The finite state projection algorithm for the solution of the chemical master equation

Brian Munsky[a] and Mustafa Khammash[b]
Mechanical and Environmental Engineering, University of California-Santa Barbara, Santa Barbara, California 93106

(Received 28 March 2005; accepted 7 November 2005; published online 25 January 2006)

This article introduces the finite state projection (FSP) method for use in the stochastic analysis of chemically reacting systems. One can describe the chemical populations of such systems with probability density vectors that evolve according to a set of linear ordinary differential equations known as the chemical master equation (CME). Unlike Monte Carlo methods such as the stochastic simulation algorithm (SSA) or τ leaping, the FSP directly solves or approximates the solution of the CME. If the CME describes a system that has a finite number of distinct population vectors, the FSP method provides an exact analytical solution. When an infinite or extremely large number of population variations is possible, the state space can be truncated, and the FSP method provides a certificate of accuracy for how closely the truncated space approximation matches the true solution. The proposed FSP algorithm systematically increases the projection space in order to meet prespecified tolerance in the total probability density error. For any system in which a sufficiently accurate FSP exists, the FSP algorithm is shown to converge in a finite number of steps. The FSP is utilized to solve two examples taken from the field of systems biology, and comparisons are made between the FSP, the SSA, and τ leaping algorithms. In both examples, the FSP outperforms the SSA in terms of accuracy as well as computational efficiency. Furthermore, due to very small molecular counts in these particular examples, the FSP also performs far more effectively than τ leaping methods. © *2006 American Institute of Physics*. [DOI: 10.1063/1.2145882]

I. INTRODUCTION

It is of great interest to the systems biology community to develop an efficient means of analyzing chemical systems in which small numbers of intracellular molecules randomly interact. In many situations, especially in the case of cell fate decisions, single molecular events may dramatically affect every subsequent process. In such cases, deterministic models fail to capture the inherent randomness of the biological processes, and stochastic models are necessary. Previous studies have shown that if a chemically reacting system is well mixed and has a fixed volume and fixed temperature, then that system is a Markov process. In such a case, the evolution of the probability density vector (pdv) of the system's chemical population can be described by the chemical master equation (CME).[1,2] In most cases the CME has not been directly solved, and analyses are often conducted using Monte Carlo algorithms such as Gillespie's stochastic stimulation algorithm (SSA).[3,4] Although the SSA can produce detailed realizations for stochastically evolving chemical systems, the method can become very computationally expensive when the system undergoes enormous numbers of individual reactions. In these cases it is often necessary to sacrifice some of the precision of the SSA for faster, yet approximate methods.

Approximations to the SSA can be grouped into two categories: time-leaping methods and system-partitioning methods. The time-leaping methods rely on the assumption that many reaction events will occur in a period of time without significantly changing the reaction propensity functions. This category includes the explicit and implicit τ leaping algorithms,[5–7] which use Poisson random variables to simulate how many times each reaction channel fires in a given time interval. A major difficulty for the τ leaping method results when too many critical reactions are included in a single leap such that the propensity functions change excessively and/or some molecular populations become negative. In Poisson τ leaping, these concerns have been addressed by adaptively restricting the size of each individual τ leap.[7,8] In another approach, Tian and Burrage proposed a τ leaping strategy based upon binomial random variables rather than unbounded Poisson random variables.[9] While these recent versions of τ leaping are more robust than their predecessors, their computational efficiency still remains severely compromised in circumstances where very small populations of interacting molecules result in fast, dramatic changes in propensity functions.

The second approach to speeding up the SSA involves separating the system into slow and fast partitions. In these approaches one analytically or numerically approximates the dynamics of the fast partition and then stochastically simulates the slow partition. In one of the first such methods, Rao and Arkin applied a quasi-steady-state assumption to the fast reactions and treated the remaining slow reactions as stochastic events.[10] In a similar study, Haseltine and Rawlings also separated the system into slow and fast reactions but

[a]Electronic mail: brianem@engineering.ucsb.edu
[b]Electronic mail: khammash@engineering.ucsb.edu

approximated the fast reactions deterministically or as Langevin equations.[11] Most recently, Cao *et al.* have partitioned the system according to fast and slow species in order to develop the slow-scale SSA (ssSSA).[12]

For any Monte Carlo method, including all of those above, computing a statistical description of the system's dynamics, such as the probability density, mean, or variance, requires a large number of individual process realizations. For some systems, important biological traits may be very rare, and extremely large numbers of simulations may be required to get sufficiently precise statistics. For example, the pyelonephritis-associated pili (Pap) epigenetic switch in *E. coli* has an OFF to ON switch rate on the order of 10^{-4} per cell per generation.[13] In order to capture this switch rate with a relative accuracy of 1%, one must run enough simulations to get a resolution of better then 10^{-6}. It may be shown that one cannot attain this resolution before completing more than 1×10^6 Monte Carlo simulations. In order to meet these stringent requirements on precision for systems such as the Pap switch, we now propose an entirely different approach. This method, known as the finite state projection (FSP) algorithm, differs from the SSA and its derivatives in that the FSP algorithm provides a direct solution or approximation of the CME without generating Monte Carlo simulations. Additionally, unlike Monte Carlo methods, the FSP provides a guarantee as to its own accuracy.

The remainder of this paper is organized into four sections. The next section establishes the basic mathematical theory needed for the FSP algorithm. The third section introduces the steps of the FSP algorithm and illustrates how to use the algorithm to describe how a chemical system evolves according to the CME. The fourth section illustrates the implementation of the FSP algorithm through two examples from ongoing computational research concerning the Pap epigenetic switch. The final section concludes with a discussion of the capabilities of the proposed algorithm especially in the context of its usefulness in the field of systems biology.

II. MATHEMATICAL BACKGROUND

Consider a well-mixed, fixed-temperature, and fixed-volume system of N distinct reacting chemical species. Define $p(\mathbf{x};t)$ to be the probability that the chemical system will have a molecular population vector $\mathbf{x}$ at time t, where $\mathbf{x}\in\mathbb{N}^N$ is a vector of integers representing a specific population of each of the N molecular species. Suppose that the system's chemical population can change through M different reaction channels and let $a_\mu(\mathbf{x})dt$ and ν_μ be the nonnegative propensity function and the stoichiometric transition vector, respectively, of each of the $\mu=(1,2,\ldots,M)$ reactions. Given that we know the probability density vector at t, the probability that the system will be in the state $\mathbf{x}$ at time $t+dt$ is equal to the sum of (i) the probability that the system begins in the state $\mathbf{x}$ at t and remains there until $t+dt$ and (ii) the probability that the system is in a different state at time t and will transition to $\mathbf{x}$ in the considered time step dt. This probability can be written as

$$p(\mathbf{x};t+dt)=p(\mathbf{x};t)\left(1-\sum_{\mu=1}^{M}a_\mu(\mathbf{x})dt\right)+\sum_{\mu=1}^{M}p(\mathbf{x}-\nu_\mu;t)a_\mu(\mathbf{x}-\nu_\mu)dt. \tag{2.1}$$

From Eq. (2.1) it is relatively simple to derive the differential equation known as the CME:[3]

$$\dot{p}(\mathbf{x};t)=-p(\mathbf{x};t)\sum_{\mu=1}^{M}a_\mu(\mathbf{x})+\sum_{\mu=1}^{M}p(\mathbf{x}-\nu_\mu;t)a_\mu(\mathbf{x}-\nu_\mu). \tag{2.2}$$

By combining all possible reactions that begin or end with the state $\mathbf{x}$, the time derivative of the probability density of state $\mathbf{x}$ can be written in vector form as

$$\dot{p}(\mathbf{x};t)=\left[-\sum_{\mu=1}^{M}a_\mu(\mathbf{x})\ a_1(\mathbf{x}-\nu_1)\ a_2(\mathbf{x}-\nu_2)\ \cdots\ a_M(\mathbf{x}-\nu_M)\right]\begin{bmatrix}p(\mathbf{x};t)\\ p((\mathbf{x}-\nu_1);t)\\ p((\mathbf{x}-\nu_2);t)\\ \vdots\\ p((\mathbf{x}-\upsilon_M);t)\end{bmatrix}. \tag{2.3}$$

We will *a priori* fix a sequence $\mathbf{x}_1,\mathbf{x}_2,\ldots$ of elements in $\mathbb{N}^N$ and define $\mathbf{X}:=[\mathbf{x}_1,\mathbf{x}_2,\ldots]^T$. The particular sequence $\mathbf{x}_1,\mathbf{x}_2,\ldots$ may be chosen to visit every element of the entire space $\mathbb{N}^N$. In this case, the choice of $\mathbf{X}$ corresponds to a particular enumeration of the space $\mathbb{N}^N$. Once $\mathbf{X}$ is selected, we can write Eq. (2.3) as a single linear expression:

$$\dot{\mathbf{P}}(\mathbf{X};t)=\mathbf{A}\cdot\mathbf{P}(\mathbf{X};t), \tag{2.4}$$

where $\mathbf{P}(\mathbf{X};t):=[p(\mathbf{x}_1,t),p(\mathbf{x}_2,t),\ldots]^T$ is the complete probability density state vector at time t and $\mathbf{A}$ is the *state reaction matrix*. The columns and rows of $\mathbf{A}$ are uniquely defined by the system's stoichiometry and the choice of $\mathbf{X}$. Beginning at any state $\mathbf{x}_i$, there can be a maximum of $\mathbf{M}$ possible reactions; each reaction leads to a different state: $\mathbf{x}_j=\mathbf{x}_i+\nu_\mu$.

The state reaction matrix contains information regarding every reaction, each weighted by the corresponding propensity function, and the elements of $\mathbf{A}$ are given as

$$\mathbf{A}_{ij} = \begin{Bmatrix} -\sum_{\mu=1}^{M} a_\mu(\mathbf{x}_i) & \text{for } i=j \\ a_\mu(\mathbf{x}_i) & \text{for all } j \text{ such that } \mathbf{x}_j = \mathbf{x}_i + \nu_\mu \\ 0 & \text{Otherwise} \end{Bmatrix}. \tag{2.5}$$

A has the properties that it is independent of t; all of its diagonal elements are nonpositive, all its off-diagonal elements are non-negative, and all its columns sum to exactly zero. The solution to the linear ordinary differential equation (ODE) beginning at $t=0$ and ending at $t=t_f$ in Eq. (2.4) is the expression

$$\mathbf{P}(\mathbf{X};t_f) = \Phi(0,t_f)\cdot\mathbf{P}(\mathbf{X};0). \tag{2.6}$$

In the case where there are only a finite number of reachable states, the operator $\Phi(0,t_f)$ is the exponential of $\mathbf{A}t_f$, and one can compute the solution: $\mathbf{P}(\mathbf{X};t_f)=\exp(\mathbf{A}t_f)\mathbf{P}(\mathbf{X};0)$.

For the examples in this study, we are interested only in the probability density at the final time t_f. This information is simply obtained by computing the exponential of $\mathbf{A}t_f$ directly and multiplying the resulting matrix by the initial probability density vector. Moler and Van Loan provide many methods for performing this computation in their celebrated 1978 paper "Nineteen Dubious Ways to Compute the Exponential of a Matrix"[14] and its revisited edition of 2003.[15] For our examples, all matrix exponentials are computed using the *expm()* function in MathWorks MATLAB. This built-in routine is based upon a scaling and squaring algorithm with a Pade approximation. As a powerful alternative to MATLAB, Roger Sidje's EXPOKIT provides a powerful matrix exponential package for programmers in *c*, *c*++, and FORTRAN. In some situations, one may wish to obtain the probability density at many intermediate times as well as the final time. For this it may be more efficient not to directly calculate the matrix exponential, but instead use a numerical stiff ODE solver such as one of MATLAB's *ode*15*s* or *ode*23*s*.

In practice there may be many simple chemical systems for which the exponential representation will produce an exact solution (see the example in Sec. IV A 1). Such cases include any system in which the number of molecules in each species is bounded through considerations such as the conservation of mass. However, when **A** is infinite dimensional or extremely large, the corresponding analytic solution is unclear or vastly difficult to compute. Even in these cases, however, one may devise a systematic means of approximating the full system using finite-dimensional subsystems. This systematic truncation approach is referred to as the finite state projection method.

The presentation of the FSP method first requires the introduction of some convenient notations. Let $J=\{j_1,j_2,j_3,\ldots\}$ denote an ordered index set corresponding to a specific set of states, $\{x_{j1},x_{j2},x_{j3},\ldots\}$. For any matrix, let $\mathbf{A}_{IJ}$ denote a submatrix of **A** such that the rows have been chosen and ordered according to I and the columns have been chosen and ordered according to J. For example, if **A** is given by

$$\mathbf{A} = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix},$$

and I and J are defined as $\{3,1,2\}$ and $\{1,3\}$, respectively, then the submatrix $\mathbf{A}_{IJ}$ is given as

$$\mathbf{A}_{IJ} = \begin{bmatrix} 7 & 9 \\ 1 & 3 \\ 4 & 6 \end{bmatrix}.$$

Similarly let $\mathbf{A}_J$ denote the principle submatrix of **A**, in which both rows and columns have been chosen and ordered according to J. We will use the notation J' to denote the complement of the set J on the entire set **X**. Define the sequence $\{J_k\}$ as a sequence of nested sets such that $J_1\subseteq J_2\subseteq J_3\subseteq\cdots$. In addition to the set notation, the vector **1** will be used to denote a column of all ones such that for any vector **v**, the product $\mathbf{1}^T\mathbf{v}$ is the sum of the elements in **v**. With this notation we can now state and prove the following two theorems.

Theorem 2.1. If $\mathbf{A}\in\mathbb{R}^{n\times n}$ has no negative off-diagonal elements, then for any pair of index sets, $J_2\supseteq J_1$,

$$[\exp(\mathbf{A}_{J_2})]_{J_1} \geqslant \exp(\mathbf{A}_{J_1}) \geqslant \mathbf{0}. \tag{2.7}$$

Proof. Because $J_2\supseteq J_1$, without loss of generality, $\mathbf{A}_{J_2}$ can be written in terms of $\mathbf{A}_{J_1}$ as

$$\mathbf{A}_{J_2} = \begin{bmatrix} \mathbf{A}_{J_1} & B \\ C & D \end{bmatrix}.$$

If one lets $\beta=\min\{0,\mathbf{A}_{ii}\}$, where $\beta\leqslant 0$, then one can write $\mathbf{A}_{J_2}=\beta\mathbf{I}+\tilde{\mathbf{A}}_{J_2}$, where $\tilde{\mathbf{A}}_{J_2}$ is a non-negative matrix. Since the matrix $\beta\mathbf{I}$ commutes with $\tilde{\mathbf{A}}_{J_2}$, the matrix exponential of $\mathbf{A}_{J_2}$ can be wrriten as

$$\exp(\mathbf{A}_{J_2}) = \exp(\beta)\exp\begin{bmatrix} \tilde{\mathbf{A}}_{J_1} & B \\ C & \tilde{D} \end{bmatrix}.$$

Expanding the matrix exponential term by term produces

$$\exp(\mathbf{A}_{J_2}) = \exp(\beta)\left(\begin{bmatrix} I & 0 \\ 0 & I \end{bmatrix} + \begin{bmatrix} \tilde{\mathbf{A}}_{J_1} & B \\ C & \tilde{D} \end{bmatrix} + \frac{1}{2}\begin{bmatrix} \tilde{\mathbf{A}}_{J_1}^2 + BC & \tilde{\mathbf{A}}_{J_1}B + B\tilde{D} \\ C\tilde{\mathbf{A}}_{J_1} + \tilde{D}C & CB + \tilde{D}^2 \end{bmatrix} + \cdots\right).$$

Using the property that $\tilde{\mathbf{A}}_{J_2}$ is a non-negative matrix, we can rewrite the exponential as

$$\exp(\mathbf{A}_{J_2}) = \exp(\beta)\begin{bmatrix} \exp(\tilde{\mathbf{A}}_{J_1}) + \tilde{Q}_{11} & \tilde{Q}_{12} \\ \tilde{Q}_{21} & \tilde{Q}_{22} \end{bmatrix}$$

$$= \begin{bmatrix} \exp(\mathbf{A}_{J_1}) + Q_{11} & Q_{12} \\ Q_{21} & Q_{22} \end{bmatrix},$$

where $Q_{ij} = \exp(\beta)\tilde{Q}_{ij}$ are non-negative matrices. Hence,

$$[\exp(\mathbf{A}_{J_2})]_{J_1} \geqslant \begin{bmatrix} \exp(\mathbf{A}_{J_1}) & 0 \\ 0 & 0 \end{bmatrix}_{J_1} = \exp(\mathbf{A}_{J_1}),$$

and the proof is complete. □

As stated above, the full state reaction matrix $\mathbf{A}$ in Eq. (2.4) has no negative off-diagonal terms and therefore satisfies the assumptions of Theorem 2.1. Consider two subsystems of the full system described by Eq. (2.4), which include the states indexed by the sets J_1 and J_2, where $J_2 \supseteq J_1$. Since the probability density vector $\mathbf{P}(\mathbf{X};t)$ is always non-negative, Theorem 2.1 assures that

$$[\exp(\mathbf{A}_{J_2} t_f)]_{J_1} \mathbf{P}(\mathbf{X}_{J_1};0) \geqslant \exp(\mathbf{A}_{J_1} t_f)\mathbf{P}(\mathbf{X}_{J_1};0),$$

where $\mathbf{X}_{J_1}$ is the vector of those elements of $\mathbf{X}$ indexed by J_1. As will be seen later, this result guarantees that as one adds more states to finite truncation of the infinite system, the solution to Eq. (2.4) increases monotonically.

In addition to always being non-negative, the solution of Eq. (2.4) at any time t_f always sums to exactly 1. These properties and the non-negativity of the off-diagonal elements of $\mathbf{A}$ allow one to prove a second theorem and relate the solution of Eq. (2.4) with a finite state description to that with the full state description.

Theorem 2.2. Consider any Markov process in which the probability density state vector evolves according to the linear ODE:

$$\dot{\mathbf{P}}(\mathbf{X};t) = \mathbf{A} \cdot \mathbf{P}(\mathbf{X};t),$$

where $\mathbf{A}$ has no negative off-diagonal entries. Let $\mathbf{A}_J$ be a principle submatrix of $\mathbf{A}$ and let $\mathbf{P}(\mathbf{X}_J;0)$ be a vector of the corresponding elements of $\mathbf{P}(\mathbf{X};0)$.

If for $\epsilon > 0$ and $t_f \geqslant 0$

$$\mathbf{1}^T \exp(\mathbf{A}_J t_f)\mathbf{P}(\mathbf{X}_J;0) \geqslant 1 - \epsilon, \tag{2.8}$$

then

$$\exp(\mathbf{A}_J t_f)\mathbf{P}(\mathbf{X}_J;0) \leqslant \mathbf{P}(\mathbf{X}_J;t_f) \leqslant \exp(\mathbf{A}_J t_f)\mathbf{P}(\mathbf{X}_J;0) + \epsilon\mathbf{1}. \tag{2.9}$$

Proof. We begin by proving the left-hand inequality. The evolution of the full probability density vector is governed by the ODE:

$$\begin{bmatrix} \dot{\mathbf{P}}(\mathbf{X}_J;t) \\ \dot{\mathbf{P}}(\mathbf{X}_{J'};t) \end{bmatrix} = \begin{bmatrix} \mathbf{A}_J & \mathbf{A}_{JJ'} \\ \mathbf{A}_{J'J} & \mathbf{A}_{J'J'} \end{bmatrix} \begin{bmatrix} \mathbf{P}(\mathbf{X}_J;t) \\ \mathbf{P}(\mathbf{X}_{J'};t) \end{bmatrix}, \tag{2.10}$$

where the submatrices $\mathbf{A}_{JJ'}$ and $\mathbf{A}_{J'J}$ are non-negative since $\mathbf{A}$ has no negative off-diagonal terms. The evolution of the probability density of the set of states indexed by J is given by the finite linear ODE:

$$\dot{\mathbf{P}}(\mathbf{X}_J;t) = \mathbf{A}_J \mathbf{P}(\mathbf{X}_J;t) + \mathbf{A}_{JJ'}\mathbf{P}(\mathbf{X}_{J'};t), \tag{2.11}$$

whose solution is given by

$$\mathbf{P}(\mathbf{X}_J;t_f) = \exp(\mathbf{A}_J t_f)\mathbf{P}(\mathbf{X}_J;0) + \int_0^{t_f} \exp(\mathbf{A}_J(t_f - \tau))\mathbf{A}_{JJ'}\mathbf{P}(\mathbf{X}_{J'};\tau)d\tau$$

$$\geqslant \exp(\mathbf{A}_J t_f)\mathbf{P}(\mathbf{X}_J;0),$$

where the last inequality follows from the non-negativity of $\mathbf{A}_{JJ'}$, the non-negativity of the vector of probability densities $\mathbf{P}(\mathbf{X}_{J'};t)$, and the non-negativity of $\exp(\mathbf{A}_J t)$ for $t \geqslant 0$ guaranteed by Theorem 2.1.

The proof of the right-hand inequality uses the fact that the probability density on the infinite space is non-negative and sums to exactly 1, and therefore the probability of the J-indexed set must also be non-negative and sum to no more than 1: $\mathbf{1}^T\mathbf{P}(\mathbf{X}_J;t_f) \leqslant 1$. Using this together with (2.8) gives

$$\mathbf{1}^T \exp(\mathbf{A}_J t_f)\mathbf{P}(\mathbf{X}_J;0) \geqslant 1 - \epsilon$$

$$\geqslant \mathbf{1}^T\mathbf{P}(\mathbf{X}_J;t_f) - \epsilon,$$

We have also shown that $\mathbf{P}(\mathbf{X}_j;t_f) \geqslant \exp(\mathbf{A}_J t_f)\mathbf{P}(\mathbf{X}_J;0)$. Hence, the above inequality holds componentwise, i.e.,

$$\mathbf{P}(\mathbf{X}_J;t_f) \leqslant \exp(\mathbf{A}_J t_f)\mathbf{P}(\mathbf{X}_J;0) + \epsilon\mathbf{1}, \tag{2.12}$$

which completes the proof. □

Theorems 2.1 and 2.2 will hereafter be referred to as the finite state projection theorems. The FSP theorems tell us two very important pieces of information. First, Theorem 2.1 shows that as we increase the size of the finite projection space, the approximation result monotonically increases. Second, Theorem 2.2 guarantees that the approximate solution never exceeds the actual solution and gives us certificate of how close the approximation is to the true solution.

To interpret the underlying intuition of the finite state projection method, it is helpful to represent all possible states of a discrete valued Markov process as nodes on an infinite N-dimensional integer lattice. For chemical systems N could correspond to the number of molecular species, and the nodes correspond to distinct population vectors $\mathbf{x}_i$. As an example, Fig. 1 (top) illustrates a two-dimensional state lattice. We are interested in projecting this infinite lattice onto the subset enclosed by the gray square, which corresponds to some index set J. This projected system is shown in Fig. 1 (bottom), where the truncated system corresponds to the set J

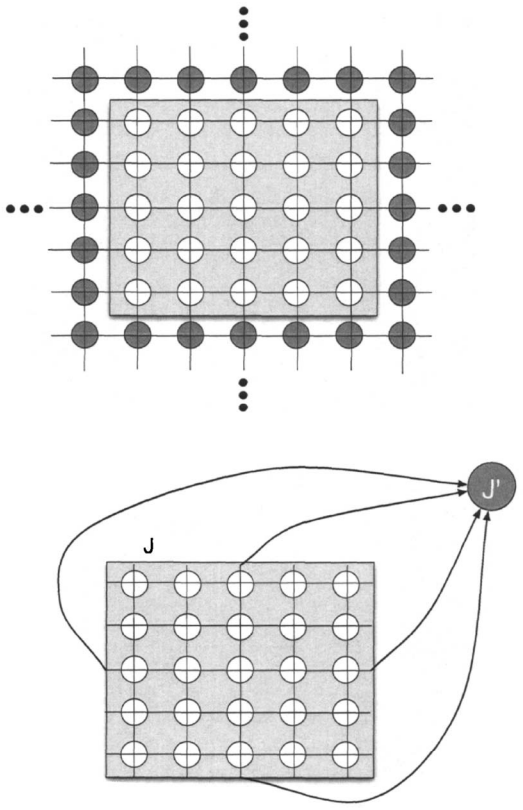

FIG. 1. Conceptual figure for the finite state projection method. (Top) Schematic of an infinite integer lattice representing all possible states in a generic discrete valued Markov process. (Bottom) Projection of the infinite state system to a finite set of states.

composed with its complement J', where J' has been aggregated to a single point. All reactions beginning within the set J are included in the projected space formulation exactly as they are in the original. These include transitions between states within J as well as reactions that begin in J and end in J'. The propensity functions of these reactions completely determine the J-indexed principle submatrix of the state reaction matrix $\mathbf{A}_J$. Since the projected system is finite dimensional, its exact solution at any time can be computed using the matrix exponential function and the truncated initial probability density vector. This FSP solution describes all trajectories fully enclosed in J. Intuitively, Theorem 2.1 shows that as the set J increases, fewer trajectories are lost to $J'$and the probability of remaining in J increases. Theorem 2.2 shows that the probability that the system is currently in J must be at least as large as the probability that the system has been in J for all times $t=0$ to $t=t_f$. In the following section, we use this intuition and the FSP theorems to develop a very useful algorithm with which many interesting chemical reaction problems may be solved.

III. THE FINITE STATE PROJECTION ALGORITHM

The results in the previous section suggest a systematic procedure to to evaluate Markov processes such as those described by the chemical master equation. This algorithm,

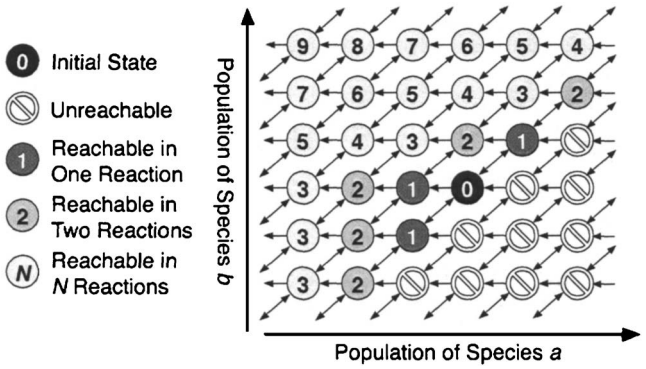

FIG. 2. Schematic of a two-dimensional integer lattice representing the infinite states of a discrete valued Markov process. Each integer valued state vector $[a,b]$ is represented by a circle and the directionality of transitions between states is shown by the connecting arrows.

which we refer to as the finite state projection algorithm, can be stated as follows

The Finite State Projection Algorithm

Step 0.

Define the propensity functions and stoichiometry for all reactions.
Choose the initial probability density vector, $\mathbf{P}(\mathbf{X};0)$.
Choose the final time of interest, t_f.
Specify the total amount of acceptable error, $\epsilon>0$.
Choose an initial finite set of states, $\mathbf{X}_{J_o}$, for the FSP.
Initialize a counter, $i=0$.

Step 1.

Use propensity functions and stoichiometry to form $\mathbf{A}_{J_i}$.
Compute $\Gamma_{J_i}=\mathbf{1}^T \exp(\mathbf{A}_{J_i} t_f)\mathbf{P}(\mathbf{X}_{J_i};0)$.

Step 2.

If $\Gamma_{J_i} \geqslant 1-\epsilon$, Stop.
$\exp(\mathbf{A}_{J_i} t_f)\mathbf{P}(\mathbf{X}_{J_i};0)$ approximates $\mathbf{P}(\mathbf{X}_{J_i};t_f)$ to within a total error of ϵ.

Step 3.

Add more states to find $\mathbf{X}_{J_{i+1}}$.
Increment i and return to Step 1.

In Step 3 of the above algorithm, the choice of how to add new states to the finite projection space is not explicitly stated. While Theorem 2.1 guarantees that adding new states can only improve the accuracy of the approximate solution, it does not explicitly state which additions are most beneficial. In practice there may be many methods of choosing how to add states to the projection, and the efficiency of each method may depend upon the class of problem. In general, the best methods will utilize knowledge of the stoichiometry of the chemical reactions and avoid including unreachable states. The following subsection illustrates one such method which introduces and utilizes a concept of *N-step reachability*.

A. State space expansion through *N*-step reachability

In order to properly introduce the process of expanding the state space through the idea of reachability, we must first introduce some additional concepts. Consider the generic two-dimensional infinite state space lattice shown in Fig. 2. In general, any chemically reacting system can be represented by an N-dimensional integer lattice, where N is the number of reacting species and where every node on the lattice is unique and can be enumerated. In Fig. 2, each circle represents a specific population vector $x^T=[a,b]$, and the initial condition is shaded in black. Reactions are shown with arrows connecting the states. For this specific system, the diagonal-oriented reactions are reversible, or bidirectional, while the horizontal reactions are irreversible.

Let I_k denote the set of all states that can be reached from the initial condition in k or fewer chemical reactions. For instance, in Fig. 2, I_o consists of only the initial condition, which is labeled with the number zero. Similarly, I_1 includes the initial condition and all nodes containing the number 1. In general, I_k contains all states in I_{k-1} combined with all states that can be reached via a single reaction beginning in I_{k-1}. Consider any finite set of states I_R which are reachable from the initial set I_0. It is not difficult to see that there will always exist a finite integer k_R such that $I_k \supseteq I_R$ for all $k \geqslant k_R$. For this method of including sequentially reachable states, the following result guarantees that if a finite state projection exists that satisfies the stopping criterion, then the FSP algorithm will converge in a finite number of steps.

Proposition 3.1. Suppose that there exists a finite set of states indexed by S for which the FSP meets the stopping criterion:

$$\mathbf{1}^T \exp(\mathbf{A}_S t_f)\mathbf{P}(\mathbf{X}_S;0) \geqslant 1-\epsilon. \tag{3.1}$$

Then there exists a number of reactions, m, such that the set of reachable states, I_k also satisfies (3.1) for all $k \geqslant m$.

Proof. The finite set S can be separated into the reachable subset R and the unreachable subset U. Without loss of generality, the state reaction matrix $\mathbf{A}_S$ can be written as

$$\mathbf{A}_S = \begin{bmatrix} \mathbf{A}_R & \mathbf{B} \\ \mathbf{C} & \mathbf{A}_U \end{bmatrix},$$

and the initial condition, which must be contained in the reachable space, can be written as

$$\mathbf{P}(\mathbf{X}_S;0) = \begin{bmatrix} \mathbf{P}(\mathbf{X}_R;0) \\ \mathbf{P}(\mathbf{X}_U;0) \end{bmatrix} = \begin{bmatrix} \mathbf{P}(\mathbf{X}_R;0) \\ 0 \end{bmatrix}.$$

Since the states in U are unreachable from the states in R, the matrix $\mathbf{C}$ is zero. Through series expansion, the exponential of $\mathbf{A}_S t_f$ can be written as

$$\exp(\mathbf{A}_S t_f) = \left(\begin{bmatrix} I & 0 \\ 0 & I \end{bmatrix} + \begin{bmatrix} \mathbf{A}_R & B \\ 0 & \mathbf{A}_U \end{bmatrix} t_f + \frac{1}{2} \begin{bmatrix} \mathbf{A}_R^2 & \mathbf{A}_R B + B\mathbf{A}_U \\ 0 & \mathbf{A}_U^2 \end{bmatrix} t_f^2 + \cdots \right).$$

Combining terms allows one to write the matrix exponential as

$$\exp(\mathbf{A}_p t_f) = \begin{bmatrix} \exp(\mathbf{A} t_f) & Q \\ \mathbf{0} & \exp(\mathbf{A}_U t_f) \end{bmatrix},$$

where Q is a positive matrix. Substituting this expression into Eq. (3.1) gives

$$1-\epsilon \leqslant \mathbf{1}^T \exp(\mathbf{A}_S t_f)\mathbf{P}(\mathbf{X}_S;0) = \mathbf{1}^T \exp(\mathbf{A}_R t_f)\mathbf{P}(\mathbf{X}_R;0). \tag{3.2}$$

Choose m large enough such that $I_m \supseteq R$, then the set indexed by I_k satisfies Eq. (3.1) for all $k \geqslant m$, completing the proof.□

Proposition 3.1 requires that there exists a finite set of states in which the system remains (with probability $1-\epsilon$) for the entire time interval, $t \in (0,t_f)$. If this assumption is satisfied, then the N-step FSP algorithm will produce an acceptable approximation within a finite number of steps. If the population of the system is bounded (i.e., by conservation of mass or volume), then such a set will obviously exist. However, one can construct some pathological examples, where the population becomes unbounded for some $t \in (0,t_f)$ (with probability greater than ϵ). For such examples, the FSP will fail to find a sufficient approximation to the entire probability density vector. Such pathological examples cannot exist in biology, but if such an example did exist, all other methods (SSA, τ leaping, and others) would similarly fail.

The next section utilizes the FSP algorithm in order to solve the CME for two chemically reacting systems taken from the field of systems biology. The first example illustrates the simplest case where there are a known, finite number of possible states and the CME can be solved exactly. In this case Step 3 of the FSP algorithm may be skipped. The second example allows for regulatory protein translation and degradation events such that there is an infinite number of reachable states. In this case the FSP of the original chemical Markov process results in an approximate solution of the CME.

IV. EXAMPLES OF THE APPLICATION OF THE FSP ALGORITHM

The following two examples illustrate the use of the finite state projection algorithm for the numerical study of the Pap epigenetic switch. This inherently stochastic genetic switch determines whether or not *E. coli* bacteria will express the hairlike surface structures known as Pap pili. These pilli enable *E. coli* to attach to host cells and establish infections and account for nearly 90% of upper urinary tract infections in women. The biological model is based upon experimental observations made by Low and Co-Workers at

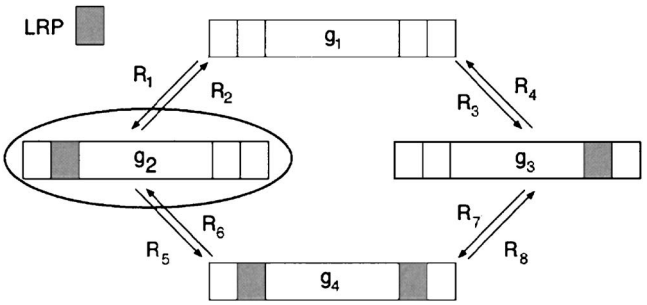

FIG. 3. Schematic of the four possible DNA-LRP binding configurations of the *pap* operon and the eight possible LRP binding and unbinding reactions. The circled state corresponds to the *production* state in which transcription of the messenger RNA's for pili production and PapI translation is possible.

UCSB,[13,16,17] and specific results of the more detailed model are presented by Munsky *et al.*[18] Figure 3 shows a simple illustration of the system consisting of the single *pap* operon with two binding sites and a regulatory protein, leucine-responsive regulatory protein (LRP). LRP binds reversibly at either or both of the *pap* binding sites such that the operon can exhibit a total of four different configurations. Each of these configurations is considered as a separate chemical species: g_1 to g_4 as defined in Fig. 3. When LRP binds to the upstream site (left) and not to the downstream (right) site the cell is considered to be in a *production* state— when in this state (circled in Fig. 3), the cell can produce the proteins necessary to begin production of pili. All other configurations do not produce the necessary proteins. In this paper, we consider only a small submodule of the full Pap system. In the more detailed model, the switch depends upon additional chemical species, and the *pap* operon can achieve 64 distinct configurations. Although we have successfully implemented the FSP algorithm on the full 64-configuration model (see Ref. 18 or some preliminary results), the objective of this paper is to solely illustrate the use of the FSP algorithm. We are interested in the state of the system at a specific instant in time t_f at which the cell's fate is decided; in our example, this decision occurs at 10 s.

In addition to the operon and LRP, the model also considers the local regulatory protein, PapI, which acts to decrease the rate at which LRP unbinds from the operon. In the real system the change in the population of PapI serves as a positive feedback loop in that larger concentrations of PapI make it more likely for the gene to express the g_2 configuration and continue to produce pili.[16,17] In the first example, the population of PapI is assumed to be constant, and the system has exactly four reachable states from the initial condition. In this case the chemical master equation can be solved exactly to find the probability density vector at any future time. In the second example the population of PapI is allowed to change according to translation and degradation events, and the resulting Markov process describing the chemical system has an infinite number of possible states. In each example, the solution scheme is first presented, followed by documentation of the specific parameters and a presentation of computed results. The FSP analyses are then compared to those obtained through use of the SSA and an explicit τ leaping algorithm, and comments are made regarding the comparative efficiency and accuracy of the three methods. In this study, we choose not to make comparisons between the FSP and partitioning methods such as the slow-scale SSA. Such a comparison is deemed unnecessary—one can also apply many of the same partitioning approximations for similar advantages to the FSP (work in preparation for submission elsewhere). All codes are implemented on a 1.50 GHz Intel Pentium 4 processor running a Linux environment; the SSA and the adaptive step τ leaping algorithm implementations are conducted using UCSB's software package STOCHKIT,[19] and all FSP analyses are conducted using Math Works MATLAB 7.0.

A. Example 1: Exact probability density vector solution for finite state problem

In the first example, we consider the Pap system shown in Fig. 3, in which it is assumed that the total concentrations of both LRP and PapI are finite integer quantities fixed at u_0 and τ_0, respectively. With these assumptions, one can uniquely write out all four possible state descriptions in terms of the populations of each of the important considered species.

$$\mathbf{X} = \{\mathbf{x}_i\} = \left\{ \begin{bmatrix} g_1 \\ g_2 \\ g_3 \\ g_4 \\ \mathrm{LRP} \\ \mathrm{PapI} \end{bmatrix} \right\} \tag{4.1}$$

$$= \left\{ \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ u_0 \\ r_0 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \\ u_0 - 1 \\ r_0 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \\ u_0 - 1 \\ r_0 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \\ u_0 - 2 \\ r_0 \end{bmatrix} \right\}.$$

The propensity function for each of the eight possible chemical reactions, $a_\mu(\mathbf{X})$ for $\mu=\{1,2,\ldots,8\}$, is given by a PapI-dependent reaction rate constant, $c_\mu([\mathrm{PapI}])$, multiplied by the product of the concentrations of the reactants. For example, reaction number 1 of the form $R_1: g_1+\mathrm{LRP} \rightarrow g_2$ has a propensity function: $a_1=c_1([\mathrm{PapI}])[g_1][\mathrm{LRP}]$, where the brackets $[\cdots]$ around a chemical species denote the population of that chemical species. Since in this case the populations of PapI and LRP are assumed to be constant, and g_i is either 0 or 1, the complete reaction matrix $\mathbf{A}$ can be written as

$$\mathbf{A} = \begin{bmatrix} -c_1u_0 - c_3u_0 & c_2 & c_4 & 0 \\ c_1u_0 & -c_2 - c_5(u_0-1) & 0 & c_6 \\ c_3u_0 & 0 & -c_4 - c_7(u_0-1) & c_8 \\ 0 & c_5(u_0-1) & c_7(u_0-1) & -c_6 - c_8 \end{bmatrix}. \tag{4.2}$$

Suppose that we know that at time $t=0$ the system is in the $\mathbf{x}_1$ state—it has the initial probability density vector, $\mathbf{P}(\mathbf{X};0)=[P(\mathbf{x}_1;0)\ P(\mathbf{x}_2;0)\ P(\mathbf{x}_3;0)\ P(\mathbf{x}_4;0)]^T=[1\ 0\ 0\ 0]^T$. Then we can exactly calculate the solution of the probability density vector at time t_f as $\mathbf{P}(\mathbf{X};t_f)=\exp(\mathbf{A}t_f)\mathbf{P}(\mathbf{X};0)$.

Table I provides the system parameters and reaction constants for this example. For the state reaction matrix $\mathbf{A}$ given in Eq. (4.2), the state transition matrix, $\exp(\mathbf{A}t_f)$, has been calculated in MATALB using the command *expm*(). Figure 4 (black bars) shows the probability density vector of the system at the final time, $t_f=10$ s as calculated using the FSP. Figure 4 also shows the same probability density vectors as averaged using 10^4 simulations of the SSA (dark gray bars) included in the software package STOCHKIT.[19] In terms of accuracy, Fig. 4 shows that the SSA and the FSP produce very similar results. However, even after 10^4 simulations, the pdv acquired with the SSA differs noticeably from the more accurate FSP solution. Suppose we are only interested in the probability that the gene will be in the g_1 configuration. From the FSP computation this state has a probability of 0.002 433. Five independent sets of 10^4 SSA simulations predicted this probability to be 0.0029, 0.0025, 0.0027, 0.0027, and 0.0016, respectively. Thus the SSA results have relative errors that range from −34% to +19%. Depending upon the needs of the researcher, such errors may be unacceptable, and more simulations will be required. As the number of simulations increases, the SSA approaches the accuracy of the FSP; however, even at 10^6 runs the relative errors of only four sets of simulations ranged as high as 0.6%. On average, each SSA run required the simulation of about 24 events. However, if one were to increase all of the rate constants by a large constant (or equivalently increase the time of simulation), then the number of reactions would increase proportionately. As more reactions occur, the computational effort of the SSA also increases, while the effort required for the FSP method remains unchanged. For a comparison of the time required, the FSP solution took less than 0.3 s while the SSA took approximately 0.4 s to simulate the system 10^4 times or 40 s to simulate the system 10^6 times.

TABLE I. Reactions and parameters used in the SSA and exact FSP solutions for the four-state, eight-reaction system describing the Pap epigenetic switch.

Reactions			
Number	Stoichiometry	Rate constant (c_μ)	Units
R_1	g_1+LRP→g_2	1	s^{-1}
R_2	g_2→g_1+LRP	2.50–2.25 ($r/(1+r)$)	s^{-1}
R_3	g_1+LRP→g_3	1	s^{-1}
R_4	g_3→g_1+LRP	1.20–0.20 ($r/(1+r)$)	s^{-1}
R_5	g_2+LRP→g_4	0.01	s^{-1}
R_6	g_4→g_2+LRP	1.20–0.20 ($r/(1+r)$)	s^{-1}
R_7	g_3+LRP→g_4	0.01	s^{-1}
R_8	g_4→g_3+LRP	2.50–2.25 ($r/(1+r)$)	s^{-1}

Parameters and initial conditions		
Parameter	Notation	Value
LRP population	u_o	100
PapI population	r_o	5
Initial time	t_o	0 s
Final time	t_f	10 s
Initial pdv	$\mathbf{P}(\mathbf{X};0)$	$[1,0,0,0]^T$

1. Comparison to τ leaping

As stated above, the use of time-leaping methods has dramatically improved the computational efficiency of the SSA in many circumstances. However, for this particular example, these methods offer no advantage. At any instant in time, each of the four molecular species, g_1 to g_4, has a population of either 0 or 1. It is not possible for *any* reaction to occur twice consecutively without resulting in negative populations. Furthermore, every propensity function switches between zero and some positive value within the space of a single reaction. In order to avoid impossible populations, therefore, no τ leap may include more than a single reaction, which is no better than an SSA step. The reader

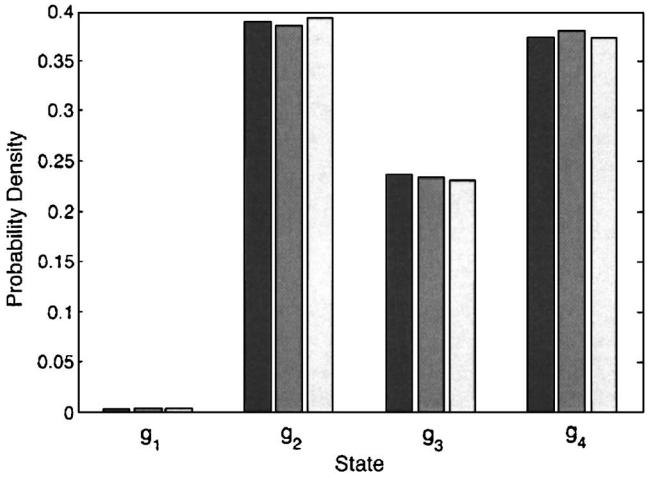

FIG. 4. Probability density vector for the simple four-state model at time $t_f=10$ s as calculated directly using the exact FSP method (black bars) and as averaged using 10^4 runs of the stochastic simulation algorithm (dark gray bars) and an adaptive τ leaping algorithm (light gray bars). Initial conditions were *pap* operon configuration, g_1, at $t_o=0$ s (see parameters and initial conditions in Table I).

should note that the statement applies to binomial τ leaping as well as Poisson τ leaping. For example, STOCHKIT's adaptive step size τ leaping code[19] automatically reverts to the SSA and takes about 0.4 s for 10^4 realizations. Figure 4 (light gray bars) illustrates the results using 10^4 τ leaping simulations.

B. Example 2: Approximate probability density vector solution for the Pap system with translation and degradation of PapI

In most realistic biological systems, the chemical concentrations of regulatory proteins are constantly changing by discrete values through, transcription, translation, degradation, and similar events. In this example we add additional such reactions to the above system and allow the population of PapI to change over time. For later convenience, we use the variable r to denote the concentration of PapI: $r \equiv [\mathrm{PapI}]$. Suppose r increases by a stochastic reaction that can occur when the gene is in the g_2 configuration. Also, let r decrease through a stochastic degradation event that is independent of the gene state. The propensity functions for these events can then be given, respectively, as $a_T = c_T[g_2]$ and $a_D = c_D r$. Because r is allowed to change, the set of all possible states becomes

$$\mathbf{X} = \left\{ \begin{bmatrix} g_1 \\ g_2 \\ g_3 \\ g_4 \\ \mathrm{LRP} \\ \mathrm{PapI} \end{bmatrix}_i \right\} = \left\{ \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ u_0 \\ r_i \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \\ u_0 - 1 \\ r \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \\ u_0 - 1 \\ r \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \\ u_0 - 2 \\ r_0 \end{bmatrix} \right\} \quad \text{for } r = \{0,1,2,\ldots\}. \tag{4.3}$$

At this point it is useful to establish a unique ordering system for the elements in the state space $\mathbf{X}$. For this particular problem, it is convenient to arrange the states according to the population of PapI:

$$j = \begin{Bmatrix} (4r+1) & \text{if } [g_1] = 1 \\ (4r+2) & \text{if } [g_2] = 1 \\ (4r+3) & \text{if } [g_3] = 1 \\ (4r+4) & \text{if } [g_4] = 1 \end{Bmatrix} = \sum_{i=1}^{4} (r + i[g_i]), \tag{4.4}$$

where j is the index of the state $\mathbf{x}_j \in \mathbf{X}$. The system changes from one state to another through three types of reactions: First, the operon configuration can change according to the reactions described above in the first example. The rates for these reactions are now dependent upon the variable concentration of PapI: $\mathbf{A}_r = \mathbf{A}(r)$, where the form of $\mathbf{A}$ is given in Eq. (4.2) The second reaction type allows for the translation of PapI only when the *pap* operon is in the g_2 configuration. The third type allows for PapI to degrade. Using the ordering defined in Eq. (4.4), all reactions can be combined to form the global reaction matrix:

$$\mathbf{A} = \begin{pmatrix} \mathbf{A}_0 - \mathbf{T}_0 - \mathbf{D}_0 & \mathbf{D}_1 & 0 & 0 & \cdots \\ \mathbf{T}_0 & \mathbf{A}_1 - \mathbf{T}_1 - \mathbf{D}_1 & \mathbf{D}_2 & 0 & \cdots \\ 0 & \mathbf{T}_1 & \mathbf{A}_2 - \mathbf{T}_2 - \mathbf{D}_2 & \mathbf{D}_3 & \ddots \\ \vdots & 0 & \mathbf{T}_2 & \mathbf{A}_3 - \mathbf{T}_3 - \mathbf{D}_3 & \ddots \\ \vdots & \ddots & 0 & \mathbf{T}_3 & \ddots \\ \vdots & \ddots & \ddots & 0 & \ddots \\ \vdots & \ddots & \ddots & \ddots & \ddots \end{pmatrix}, \tag{4.5}$$

where the transcription and the degradation matrices $\mathbf{T}$ and $\mathbf{D}$, respectively, are given by

$$\mathbf{T} = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & c_T & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \text{ and } \mathbf{D} = \begin{pmatrix} c_D[r] & 0 & 0 & 0 \\ 0 & c_D[r] & 0 & 0 \\ 0 & 0 & c_D[r] & 0 \\ 0 & 0 & 0 & c_D[r] \end{pmatrix}. \tag{4.6}$$

The production and degradation of PapI are modeled as stochastic events, such that it is possible (although with zero probability if c_T is finite and c_D is nonzero) that infinitely more PapI-production events will occur than PapI-degradation events in finite time. This suggests that the value of r must be allowed to grow unbounded, and we cannot compute an exact analytical solution as in the previous example. In this case it will be necessary to truncate $\mathbf{A}$ using the FSP algorithm.

Suppose that at time $t=0$ it is known that the gene is in the g_1 configuration and there are exactly r_0 molecules of PapI present in the system:

$$\mathbf{P}(X_{J_o};0)=\begin{bmatrix}1\\0\\0\\0\end{bmatrix},$$

where

$$J_0=\begin{bmatrix}4r_o+1\\4r_o+2\\4r_o+3\\4r_o+4\end{bmatrix}.$$

Then, using the FSP algorithm, if we can find a principle submatrix $\mathbf{A}_{J_k}$, such that

$$\mathbf{1}^T\exp(\mathbf{A}_{J_k}t_f)\mathbf{P}(\mathbf{X}_{J_k};0)\geqslant 1-\epsilon, \tag{4.7}$$

then we are guaranteed that the probability density for every state at time $t=t_f$ is within the bounds given by

$$\exp(\mathbf{A}_{J_k}t_f)\mathbf{P}(\mathbf{X}_{J_k};0)\leqslant\mathbf{P}(\mathbf{X}_{J_k};t_f)\leqslant\exp(\mathbf{A}_{J_k}t_f)\mathbf{P}(\mathbf{X}_{J_k};0)+\epsilon\mathbf{1}. \tag{4.8}$$

For this problem, it is easy to choose a searching algorithm to dictate the expansion of the set J_k until the condition specified by Eq. (3.1) is met. The most reasonable search algorithm is to simply continue adding adjacent block structures of the form given in Eq. (4.5)—this corresponds to increasing the space of sets that are sequentially reachable from J_o through PapI translation and degradation events.

Table II provides the reaction parameters that, in addition to those in Table I, have been used for this example. We have specified a total error tolerance of $\epsilon=10^{-6}$ for the probability density vector at time t_f. We chose this error tolerance in order to predict a switch rate on the order of 10^{-4} to within a 1% relative error. Figure 5 shows the lower bound on the probability density vector at the final time as computed with the FSP algorithm. In this figure, the states have been arranged according to their index as specified in Eq. (4.4). Recall that although inclusion of states is based upon reachability, the choice of enumeration is arbitrary, such that it is often necessary to reorder and combine states to illustrate more meaningful results. For instance, in this example we may be most interested in the distribution of the different operon states: g_1 through g_4 or the distribution of the population of PapI. Figure 6 shows the partial probability density vectors for the population of PapI as separated for each possible operon configuration. From the figure, one can observe that the production operon configuration, g_2 (top right), has a different distribution shape than do the the other states. In particular, the median population of PapI is much larger when the operon is in the g_2 configuration. In this Pap system, the population of PapI can be related to amount of pili expression found on the bacteria, and it might not actually be interesting to know the gene configuration of the system. In this case, it is helpful to consider the distribution in the format of Fig. 7, which shows the probability density of the total amount of PapI. For these results, the FSP required the inclusion of all values of r from 0 to 30 (corresponding to a total of 124 states), and the sum of the probability density was found to be greater than 0.999 999. The results provide us a guarantee that the probability of every state (including those with more than 30 copies of r) is known within a positive error of 10^{-6}. We also have a guarantee that the error in the full probability density vector is non-negative and sums to less than 10^{-6}.

TABLE II. Reactions and parameters used in the SSA and exact FSP solutions for the Pap epigenetic switch in which the population of the regulatory protein PapI may change according to stochastic translation and degradation events. See also Table I.

Reactions			
Number	Stoichiometry	Rate constant (c_μ)	Units
R_T	$g_2\rightarrow g_2+r$	10	s^{-1}
R_D	$r\rightarrow\phi$	1	s^{-1}

Parameters and initial conditions		
Parameter	Notation	Value
Initial catalyst protein	r_o	5
Initial *pap* operon state	g_1	⋯
Initial state	$j_o=4r_0+1$	21
Initial pdv	$\mathbf{P}(\mathbf{X}_{J_o};0)=1$	⋯
Allowable error in pdv	ϵ	10^{-6}

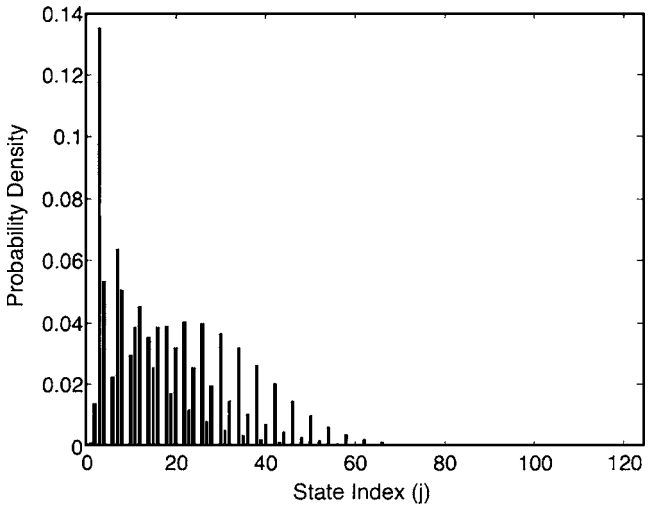

FIG. 5. Probability density vector solution for the Pap model in which PapI is allowed to change through stochastic translation and degradation events. The states are ordered according to Eq. (4.4), and the density vector is shown as time $t_f=10$ s for the initial condition of state $j=21$ ([PapI]=5 molecules and *pap* operon in state g_1) at time $t_o=0$ s (see also parameters and initial conditions in Tables I and II).

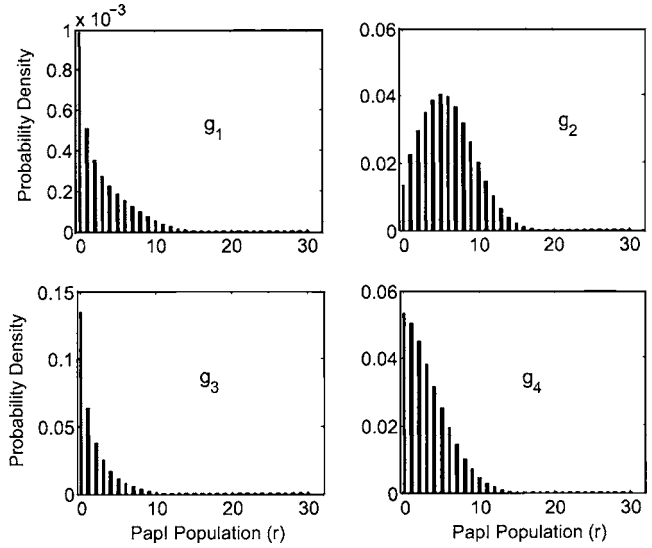

FIG. 6. Solution for the Pap model in which PapI is allowed to change through stochastic translation and degradation events. The probability density vector from Fig. 5 is separated into four components according to when the *pap* operon is in (top left) g_1, (top right) g_2, (bottom left) g_3, and (bottom right) g_4 (see also Fig. 5).

The most biologically interesting results correspond to cells in which there is a large amount of PapI; these are the cells that will actually succeed in turning ON and express pili. For our model, we define an ON cell as a cell that contains at least 20 molecules of PapI. In Fig. 7, ON cells are all those to the right of the dashed line. From the figure one can immediately see that the probability turning ON is very low; using the FSP, this probability is guaranteed to be within the interval of $[1.376, 1.383] \times 10^{-4}$. We also conducted five sets of 10^5 SSA simulations—each with a different seed for the random number generator. In these numerical experiments, the SSA computed the probability of having more than 20 molecules of PapI to be $\{1.3, 1.8, 1.7, 1.3, \text{and } 0.9\} \times 10^{-4}$. For the five sets of 10^5 SSA simulations, the relative error ranged between −35% and +30%. For comparison, the relative error of the FSP is guaranteed to be in the range of −0.46%–0.00% (more than three orders of magnitude more precise than 10^5 simulations of the SSA). Figure 8 (light line) plots the average number of times the SSA produces the result that there are more than 20 molecules of PapI at time t_f as a function of the number of simulation runs. The horizontal line in the figure shows the probability as calculated using the FSP algorithm, where the thickness of the line exceeds the difference between the computed upper and lower bounds. As in the previous example, more SSA simulations allow for better accuracy at the cost of additional computational expense. For a comparison of the methods' efficiency and accuracy, Table III provides the computational time and relative error in the prediction of the Pap OFF to ON switching rate after $\{1.25, 2.50, 5.00, \text{and } 10.00\} \times 10^5$ simulations. From the table one can immediately see that the performance of the FSP is superior to that of the SSA for this example.

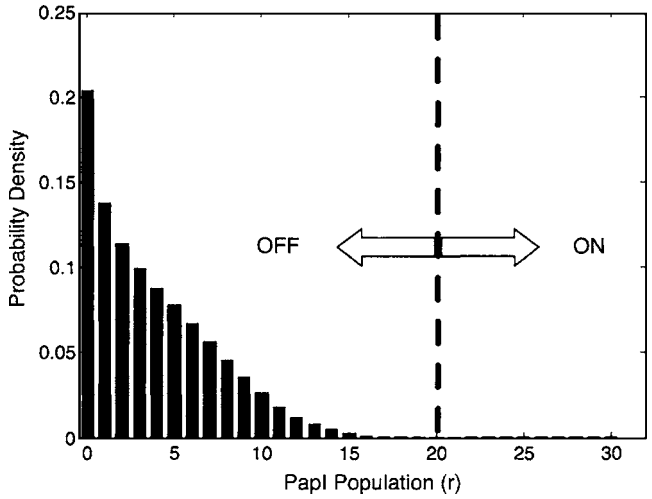

FIG. 7. The probability density vector of the population of PapI as calculated in Sec. IV B at final time $t_f = 10$ s. All cells that contain more than 20 molecules of PapI are considered to be ON (see also Figs. 5 and 6).

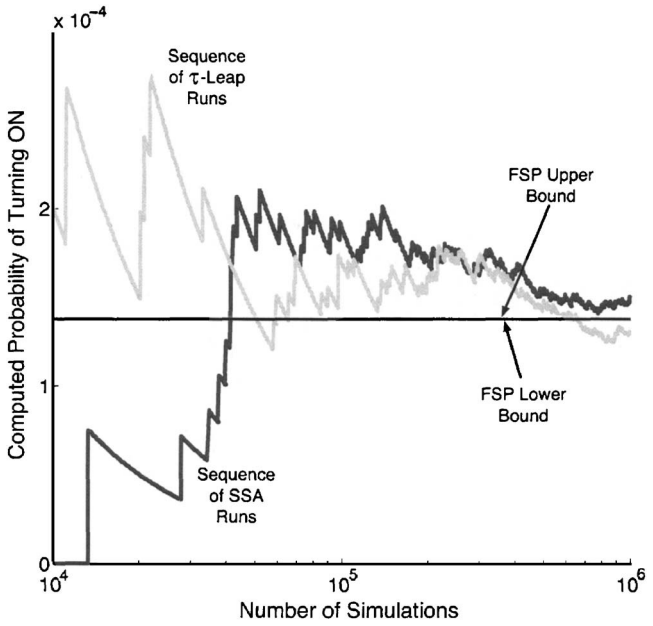

FIG. 8. Predictions of Pap OFF to ON switching rate using the SSA (light gray) and an explicit, adaptive step size τ leaping algorithm (dark gray) from STOCHKIT. The x axis shows the number of simulations that have been conducted and the y axis shows the corresponding computed probability of turning ON. As the number of simulations increases, the probabilities computed with the Monte Carlo methods converge toward the more precise FSP solution (horizontal line). However, a large relative error persists even after 10^6 simulations. For comparison, the thickness of the horizontal line corresponds to the upper and lower bounds on the switch rate as computed using the FSP algorithm. See also Table III.

1. Comparison to τ leaping

As above, the use of time-leaping methods can do little to improve the computational efficiency of the SSA for this example. In this case, negative molecular populations will always result if any LRP binding/unbinding reaction is simulated twice consecutively before a different LRP binding/unbinding event. In order to avoid impossible populations, therefore, one must use an adaptive step size algorithm, and no τ leap may be allowed to include more than a single reaction from the set R_1 to R_8. Exploring the SSA simulations, we found that more than one quarter of all of the

TABLE III. A comparison of the efficiency and accuracy of the FSP, SSA, and adaptive explicit τ leaping methods for the prediction of the Pap OFF to ON switching rate. Using the FSP, it takes less than 4 s to guarantee that the OFF to ON switch rate is within the interval of $[1.376, 1.383]\times 10^{-4}$, a relative error of less than 0.5%. The table shows the results of a single set of 10^6 statistically independent simulations for each of the SSA and the τ leaping methods. The relative errors have been calculated after $\{1.25, 2.50, 5.00, \text{and } 10.00\}\times 10^5$ simulations. Simulation sets with different random number generator seed values will produce different results (some are better and some are worse—results not shown). In contrast, every run of the FSP algorithm always produces the exact same result. All codes are run on the same 1.50 GHz Intel Pentium 4 processor running a Linux environment. See also Fig. 8.

Method	Number of simulations	Time (s)	Relative error in switch rate (%)
FSP	Does not apply[a]	<4	<0.5
SSA[b]	1.25×10^5	≈ 18	38.8
SSA	2.5×10^5	≈ 35	27.3
SSA	5.0×10^5	≈ 70	9.9
SSA	10.0×10^5	≈ 140	8.5
τ leaping	1.25×10^5	≈ 18	9.9
τ leaping	2.5×10^5	≈ 35	24.4
τ leaping	5.0×10^5	≈ 70	7.0
τ leaping	10.0×10^5	≈ 140	6.0

[a]The FSP is run only once with a specified allowable total error of 10^{-6}.
[b]SSA and τ leaping simulations have been conducted using STOCHKIT (Ref. 19).

reactions involved operon configuration changes. Therefore, if we make the liberal assumptions that a single τ leap step is as fast as a single SSA step and that there is exactly one R_1 to R_8 reaction included in each τ leap, then a τ leaping method can boost the speed of the SSA by a maximum factor of less than 4. It must be mentioned, however, that PapI production and degradation reactions can also result in excessively large changes in propensity functions, thus further restricting the size of allowable time leaps. In practice τ leap steps may take far longer to compute than individual SSA steps, and one would expect that τ leaping will provide far less benefit over the SSA in this example. As in the previous example, it does not matter what type of τ leaping is chosen (Poisson or binomial); the leap size will be similarly restricted in each. As an example of the failure of τ leaping to handle this example, we have again utilized STOCHKIT[19] and we have set the program to use an adaptive explicit τ leaping algorithm.[8] For this algorithm, computation took about 14 seconds for 10^5 runs (the same as the direct step SSA), and the accuracy was similar to that of the SSA. The dark gray line in Fig. 8 illustrates the convergence of the τ leaping predictions as more and more simulations have been conducted (see also Table III).

V. CONCLUSIONS

This paper introduced the finite state projection (FSP) method and the FSP algorithm for the approximate solution of the chemical master equation. Unlike previous Monte Carlo-type analyses, the FSP directly computes the system's probability density vector at a given time without requiring the computation of large numbers of process realizations. In the case of any Markov process containing only a finite number of states, the FSP method provides an exact analytical solution. When the number of possible states is infinite, the approximate solution on the projected space guarantees upper and lower bounds on the solution of the true system. The FSP algorithm provides a systematic means of increasing the size of the finite state projection until these bounds are within any prespecified error tolerance. The FSP method was effectively demonstrated on a real biological example problem: the Pap epigenetic switch in *E. coli*. Given a known initial state, we have used the FSP to generate the probability density vector at a later time of interest. We have compared the accuracy and efficiency of the FSP and two popular Monte Carlo methods: the SSA and an adaptive step size explicit τ-leaping algorithm. In these examples, we have shown that the FSP algorithm outperforms both of the two Monte Carlo methods, especially when computing the probability of unlikely events, such as Pap OFF to ON switching. This example and others suggest that the FSP is a very promising method—especially in the field of system's biology, where very small chemical populations are common and in which unlikely events may be of critical importance.

For biological problems such as the Pap example, the nature of the FSP, in addition to being more efficient and more accurate than existing Monte Carlo methods, provides a wealth of information and opens new doors for stochastic analyses. For example, one could easily adjust the FSP to precisely compute the probability of avoiding undesirable system trajectories throughout prespecified periods of time. One may also use the FSP to easily and efficiently perform parametric sensitivity and robustness analyses on the dynamics of the biological system. However, in its current basic form, the FSP method is not yet feasible for all classes of systems. For example, when there is a high probability that species populations will undergo large excursions in small periods of time, the basic FSP algorithm will require solutions to very large numbers of coupled linear ODEs.

While such computations present a significant challenge, there are several approaches that will help the FSP to meet these challenges. Krylov subspace methods for sparse systems could effectively enable the computation of the matrix exponential times a vector for very large systems (tens of thousands). Furthermore, several readily available tools facilitate lower-order approximations of larger systems and promise significant reductions in computational cost. State aggregation and multiscale partitioning also provide enormous benefits over the original FSP. These approaches along with advanced approaches for updating the finite projection subset are currently being developed and will be presented elsewhere. While the practical limits of the finite-projection-based approach are yet unknown, future implementations will greatly expand the class of problems for which the FSP is an efficient and versatile tool for stochastic analysis.

ACKNOWLEDGMENTS

This material is based upon the work supported by the National Science Foundation under Grant NSF-ITR CCF-0326576 and the Institute for Collaborative Biotechnologies

through Grant DAAD19-03-D-0004 from the U.S. Army Research Office.

[1] D. McQuarrie, J. Appl. Probab. **4**, 413 (1967).
[2] D. T. Gillespie, Physica A **188**, 404 (1992).
[3] D. T. Gillespie, J. Phys. Chem. **81**, 2340 (1977).
[4] M. A. Gibson and J. Bruck, J. Phys. Chem. **104**, 1876 (2000).
[5] D. T. Gillespie, J. Chem. Phys. **115**, 1716 (2001).
[6] M. Rathinam, L. R. Petzold, Y. Cao, and D. T. Gillespie, J. Chem. Phys. **119**, 12784 (2003).
[7] D. T. Gillepsie and L. R. Petzold, J. Chem. Phys. **119**, 8229 (2003).
[8] Y. Cao, D. T. Gillespie, and L. R. Petzold, J. Chem. Phys. **123**, 054104 (2005).
[9] T. Tian and K. Burrage, J. Chem. Phys. **121**, 10356 (2004).
[10] C. V. Rao and A. P. Arkin, J. Chem. Phys. **118**, 4999 (2003).
[11] E. L. Haseltine and J. B. Rawlings, J. Chem. Phys. **117**, 6959 (2002).
[12] Y. Cao, D. Gillespie, and L. Petzold, J. Chem. Phys. **122**, 014116 (2005).
[13] L. B. Blyn, B. A. Braaten, C. A. White-Ziegler, D. H. Rolfson, and D. A. Low, EMBO J. **8**, 613 (1989).
[14] C. Moler and C. Van Loan, SIAM Rev. **20**, 801 (1978).
[15] C. Moler and C. Van Loan, SIAM Rev. **45**, 3 (2003).
[16] A. Hernday, M. Krabbe, B. Braaten, and D. Low, Proc. Natl. Acad. Sci. U.S.A. **99**, 16470 (2002).
[17] A. D. Hernday, B. A. Braaten, and D. A. Low, Mol. Cell **12**, 947 (2003).
[18] B. Munsky, A. Hernday, D. Low, and M. Khammash, *Proceedings of FOSBE Conference*, (unpublished), pp. 145–148.
[19] L. Petzold, *et al.* STOCHKIT Beta Version, November 2004. Downloaded from the internet at http://www. engineering.ucsb.edu/cse/StochKit/index.html