ ps_{k+1} = (ps_k + r_{k-1}) \cdot J_k \cdot r_k \text{ for } k = 2, \ldots, N.$$

The vectorized core, referenced through the notation $\mathbf{w}$.core, is of length $ps_{N+1} - 1$, and the part of the vectorized core that corresponds to the sth core of $\mathbf{w}$, $G_s^{(w)}(j_s), j_s = 1 : J_s$, is in the range $(ps_s : ps_{s+1} - 1)$ of $\mathbf{w}$.core and satisfies (17).

The initial window $\mathbf{H}_0$ is generated by a number of SSA trajectories from the initial state. The initial distribution $\mathbf{p}_0$ is defined on $\mathbf{H}_0$ with value 1 at the initial state and 0 elsewhere.

Table 6. Michaelis–Menten reactions and propensities. $[X]$ is the number of copies of the species X.)

	Reaction	Propensities	Rate constant (s^{-1})
1	$S + E \xrightarrow{k_1} ES$	$\alpha_1 = k_1[S][E]$	$k_1 = 1.0$
2	$ES \xrightarrow{k_2} E + S$	$\alpha_2 = k_2[ES]$	$k_2 = 1.0$
3	$ES \xrightarrow{k_3} P + E$	$\alpha_3 = k_3[ES]$	$k_3 = 0.1$

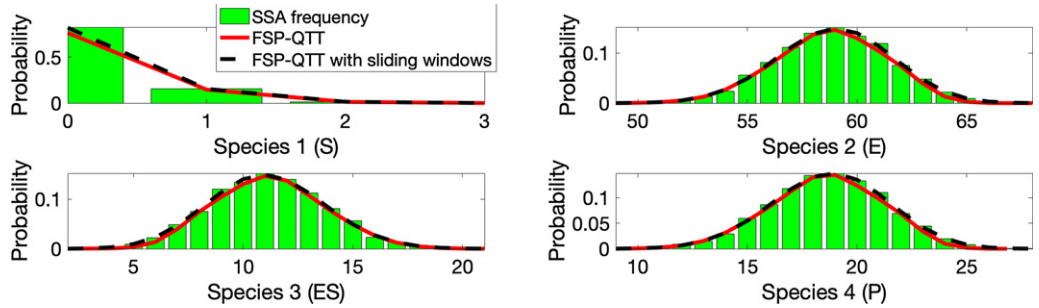

Figure 2. Marginal distributions for the Michaelis–Menten enzyme kinetics model, from the SSA frequency, original FSP–QTT algorithm, and our FSP–QTT algorithm with sliding window.

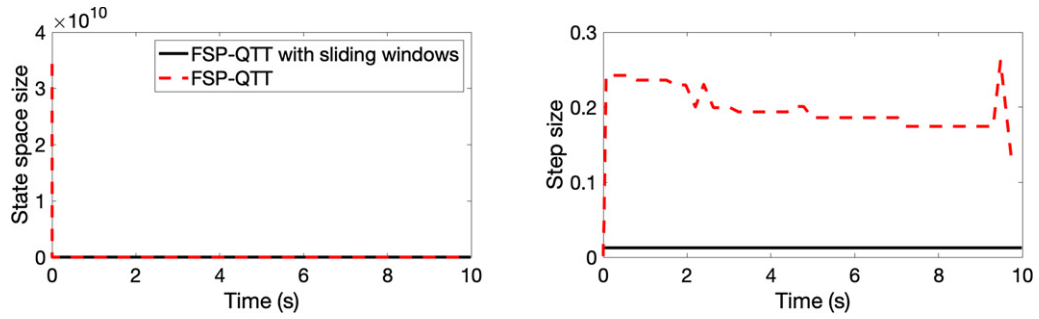

Figure 3. Size of the state space (left) and stepsize (right) of the original FSP–QTT algorithm and our FSP–QTT algorithm with sliding window over time, for the Michaelis–Menten enzyme kinetics model.

Table 7. Reaction channels of the genetic toggle switch example. We set the parameters $d_1 = d_2 = 1.0, s = 0.1, K_1 = 1.0 \times$ scale, $K_2 = 1.0 \times$ scale, $\beta_1 = 4.0 \times$ scale, $\beta_2 = 4.0 \times$ scale, $\alpha_1 = \alpha_2 = 0.2 \times$ scale, $\gamma = 1.0, \eta = 1.0,$ scale $= 100$. ($[X]$ is the number of copies of the species X.)

	Reaction	Propensities
1	$U \to \emptyset$	$\left(d_1 + \frac{s\gamma}{1+s}\right)[U]$
2	$\emptyset \to U$	$\eta\left(\alpha_1 + \frac{\beta_1 K_1^3}{K_1^3 + [V]^3}\right)$
3	$V \to \emptyset$	$d_2[V]$
4	$\emptyset \to V$	$\eta\left(\alpha_2 + \frac{\beta_2 K_2^3}{K_2^3 + [U]^3}\right)$

Now, given the time points $t_0 = 0 < t_1 < \cdots < t_K := t_f$, we shall construct by recurrence the windows of states,

$$\mathbf{H}_k = \left[lb_k^{(1)} : ub_k^{(1)}\right] \times \cdots \times \left[lb_k^{(N)} : ub_k^{(N)}\right],$$

and the distributions $\mathbf{p}_k$ on $\mathbf{H}_k$, for $k = 0, 1, \ldots, K$. Without loss of generality, by expanding the sides of $\mathbf{H}_k$, we can assume that $ub_k^{(s)} - lb_k^{(s)} + 1 = 2^{L_k}$, with $L_k \in \mathbb{N}$, $k = 1, \ldots, N$.

Within each time interval $\left[t_k, t_{k+1}\right]$, we first update the state space via the following stages. The current state space is reduced by eliminating states with probability below a certain threshold (table 1).

The state space is then expanded by running a set of SSA trajectories from t_k to t_{k+1} a fixed number of runs (10 in our numerical examples). Therefore, the share of the SSA on the overall runtime of our algorithm is negligible compared to other

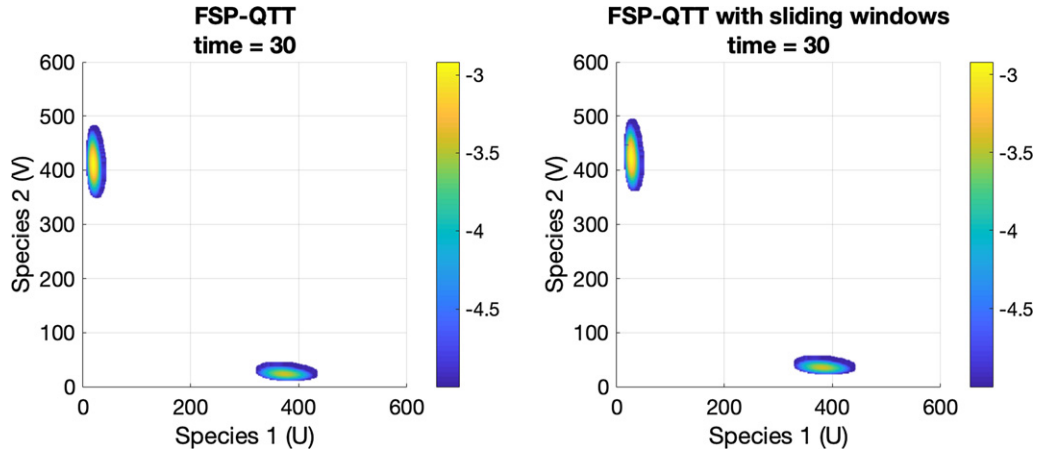

Figure 4. Joint distributions for the gene toggle model from the original FSP–QTT algorithm, and our FSP–QTT algorithm with sliding windows.

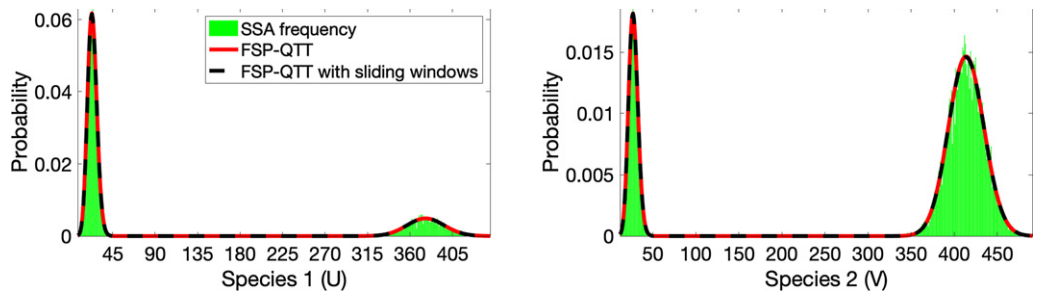

Figure 5. Marginal distributions for the gene toggle model, from the SSA frequency, original FSP–QTT algorithm, and our FSP–QTT algorithm with sliding windows.

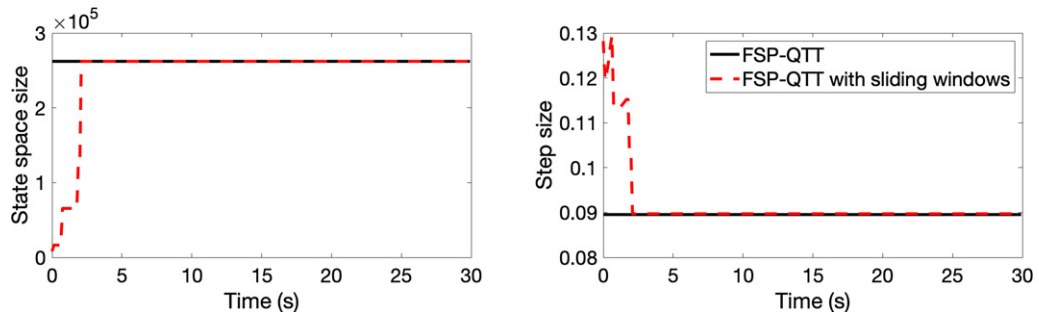

Figure 6. Size of the state space (left) and stepsize (right) of the original FSP–QTT algorithm and our FSP–QTT algorithm with sliding windows over time, for the gene toggle model.

subtasks such as the AMEn solver (see tables 2–4). There is a set of SSA runs seeded with the lower bounds and the maximum species counts across these runs are taken as the new lower bounds. There is another set of SSA runs with the upper bounds and the minimum species counts across these runs are taken as the new upper bounds. Figure 1 illustrates the effectiveness of our state space expansion compared with the original FSP–QTT and other fixed non-adaptive approaches.

Once the state space for the time interval $[t_k, t_{k+1}]$ is determined, the algorithm then constructs the truncated transition matrix $\mathbf{A}_{\mathbf{H}_k}$ and the current probability vector $\mathbf{p}_k$ in QTT format, via algorithm 1. Finally, the algorithm finds the approximate solution $\mathbf{p}_{k+1} := \exp\left(\tau_k \mathbf{A}_{\mathbf{H}_k}\right) \mathbf{p}_k$ at the time point t_{k+1} by applying the inexact uniformization method, discussed previously. Before moving on to the next time step, the algorithm checks if the L^1 error of $\mathbf{p}_{k+1}$ satisfies the tolerance. If the step size is rejected, the state space expansion is repeated until the L^1 tolerance is satisfied. Because the L^1 norm is computed on the whole distribution, its substraction from 1 produces the exact loss of probability mass. This differs from

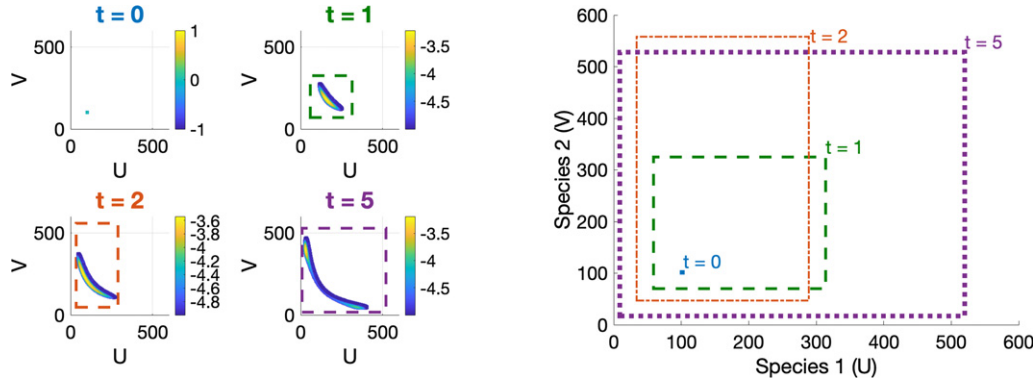

Figure 7. Evolution of the state space in the FSP–QTT with sliding windows when solving the gene toggle example. The right plot superimposes the sliding windows over time.

Table 8. Stochastic model of p53 regulation. ($[X]$ is the number of copies of the species X.)

	Reaction	Propensities	Parameters
1	$\emptyset \to \text{p53}$	k_p	$k_p = 0.5$
2	$\text{p53} \to \emptyset$	$d_p[\text{p53}] + k_1[\text{p53}][\text{MDM2}_{\text{nuc}}]$	$d_p = 1.925 \times 10^{-5}$
			$k_1 = 9.963 \times 10^{-6}$
3	$\emptyset \to \text{RNA}_{\text{nuc}}$	$k_m + k_2 \frac{[\text{p53}]^{1.8}}{k_D^{1.8}+[\text{p53}]^{1.8}}$	$k_m = 1.5 \times 10^{-3}$
			$k_2 = 1.5 \times 10^{-2}$
			$k_D = 740$
4	$\text{RNA}_{\text{nuc}} \to \text{RNA}_{\text{cyt}}$	$k_0[\text{RNA}_{\text{nuc}}]$	$k_0 = 8.0 \times 10^{-4}$
5	$\text{RNA}_{\text{cyt}} \to \emptyset$	$d_{rc}[\text{RNA}_{\text{cyt}}]$	$d_{rc} = 1.444 \times 10^{-4}$
6	$\text{RNA}_{\text{cyt}} \to \text{RNA}_{\text{cyt}} + \text{MDM2}_{\text{cyt}}$	$k_T[\text{RNA}_{\text{cyt}}]$	$k_T = 1.66 \times 10^{-2}$
7	$\text{MDM2}_{\text{cyt}} \to \text{MDM2}_{\text{nuc}}$	$k_i[\text{MDM2}_{\text{cyt}}]$	$k_i = 9.0 \times 10^{-4}$
8	$\text{MDM2}_{\text{nuc}} + \text{MDM2}_{\text{nuc}} \to \text{MDM2}_{\text{nuc}}$	$d_{mn}[\text{MDM2}_{\text{nuc}}]([\text{MDM2}_{\text{nuc}}] - 1)$	$d_{mn} = 1.66 \times 10^{-7}$
9	$\text{MDM2}_{\text{nuc}} + \text{ARF} \to \text{MDM2}_{\text{nuc}}.\text{ARF}$	$k_3[\text{MDM2}_{\text{nuc}}][\text{ARF}]$	$k_3 = 9.963 \times 10^{-6}$
10	$\emptyset \to \text{ARF}$	k_a	$k_a = 0.5$
11	$\text{ARF} \to \emptyset$	$d_a[\text{ARF}]$	$d_a = 3.209 \times 10^{-5}$

the original FSP–QTT algorithm [35], where only the marginal probabilities are checked.

Algorithm 2 is a pseudo-code of the overall algorithm.

4. Numerical comparisions

The computing environment is MATLAB 2018b on a Macbook pro Mid 2014 with 8 GB of RAM and 2.8 GHz dual-core i5 CPU. We use the TT-Toolbox version 2.2.[1] We specifically use the MEX-Fortran implementation of AMEn in the file `fort_amen_solve.f` for fast executions of the tensor linear system solves. The safety parameter in uniformization is chosen to be 100 for the Michaelis–Menten and gene toggle example and 10 for p53 example.

For each biological model, we compare the marginal distributions of the final probability vectors from the original FSP–QTT algorithm and our FSP–QTT with sliding windows, against the frequency distribution from 10 000 SSA trajectories. We also look at the history of the size of the state space and step size of each algorithm at every time point. The runtime for each biological problem and algorithm is listed in table 5.

[1] https://github.com/oseledets/TT-Toolbox/blob/master/quick_start.pdf .

4.1. Michaelis–Menten enzyme kinetics

The reaction rates and propensities for the Michaelis–Menten enzyme kinetics [16] are presented in table 6. The initial state is $[S, E, ES, P] = [0, 0, 30, 70]$ and the final time is $t_f = 10$. The initial state space guess for the original FSP–QTT algorithm is $[2^5, 2^8, 2^5, 2^5]$.

As can be seen in figure 2, the probability distributions from both the FSP–QTT algorithm and the FSP–QTT with sliding windows agree with each other and the SSA frequency. However, the state space size of the FSP–QTT algorithm with sliding windows is consistently smaller than that of the original FSP–QTT algorithm (figure 3-left). This results in two advantages for our algorithm. First, the step-size in the FSP–QTT depends on the largest entry on the diagonal of the CME matrix [as discussed after equation (11)]. Hence smaller CME matrices can potentially lead to larger step-sizes (figure 3-right). Our algorithm can therefore finish within fewer iterations. Second, smaller CME matrices result in smaller AMEn problems, decreasing the runtime to 39% of the original FSP–QTT algorithm (table 5).

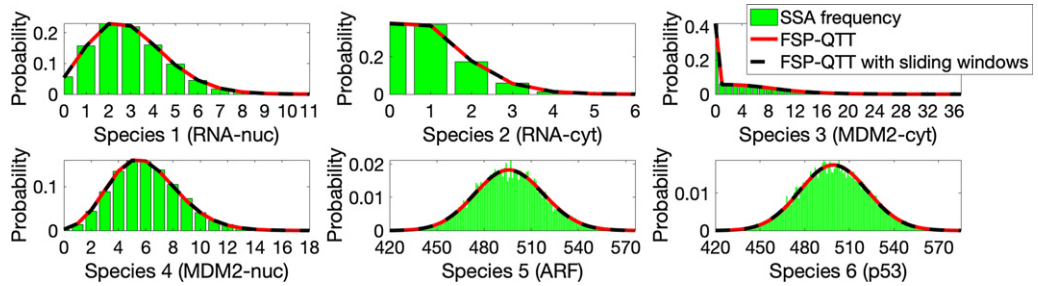

Figure 8. Marginal distributions for the p53 model, from the SSA frequency, original FSP–QTT algorithm, and our FSP–QTT algorithm with sliding windows.

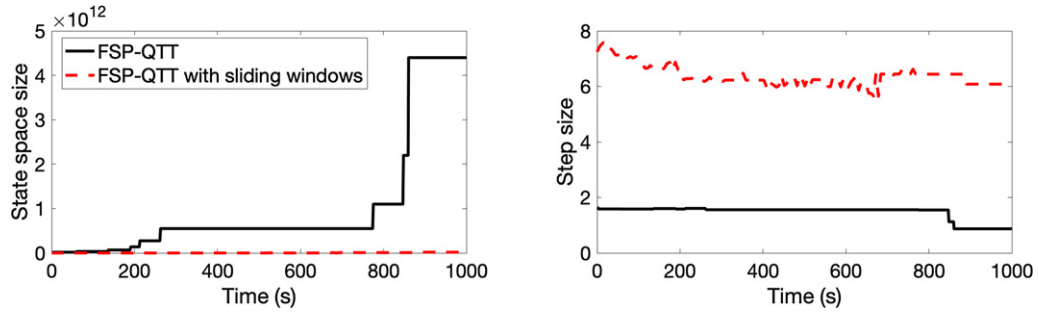

Figure 9. Size of the state space (left) and stepsize (right) of the original FSP–QTT algorithm and our FSP–QTT algorithm with sliding window over time, for the p53 model.

4.2. Gene toggle

The reaction rates and propensities for the gene toggle example [11] are presented in table 7. The initial state is $[U, V] = [100, 100]$ and the final time is $t_f = 30$. The initial state space for the original FSP–QTT algorithm is $[2^9, 2^9]$.

Figure 4 demonstrates the similarity between the joint distributions of the FSP–QTT with sliding windows and the original FSP–QTT. The marginal distributions produced by both of the FSP–QTT algorithms also agree with 10 000 SSA trajectories (figure 5). The FSP–QTT with sliding windows is 60 times faster than the SSA (table 5).

Figure 6 also shows that the size of the state space of the FSP–QTT algorithm with sliding windows is much smaller than that of the original FSP–QTT during the first four seconds (figure 6-left). This leads to larger step-sizes for the FSP–QTT with sliding windows during this time frame (figure 6-right). However, the size of the state space and the step size in both FSP–QTT algorithms completely match up as time progresses. The evolution of the state space is also illustrated in figure 7. Both algorithms end up using hyper rectangles that are big enough to account for the bimodality of the gene toggle. We envision further extending the flexibility of our algorithm to handle multiple, smaller disjoint regions to improve its efficiency.

4.3. p53

The reaction rates and propensities for p53 [23] are presented in table 8.

The initial state is

$$[\mathrm{RNA}_{\mathrm{nuc}}, \mathrm{RNA}_{\mathrm{cyt}}, \mathrm{MDM2}_{\mathrm{cyt}}, \mathrm{MDM2}_{\mathrm{nuc}}, \mathrm{ARF}, \mathrm{p53}] = [0, 0, 0, 100, 100, 100]$$

and the final time is $t_f = 1000$. The initial state space for the original FSP–QTT algorithm is $[2^2, 2^2, 2^2, 2^9, 2^9, 2^9]$.

Comparison of the marginal distributions against the SSA frequency shows that both of the FSP–QTT algorithms produce accurate results (figure 8). The FSP–QTT with sliding windows is 17 times faster than the SSA, and 27% faster than the original FSP–QTT (table 5). This is because the algorithm keeps the state space at the minimum size while retaining accuracy in the probability distribution. For instance, as time increases, species p53 increases in copy number. At the end, its copy number is contained within the range 420–584. The FSP–QTT with sliding windows finds the smallest window of exponential base two that covers this range, while the original FSP–QTT algorithm finds the smallest window of exponential base two that covers the range 0–580. The same situation happens with species ARF (figure 8). The state space size for the FSP–QTT with sliding windows is

therefore consistently smaller (figure 9-left), resulting in larger stepsizes (figure 9-right) and faster AMEn executions.

5. Conclusion

Efficient algorithms for solving the CME directly remain of importance. Many parameter fitting procedures require the computation of the transient probability distribution of a model under various parameter sets. There are parameter inference techniques based on maximum log-likelihood [3, 10, 17, 29, 30, 34, 37] that fit the entire distribution, instead of statistics such as means or variance [26]. For these methods, it is therefore important to compute trustworthy probability distributions that not only retain the true solutions' statistics but also their shapes. Building these distributions would require many SSA runs, and particularly for biological models with small species counts, solving the CME directly using FSP-based methods such as ours can be efficient. Moreover, for models displaying bimodality such as the gene toggle model, a large number of SSA trajectories are required to compute the probabilities especially to detect rare states [33]. For these problems, direct CME solvers can be a time-saving alternate [6]. Direct computation of the solution to the CME also has applications in uncertainty quantification, sensitivity analysis, and experimental design [14, 19, 22, 38].

Current developments using the tensor format promise to offer an approach to solve the high-dimensional CME problems, which have remained difficult and time-consuming due to the 'curse of dimensionality'. Our work has furthered the work in [35], and has explored the use of sliding windows that can slide, shrink or expand to keep the state space at a reduced size while retaining accuracy in the resulting probability vector. We updated our hyper rectangles adaptively at each time step by first excluding states with small probability from the previous time step, then adding states from some SSA runs. Numerical experiments with biological models of varying sizes and difficulties show that our algorithm decreased the total runtime from the original FSP–QTT algorithm.

As shown in tables 2–4, the computational bottleneck of our algorithm remains to be the AMEn solver for computing the probability distribution. There are several approaches to optimize the algorithm in future work. First, we can explore other techniques of solving for the probability distribution in QTT format, such as those in [7, 20, 28]. Second, for multimodal problems such as the gene toggle, the state space can be divided into multiple non-overlapping hyper rectangles, each following one mode of the distribution. The AMEn algorithm or other solvers could then be implemented in parallel across those disjoint hyper rectangles.

In its current form, our algorithm cannot be employed to solve the CME for biological problems where the propensities are time-dependent. Recall that for these models, the Gillespie algorithm also had to be modified [1, 36]. For direct FSP-based CME solvers, a prominent approach is to utilize the Magnus expansion, which evaluates the transition matrix as a time-dependent operator at each time step [5]. Investigating this technique using the current QTT format is of future interest.

Acknowledgments

The authors thank the anonymous referees for their comments.

ORCID iDs

Trang Dinh https://orcid.org/0000-0002-4202-7143
Roger B Sidje https://orcid.org/0000-0001-5353-3642

References

[1] Anderson D F 2007 A modified next reaction method for simulating chemical systems with time dependent propensities and delays *J. Chem. Phys.* **127** 214107
[2] Burrage K, Hegland M, MacNamara S and Sidje R B 2006 A Krylov-based finite state projection algorithm for solving the chemical master equation arising in the discrete modelling of biological systems *150th Markov Anniversary Meeting* ed A N Langville and W J Stewart (Charleston, SC: Boson Books) pp 21–38
[3] Daigle B J, Roh M K, Petzold L R and Niemi J 2012 Accelerated maximum likelihood parameter estimation for stochastic biochemical systems *BMC Bioinform.* **13** 68
[4] Dinh K N and Sidje R B 2016 Understanding the finite state projection and related methods for solving the chemical master equation *Phys. Biol.* **13** 035003
[5] Dinh K N and Sidje R B 2017 An adaptive Magnus expansion method for solving the chemical master equation with time-dependent propensities *J. Coupled Syst. Multiscale Dyn.* **5** 119–31
[6] Dinh K N and Sidje R B 2017 An application of the Krylov-FSP–SSA method to parameter fitting with maximum likelihood *Phys. Biol.* **14** 065001
[7] Dolgov S and Khoromskij B 2015 Simultaneous state-time approximation of the chemical master equation using tensor product formats *Numer. Linear Algebr. Appl.* **22** 197–219
[8] Dolgov S V and Savostyanov D 2014 Alternating minimal energy methods for linear systems in higher dimensions *SIAM J. Sci. Comput.* **36** A2248–71
[9] Drawert B, Lawson M J, Petzold L and Khammash M 2010 The diffusive finite state projection algorithm for efficient simulation of the stochastic reaction–diffusion master equation *J. Chem. Phys.* **132** 074101
[10] Fox Z, Neuert G and Munsky B 2016 Finite state projection based bounds to compare chemical master equation models using single-cell data *J. Chem. Phys.* **145** 074101
[11] Gardner T, Cantor C and Collins J 2000 Construction of a genetic toggle switch in *Escherichia coli Nature* **403** 339–42
[12] Gillespie D T 1976 A general method for numerically simulating the stochastic time evolution of coupled chemical reactions *J. Comput. Phys.* **22** 403–34

[13] Gillespie D T 1992 A rigorous derivation of the chemical master equation *Physica* A **188** 404–25
[14] Goutsias J and Jenkinson G 2013 Markovian dynamics on complex reaction networks *Phys. Rep.* **529** 199–264
[15] Hegland M and Garcke J 2011 On the numerical solution of the chemical master equation with sums of rank one tensors *ANZIAM J.* **52** 628–43
[16] Higham D J 2008 Modeling and simulating chemical reactions *SIAM Rev.* **50** 347–68
[17] Horvath A and Manini D 2008 Parameter estimation of kinetic rates in stochastic reaction networks by the EM method *Proceedings of the International Conference on Biomedical Engineering and Informatics* vol 1 pp 713–7
[18] Jahnke T and Huisinga W 2007 Solving the chemical master equation for monomolecular reaction systems analytically *J. Math. Biol.* **54** 1–26
[19] Jimenez M N, Le Maître O P and Knio O M 2016 Global sensitivity analysis in stochastic simulators of uncertain reaction networks *J. Chem. Phys.* **145** 244106
[20] Kazeev V, Khammash M, Nip M and Schwab C 2014 Direct solution of the chemical master equation using quantized tensor trains *PLoS Comput. Biol.* **10** e1003359
[21] Kazeev V, Reichmann O and Schwab C 2012 *Hp-Dg-Qtt Solution of High-Dimensional Degenerate Diffusion Equations Research Report 11* (ETH Zurich)
[22] Komorowski M, Costa M J, Rand D A and Stumpf M P H 2011 Sensitivity, robustness, and identifiability in stochastic chemical kinetics models *Proc. Natl Acad. Sci.* **108** 8645–50
[23] Leenders G B and Tuszynski J A 2013 Stochastic and deterministic models of cellular p53 regulation *Frontiers Oncol.* **3** 64
[24] Munsky B and Khammash M 2006 The finite state projection algorithm for the solution of the chemical master equation *J. Chem. Phys.* **124** 044104
[25] Munsky B and Khammash M 2007 A multiple time interval finite state projection algorithm for the solution to the chemical master equation *J. Comput. Phys* **226** 818–35
[26] Munsky B and Khammash M 2010 Identification from stochastic cell-to-cell variation: a genetic switch case study *IET Syst. Biol.* **4** 356–66
[27] Neubrander M and Roger B S 2020 Stochastic automata networks and tensors with application to chemical kinetics (SIAM Undergraduate Research Online) **13** 7
[28] Oseledets I V 2011 Tensor-train decomposition *SIAM J. Sci. Comput.* **33** 2295–317
[29] Poovathingal S and Gunawan R 2010 Global parameter estimation methods for stochastic biochemical systems *BMC Bioinform.* **11** 414
[30] Reinker S, Altman R and Timmer J 2006 Parameter estimation in stochastic biochemical reactions *Syst. Biol.* **153** 168–78
[31] Schnoerr D, Guido S and Grima R 2017 Approximation and inference methods for stochastic biochemical kinetics-a tutorial review *J. Phys. A: Math. Theor.* **50** 093001
[32] Sidje R B, Burrage K and MacNamara S 2007 Inexact uniformization method for computing transient distributions of Markov chains *SIAM J. Sci. Comput.* **29** 2562–80
[33] Sidje R B and Vo H D 2015 Solving the chemical master equation by a fast adaptive finite state projection based on the stochastic simulation algorithm *Math. Biosci.* **269** 10–6
[34] Tian T, Xu S, Gao J and Burrage K 2007 Simulated maximum likelihood method for estimating kinetic rates in gene expression *Bioinformatics* **23** 84–91
[35] Vo H D and Sidje R B 2017 An adaptive solution to the chemical master equation using tensors *J. Chem. Phys.* **147** 044102
[36] Voliotis M, Thomas P, Grima R and Bowsher C G 2016 Stochastic simulation of biomolecular networks in dynamic environments *PLoS Comput. Biol.* **12** e1004923
[37] Wang Y, Christley S, Mjolsness E and Xie X 2010 Parameter inference for discretely observed stochastic kinetic models using stochastic gradient descent *BMC Syst. Biol.* **4** 99
[38] Weber L, Raymond W and Munsky B 2018 Identification of gene regulation models from single-cell data *Phys. Biol.* **15** 055001
[39] Wolf V 2007 Modelling of biochemical reactions by stochastic automata networks *Electron. Notes Theor. Comput. Sci.* **171** 197–208
[40] Wolf V, Goel R, Mateescu M and Henzinger T A 2010 Solving the chemical master equation using sliding windows *BMC Syst. Biol.* **4** 42