Error analysis and efficient sampling in Markovian state models for molecular dynamics

Nina Singhal
Department of Computer Science, Stanford University, Stanford, California 94305

Vijay S. Pande[a)]
Department of Chemistry, Stanford University, Stanford, California 94305

(Received 7 June 2005; accepted 14 September 2005; published online 29 November 2005)

In previous work, we described a Markovian state model (MSM) for analyzing molecular-dynamics trajectories, which involved grouping conformations into states and estimating the transition probabilities between states. In this paper, we analyze the errors in this model caused by finite sampling. We give different methods with various approximations to determine the precision of the reported mean first passage times. These approximations are validated on an 87 state toy Markovian system. In addition, we propose an efficient and practical sampling algorithm that uses these error calculations to build a MSM that has the same precision in mean first passage time values but requires an order of magnitude fewer samples. We also show how these methods can be scaled to large systems using sparse matrix methods. © *2005 American Institute of Physics*.
[DOI: 10.1063/1.2116947]

I. INTRODUCTION

The kinetics of proteins and other biological molecules is fundamental to biology. However, while experiments can give information about the rate of folding, they typically cannot give details about molecular motion at atomic resolution or at very short time scales. Therefore, it is natural to turn to molecular dynamics to simulate how molecules move in atomic detail. While molecular simulations are computationally expensive, by using distributed computing methods such as Folding@Home,[1] it is now possible to simulate many independent molecular-dynamics trajectories in a parallel fashion.

After generating large ensembles of molecular-dynamics simulations, we wish to analyze these trajectories to find thermodynamic properties such as the equilibrium distribution of the protein and kinetic properties such as the rate and mechanism of folding. Until recently, it was only possible to generate small ensembles of molecular-dynamics simulations, so most approaches to analyzing these trajectories dealt with finding and comparing different order parameters along the trajectory, such as the distance to the native state, or the order of secondary structure formation.

Some new approaches involve graph-based models for protein kinetics that divide the conformation space into discrete states and calculate transition probabilities or rates between states based on molecular-dynamics trajectories.[2–4] It is efficient to calculate kinetic properties such as the probability that a conformation will fold (P_{fold}) or the mean first passage time (MFPT), the average time taken for a given conformation to fold, from this graph.[2] This model allows one to easily combine and analyze simulation data started from various conformations and naturally handles intermediate states and traps. This approach has been applied to small protein systems,[2,4] a nonbiological polymer,[5,6] and vesicle fusion[7] with good agreement with experimental rates.

While these graph-based models agree well with experiments, only a single value for the rate was used in the comparison. It is also important to determine the uncertainty in this value, so one can know the confidence of the results. One main source of error is caused by grouping conformations into states and assuming that transitions between these states are Markovian. If we look at a protein and consider each conformation as its own state, on the tens of picosecond and longer time scale, the transitions follow a Markovian pattern. Unfortunately, sampling transitions between an infinite number of states is impractical; therefore, the Markovian state model groups conformations into a finite number of discrete states. However, it has been shown that if the conformations are grouped incorrectly, the state space is no longer Markovian, and any analysis that assumes a Markovian process may produce incorrect results.[3] Even if the states are defined such that the transitions between them are Markovian, the results could still be in error. This second source of error results from the finite sampling of transitions between states, which gives uncertainties in the transition probability estimates and in turn leads to uncertainties in the values we calculate, such as the MFPT.

There has been some recent work on error analysis in these graph-based models of a protein conformation space. Swope *et al.* focused on the problem of defining states which meet the Markovian criteria. They provided tests for whether or not a given state space definition is history independent.[3] Here, we will focus on the error caused by finite sampling. Some recent work has involved a Bayesian approach to sampling possible transition probability matrices and solving each sample for the value of interest.[8] While this approach can estimate errors, it does not scale well for systems with

[a)]Electronic mail: pande@stanford.edu

large numbers of states. In addition, we want to determine the transitions that contribute the most to the uncertainty, which the current techniques do not allow. If we can identify these transitions, additional simulations can be started from them to increase the overall precision.

In this paper, we will discuss novel methods for computing the error in a Markovian state model for molecular dynamics caused by finite sampling of transitions. We will give different methods for calculating the error from finite sampling and how it translates into errors in the mean first passage time and other estimates. The methods employ a set of approximations and lead to an efficient and practical closed-form solution for the uncertainty. We will also present a new sequential sampling algorithm that uses these error estimates to improve the sampling efficiency by over an order of magnitude. In addition, we discuss how the use of sparse matrix techniques will allow these methods to scale to systems with large numbers of states. These algorithms are then tested and the approximations validated a toy Markovian system.

II. METHODS

Molecular-dynamics simulations are often used to understand protein kinetics. A question then arises as how to best analyze these molecular-dynamics trajectories. In a previous paper,[2] we clustered nearby conformations from the trajectories to group the conformation space of the protein into discrete states. We assumed that transitions between these states were Markovian, or history independent, and we estimated the transition probabilities between states by counting the number of times we see each transition. From this Markovian state model (MSM), it was possible to efficiently calculate kinetic properties such as the P_{fold} and the MFPT.

In this work, we are interested in determining the uncertainty in the kinetic results that can be calculated from this graph-based model. We will assume that we can define a Markovian state space for the protein, though forming states that meet this criterion is not a trivial task.[2,3] Even with this assumption, one can still have errors in the results. Since we can only finitely sample the transitions between states, we will have some statistical uncertainty in the transition probabilities. Therefore, any value we calculate from the transition probabilities will also have an uncertainty associated with it.

In this section, we will first discuss how to calculate the MFPT from the transition probabilities. We then derive the distribution for the transition probabilities, and define both sampling- and non-sampling-based methods for calculating the distribution of the MFPT. From these error estimates, we develop an efficient adaptive sampling technique designed to increase precision. Lastly, we will show how to use sparse matrix manipulations that permit the scaling of these methods to systems with large numbers of states.

A. Mean first passage times

In a Markovian state model, we represent the conformation space by K discrete states, each of which corresponds to some distinct group of protein conformations. Let us define the probability of transitioning from state i to state j at a time step of Δt as p_{ij}. We also assume that these states are Markovian, i.e., that the transitions between them are history independent at Δt. We can use the transition probabilities to calculate kinetic properties of the system such as the probability of folding, mean first passage time to reach the final state, and the equilibrium distribution. These quantities are defined by sets of linear equations that are based on the transition probabilities.

For example, the equations for the mean first passage time, $\mathbf{x}$, from any state to the final state, are of the form[2]

$$x_i = \begin{cases} \Delta t + \sum_{j=1}^{K} x_j p_{ij}, & i \neq K, \\ 0, & i = K, \end{cases} \tag{1}$$

where the Kth state represents the final state. Writing this in matrix form, we have

$$\begin{bmatrix} p_{11}-1 & p_{12} & \cdots & p_{1K} \\ p_{21} & p_{22}-1 & \cdots & p_{2K} \\ & \ddots & & \\ p_{(K-1)1} & \cdots & p_{(K-1)(K-1)}-1 & p_{(K-1)K} \\ 0 & \cdots & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_{K-1} \\ x_K \end{bmatrix} = \begin{bmatrix} -\Delta t \\ -\Delta t \\ \vdots \\ -\Delta t \\ 0 \end{bmatrix}, \tag{2}$$

where the last line is the boundary condition that the mean first passage time from the final state is zero. The matrix on the left side of Eq. (2) will be referred to as $\mathbf{A}$, with rows $\mathbf{a_i}$. We will use these mean first passage time equations as an example throughout the paper.

B. Transition probability distribution

Finite sampling causes uncertainties in the estimates of the transition probabilities between states. In this section, we derive a distribution over possible transition probability vectors. Define p_{ij}^* as the actual transition probability from state i to j at a time step of Δt. The sum of the transition probabilities from state i is equal to 1,

$$\sum_{j=1}^{K} p_{ij}^* = 1. \tag{3}$$

We do not know these actual transition probabilities, but we can estimate them by sampling transitions between states. Since we make the assumption that our state space is Markovian, each transition sample originating from state i will be a random variable with K possible values occurring with probabilities p_{ij}^* for $j=1\cdots K$. Define the transition count z_{ij} as the total number of transition samples which start in state i and end in state j, and define n_i as the total number of samples originating from state i,

$$\sum_{j=1}^{K} z_{ij} = n_i. \tag{4}$$

The distribution of the z_{ij} variables follows the multinomial distribution with parameters $n_i, p^*_{i1}, p^*_{i2}, \ldots, p^*_{iK}$.[9] From these transition counts, we can calculate the maximum-likelihood estimates of the transition probabilities, $\hat{p}_{ij}$, which, for the multinomial distribution, are simply the number of transitions from state i to state j, z_{ij}, divided by the total number of transitions from state i, n_i.[9] Thus,

$$\hat{p}_{ij} = \frac{z_{ij}}{n_i}. \tag{5}$$

However, the maximum-likelihood estimates give no indication of the uncertainties in the transition probabilities. Using Bayesian analysis, we can compute the distribution over all possible vectors of transition probabilities, as opposed to simply calculating the most likely transition probability vector. Each set of possible transition probabilities $p_{i1}, p_{i2}, \ldots, p_{iK}$, where $0 \leq p_{ij} \leq 1$ and $\sum_{j=1}^{K} p_{ij} = 1$, has some chance of producing the transition counts z_{ij} that we observed. The probability of a particular $\mathbf{p_i}$ being the true transition probability vector, given the observed transition counts, is, from Bayes' rule,

$$P(\mathbf{p_i}|\mathbf{z_i}) \propto P(\mathbf{z_i}|\mathbf{p_i})P(\mathbf{p_i}) = p_{i1}^{z_{i1}} p_{i2}^{z_{i2}} \cdots p_{iK}^{z_{iK}} P(\mathbf{p_i}), \tag{6}$$

where $P(\mathbf{p_i})$ is the prior probability over the transition probability vectors, i.e., the distribution of transition probability vectors before observing any data.

A typical choice for the prior is the Dirichlet distribution, the conjugate prior of the multinomial distribution. This means that if the prior $P(\mathbf{p_i})$ is a Dirichlet distribution, then the posterior $\mathbf{P(p_i|z_i)}$ is also a Dirichlet distribution.[10] The Dirichlet distribution with variables $\mathbf{p}$ and parameters $\mathbf{u}$ is defined as

$$\text{Dirichlet}(\mathbf{p};\mathbf{u}) = \frac{1}{Z(\mathbf{u})} \prod_{i=1}^{K} p_i^{u_i - 1}, \tag{7}$$

where $Z(\mathbf{u})$ is a normalizing constant defined in Appendix A. If we define the prior of the transition probabilities as a Dirichlet distribution with parameters $\alpha_{i1}, \alpha_{i2}, \ldots, \alpha_{iK}$ and we observe transition counts $z_{i1}, z_{i2}, \ldots, z_{iK}$, the posterior of the transition probabilities is a Dirichlet distribution with parameters $\alpha_{i1}+z_{i1}, \alpha_{i2}+z_{i2}, \ldots, \alpha_{iK}+z_{iK}$. For notational convenience, we define the Dirichlet counts as

$$u_{ij} = \alpha_{ij} + z_{ij}. \tag{8}$$

Therefore, assuming a Dirichlet prior, the distribution of the transition probabilities $\mathbf{p_i}$ given the observed data counts is Dirichlet$(\mathbf{p_i};\mathbf{u_i})$.

Choosing the parameters for the prior completes the description of the distribution. The Dirichlet distribution is noninformative for any parameter $\alpha_{ij}=0$. If we set $\alpha_{ij}=0$ and do not observe any transitions from i to j, the posterior of p_{ij} will always equal zero. However, for molecular dynamics, not seeing a particular transition over some finite sampling does not imply that the transition can never occur. So, we will restrict ourselves to positive priors. Possible choices for the prior distribution are the uniform distribution $\alpha_{i1}=\alpha_{i2}=\cdots=\alpha_{iK}=1$ and the symmetric Dirichlet $\alpha_{i1}=\alpha_{i2}=\cdots=\alpha_{iK}$. In the limit, as the sampling (and therefore transition counts) increases, the distribution of the transition probabilities will not depend on the choice of the prior distribution, therefore making further calculations insensitive to the choice of prior distribution.

It will be useful to state the expected values of the posterior distribution of the transition probabilities for future reference, where w_i is a normalizing weight variable:[10]

$$\bar{p}_{ij} = E(p_{ij}) = \frac{u_{ij}}{w_i}, \quad w_i = \sum_{j=1}^{K} u_{ij}. \tag{9}$$

C. Sampling-based error analysis methods

In our previous paper,[2] we used $\hat{p}_{ij}$ in Eq. (2) to calculate an estimate of the true values of the mean first passage times $\mathbf{x}$. We now wish to calculate the *distribution* of $\mathbf{x}$ given the distribution of the $\mathbf{p_i}$ values. In particular, we are interested in the MFPT from the initial state to the final state, since this value can be converted to the rate of folding and compared with experiments. Therefore, we are interested in the distribution of the term x_1.

We propose a number of sampling-based methods for calculating the distribution of x_1. All of these methods involve repeatedly generating a sample of transition probabilities and converting to a sample of x_1. We can either sample the transition probabilities from the Dirichlet distributions or from approximate multivariate normal distributions (MVN). Then, we can either solve the above set of linear equations or we can substitute into a first-order Taylor series approximation to the set of equations. Each of these four options will be described below, and can be combined to give four sampling-based methods for calculating error, which are summarized below.

1. Sampling from the Dirichlet distribution

As shown in Sec. II B, if we assume a Dirichlet prior, the posterior distribution of $\mathbf{p_i}$, given the transition counts, is a Dirichlet distribution with parameters $\mathbf{u_i}$, as defined in Eq. (8). A method for generating samples from the Dirichlet distribution is given in Appendix A.

A sample of the $\mathbf{A}$ matrix, defined in Eq. (2), consists of K independent samples from Dirichlet distributions, each corresponding to one row of the matrix. As was shown in Appendix A, each sample from a Dirichlet distribution takes expected time $O(8KQ)$, where Q is the time to sample from a normal distribution. Therefore, each sample of transition probabilities requires time $O(8K^2Q)$.

2. Sampling from a multivariate normal distribution

Sampling from the Dirichlet distribution is very expensive. In an attempt to reduce this cost, the true Dirichlet distribution of the $\mathbf{p_i}$ parameters is approximated by a MVN distribution. In addition, the MVN has some nice properties that we will exploit in Sec. II D.

If $\mathbf{p_i}$ is distributed as Dirichlet $(\mathbf{p_i};\mathbf{u_i})$, then by the central limit theorem, the distribution of $\mathbf{p_i}$ converges to a multivariate normal distribution[11] with mean $\boldsymbol{\mu}_\mathbf{i}$ and covariance matrix $\boldsymbol{\Sigma}_\mathbf{i}$ given by

$$\boldsymbol{\mu}_\mathbf{i} = \frac{\mathbf{u_i}}{w_i}, \tag{10}$$

$$\boldsymbol{\Sigma}_\mathbf{i} = \frac{1}{w_i^2(w_i+1)}[w_i \mathrm{Diag}(\mathbf{u_i}) - \mathbf{u_i}\mathbf{u_i^T}], \tag{11}$$

where the superscript "**T**" denotes the transpose and $\mathrm{Diag}(\mathbf{u_i})$ represents a matrix with entries u_{ij} along the diagonal. A method for creating samples of the $\mathbf{p_i}$ variables from this approximate MVN distribution is given in Appendix B.

For each sample of the **A** matrix, we must generate K independent samples from the MVN distributions. As described in Appendix B, each sample of a MVN distribution requires time $O(KQ+K)$, where Q is the time to sample from a normal distribution. Therefore, each sample of transition probabilities takes time $O(K^2Q+K^2)$. In addition, there is a one-time cost of $O(K)$ for each of the K MVN distributions.

This approximation assumes that the central limit theorem holds and that the transition probabilities are well approximated by multivariate normal distributions. One drawback of the MVN approximation is that it permits negative values of the transition probabilities. While these negative values are invalid from a physical perspective, they generally do not affect the error calculations.

3. Solving sets of linear equations

Once we have generated a sample of transition probabilities from either Dirichlet or MVN distributions, we can simply solve Eq. (2) to find the MFPT vector. This can be done by factoring the matrix into the form $\mathbf{A}=\mathbf{LU}$ where $\mathbf{L}$ is a lower triangular matrix and $\mathbf{U}$ is an upper triangular matrix with unit entries along the diagonal followed by forward and back substitutions.[12] The cost of this algorithm is $(1/3)K^3$ for the factoring plus $O(K^2)$ for the substitutions.

4. Taylor series approximation

Solving the set of linear equations for each sample directly is expensive. Instead, we can approximate the solution using a first-order Taylor series expansion. The mean first passage time from the initial state, as given by Eq. (2), depends on all the parameters a_{ij} as well as the parameter Δt. We assume for now that Δt is a constant, but can modify this if we allow variable time steps in the simulation data. Thus,

$$x_1 = f(a_{11}, a_{12}, \ldots, a_{KK}) = f(\mathbf{A}). \tag{12}$$

The function f does not, in general, have a simple form. We therefore perform a first-order Taylor series expansion to x_1 around the expected values of the parameters, as given in Eq. (9):

$$x_1 = \bar{x}_1 + \Delta x_1 = f(\bar{\mathbf{A}}) + \left.\frac{\partial f}{\partial a_{11}}\right|_{\bar{\mathbf{A}}}\Delta a_{11} + \left.\frac{\partial f}{\partial a_{12}}\right|_{\bar{\mathbf{A}}}\Delta a_{12} + \cdots + \left.\frac{\partial f}{\partial a_{KK}}\right|_{\bar{\mathbf{A}}}\Delta a_{KK}, \tag{13}$$

where $\bar{x}_1 = f(\bar{\mathbf{A}})$ and the Δa_{ij} are small perturbations in the parameters. Thus,

$$\Delta x_1 = \left.\frac{\partial f}{\partial a_{11}}\right|_{\bar{\mathbf{A}}}\Delta a_{11} + \left.\frac{\partial f}{\partial a_{12}}\right|_{\bar{\mathbf{A}}}\Delta a_{12} + \cdots + \left.\frac{\partial f}{\partial a_{KK}}\right|_{\bar{\mathbf{A}}}\Delta a_{KK}. \tag{14}$$

Appendix C gives an efficient way for computing all the terms of the form $\partial f/\partial a_{ij}|_{\bar{\mathbf{A}}}$ in Eq. (14).

Once we have generated a sample of the transition probabilities from either Dirichlet or MVN distributions, we convert to a sample of a_{ij} using Eq. (2) and then a sample of Δa_{ij} by subtracting the expected values $\bar{a}_{ij}$. We next substitute these values into Eq. (13) to find a sample from the distribution of x_1. There are K^2 terms in Eq. (13), so the substitution will take time $O(K^2)$ per sample. In addition, as shown in Appendix C, there is a one-time cost of $O((1/3)K^3+3K^2)$ to both solve for $\bar{x}_1 = f(\bar{\mathbf{A}})$ and generate the partial derivative terms in the Taylor series.

5. Sampling method summary

Combining the techniques for sampling and the solution of the linear system given above, we get four sampling-based methods for finding the uncertainty in the MFPT from the initial state. Assume that we take a total of N samples to estimate the distribution of x_1. Table I shows the running times and various assumptions of each of the four sampling-based methods, where Q is the time to sample from a normal distribution.

Method 1 that we propose is similar to previous methods to calculate error in these models.[8] We have also introduced new methods that use different approximations and improve the running time of the error analysis.

D. Non-sampling-based error analysis method

The methods in the previous section all relied on sampling possible transition probabilities from either the Dirichlet or MVN distributions to get samples from the distribution of x_1. If we make both the MVN approximation and the Taylor series expansion, we can derive a *closed-form* representation for the distribution of x_1.

First, we will rewrite Eq. (14) by grouping K terms at a time as

TABLE I. Methods for calculating the error of the MFPT from the initial state due to sampling. The columns give the distribution from which to sample transition probabilities, and the rows give the method by which to solve the set of linear equations. Each cell gives the running time and advantages for the method.

	Dirichlet distribution	Multivariate normal approximation
Linear algebra	**Method 1** -Running time $N\left(8K^2Q+\frac{1}{3}K^3\right)=O(NK^3)$ -No assumptions	**Method 2** -Running time $K^2+N\left(K^2Q+K^2+\frac{1}{3}K^3\right)=O(NK^3)$ -Assumes central limit theorem holds -Permits negative transition probabilities
Taylor series approximation	**Method 3** -Running time $\frac{1}{3}K^3+K^2+N(8K^2Q+K^2)$ $=O(K^3+NK^2)$ -Ignores higher-order terms in Taylor series	**Method 4** -Running time $\frac{1}{3}K^3+K^2+K^2+N(K^2Q+K^2+K^2)$ $=O(K^3+NK^2)$ -Assumes central limit theorem holds -Ignores higher-order terms in Taylor series

$$\Delta x_1 = \left[\left.\frac{\partial f}{\partial a_{11}}\right|_{\bar{\mathbf{A}}} \cdots \left.\frac{\partial f}{\partial a_{1K}}\right|_{\bar{\mathbf{A}}}\right]\begin{bmatrix}\Delta a_{11}\\ \vdots \\ \Delta a_{1K}\end{bmatrix} + \cdots + \left[\left.\frac{\partial f}{\partial a_{K1}}\right|_{\bar{\mathbf{A}}} \cdots \left.\frac{\partial f}{\partial a_{KK}}\right|_{\bar{\mathbf{A}}}\right]\begin{bmatrix}\Delta a_{K1}\\ \vdots \\ \Delta a_{KK}\end{bmatrix}. \quad (15)$$

For notational convenience, we define the above vectors as the sensitivity $\mathbf{s_i}$ and deviation $\Delta\mathbf{a_i}$,

$$\mathbf{s_i^T} = \left[\left.\frac{\partial f}{\partial a_{i1}}\right|_{\bar{\mathbf{A}}} \cdots \left.\frac{\partial f}{\partial a_{iK}}\right|_{\bar{\mathbf{A}}}\right], \qquad \Delta\mathbf{a_i^T} = [\Delta a_{i1} \cdots \Delta a_{iK}]. \quad (16)$$

Therefore,

$$\Delta x_1 = \sum_{i=1}^{K} \mathbf{s_i^T}\Delta\mathbf{a_i}. \quad (17)$$

The vector $\Delta\mathbf{a_i}$ is equal to $\mathbf{a_i}-\bar{\mathbf{a}}_\mathbf{i}$ and, with the MVN approximation, has mean $\mathbf{0}$ and covariance matrix $\mathbf{\Sigma_i}$ given by Eq. (11). As described in Appendix B, linear combinations of MVN distributions are also MVN distributions, and Eq. (B3) gives that Δx_1 is distributed as normal with mean 0 and variance σ^2, where

$$\sigma^2 = \sum_{i=1}^{K} \mathbf{s_i^T}\mathbf{\Sigma_i}\mathbf{s_i}. \quad (18)$$

Substituting Eq. (11) for $\mathbf{\Sigma_i}$, we see that

$$\sigma^2 = \sum_{i=1}^{K} \frac{1}{w_i^2(w_i+1)}\mathbf{s_i^T}[w_i\,\mathrm{Diag}(\mathbf{u_i}) - \mathbf{u_i}\mathbf{u_i^T}]\mathbf{s_i}$$
$$= \sum_{i=1}^{K} \frac{1}{w_i^2(w_i+1)}[w_i\mathbf{s_i^T}\,\mathrm{Diag}(\mathbf{u_i})\mathbf{s_i} - (\mathbf{s_i^T}\mathbf{u_i})(\mathbf{u_i^T}\mathbf{s_i})]. \quad (19)$$

Therefore, x_1, which equals $\bar{x}_1+\Delta x_1$, has a normal distribution with mean $\bar{x}_1$ and variance given by Eq. (19). These closed-form expressions for the mean and variance of x_1 give the closed-form distribution of x_1.

In addition, the distribution of x_1 can be computed efficiently. As described in Sec. II C 4, we can calculate $\bar{x}_1$ and all the partial derivative terms in the sensitivity vectors in time $O((1/3)K^3+3K^2)$. Since the variance is the sum of vector dot products (rather than matrix vector products), we can calculate the sum in Eq. (19) in time $O(K^2)$. The running time for calculating the distribution of x_1 is thus $O((1/3)K^3+4K^2)$, a large improvement over the running time for any of the sampling-based methods.

E. Adaptive sampling algorithm

To generate molecular-dynamics data, the typical method is either to start all simulations from one state or to generate a representative set of starting conformations, for example, from high-temperature unfolding[13] or replica exchange,[14] and start a number of simulations from each of these conformations. If our trajectories are sufficiently long or we have enough trajectories from relevant conformations, we can hope to sample all the important transitions. In the framework of a Markovian state model with the state space defined, we are sampling transitions with no guidance from which ones are more or less uncertain given our current data.

In this section, we present an algorithm that uses the error analysis techniques described in the previous sections to achieve much higher precision in the quantities of interest for the same number of total simulations. In addition to the total error in the values, which we have already discussed, we also want to determine the main contributors to this error so that we can selectively add simulations to those regions that give rise to the greatest uncertainties. One advantage of the Taylor series methods (sampling-based methods 3 and 4 and the non-sampling-based method) outlined above is that they naturally decompose the contribution of each element in the matrix to the variation in x_1. Each row of $\mathbf{A}$ corresponds to transitions from a single state, so if we find that one row contributes the most to the uncertainty in x_1, we can decrease the uncertainty from that row by generating new transitions from that state.

In principle, we could also find the main error contributors directly from the set of linear equations using techniques such as analysis of variance and statistical design of experiments.[15] However, these are computationally expensive when the problem dimension is high. In this section, we will focus only on the non-sampling-based method for calculating the error. First, we will show how to minimize the variance given the actual transition probability matrix. In practice, we do not know this matrix, but it will be useful to compare the variance achieved by different sampling algorithms to this "optimal" variance.

Assume that we know the actual transition probability matrix $\mathbf{P}^*$. With a total of M simulations, we can calculate the optimal allocation of simulations per row that minimizes the variance in the MFPT from the initial state. If we allocate w_i simulations to row i, the expected counts for that row are $u_{ij}^*=p_{ij}^* w_i$. Substituting into Eq. (19), the variance of x_1 is

$$\sigma^2 = \sum_{i=1}^{K} \frac{\nu_i^*}{w_i+1},$$
$$\nu_i^* = \frac{1}{w_i^2}\mathbf{s_i^T}[w_i\,\mathrm{Diag}(\mathbf{u_i^*}) - \mathbf{u_i^*}\mathbf{u_i^{*T}}]\mathbf{s_i} = \mathbf{s_i^T}[\mathrm{Diag}(\mathbf{p_i^*}) - \mathbf{p_i^*}\mathbf{p_i^{*T}}]\mathbf{s_i}, \tag{20}$$

where we separate out the ν_i^* terms which do not depend on the allocation of simulations, w_i.

We can minimize the quantity σ^2 with respect to the variables w_i subject to the constraint that the total number of simulations is equal to M. Solving this minimization problem gives

$$w_i = \frac{(M+K)\sqrt{\nu_i}}{\sum_{j=1}^{K}\sqrt{\nu_j}} - 1. \tag{21}$$

If we know the transition probability matrix and make the MVN and Taylor series approximations, Eq. (21) gives the optimal number of simulations per row that minimizes the variance of the MFPT from the initial state. Strictly speaking, the variables w_i should be constrained to be positive integers. However, for large sample size M, we can round the values to get good approximations.

However, in general, we do not know the transition probability matrix. We now outline an adaptive sampling algorithm that attempts to approximate the optimal number of simulations per row. Assume that we have observed some transitions and have the Dirichlet transition counts of u_{ij}. From Eq. (19), the variance of x_1 is

$$\sigma^2 = \sum_{i=1}^{K} \frac{\bar{\nu}_i}{w_i+1},$$
$$\bar{\nu}_i = \mathbf{s_i^T}[\mathrm{Diag}(\bar{\mathbf{p}}_\mathbf{i}) - \bar{\mathbf{p}}_\mathbf{i}\bar{\mathbf{p}}_\mathbf{i}^\mathbf{T}]\mathbf{s_i}, \tag{22}$$

since $\bar{p}_{ij}=u_{ij}/w_i$. This equation is similar to Eq. (20), but we have replaced the actual values of the transition probabilities, $\mathbf{p_i^*}$, with the expected values of the transition probabilities, $\bar{\mathbf{p}}_\mathbf{i}$.

Our goal is to decrease the variance given a total number of simulations M. Since the expected values of the transition probabilities are just estimates to the actual values, they may change as we add new simulations. Therefore, we will use these estimates to start a few new simulations, reevaluate the expected values, and repeat.

Let us assume that with our current expected value estimates, we can start m more simulations from any states. In the simplest implementation of the adaptive sampling algorithm, we will start all m simulations from the same state j, but we could easily modify this using the analysis above to start a total of m simulations from different states. The expected values of the transition probabilities may change after these m simulations, but our best guess for them are the current expected values. Therefore, the only term in Eq. (22) that changes with these additional simulations is the term corresponding to the jth row. The expected change of variance in x_1 is

$$\Delta\sigma^2 = \frac{\bar{\nu}_j}{w_j+m+1} - \frac{\bar{\nu}_j}{w_j+1}, \tag{23}$$

which is simply the difference of the jth term with m additional simulations and the original jth term. Using the above equation, we calculate the expected decrease in variance caused by adding m more simulations to any specified row, select the row that reduces the variance the most, and start m more simulations from that state. Repeating this process, we adaptively add samples to our transition counts.

The adaptive sampling algorithm is therefore as follows:

```
(1) Generate initial simulations and transition counts.
(2) While (some tolerance criteria)
(3)     $\bar{\nu}_i \leftarrow \mathbf{s_i^T}[\mathrm{Diag}(\bar{\mathbf{p}}_\mathbf{i}) - \bar{\mathbf{p}}_\mathbf{i}\bar{\mathbf{p}}_\mathbf{i}^\mathbf{T}]\mathbf{s_i}$,
(4)     $best \leftarrow \arg\max_j([\bar{\nu}_j/(w_j+1)] - [\bar{\nu}_j/(w_j+m+1)])$.
(5)     Start $m$ more simulations from state $best$.
```

The tolerance criteria for the *while* loop could be that the total number of simulations is less than some maximum (as we motivated above), the total variance σ^2 is larger than some tolerance, or the decrease in σ^2 is larger than some value. Using this algorithm, we either decrease the total variance with the same number of simulations or decrease the number of simulations necessary for a given precision.

F. Extension to large systems

As the simulated system becomes large, or if we include spatial degrees of freedom in the MSM, the number of states required may become unwieldy. In these cases, both the storage requirements and the cost associated with the linear algebra of Eq. (2) become prohibitive.

In general, since we are looking at molecular dynamics at a small time step, we will not see transitions between all states. We only expect to see transitions between states that are sufficiently close conformationally to move between each other in a small amount of time. Therefore, we expect that the observed transition counts z_{ij} will be sparse, i.e., most of them will equal zero. However, the Dirichlet distribution of the transition probabilities also depends on the prior probability distribution, which may not be sparse. In this section, we will describe how to maintain the sparsity of the transition counts in our calculations, even with a dense prior, since sparse matrix calculations are much more efficient in terms of both storage and computation.

First we show how to decompose the transition probabilities into a dense term and a sparse term and how to write this as a bordered sparse matrix.[16] Then, we show how to efficiently solve this system using sparse matrix techniques. We also discuss the implementation of adaptive sampling algorithm using similar techniques.

We will focus first on solving Eq. (2) at the expected transition probability values $\bar{p}_{ij}$. In our system, the matrix $\mathbf{Z}$, with elements z_{ij}, is sparse. Assume that the prior probabilities α_{ij} are symmetric for each row.

$$\begin{aligned} &\alpha_{11} = \alpha_{12} = \cdots = \alpha_{1K} = c_1, \\ &\alpha_{21} = \alpha_{22} = \cdots = \alpha_{2K} = c_2, \\ &\cdots, \\ &\alpha_{K1} = \alpha_{K2} = \cdots = \alpha_{KK} = c_K. \end{aligned} \tag{24}$$

We can represent this prior compactly as the product of two vectors,

$$\boldsymbol{\alpha} = \mathbf{c}\mathbf{1}^{\mathbf{T}}, \tag{25}$$

where $\mathbf{1}$ is a column vector with all unit entries. Using Eq. (9), the expected values of the transition probabilities are

$$\bar{\mathbf{P}} = \mathbf{W}^{-1}(\mathbf{Z} + \boldsymbol{\alpha}), \tag{26}$$

where $\mathbf{W}$ is a diagonal matrix with entries w_i along the diagonal. Equation (2) can be rewritten as

$$\begin{aligned} &(-\mathbf{I} + \bar{\mathbf{P}})\mathbf{x} = \mathbf{b}, \\ &(-\mathbf{I} + \mathbf{W}^{-1}(\mathbf{Z} + \mathbf{c}\mathbf{1}^{\mathbf{T}}))\mathbf{x} = \mathbf{b}, \end{aligned} \tag{27}$$

where $\mathbf{I}$ is the identity matrix and $\mathbf{Z}$ is sparse. Technically, we need the last row of the matrix to correspond to the boundary condition. This does not change the sparse structure of the matrix, so we will ignore it for notational simplicity.

The matrix in Eq. (27) is generally dense. However, noting that it is a rank-one update of a matrix with sparse structure $\mathbf{Z}$, simple algebra gives the augmented system

$$\begin{bmatrix} \mathbf{F} & \mathbf{c} \\ \mathbf{1}^{\mathbf{T}} & -1 \end{bmatrix} \begin{bmatrix} \mathbf{x} \\ y \end{bmatrix} = \begin{bmatrix} \mathbf{W}\mathbf{b} \\ 0 \end{bmatrix}, \tag{28}$$

where

$$\mathbf{F} = \mathbf{Z} - \mathbf{W} \tag{29}$$

and is a sparse matrix. We have thus shown how to rewrite Eq. (2) as a bordered sparse matrix.

Using standard $\mathbf{LU}$ decomposition to solve the system of equations in Eq. (28) takes time $O((1/3)K^3)$. However, using sparse matrix techniques, it is possible to store only the nonzero entries and solve for the $\mathbf{LU}$ factors of the matrix $\mathbf{F}$ efficiently, where both $\mathbf{L}$ and $\mathbf{U}$ are sparse.[17] The complexity of solving the sparse matrix is very system dependent; however, the running time is much less than $O((1/3)K^3)$. Appendix D shows how to efficiently solve the system given in Eq. (28) by using the $\mathbf{LU}$ factors of $\mathbf{F}$. Thus, we can leverage sparse matrix algorithms even with a dense prior.

Recall that this decomposition is possible because the expected values of the transition probabilities can be separated into a sparse term and a dense term. We cannot use these sparse matrix schemes for methods 1 and 2 outlined above. In those cases, we need to solve the system of linear equations after generating a sample of the transition probabilities, which is unlikely to be the sum of a sparse term and a low-rank dense term. However, we can use these schemes for the Taylor series methods, since they rely on solving Eq. (2) at the expected values of the transition probabilities $\bar{p}_{ij}$, which is what we have outlined above. This reduces the time for finding $\bar{x}_1 = f(\bar{\mathbf{A}})$ and the partial derivative terms from $O(K^3)$ to $O(K^2+\text{sparse matrix time})$. In particular, this reduces the running time of the non-sampling-based method to $O(K^2+\text{sparse matrix time})$.

The above discussion assumes a symmetric prior, but we can generalize these results to any prior that is the outer product of two vectors. In addition to using sparse matrices for the error analysis, we can also use these techniques for the adaptive sampling algorithm. Since each iteration of the adaptive sampling algorithm only adds simulations from a single state i, only the ith row of the matrix $\mathbf{F}$, as defined by Eq. (29), is updated in this iteration. We can represent this change as a rank-one update to $\mathbf{F}$ and use the previously described techniques for converting to a bordered matrix to reuse the $\mathbf{LU}$ factors and reduce the computation time. After some number of adaptive iterations, it will be worthwhile to add the updated counts to the $\mathbf{F}$ matrix and re-factor this matrix, since each update increases the size of the system by one and the updated rows and columns of the factors are generally dense.

III. RESULTS

The error analysis methods presented in this paper assume that the defined state space is Markovian. For molecular systems, it is difficult to define states that meet this criterion. Though there are tests for Markovian behavior in a system,[3] it is unclear whether these tests are both necessary and sufficient. Therefore, since we have assumed that the state space is Markovian in order to calculate the error from sampling, we test the methods given above on a toy system with 87 states and Markovian transitions between the states.

We want the transition probabilities to be representative of molecular kinetics, which random transition probabilities are not. Therefore, we construct the transition probabilities of the toy system p^*_{ij} from existing simulation data of a small protein, the 12-residue tryptophan zipper beta hairpin, TZ2.[18] We took a subset of 1750 independent molecular-dynamics trajectories generated by Snow *et al.*[19] which were started from the unfolded state and taken at a resolution of 10 ns (a total of approximately 12 000 conformations). These conformations were then clustered using hierarchical clustering with a cutoff of 3.25 Å to result in a total of 87 states.[2]

We define the transition count z_{ij} as the sum over all trajectories of the number of transitions from state i to state j at a time step of 10 ns. We define the transition probability matrix $\mathbf{P}^*$ of our toy system as the expected transition probabilities as defined by Eq. (9),

$$\mathbf{P}^* = \begin{bmatrix} \frac{u_{11}}{w_1} & \cdots & \frac{u_{1K}}{w_1} \\ \vdots & \ddots & \\ \frac{u_{K1}}{w_K} & & \frac{u_{KK}}{w_K} \end{bmatrix}, \tag{30}$$

where $u_{ij}=\alpha_{ij}+z_{ij}$, we use a symmetric Dirichlet prior of $\alpha_{i1}=\alpha_{i2}=\cdots=\alpha_{iK}=1/K$, and w_i are the normalization constants. Since this cluster space is not Markovian on the time scales of the source trajectories, the transition matrix does not represent a Markovian model for protein folding and thus we will not use the analysis presented below to draw conclusions about the protein system. However, there certainly exists a Markovian model with these transition probabilities, and thus we can use this matrix in our analysis as long as we restrict our comments to the nature of the error analysis and sampling, which is the goal of this work. The results presented below are on the toy Markovian system with 87 states, transition probabilities given by Eq. (30), and a time step Δt of 10 ns.

A. Demonstration of method 1

Given a transition probability matrix $\mathbf{P}^*$, we can calculate the MFPT from the initial state, x^*_1, using Eq. (2). In this section we will demonstrate that if we sample transitions from the matrix $\mathbf{P}^*$ and use method 1 on these transition counts, the distribution of x_1 which we calculate is a good approximation to x^*_1.

For our toy system, the transition probability matrix $\mathbf{P}^*$ is given by Eq. (30). We sample transitions from this matrix by first selecting a row i and then choosing a transition j, each with probability p^*_{ij}. We generate transition counts by sampling transitions from each row of $\mathbf{P}^*$ independently and summing the number of transitions from state i to state j. For each of these transition counts, we use method 1 to estimate

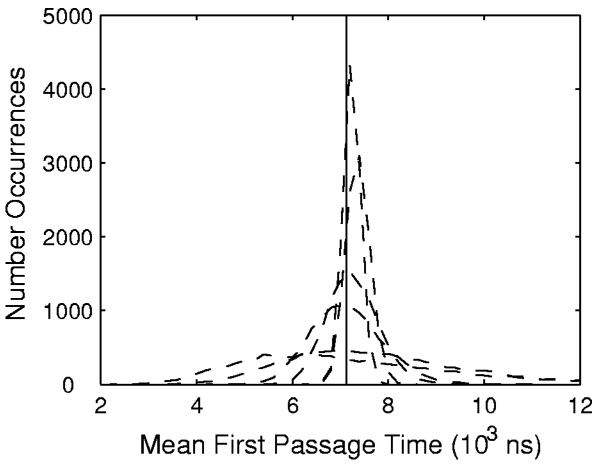

FIG. 1. Distributions of mean first passage time as generated by method 1. The solid line shows the true value of the mean first passage time for the toy system given by the transition probability matrix $\mathbf{P}^*$. The dotted distributions were generated using method 1 for transition count matrices sampled from $\mathbf{P}^*$ with total numbers of 500, 1000, 5000, 10 000, 50 000, or 100 000 samples per row.

the distribution of x_1. We have taken 10 000 independent samples of possible transition probability matrices and solved each for x_1. Figure 1 shows the actual value of x^*_1 for the matrix $\mathbf{P}^*$ as well as the distributions of x_1 for six different transition count matrices, generated with either 500, 1000, 5000, 10 000, 50 000, or 100 000 independent transition samples per row.

It is easy to verify that the actual value x^*_1 falls within each distribution. Also, as the number of transition samples increases, the distribution of x_1 becomes narrower and centers around the actual value, x^*_1. While we have only shown one distribution for each number of samples per row in the figure, these results are typical. Also, we repeated these experiments for random transition probability matrices $\mathbf{P}^*$ and found similar results (data not shown).

B. Validity of approximations

We have demonstrated above that given transition counts, we can calculate the distribution of x_1 by sampling from possible transition matrices that could have produced our observed data and solving the system of linear equations for each sample. But, as described in Sec. II, this procedure is computationally expensive. Therefore, we proposed two approximations, the MVN approximation to the Dirichlet distribution and the Taylor series approximation to the set of linear equations, which when taken together give an efficient closed-form approximation to the distribution of x_1. We now demonstrate the validity of these approximations.

First, we generated transition counts from 2000 independent samples per row of the toy system as described above. We then ran methods 1–4, as well as the non-sampling-based method to calculate the distribution of x_1 from these transition counts. For each of methods 1–4, we used 10 000 independent samples of transition probability matrices. Figure 2 shows the resulting distributions of x_1 for each of the four sampling-based methods, as well as the density for the non-sampling-based method.

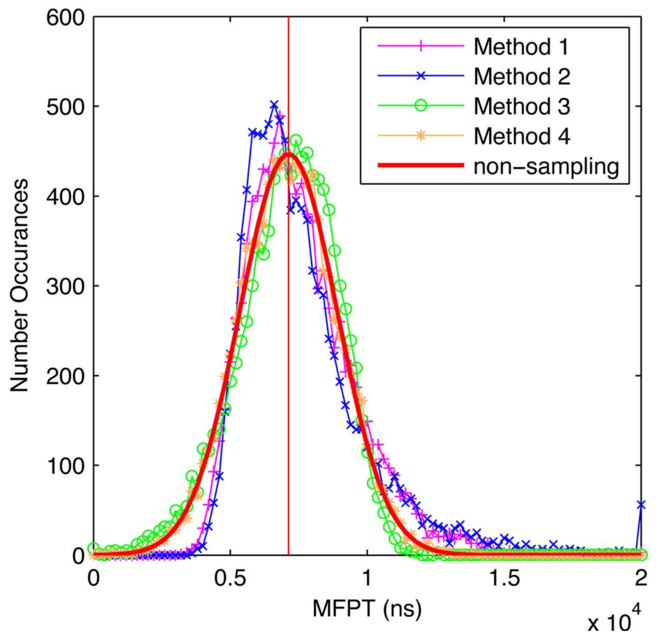

FIG. 2. Distribution of x_1 as calculated by the five methods outlined above. The vertical line indicates the mean first passage time at the expected values of the transition probabilities, $\bar{x}_1$.

Methods 1 and 2 and methods 3 and 4 overlay almost exactly, showing that the MVN approximation to the Dirichlet distribution is valid. Between the linear equation methods (1 and 2) and the Taylor series methods (3 and 4), there is a slight difference since the Taylor series method ignores higher-order terms. However, this difference is mostly in the tails of the distributions and is minimal if one only cares about the mean and variance of the distribution (Table II). In addition, it is clear that the non-sampling-based method overlays sampling-based method 4 exactly, which is expected since they solve the same approximations to the problem.

For this example, we compared the running times of the various methods. The code was implemented in MATLAB and run on a Dual Athelon MP 2200+ (1.8 GHz) computer. Table III gives the running times for the five different methods required to generate the histograms shown in Fig. 2. While we did not fully optimize the code for sampling from the Dirichlet and MVN distributions and did not yet implement the sparse matrix solver, these running times clearly demonstrate the superiority of the non-sampling-based method for error analysis. These tests were repeated on random matrices

TABLE II. Means and standard deviations of the distributions generated for different methods (all units are in nanoseconds). All methods show reasonable agreement for these quantities, but differ between the linear equation methods and the Taylor series methods.

	Mean	Standard deviation
Method 1	6178	973
Method 2	6199	1023
Method 3	6026	940
Method 4	6041	930
Nonsampling	6044	931

TABLE III. Running times for the different methods on an 87 state example with 10 000 independent samples of the transition probabilities for the sampling-based methods 1–4.

	Preprocessing	Sampling	Solving	Total for 10 000 samples
Method 1	NA	4164 s	14.5 s	4178.50 s
Method 2	3.05 s	273 s	15.2 s	291.25 s
Method 3	NA	4164 s	4.6 s	4168.60 s
Method 4	3.05 s	273 s	5.1 s	281.15 s
Nonsampling	0.05 s	NA	0.02 s	0.07 s

and the toy system with varying levels of total transitions per row with similar results (data not shown). In addition, we tried different prior probability distributions and found no noticeable change in the results for small priors.

C. Adaptive sampling

Our goals for this work were to both calculate the error in MFPT as well as use this error to improve the results. Above, we have shown how to efficiently calculate the error in MFPT, and we have demonstrated that the approximations we made were reasonable. Now, we show how using these error estimates improves the sampling through an adaptive algorithm. We demonstrate how the adaptive sampling algorithm compares to both an even allocation of samples and the optimal allocation of samples.

Assume we can take m transition samples in each round, we can decide where to allocate the samples before each round, and that we have a limit on the total number of samples. An even sampling algorithm will always take the same number of samples from each state in each round, m/K. In the simplest implementation of the adaptive sampling algorithm, we calculate the contribution to the variance of x_1 for each row and add all m samples to the row that will decrease the variance the most.

We ran both the even and adaptive sampling algorithms by generating samples from the matrix $\mathbf{P}^*$. We started with the Dirichlet prior of $\alpha_{ij}=1/K$ and an initial 10 samples per row and added 870 samples in each round until we had a maximum of 500 000 samples. Figure 3 shows the variance of x_1 versus the total number of samples over 20 independent runs of each algorithm, with the dark gray points from the even sampling algorithm and the light gray points from the adaptive sampling algorithm. The solid gray lines on the figure represent the variance in x_1 for the even and adaptive sampling schemes generated by keeping the transition counts proportional to $\mathbf{P}^*$. The black line on the figure is the variance of x_1 when the samples are distributed optimally, as described in Sec. II E. These solid lines were generated by scaling each row in the matrix by the desired number of samples for that row. We can see that the adaptive sampling algorithm rapidly achieves the optimal variance.

It is clear that the adaptive sampling algorithm achieves either a much higher precision in MFPT with the same number of samples or, conversely, requires many less samples to achieve a given precision. Figure 4 shows these relations and demonstrates that the adaptive sampling algorithm, for this

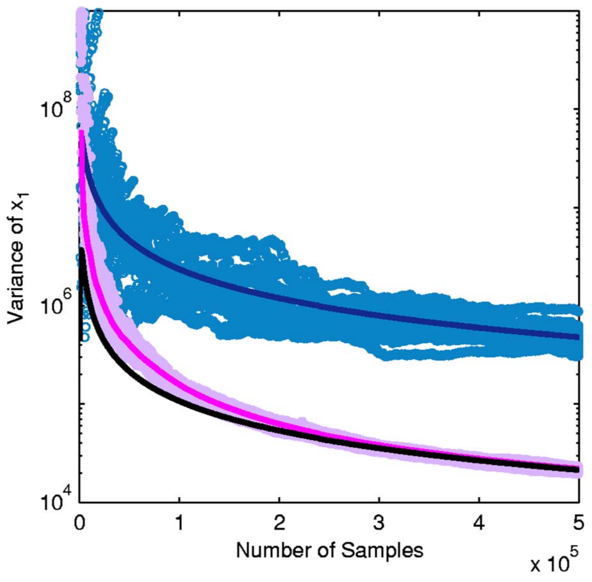

FIG. 3. Effect of adaptive sampling on the variance of x_1. The dark gray points are the variance generated from the even sampling algorithms and the light gray points are the variance from the adaptive sampling algorithm. The black line shows the variance when the sampling is distributed optimally.

data set, achieved a greater than 20-fold reduction in the number of samples or increase in precision. If we look at the optimal allocation of samples per row, we see that the distribution is far from uniform across the states (Fig. 5). Again, this algorithm was tested on random matrices with similar results (data not shown).

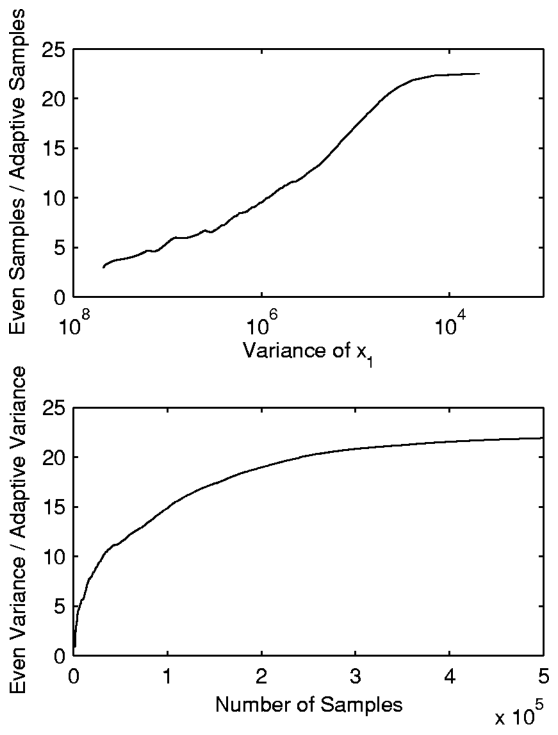

FIG. 4. Relationship between the number of samples required and the variance for the even and adaptive sampling algorithms. The first panel shows the ratio of the number of even samples to the number of adaptive samples required for a desired variance. The second panel shows the ratio of the variance of the even sampling to the variance of the adaptive sampling for the same number of samples.

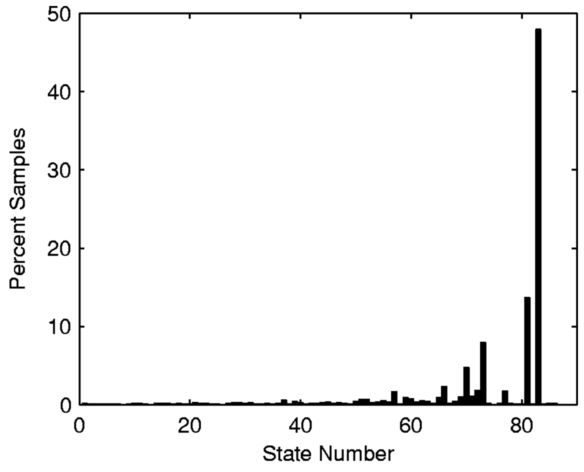

FIG. 5. Percent of samples required for each state in the optimal allocation of samples per state.

IV. DISCUSSION AND CONCLUSION

Given that we can generate a large number of molecular-dynamics trajectories using distributed computing methods, such as Folding@Home, it is important to develop efficient techniques for analyzing the data. In previous work, we described a technique for building a graph of the important states of a protein and estimating transition probabilities between these states. In this work, we discuss methods for ascertaining the uncertainties in important kinetic properties that can be calculated from this graph.

A main contribution of this work is in the computation of error from finite sampling. Given that a state definition is Markovian, we have shown that the distribution of transition probabilities, estimated from molecular-dynamics data, is Dirichlet if one assumes a Dirichlet prior. From these distributions, we gave a method which, given sufficient samples of transition probabilities, can calculate the distribution of a desired quantity, such as the mean first passage time. We also presented and validated two approximations that have little effect on the accuracy of the results and improve the efficiency of the first method. When taken together, these approximations yield an efficient closed-form approximation to the uncertainty, which can be calculated in the same amount of time as *one* solution to the set of linear equations.

While we have not gone into detail in this paper, it should be noted that the analysis presented above could easily be modified to calculate errors in P_{fold} values, stationary distribution, or any other quantity that can be represented by a set of linear equations. Similarly, it is possible to modify the sensitivity analysis to calculate the error of other functions of the MFPT vector, such as the sum of errors or the norm.

The analysis presented here assumes that it is possible to define states that behave in a Markovian manner at a given time step. This is not a trivial task, and incorrect state definitions may lead to unpredictable results. In this work, we did not address the errors that may arise from incorrect state definitions. Instead, we cite tests that can be performed on a specific state definition to see whether or not it is Markovian. Some of these tests rely on finding eigenvalues or other properties of transition probability matrices. The methods that we presented here for finding uncertainties in solutions to linear equations can also be generalized to finding uncertainties in eigenvalues. Methods 1 and 2 are trivial; methods 3 and 4 are more complex but can be done.[20] It is important when running these Markovian tests to find the errors from finite sampling, since it may be possible to pass the tests within the error, or we may find that the sampling errors are too large to draw any meaningful conclusions about the Markovian-ness of the system.

For systems that can be reduced to a small number of states, the efficiency gains in the uncertainty estimates are minimal compared to the time it takes to generate the molecular-dynamics data and cluster the conformations. However, systems may have important transitional and rotational degrees of freedom, i.e., two proteins moving in relation to one another or a binding event. In these cases, for each relevant spatial state of the protein, we would need separate states for each conformation of the protein and a MSM with tens of thousands of states may be necessary. The sampling-based methods will hardly be practical for such large systems. Our non-sampling-based, closed-form solution for uncertainty, when taken together with the sparse matrix manipulations given in Sec. II F, would provide efficient ways to measure uncertainties.

In addition to the error analysis techniques, we also presented an adaptive sampling method that can produce a given precision with over an order of magnitude reduction in the number of samples required by a naive sampling algorithm. We outlined how this algorithm can be applied efficiently for systems with many states and demonstrated the large gains in either sampling time or precision. We also outlined sparse matrix techniques that can improve the efficiency of both the error analysis and adaptive sampling.

In conclusion, we have developed efficient and practical computational tools and algorithms that find the main sources of error in a Markovian state model caused by finite sampling. We have shown that our approximate solutions are in good agreement with the actual distributions, and are computationally far more efficient. In addition, we gave an algorithm that uses the error analysis to greatly reduce the number of samples necessary to build a MSM with the same precision. Lastly, we gave techniques for using sparse matrix manipulations that will allow the handling of systems with large numbers of states.

V. FUTURE WORK

For protein systems, it is a significant challenge to define states that meet the Markovian criteria. We are currently working on methods to properly define states and on new tests for Markovian behavior. When these are complete, we plan to apply the error analysis techniques and adaptive sampling algorithm to molecular systems, including alanine dipeptide, TrpZip, and lipid vesicles and address some of the more practical concerns of these methods. For example, while running the adaptive sampling algorithm, it is likely that we will need to recluster the conformations as we gather new data in order to meet the Markovian criteria. Also, if we begin new simulations from a given state, we should pick the starting conformation at random from all the conformations in the state, so that we do not bias the transitions from that state. In addition, we plan to use these uncertainty techniques on systems with many states to demonstrate scalability.

ACKNOWLEDGMENTS

One of the authors (N.S.) was supported through the National Defense Science and Engineering Graduate Fellowship. The computation was supported by grants from NIH and NSF and a gift from Google. This paper has greatly benefited from many useful discussions with K. Singhal.

APPENDIX A: SAMPLING FROM THE DIRICHLET DISTRIBUTION

If $\mathbf{p}$ follows a Dirichlet distribution with parameters $\mathbf{u}$, then

$$\text{Dirichlet}(\mathbf{p};\mathbf{u}) = \frac{1}{Z(\mathbf{u})}\prod_{i=1}^{K} p_i^{u_i-1}, \tag{7'}$$

$$Z(\mathbf{u}) = \frac{\prod_{i=1}^{K}\Gamma(u_i)}{\Gamma(\sum_{i=1}^{K} u_i)}. \tag{A1}$$

We can sample from $\mathbf{p}$ by using the fact that if $Y_1,\ldots,Y_K$ are independent gamma random variables with parameters $u_i > 0$, respectively, and $Y_0=\Sigma_{i=1}^{K}Y_i$, then $p_1=Y_1/Y_0,\ldots,p_K=Y_K/Y_0$ are distributed by the Dirichlet distribution with parameters $u_1,u_2,\ldots,u_K$.[21] There are many algorithms to sample from a gamma distribution. We use Best's rejection algorithm for parameters greater than one (1978) and Best's RGS algorithm (1983) for parameters less than or equal to one.[21]

Each sample from a Dirichlet distribution needs K gamma distribution samples. For the gamma sampling algorithms we chose, each gamma sample requires two samples from normal distributions and has a rejection constant of less than 4. We ignore the time taken for mathematical functions such as exponentiation and square roots. Therefore, the maximum expected time to generate one gamma sample is $O(8Q)$, and the expected time to generate one Dirichlet sample is $O(8KQ)$, where Q is the time taken to sample from a normal distribution.

APPENDIX B: SAMPLING FROM THE MULTIVARIATE NORMAL DISTRIBUTION

To sample from $\text{MVN}(\boldsymbol{\mu},\boldsymbol{\Sigma})$, we use the fact that if we have a vector

$$\mathbf{y} = \text{MVN}(\boldsymbol{\mu},\boldsymbol{\Sigma}) \tag{B1}$$

and we define a new vector

$$\mathbf{y}' = \mathbf{R}\mathbf{y} + \mathbf{b}, \tag{B2}$$

then $\mathbf{y}'$ is distributed as[11]

$$\mathbf{y}' \approx \mathrm{MVN}(\mathbf{R}\boldsymbol{\mu}+\mathbf{b},\mathbf{R}\boldsymbol{\Sigma}\mathbf{R}^{\mathbf{T}}). \tag{B3}$$

So, to generate samples from $\mathrm{MVN}(\boldsymbol{\mu}_{\mathbf{i}},\boldsymbol{\Sigma}_{\mathbf{i}})$, we first generate a sample

$$\mathbf{y} \approx \mathrm{MVN}(\mathbf{0},\mathbf{I}) \tag{B4}$$

and then calculate

$$\mathbf{p}_{\mathbf{i}} = \mathbf{L}_{\mathbf{i}}\mathbf{y} + \boldsymbol{\mu}_{\mathbf{i}}, \tag{B5}$$

$$\mathbf{L}_{\mathbf{i}}\mathbf{L}_{\mathbf{i}}^{\mathbf{T}} = \boldsymbol{\Sigma}_{\mathbf{i}}. \tag{B6}$$

There will be an initial cost of $(1/6)K^3$ to perform the decomposition given in Eq. (B6). Then, for each desired sample, we need K independent samples from the normal distribution $N(0,1)$, and K^2 multiplications in Eq. (B5). Thus, the total time per sample is $O(KQ+K^2)$, where Q is the time to sample from the normal distribution.

However, the covariance matrices of the MVN distributions in which we are interested are structured; they are rank-one updates of diagonal matrices [Eq. (11)]. Therefore, it is possible to implicitly calculate the matrix $\mathbf{L}_{\mathbf{i}}$ in Eq. (B6) in time $O(K)$, and to perform the multiplication in Eq. (B5) in time $O(K)$ using slight modifications of the methods described by Gill *et al.*[22] Thus, the revised total time per sample is $O(KQ+K)$.

APPENDIX C: SENSITIVITY ANALYSIS

In this appendix we show how the terms in the Taylor series expansion in Eq. (14) can be computed efficiently using adjoint systems. For details, see Vlach and Singhal.[20]

Consider a system of linear equations of the form

$$\mathbf{A}\mathbf{x} = \mathbf{b}. \tag{C1}$$

We are interested in the sensitivity of $\mathbf{x}$ to small perturbations in the elements of $\mathbf{A}$, a_{ij}. Let us differentiate each side of the equation with respect to a_{ij}:

$$\frac{\partial \mathbf{A}}{\partial a_{ij}}\mathbf{x} + \mathbf{A}\frac{\partial \mathbf{x}}{\partial a_{ij}} = \mathbf{0}. \tag{C2}$$

The derivative of the $\mathbf{A}$ matrix with respect to one of the entries is simply a matrix with a unit value at position (i,j) and zeros elsewhere. This is compactly written as

$$\frac{\partial \mathbf{A}}{\partial a_{ij}} = \mathbf{e}_{\mathbf{i}}\mathbf{e}_{\mathbf{j}}^{\mathbf{T}}, \tag{C3}$$

where $\mathbf{e}_{\mathbf{i}}$ is the basis vector given by the ith column of the identity matrix. Thus, substituting Eq. (C3) into Eq. (C2) gives

$$\frac{\partial \mathbf{x}}{\partial a_{ij}} = -\mathbf{A}^{-1}\mathbf{e}_{\mathbf{i}}\mathbf{e}_{\mathbf{j}}^{\mathbf{T}}\mathbf{x}. \tag{C4}$$

For our application, we are only interested in the sensitivity of the first element of $\mathbf{x}$, x_1.

$$\frac{\partial x_1}{\partial a_{ij}} = \mathbf{e}_{\mathbf{1}}^{\mathbf{T}}\frac{\partial \mathbf{x}}{\partial a_{ij}} = -\mathbf{e}_{\mathbf{1}}^{\mathbf{T}}\mathbf{A}^{-1}\mathbf{e}_{\mathbf{i}}\mathbf{e}_{\mathbf{j}}^{\mathbf{T}}\mathbf{x}. \tag{C5}$$

Let us define the column vector $\boldsymbol{\varphi}$ as

$$\boldsymbol{\varphi}^{\mathbf{T}} = \mathbf{e}_{\mathbf{1}}^{\mathbf{T}}\mathbf{A}^{-1}. \tag{C6}$$

Substituting Eq. (C6) into Eq. (C5) gives

$$\frac{\partial x_1}{\partial a_{ij}} = -\boldsymbol{\varphi}^{\mathbf{T}}\mathbf{e}_{\mathbf{i}}\mathbf{e}_{\mathbf{j}}^{\mathbf{T}}\mathbf{x} = -\varphi_i x_j. \tag{C7}$$

The vector $\boldsymbol{\varphi}$ is called the adjoint vector and is the solution of the *adjoint* system of equations

$$\mathbf{A}^{\mathbf{T}}\boldsymbol{\varphi} = \mathbf{e}_{\mathbf{1}}. \tag{C8}$$

We wish to evaluate the $\partial x_1/\partial a_{ij}$ terms at the matrix $\bar{\mathbf{A}}$, which corresponds to the expected values of the parameters. Determination of the vectors $\bar{\mathbf{x}} = \mathbf{x}|_{\bar{\mathbf{A}}}$ and $\bar{\boldsymbol{\varphi}} = \boldsymbol{\varphi}|_{\bar{\mathbf{A}}}$ involves solving the sets of linear equations,

$$\bar{\mathbf{A}}\bar{\mathbf{x}} = \mathbf{b}, \tag{C9}$$

$$\bar{\mathbf{A}}^{\mathbf{T}}\bar{\boldsymbol{\varphi}} = \mathbf{e}_{\mathbf{1}}. \tag{C10}$$

We can then find all the $\partial x_1/\partial a_{ij}|_{\bar{\mathbf{A}}}$ terms by simply taking the outer product of $\bar{\boldsymbol{\varphi}}$ and $\bar{\mathbf{x}}$: $\bar{\boldsymbol{\varphi}}\bar{\mathbf{x}}^{\mathbf{T}}$.

We can calculate the vector $\bar{\mathbf{x}}$ by solving Eq. (C9) in $O((1/3)K^3)$ operations by factoring $\bar{\mathbf{A}}$ into its $\mathbf{LU}$ factors.[12] The solution of the adjoint system in Eq. (C10) only requires $O(K^2)$ operations as the same $\mathbf{LU}$ factors can be used. All the terms $\partial x_1/\partial a_{ij}|_{\bar{\mathbf{A}}}$ can be computed from these two solutions in $O(K^2)$ operations.

APPENDIX D: SOLVING A BORDERED SPARSE MATRIX

In this section, we show how to efficiently solve the system of linear equations

$$\begin{bmatrix} \mathbf{F} & \mathbf{c} \\ \mathbf{1}^{\mathbf{T}} & -1 \end{bmatrix}\begin{bmatrix} \mathbf{x} \\ y \end{bmatrix} = \begin{bmatrix} \mathbf{W}\mathbf{b} \\ 0 \end{bmatrix}, \tag{28$'$}$$

where $\mathbf{F}$ is a sparse matrix by using the $\mathbf{LU}$ factors of $\mathbf{F}$, $\mathbf{L}_{\mathbf{F}}$ and $\mathbf{U}_{\mathbf{F}}$, to find the $\mathbf{LU}$ factors of the matrix on the left-hand side of Eq. (28), which we will name $\mathbf{G}$.

We are looking for factors $\mathbf{L}_{\mathbf{G}}$ and $\mathbf{U}_{\mathbf{G}}$ such that

$$\underbrace{\begin{bmatrix} \mathbf{L}_{\mathbf{11}} & 0 \\ \mathbf{l}_{\mathbf{21}} & l_{22} \end{bmatrix}}_{\mathbf{L}_{\mathbf{G}}}\underbrace{\begin{bmatrix} \mathbf{U}_{\mathbf{11}} & \mathbf{u}_{\mathbf{12}} \\ 0 & 1 \end{bmatrix}}_{\mathbf{U}_{\mathbf{G}}} = \underbrace{\begin{bmatrix} \mathbf{F} & \mathbf{c} \\ \mathbf{1}^{\mathbf{T}} & -1 \end{bmatrix}}_{\mathbf{G}}, \tag{D1}$$

where we place zeros and ones such that $\mathbf{L}_{\mathbf{G}}$ and $\mathbf{U}_{\mathbf{G}}$ are lower and upper triangular, respectively, and $\mathbf{U}_{\mathbf{G}}$ has unit entries along the diagonal. We have the following four conditions from Eq. (D1):

$$\begin{aligned} &\mathbf{L}_{\mathbf{11}}\mathbf{U}_{\mathbf{11}} = \mathbf{F}, \\ &\mathbf{L}_{\mathbf{11}}\mathbf{u}_{\mathbf{12}} = \mathbf{c}, \\ &\mathbf{l}_{\mathbf{21}}\mathbf{U}_{\mathbf{11}} = \mathbf{1}^{\mathbf{T}}, \\ &\mathbf{l}_{\mathbf{21}}\mathbf{u}_{\mathbf{12}} + l_{22} = -1. \end{aligned} \tag{D2}$$

To satisfy the first equation, we simply set $\mathbf{L}_{\mathbf{11}}$ and $\mathbf{U}_{\mathbf{11}}$ to the $\mathbf{LU}$ factors of $\mathbf{F}$, $\mathbf{L}_{\mathbf{F}}$, and $\mathbf{U}_{\mathbf{F}}$. The other three equations are then easily solved for $\mathbf{u_{12}}$, $\mathbf{l_{21}}$, and l_{22} via either forward or back substitution. From these **LU** factors, we can efficiently solve Eq. (28) and therefore our original equations for the mean first passage times.

[1] M. Shirts and V. Pande, Science **290**, 7220 (2000).
[2] N. Singhal, C. Snow, and V. S. Pande, J. Chem. Phys. **121**, 415 (2004).
[3] W. C. Swope, J. W. Pitera, and F. Suits, J. Phys. Chem. B **108**, 6571 (2004).
[4] W. C. Swope, J. W. Pitera, F. Suits *et al.*, J. Phys. Chem. B **108**, 6582 (2004).
[5] S. P. Elmer and V. S. Pande, J. Chem. Phys. **121**, 12760 (2004).
[6] S. P. Elmer, S. Park, and V. S. Pande, J. Chem. Phys. **123**, 114902 (2005).
[7] P. Kasson, N. W. Kelley, N. Singhal, M. Vrljic, A. T. Brunger, and V. S. Pande (unpublished).
[8] S. Sriraman, I. G. Kevrekidis, and G. Hummer, J. Phys. Chem. B **109**, 6479 (2005).
[9] N. L. Johnson, S. Kotz, and N. Balakrishnan, *Discrete Multivariate Distributions* (Wiley, New York, 1997).
[10] S. Kotz, N. Balakrishnan, and N. L. Johnson, *Continuous Multivariate Distributions* (Wiley, New York, 2000).
[11] C. R. Rao, *Linear Statistical Inference and Its Applications*, 2nd ed. (Wiley, New York, 1973).
[12] G. H. Golub and C. van Loan, *Matrix Computations*, 3rd ed. (The Johns Hopkins University Press, London, 1996).
[13] V. Daggett and M. Levitt, J. Mol. Biol. **232**, 600 (1993).
[14] Y. Sugita and Y. Okamoto, Chem. Phys. Lett. **314**, 141 (1999).
[15] E. P. Box, W. G. Hunter, and J. S. Hunter, *Statistics for Experimenters: An Introduction to Design, Data Analysis, and Model Building* (Wiley, New York, 1978).
[16] J. R. Bunch and D. J. Rose, J. Math. Anal. Appl. **48**, 574 (1974).
[17] I. S. Duff, A. M. Erisman, and J. K. Reid, *Direct Methods for Sparse Matrices* (Oxford University Press, New York, 1986).
[18] A. G. Cochran, N. J. Skelton, and M. A. Starovasnik, Proc. Natl. Acad. Sci. U.S.A. **98**, 5578 (2001).
[19] C. D. Snow, L. Qui, D. Du, F. Gai, S. J. Hagen, and V. S. Pande, Proc. Natl. Acad. Sci. U.S.A. **101**, 4077 (2004).
[20] J. Vlach and K. Singhal, *Computer Methods for Circuit Analysis and Design* (Van Nostrand Reinhold, New York, 1983).
[21] Luc Devroye, *Non-Uniform Random Variate Generation* (Springer-Verlag, New York, 1986).
[22] P. E. Gill, G. H. Golub, W. Murray, and M. A. Saunders, Math. Comput. **28**, 505 (1974).