RESEARCH ARTICLE | JULY 06 2023

# Inexact iterative numerical linear algebra for neural network-based spectral estimation and rare-event prediction

**Special Collection:** Machine Learning Hits Molecular Simulations

John Strahan; Spencer C. Guo ; Chatipat Lorpaiboon ; Aaron R. Dinner ; Jonathan Weare

*J. Chem. Phys.* 159, 014110 (2023)
https://doi.org/10.1063/5.0151309

# Inexact iterative numerical linear algebra for neural network-based spectral estimation and rare-event prediction

Cite as: J. Chem. Phys. **159**, 014110 (2023); doi: 10.1063/5.0151309
Submitted: 20 March 2023 • Accepted: 2 June 2023 •
Published Online: 6 July 2023

View Online Export Citation CrossMark

John Strahan,[1] Spencer C. Guo,[1] Chatipat Lorpaiboon,[1] Aaron R. Dinner,[1,a)] and Jonathan Weare[2,a)]

## AFFILIATIONS

[1] Department of Chemistry and James Franck Institute, University of Chicago, Chicago, Illinois 60637, USA
[2] Courant Institute of Mathematical Sciences, New York University, New York, New York 10012, USA

**Note:** This paper is part of the JCP Special Topic on Machine Learning Hits Molecular Simulations.
[a)]**Authors to whom correspondence should be addressed:** dinner@uchicago.edu and weare@nyu.edu

## ABSTRACT

Understanding dynamics in complex systems is challenging because there are many degrees of freedom, and those that are most important for describing events of interest are often not obvious. The leading eigenfunctions of the transition operator are useful for visualization, and they can provide an efficient basis for computing statistics, such as the likelihood and average time of events (predictions). Here, we develop inexact iterative linear algebra methods for computing these eigenfunctions (spectral estimation) and making predictions from a dataset of short trajectories sampled at finite intervals. We demonstrate the methods on a low-dimensional model that facilitates visualization and a high-dimensional model of a biomolecular system. Implications for the prediction problem in reinforcement learning are discussed.

## I. INTRODUCTION

Modern observational, experimental, and computational approaches often yield high-dimensional time series data (trajectories) for complex systems. In principle, these trajectories contain rich information about dynamics and, in particular, the infrequent events that are often most consequential. In practice, however, high-dimensional trajectory data are often difficult to parse for useful insight. The need for more efficient statistical analysis tools for trajectory data is critical, especially when the goal is to understand rare-events that may not be well represented in the data.

We consider dynamics that can be treated as Markov processes. A common starting point for statistical analyses of Markov processes is the transition operator, which describes the evolution of function expectations. The eigenfunctions of the transition operator characterize the most slowly decorrelating features (modes) of the system.[1–5] These can be used for dimensionality reduction to obtain a qualitative understanding of the dynamics,[6,7] or they can be used as the starting point for further computations.[8–10] Similarly, prediction functions, which provide information about the likelihood and timing of future events as a function of the current state, are defined through linear equations of the transition operator.[10,11]

A straightforward numerical approach to obtaining these functions is to convert the transition operator to a matrix by projecting onto a finite basis for Galerkin approximation.[1,2,10–15] The performance of such a linear approximation depends on the choice of basis,[10,11,15] and previous work often resorts to a set of indicator functions on a partition of the state space (resulting in a Markov state model or MSM[14]) for lack of a better choice. While Galerkin approximation has yielded many insights,[16,17] the limited expressivity of the basis expansion has stimulated interest in (nonlinear) alternatives.

In particular, artificial neural networks can be harnessed to learn eigenfunctions of the transition operator and prediction functions from data.[5,18–25] However, existing approaches based on neural networks suffer from various drawbacks. As discussed in Ref. 5, their performance can often be very sensitive to hyperparameters, requiring extensive tuning and varying with random initialization. Many use loss functions that are estimated against the stationary distribution,[25–30] so that metastable states contribute most heavily, which negatively impacts performance.[24,30] Assumptions about the dynamics (e.g., microscopic reversibility) limit applicability. In Ref. 24, we introduced an approach that overcomes the issues above, but it uses multiple trajectories from each initial condition; this limits the approach to analysis of simulations and moreover requires specially prepared datasets.

The need to compute prediction functions from observed trajectory data also arises in reinforcement learning. There the goal is to optimize an expected future reward (the prediction function) over a policy (a Markov process). For a fixed Markov process, the prediction problem in reinforcement learning is often solved by temporal difference (TD) methods, which allow the use of arbitrary ensembles of trajectories without knowledge of the details of the underlying dynamics.[31] TD methods have a close relationship with an inexact form of power iteration, which, as we describe, can perform poorly on rare-event related problems.

Motivated by this relationship, as well as by an inexact power iteration scheme previously proposed for approximating the stationary probability distribution of a Markov process using trajectory data,[32] we propose a computational framework for spectral estimation and rare-event prediction based on inexact iterative numerical linear algebra. Our framework includes an inexact Richardson iteration for the prediction problem, as well as an extension to inexact subspace iteration for the prediction and spectral estimation problems. The theoretical properties of exact subspace iteration suggest that eigenfunctions outside the span of the approximation will contribute significantly to the error of our inexact iterative schemes.[33] Consistent with this prediction, we demonstrate that learning additional eigenvalues and eigenfunctions simultaneously through inexact subspace iteration accelerates convergence dramatically relative to inexact Richardson and power iteration in the context of rare events. While we assume the dynamics can be modeled by a Markov process, we do not require knowledge of their form or a specific underlying model. The method shares a number of further advantages with the approach discussed in Ref. 24 without the need for multiple trajectories from each initial condition in the dataset. This opens the door to treating a wide range of observational, experimental, and computational datasets.

The remainder of the paper is organized as follows: In Sec. II, we describe the quantities that we seek to compute in terms of linear operators. In Secs. III and IV, we introduce an inexact subspace iteration algorithm that we use to solve for these quantities. Section V illustrates how the loss function can be tailored to the known properties of the desired quantity. Section VI summarizes the two test systems that we use to illustrate our methods: a two-dimensional potential, for which we can compute accurate reference solutions, and a molecular example that is high-dimensional but still sufficiently tractable that statistics for comparison can be computed from long trajectories. In Sec. VII, we explain the details of the invariant subspace iteration and then demonstrate its application to our two examples. Finally, Sec. VIII details how the subspace iteration can be modified to compute prediction functions and compares the effect of different loss functions, as well as the convergence properties of power iteration and subspace iteration. We conclude with implications for reinforcement learning.

## II. SPECTRAL ESTIMATION AND THE PREDICTION PROBLEM

We have two primary applications in mind in this article. First, we would like to estimate the *dominant eigenfunctions and eigenvalues* of the transition operator $\mathcal{T}^t$ for a Markov process $X^t \in \mathbb{R}^d$, defined as

$$\mathcal{T}^t f(x) = \mathbb{E}_x\big[f(X^t)\big], \tag{1}$$

where $f$ is an arbitrary real-valued function and the subscript $x$ indicates the initial condition $X^0 = x$. The transition operator encodes how expectations of functions evolve in time. The right eigenfunctions of $\mathcal{T}^t$ with the largest eigenvalues characterize the most slowly decorrelating features (modes) of the Markov process.[1,2,4,5]

Our second application is to compute *prediction functions* of the general form

$$u(x) = \mathbb{E}_x\left[\Psi(X^T) + \sum_{t=0}^{T-1}\Gamma(X^t)\right], \tag{2}$$

where $T$ is the first time $X^t \notin D$ from some domain $D$, and $\Psi$ and $\Gamma$ are functions associated with the escape event (rewards in reinforcement learning). Prototypical examples of prediction functions that appear in our numerical results are the mean first passage time (MFPT) from each $x$

$$m(x) = \mathbb{E}_x[T] \tag{3}$$

and the committor

$$q(x) = \mathbb{E}_x\Big[\mathbb{1}_B(X^T)\Big] = \mathbb{P}_x\Big[X^T \in B\Big], \tag{4}$$

where $A$ and $B$ are disjoint sets ("reactant" and "product" states) and $D = (A \cup B)^{\mathsf{C}}$. The MFPT is important for estimating rates of transitions between regions of state space, while the committor can serve as a reaction coordinate[29,34–36] and as a key ingredient in transition path theory statistics.[24,37,38] For any $\tau > 0$, the prediction function $u(x)$ satisfies the linear equation

$$(\mathcal{I} - \mathcal{S}^\tau)u(x) = \mathbb{E}_x\left[\sum_{t=0}^{(\tau\wedge T)-1}\Gamma(X^t)\right] \tag{5}$$

for $x \in D$, with boundary condition

$$u(x) = \Psi(x) \tag{6}$$

for $x \notin D$. In (5), $\mathcal{I}$ is the identity operator and

$$\mathcal{S}^\tau f(x) = \mathbb{E}_x\Big[f(X^{\tau\wedge T})\Big] \tag{7}$$

is the "stopped" transition operator.[10] We use the notation $\tau \wedge T = \min\{\tau, T\}$. The parameter $\tau$ is known as the lag time. While it is an integer corresponding to a number of integration steps, in our numerical examples, we often express it in terms of equivalent time units.

Our specific goal is to solve both the eigenproblem and the prediction problem for $X^t$ in high dimensions and without direct access to a model governing its evolution. Instead, we have access to trajectories of $X^t$ of a fixed, finite duration $\tau$. A natural and generally applicable approach to finding an approximate solution to the prediction problem is to attempt to minimize the residual of (5) over parameters $\theta$ of a neural network $u_\theta(x)$ representing $u(x)$. For example, we recently suggested an algorithm that minimizes the residual norm

$$\frac{1}{2}\left\|(\mathcal{I}-\mathcal{S}^\tau)u_\theta - r\right\|_\mu^2, \tag{8}$$

where $r(x)$ is the right-hand side of (5) and $\mu$ is an arbitrary distribution of initial conditions $X^0$ (boundary conditions were enforced by an additional term).[24] The norm $\|\cdot\|_\mu$ is defined by $\|f\|_\mu{}^2 = \langle f, f\rangle_\mu$, where $\langle f, g\rangle_\mu = \int f(x)g(x)\mu(dx)$ for arbitrary functions $f$ and $g$. The gradient of the residual norm in (8) with respect to neural-network parameters $\theta$ can be written as

$$\left\langle(\mathcal{I}-\mathcal{S}^\tau)u_\theta - r, \nabla_\theta u_\theta\right\rangle_\mu - \left\langle(\mathcal{I}-\mathcal{S}^\tau)u_\theta - r, \mathcal{S}^\tau\nabla_\theta u_\theta\right\rangle_\mu. \tag{9}$$

Given a dataset of initial conditions $\{X_j^0\}_{j=1}^n$ drawn from $\mu$ and a single sample trajectory $\{X_j^t\}_{t=0}^\tau$ of $X^t$ for each $X_j^0$, the first term in the gradient (9) can be approximated without bias as

$$\begin{aligned}&\left\langle(\mathcal{I}-\mathcal{S}^\tau)u_\theta - r, \nabla_\theta u_\theta\right\rangle_\mu\\ &\approx \frac{1}{n}\sum_{j=1}^n\left(u_\theta(X_j^0) - u_\theta(X_j^{\tau\wedge T_j}) - \sum_{t=0}^{(\tau\wedge T_j)-1}\Gamma(X_j^t)\right)\nabla_\theta u_\theta(X_j^0),\end{aligned} \tag{10}$$

where $T_j$ is the first time $X_j^t \notin D$.

In contrast, unbiased estimation of the second term in (9) requires access to two independent trajectories of $X^t$ for each sample initial condition since it is quadratic in $\mathcal{S}^\tau$.[24,31] In the reinforcement learning community, TD methods were developed for the purpose of minimizing a loss of a very similar form to (8), and they perform a "semigradient" descent by following only the first term in (9).[31] As in the semigradient approximation, the algorithms proposed in this paper only require access to inner products of the form $\langle f, \mathcal{A}g\rangle_\mu$ for an operator $\mathcal{A}$ related to $\mathcal{T}^\tau$ or $\mathcal{S}^\tau$, and they avoid terms that are non-linear in $\mathcal{A}$. As we explain, such inner products can be estimated using trajectory data. However, rather than attempting to minimize the residual directly by an approximate gradient descent, we view the eigenproblem and prediction problems through the lens of iterative numerical linear algebra schemes.

## III. AN INEXACT POWER ITERATION

To motivate the iterative numerical linear algebra point of view, observe that the first term in (9) is the exact gradient with respect to $\theta'$ of the loss

$$\frac{1}{2}\left\|u_{\theta'} - \mathcal{S}^\tau u_\theta - r\right\|_\mu^2, \tag{11}$$

evaluated at $\theta' = \theta$. The result of many steps of gradient descent (later, "inner iterations") on this loss with $\theta$ held fixed can then be viewed as an inexact Richardson iteration for (5), resulting in a sequence

$$u_{\theta^{s+1}} \approx \mathcal{S}^\tau u_{\theta^s} + r, \tag{12}$$

where, for each $s$, $u_{\theta^s}$ is a parameterized neural-network approximation of the solution to (5). To unify our discussion of the prediction and spectral estimation problems, it is helpful to observe that (12) can be recast as an equivalent inexact power iteration

$$\bar{u}_{\theta^{s+1}} \approx \mathcal{A}\bar{u}_{\theta^s}, \tag{13}$$

where

$$\bar{u}_\theta = \begin{pmatrix}1\\ u_\theta\end{pmatrix} \quad\text{and}\quad \mathcal{A} = \begin{bmatrix}1 & 0\\ r & \mathcal{S}^\tau\end{bmatrix}. \tag{14}$$

Note that $(1, u)^\top$ is the dominant eigenfunction of $\mathcal{A}$ and has eigenvalue equal to 1.

Reference 32 introduced an inexact power iteration to compute the stationary probability measure of $\mathcal{T}^\tau$, i.e., its dominant left eigenfunction. As those authors note, an inexact power update, such as (13), can be enforced using a variety of loss functions. In our setting, the $L_\mu^2$ norm in (11) can be replaced by any other measure of the difference between $u_{\theta'}$ and $\mathcal{S}^\tau u_\theta + r$, as long as an unbiased estimator of its gradient with respect to $\theta'$ is available. This flexibility is discussed in more detail in Sec. V, and we exploit it in applications presented later in this article. For now, we focus instead on another important implication of this viewpoint: the flexibility in the form of the iteration itself.

We will see that when the spectral gap of $\mathcal{A}$, the difference between its largest and second largest eigenvalues, is small, the inexact power iteration (or the equivalent Richardson iteration) described above fails to reach an accurate solution. The largest eigenvalue of $\mathcal{A}$ in (14) is 1 and its next largest eigenvalue is the dominant eigenvalue of $\mathcal{S}^\tau$. For rare-event problems, the difference between these two eigenvalues is expected to be very small. Indeed, when $X^0$ is drawn from the quasi-stationary distribution of $X^t$ in $D$, the logarithm of the largest eigenvalue of $\mathcal{S}^\tau$ is exactly $-\tau/\mathbb{E}[T]$.[39] Thus, when the mean exit time from $D$ is large, we can expect the spectral gap of $\mathcal{A}$ in (14) to be very small. In this case, classical convergence results tell us that exact power iteration converges slowly, with the largest contributions to the error coming from the eigenfunctions of $\mathcal{S}^\tau$ with largest magnitude eigenvalues.[33] Iterative schemes that approximate multiple dominant eigenfunctions simultaneously, such as subspace iteration and Krylov methods, can converge much more rapidly.[33] In Sec. IV, we introduce an inexact form of subspace iteration to address this shortcoming. Beyond dramatically improving performance on the prediction problem for rare-events when applied with $\mathcal{A}$ in (14), it can also be applied with $\mathcal{A} = \mathcal{T}^\tau$ to compute multiple dominant eigenfunctions and values of $\mathcal{T}^\tau$ itself.

## IV. AN INEXACT SUBSPACE ITERATION

Our inexact subspace iteration for the $k$ dominant eigenfunctions/values of $\mathcal{A}$ proceeds as follows. Let $\left\{\varphi_{\theta^s}^a\right\}_{a=1}^k$ be a sequence of $k$ basis functions parameterized by $\theta^s$ (these can be scalar or vector valued functions depending on the form of $\mathcal{A}$). We can represent this basis as the vector valued function

$$U_{\theta^s} = \left(\varphi_{\theta^s}^1, \varphi_{\theta^s}^2, \ldots, \varphi_{\theta^s}^k\right). \tag{15}$$

Then, we can obtain a new set of $k$ basis functions by approximately applying $\mathcal{A}$ to each of the components of $U_{\theta^s}$,

$$U_{\theta^{s+1}} K^{s+1} \approx \mathcal{A}\, U_{\theta^s}, \tag{16}$$

where $K^{s+1}$ is an invertible, upper-triangular $k \times k$ matrix that does not change the span of the components of $U_{\theta^{s+1}}$ but is included to facilitate training. One way the approximate application of $\mathcal{A}$ can be accomplished is by minimizing

$$\frac{1}{2}\sum_{a=1}^{k} \left\| \sum_{b=1}^{k} \varphi_\theta^b K_{ba} - \mathcal{A}\, \varphi_{\theta^s}^a \right\|_\mu^2 \tag{17}$$

over $\theta$ and $K$ with $\theta^s$ held fixed.

The eigenvalues and eigenfunctions of $\mathcal{A}$ are then approximated by solving the finite-dimensional generalized eigenproblem

$$C^\tau W = C^0 W \Lambda, \tag{18}$$

where

$$C_{ab}^\tau = \langle \varphi_{\theta^s}^a, \mathcal{A}\varphi_{\theta^s}^b \rangle_\mu, \tag{19}$$

$$C_{ab}^0 = \langle \varphi_{\theta^s}^a, \varphi_{\theta^s}^b \rangle_\mu, \tag{20}$$

each inner product is estimated using trajectory data, and $W$ and $\Lambda$ are $k \times k$ matrices. The matrix $\Lambda$ is diagonal, and the $a$th eigenvalue $\lambda_a$ of $\mathcal{A}$ is approximated by $\Lambda_{aa}$; the corresponding eigenfunction $v_a$ is approximated by $\sum_{b=1}^k W_{ab}\, \varphi_{\theta^s}^b$.

Even when sampling is not required to estimate the matrices in (19) and (20), the numerical rank of $C^\tau$ becomes very small as the eigenfunctions become increasingly aligned with the single dominant eigenfunction. To overcome this issue, we apply an orthogonalization step between iterations (or every few iterations). Just as the matrices $C^0$ and $C^\tau$ can be estimated using trajectory data, the orthogonalization procedure can also be implemented approximately using data.

Finally, in our experiments, we find it advantageous to damp large parameter fluctuations during training by mixing the operator $\mathcal{A}$ with a multiple of the identity, i.e., we perform our inexact subspace iteration on the operator $(1-\alpha_s)\mathcal{I} + \alpha_s \mathcal{A}$ in place of $\mathcal{A}$ itself. This new operator has the same eigenfunctions as $\mathcal{A}$. In our experiments, decreasing the parameter $\alpha_s$ as the number of iterations increases results in better generalization properties of the final solution and helps ensure convergence of the iteration. For our numerical experiments, we use

$$\alpha_s = \begin{cases} 1 & s < \sigma \\ 1/\sqrt{s+1-\sigma} & s \geq \sigma, \end{cases} \tag{21}$$

where $\sigma$ is a user chosen hyperparameter that sets the number of iterations performed before damping begins.

The details, including estimators for all required inner products, in the case of the eigenproblem $(\mathcal{A} = \mathcal{T}^\tau)$ are given in Sec. VII and Algorithm 1. For the prediction problem with $\mathcal{A}$ as in (14), they are given in Sec. VIII and Algorithm 2.

**ALGORITHM 1.** Inexact subspace iteration (with $L^2_\mu$ loss) for spectral estimation.

```
Require: Subspace dimension $k$, transition data $\{X_j^0, X_j^\tau\}_{j=1}^n$, batch size $B$, learning rate $\eta$, number of subspace iterations $S$, number of inner
    iterations $M$, regularization parameters $\gamma_1$ and $\gamma_2$
 1: Initialize $\{\varphi_\theta^a\}_{a=1}^k$ and $\{\tilde\varphi_1^a\}_{a=1}^k$
 2: for $s = 1 \dots S$ do
 3:     for $m = 1 \dots M$ do
 4:         Sample a batch of data $\{X_j^0, X_j^\tau\}_{j=1}^B$
 5:         $\hat{\mathcal{L}}_1 \leftarrow \frac{1}{B}\sum_{j=1}^B \sum_{a=1}^k \left[\frac{1}{2}(\sum_{b=1}^k \varphi_\theta^b(X_j^0)K_{ba})^2 - \sum_{b=1}^k \varphi_\theta^b(X_j^0)K_{ba}\{\alpha_s\tilde\varphi_s^a(X_j^\tau) + (1-\alpha_s)\tilde\varphi_s^a(X_j^0)\}\right]$
 6:         $\hat{\mathcal{L}}_K \leftarrow \gamma_1\|K - \mathrm{diag}\,(K)\|_{\mathrm{F}}^2$
 7:         $\hat{\mathcal{L}}_{\mathrm{norm}} \leftarrow \gamma_2\sum_{a=1}^k (2\nu_a(\frac{1}{B}\sum_{j=1}^B \varphi_\theta^a(X_j^0)^2 - 1) - \nu_a^2)$
 8:         $\hat{\mathcal{L}} \leftarrow \hat{\mathcal{L}}_1 + \hat{\mathcal{L}}_K + \hat{\mathcal{L}}_{\mathrm{norm}}$
 9:         $\theta \leftarrow \theta - \eta\nabla_\theta\hat{\mathcal{L}}$
10:         $K \leftarrow K - \eta\,\mathrm{triu}(\nabla_K\hat{\mathcal{L}})$
11:         $\nu_a \leftarrow \nu_a + \eta\nabla_{\nu_a}\hat{\mathcal{L}}$
12:     end for
13:     Compute the matrix $\Phi_{ia} = \varphi_\theta^a(X_i^0)$                                    ▷ $\Phi \in \mathbb{R}^{n\times k}$
14:     Compute diagonal matrix $N_{aa}^2 = \sum_i \varphi_\theta^a(X_i^0)^2$
15:     Compute QR-decomposition $\Phi = QR$                                 ▷ $Q \in \mathbb{R}^{n\times k}$ and $R \in \mathbb{R}^{k\times k}$
16:     $\tilde\varphi_s^a \leftarrow \sum_{b=1}^k \varphi_\theta^b\,(R^{-1}N)_{ba}$
17:     $K_{1:i,i} \leftarrow \arg\min_c \frac{1}{n}\sum_{j=1}^n \left(\sum_{a=1}^i \varphi_\theta^a(X_j^0)c_a - \tilde\varphi_s^a(X_j^\tau)\right)^2 + \gamma_2\sum_{a=1}^{i-1} c_a^2$
18: end for
19: Compute the matrices $C_{ab}^t = \frac{1}{n}\sum_{j=1}^n \tilde\varphi_s^a(X_j^0)\tilde\varphi_s^b(X_j^t)$ for $t = 0, \tau$       ▷ $C^t \in \mathbb{R}^{k\times k}$
20: Solve the generalized eigenproblem $C^\tau W = C^0 W\Lambda$ for $W$ and $\Lambda$
21: return eigenvalues $\Lambda$, eigenfunctions $v_a = \sum_{b=1}^k W_{ab}\tilde\varphi_s^b$
```

**ALGORITHM 2.** Inexact subspace iteration (with $L^2_\mu$ loss) for prediction functions.

```
Require: Subspace dimension $k$, stopped transition data $\left\{X_j^0, X_j^{\tau\wedge T_j}\right\}_{j=1}^n$, reward data $\{\rho(X_j)\}_{j=1}^n$, batch size $B$, learning rate $\eta$, number of subspace iterations $S$, number of inner iterations $M$, regularization parameters $\gamma_1$ and $\gamma_2$
1:  Initialize $\{\varphi_\theta^a\}_{a=1}^k$ and $\{\tilde{\varphi}_1^a\}_{a=1}^k$
2:  for $s = 1 \ldots S$ do
3:      for $m = 1 \ldots M$ do
4:          Sample a batch of data $\{X_j^0, X_j^{\tau\wedge T_j}\}_{j=1}^B$, $\{\rho(X_j)\}_{j=1}^B$
5:          $\hat{\mathcal{L}}_1 \leftarrow \frac{1}{B}\sum_{j=1}^B \sum_{a=1}^k \left[\frac{1}{2}(\sum_{b=1}^k \varphi_\theta^b(X_j^0)K_{ba})^2 - \alpha_s\left(\sum_{b=1}^k \varphi_\theta^b K_{ba}(\tilde{\varphi}_s^a(X_j^{\tau\wedge T_j}) + \rho(X_j)I_{a1})\right)\right]$
6:          $\hat{\mathcal{L}}_2 \leftarrow -\frac{1}{B}\sum_{j=1}^B \sum_{a=1}^k (1-\alpha_s)\sum_{b=1}^k \varphi_\theta^b K_{ba}\tilde{\varphi}_s^a(X_j^0)$
7:          $\hat{\mathcal{L}}_K \leftarrow \gamma_1\|K - \mathrm{diag}\,(K)\|_{\mathrm{F}}^2$
8:          $\hat{\mathcal{L}}_{\mathrm{norm}} \leftarrow \gamma_2\sum_{a=2}^k (2\nu_a(\frac{1}{B}\sum_{j=1}^B \varphi_\theta^a(X_j^0)^2 - 1) - \nu_a^2)$
9:          $\hat{\mathcal{L}} \leftarrow \hat{\mathcal{L}}_1 + \hat{\mathcal{L}}_2 + \hat{\mathcal{L}}_K + \hat{\mathcal{L}}_{\mathrm{norm}}$
10:         $\theta \leftarrow \theta - \eta\nabla_\theta\hat{\mathcal{L}}$
11:         $K \leftarrow K - \eta\,(\mathrm{triu}(\nabla_K\hat{\mathcal{L}}))$
12:         $\nu_a \leftarrow \nu_a + \eta\nabla_{\nu_a}\hat{\mathcal{L}}$
13:     end for
14:     if $\Psi(x) = 0$ then
15:         Compute the matrix $\Phi_{ia} = \varphi_\theta^a(X_i^0)$            ▷ $\Phi \in \mathbb{R}^{n\times k}$
16:     else
17:         Compute the matrix $\Phi_{ia} = \varphi_\theta^a(X_i^0)$ for $a > 1$            ▷ $\Phi \in \mathbb{R}^{n\times(k-1)}$
18:     end if
19:     Compute QR-decomposition $\Phi = QR$
20:     Compute diagonal matrix $N_{aa}^2 = \sum_i \varphi_\theta^a(X_i^0)^2$
21:     $\tilde{\varphi}_s^a \leftarrow \sum_{b=1}^k \varphi_\theta^b\,(R^{-1}N)_{ba}$            ▷ if $\Psi(x) = 0$ exclude $a = 1$
22:     $K_{1:i,i} \leftarrow \arg\min_c \frac{1}{n}\sum_{j=1}^n \left(\sum_{a=1}^i \varphi_\theta^a(X_j^0)c_a - \tilde{\varphi}_s^a(X_j^{\tau\wedge T_j})\right)^2 + \gamma_2\sum_{a=1}^{i-1} c_a^2$
23: end for
24: Compute the matrix $C_{ab}^t = \frac{1}{n}\sum_{j=1}^n \varphi_\theta^a(X_j^0)\varphi_\theta^b(X_j^t)$ for $t = 0, \tau\wedge T_j$            ▷ $C^t \in \mathbb{R}^{k\times k}$
25: Compute the vector $p_a = \frac{1}{n}\sum_{j=1}^n \varphi_\theta^a(X_j^0)\rho(X_j)$
26: Solve linear system $(C^0 - C^\tau)w = p$            ▷ if $\Psi(x) = 0$ enforce $w_1 = 1$
27: return $u = \sum_{a=1}^k w_a\varphi_\theta^a$
```

## V. ALTERNATIVE LOSS FUNCTIONS

As mentioned above, the inexact application of the operator $\mathcal{A}$ can be accomplished by minimizing loss functions other than (17). The key requirement for a loss function in the present study is that $\mathcal{A}$ appears in its gradient only through terms of the form $\langle f, \mathcal{A}g\rangle_\mu$ for some functions $f$ and $g$, so that the gradient can be estimated using trajectory data. As a result, we have flexibility in the choice of loss and, in turn, the representation of $u$. In particular, we consider the representation $u_\theta = \zeta(z_\theta)$, where $\zeta$ is an increasing function, and $z_\theta$ is a function parameterized by a neural network. An advantage of doing so is that the function $\zeta$ can restrict the output values of $u_\theta$ to some range. For example, when computing a probability such as the committor, a natural choice is the sigmoid function $\zeta(x) = (1 + e^{-x})^{-1}$.

Our goal is to train a sequence of parameter values so that $u_{\theta^s}$ approximately follows a subspace iteration, i.e., so that $\zeta(z_{\theta^{s+1}}) \approx \mathcal{A}u_{\theta^s}$. To this end, we minimize with respect to $\theta$ the loss function

$$\mathbb{E}_{X^0\sim\mu}[V(z_\theta) - z_\theta\mathcal{A}u_{\theta^s}], \tag{22}$$

where $V$ is an antiderivative of $\zeta$, and $\theta^s$ is fixed. The subscript $X^0 \sim \mu$ in this expression indicates that $X^0$ is drawn from $\mu$. Note that, as desired, $\mathcal{A}$ appears in the gradient of (22) only in an inner product of the required form, and an approximate minimizer, $\theta^{s+1}$, of this loss satisfies $\zeta(z_{\theta^{s+1}}) \approx \mathcal{A}u_{\theta^s}$. This general form of loss function is adapted from variational expressions for the divergence of two probability measures.[32,40]

The $L^2_\mu$ loss in (17), which we use in several of our numerical experiments, corresponds to the choice $\zeta(x) = x$ and $V(x) = x^2/2$. The choice of $\zeta(x) = (1 + e^{-x})^{-1}$ mentioned above corresponds to $V(x) = \log(1 + e^x)$; we refer to the loss in (22) with this choice of $V$ as the "softplus" loss. We note that in the context of committor function estimation, in the limit of infinite $\tau$ the "softplus" loss corresponds directly to the maximum log-likelihood loss (for independent Bernoulli random variables) that defines the logistic regression estimate of the probability of reaching $B$ before $A$. Logistic regression has previously been used in conjunction with datasets of trajectories integrated all the way until reaching $A$ or $B$ to estimate the committor function.[41–46]

## VI. TEST PROBLEMS

We illustrate our methods with two well-characterized systems that exemplify features of molecular transitions. In this section, we provide key details of these systems.

### A. Müller-Brown potential

The first system is defined by the Müller–Brown potential[47] (Fig. 1)

$$V_{\mathrm{MB}}(y,z) = \frac{1}{20}\sum_{i=1}^{4} C_i \exp\left[a_i(y-y_i)^2 + b_i(y-y_i)(z-z_i) + c_i(z-z_i)^2\right]. \tag{23}$$

The two-dimensional nature of this model facilitates visualization. The presence of multiple minima and saddlepoints that are connected by a path that does not align with the coordinate axes makes the system challenging for both sampling and analysis methods. In Secs. VII A and VIII A, we use $C_i = \{-200, -100, -170, 15\}$, $a_i = \{-1, -1, -6.5, 0.7\}$, $b_i = \{0, 0, 11, 0.6\}$, $c_i = \{-10, -10, -6.5, 0.7\}$, $y_i = \{1, -0.27, -0.5, -1\}$, and $z_i = \{0, 0.5, 1.5, 1\}$. In Sec. VIII B, we tune the parameters to make transitions between minima rarer; there, the parameters are $C_i = \{-250, -150, -170, 15\}$, $a_i = \{-1, -3, -6.5, 0.7\}$, $b_i = \{0, 0, 11, 0.6\}$, $c_i = \{-10, -30, -6.5, 0.7\}$, $y_i = \{1, -0.29, -0.5, -1\}$, and $z_i = \{0, 0.5, 1.5, 1\}$. For prediction, we analyze transitions between the upper left minimum ($A$) and the lower right minimum ($B$) in Fig. 1; these states are defined as

$$A = \{y,z : 6.5(y+0.5)^2 - 11(y+0.5)(z-1.5) + 6.5(z-1.5)^2 < 0.3\}$$
$$B = \{y,z : (y-0.6)^2 + 0.5(z-0.02)^2 < 0.2\}. \tag{24}$$

To generate a dataset, we randomly draw 50 000 initial conditions $X_j^0$ uniformly from the region

$$\Omega = \{y,z : -1.5 < y < 1.0,\ -0.5 < z < 2.5,\ V_{\mathrm{MB}}(y,z) < 12\} \tag{25}$$

and, from each of these initial conditions, generate one trajectory according to the discretized overdamped Langevin dynamics

$$X_j^t = X_j^{t-1} - \delta_t \nabla V_{\mathrm{MB}}(X_j^{t-1}) + \sqrt{\delta_t\, 2\beta^{-1}}\, \xi_j^t, \tag{26}$$

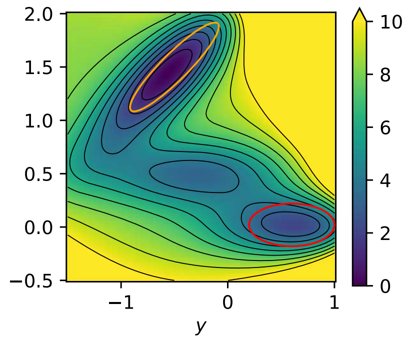

FIG. 1. Müller–Brown potential energy surface. The orange and red ovals indicate the locations of states $A$ and $B$, respectively, when computing predictions. Contour lines are drawn every 1 $\beta^{-1}$.

where $1 \le t \le \tau$, the $\xi_j^t$ are independent standard Gaussian random variables, and the timestep is $\delta_t = 0.001$. We use an inverse temperature of $\beta = 2$, and we vary $\tau$ as indicated below (note that $\tau$ is an integer number of steps, but we present our results for the Müller–Brown model in terms of the dimensionless scale used for $\delta_t$ immediately above). For each test, we use independent random numbers and redraw the initial conditions because it is faster to generate the trajectories than to read them from disk in this case. All error bars are computed from the empirical standard deviation over ten replicate datasets.

To validate our results, we compare the neural-network results against grid-based references, computed as described in Sec. S4 of the supplementary material of Ref. 11 and the appendix of Ref. 48 (our notation here follows the former more closely). The essential idea is that the terms in the infinitesimal generator of the transition operator can be approximated on a grid to second order in the spacing by expanding both the probability for transitions between sites and a test function. To this end, we define the transition matrix for neighboring sites $x = (y,z)$ and $x' = (y \pm \delta_x, z)$ or $(y, z \pm \delta_x)$ on a grid with spacings $\delta_x$,

$$P(x'|x) = \frac{1}{1 + e^{\beta[V(x') - V(x)]}} \tag{27}$$

(this definition differs from that in Ref. 11 by a factor of 1/4, and we scale $P - I$, where $I$ is the identity matrix, accordingly to set the time units below). The diagonal entry $P(x|x)$ is fixed to make the transition matrix row-stochastic. We use $\delta_x = 0.0125$; the grid is a rectangle with $y \in [-1.5, 1.0]$, and $z \in [-0.5, 2.0]$. The reference invariant subspaces are computed by diagonalizing the matrix $2(P-I)/\beta\delta_x^2$ with a sparse eigensolver; we use scipy.sparse.linalg. We obtain the reference committor $\hat{q}_+$ for grid sites in $(A \cup B)^{\mathsf{C}}$ by solving

$$\operatorname{diag}(\hat{\mathbb{1}}_{(A\cup B)^{\mathsf{C}}})(P-I)\hat{q}_+ = -\operatorname{diag}(\hat{\mathbb{1}}_{(A\cup B)^{\mathsf{C}}})(P-I)\hat{\mathbb{1}}_B \tag{28}$$

and for grid sites in $A \cup B$ by setting $\hat{q}_+ = \hat{\mathbb{1}}_B$. Here, $\hat{q}_+$ and $\hat{\mathbb{1}}_B$ are vectors corresponding to the functions evaluated at the grid sites.

### B. AIB$_9$ helix-to-helix transition

The second system is a peptide of nine $\alpha$-aminoisobutyric acids (AIB$_9$; Fig. 2). Because AIB is achiral around its $\alpha$-carbon atom, AIB$_9$ can form left- and right-handed helices with equal probabilities, and we study the transitions between these two states. This transition was previously studied using MSMs and long unbiased molecular dynamics simulations.[49,50] AIB$_9$ poses a stringent test due to the presence of many metastable intermediate states.

The states are defined in terms of the internal $\phi$ and $\psi$ dihedral angles. We classify an amino acid as being in the "left" state if its dihedral angle values are within a circle of radius 25° centered at (41°, 43°), that is,

$$(\phi - 41^\circ)^2 + (\psi - 43^\circ)^2 \le (25^\circ)^2.$$

Amino acids are classified as being in the "right" state using the same radius, but centered instead at (−41°, −43°). States $A$ and $B$ are defined by the amino acids at sequence positions 3–7 being all left or all right, respectively. We do not use the two residues on each

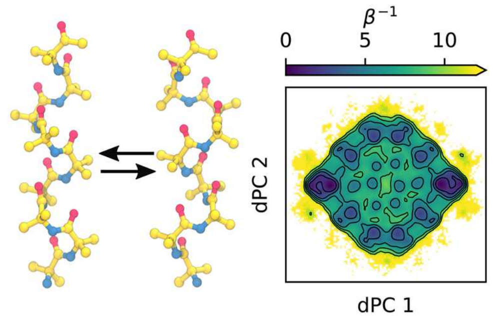

**FIG. 2.** Helix-to-helix transition of AIB$_9$. (Left) Left- and right-handed helices, which we use as states $A$ and $B$, respectively, when computing predictions. Carbon, nitrogen, and oxygen atoms are shown in yellow, blue, and red, respectively; hydrogen atoms are omitted. (Right) Potential of mean force constructed from the histogram of value pairs of the first two dihedral angle principal components; data are from the 20 trajectories of 5 $\mu$s that we use to construct reference statistics (see text). The left-handed helix corresponds to the left-most basin, and the right-handed helix corresponds to the right-most basin. Contour lines are drawn every 2 $\beta^{-1}$ corresponding to a temperature of 300 K.

end of AIB$_9$ in defining the states as these are typically more flexible.[50] The states can be resolved by projecting onto dihedral angle principal components (dPCs; Fig. 2, right) as described previously.[51]

Following a procedure similar to that described in Ref. 50, we generate a dataset of short trajectories. From each of the 691 starting configurations in Ref. 50, we simulate ten trajectories of duration 20 ns with initial velocities drawn randomly from a Maxwell–Boltzmann distribution at a temperature of 300 K. The short trajectory dataset thus contains 6910 trajectories, corresponding to a total sampling time of 138.2 $\mu$s. We use a timestep of 4 fs together with a hydrogen mass repartitioning scheme,[52] and configurations are saved every 40 ps. We employ the AIB parameters from Forcefield_NCAA[53] and the GBNeck2 implicit-solvent model.[54] Simulations are performed with the Langevin integrator in OpenMM 7.7.0[55] using a friction coefficient of 1 ps$^{-1}$. To generate a reference for comparison to our results, we randomly select 20 configurations from the dataset above and, from each, run a simulation of 5 $\mu$s with the same simulation parameters. For all following tests on AIB$_9$, batches consist of pairs of frames separated by $\tau$ drawn randomly with replacement from the short trajectories (i.e., from all possible such pairs in the dataset). From each frame, we use only the atom positions because the momenta decorrelate within a few picoseconds, which is much shorter than the lag times that we consider. However, in principle, the momenta could impact the prediction functions[56] and be used as neural-network inputs as well.

## VII. SPECTRAL ESTIMATION

In this section, we provide some further numerical details for the application of our method to spectral estimation and demonstrate the method on the test problems. For our subspace iteration with $\mathcal{A} = \mathcal{T}^\tau$, we require estimators for inner products of the form $\langle f, \mathcal{T}^\tau g\rangle_\mu$. For example, the gradient of the loss function (17) involves inner products of the form

$$\left\langle \nabla_\theta \varphi_\theta^a, \mathcal{T}^\tau \varphi_\theta^b \right\rangle_\mu. \tag{29}$$

For these, we use the unbiased data-driven estimator

$$\langle f, \mathcal{T}^\tau g\rangle_\mu \approx \frac{1}{n}\sum_{j=1}^{n} f(X_j^0) g(X_j^\tau). \tag{30}$$

As discussed in Sec. IV, applying the operator $\mathcal{T}^\tau$ repeatedly causes each basis function to converge to the dominant eigenfunction and leads to numerical instabilities. To avoid this, we orthogonalize the outputs of the networks with a QR decomposition at the end of each subspace iteration by constructing the matrix $\Phi_{ia} = \varphi_\theta^a(X_i^0)$ and then computing the factorization $\Phi = QR$, where $Q$ is an $n \times k$ matrix with orthogonal columns and $R$ is an upper triangular $k \times k$ matrix. Finally, we set $\tilde{\varphi}_s^a = \sum_{b=1}^k \varphi_\theta^b (R^{-1}N)_{ba}$, where $N$ is a diagonal matrix with entries equal to the norms of the columns of $\Phi$ (before orthogonalization). To ensure that the networks remain well-separated (i.e., the eigenvalues of $C^0$ remain away from zero), we penalize large off-diagonal entries of $K$ by adding to the loss

$$\gamma_1 \| K - \operatorname{diag}(K) \|_F^2, \tag{31}$$

where $\gamma_1$ allows us to tune the strength of this term relative to others, and $\|\cdot\|_F$ is the Frobenius norm. We control the scale of each network output using the strategy from Ref. 32; that is, we add to the loss a term of the form

$$\gamma_2 \sum_{a=1}^{k} \left[ 2\nu_a \left( \frac{1}{n} \sum_{j=1}^{n} \varphi_\theta^a (X_j^0)^2 - 1 \right) - \nu_a^2 \right], \tag{32}$$

where we have introduced the conjugate variables $\nu_a$, which we maximize with gradient ascent (or similar optimization). In general, our numerical experiments suggest that it is best to keep $\gamma_1$ and $\gamma_2$ relatively small. We find that stability of the algorithm over many subspace iterations is improved if the matrix $K$ is set at its optimal value before each inner loop. To do this, we set

$$K_{1:i,i} = \arg\min_c \frac{1}{n} \sum_{j=1}^{n} \left( \sum_{a=1}^{i} \varphi_\theta^a(X_j^0) c_a - \tilde{\varphi}_s^a(X_j^\tau) \right)^2 + \gamma_2 \sum_{a=1}^{i-1} c_a^2. \tag{33}$$

The above minimization can be solved with linear least squares. Finally, we note that, in practice, any optimizer can be used for the inner iteration steps, though the algorithm below implements stochastic gradient descent. In this work, we use Adam[57] for all numerical tests. We summarize our procedure for spectral estimation in Algorithm 1.

### A. Müller–Brown model

As a first test of our method, we compute the $k = 3$ dominant eigenpairs for the Müller–Brown model. Since we know that the dominant eigenfunction of the transition operator is the constant function $v_1(y, z) = 1$ with eigenvalue $\lambda_1 = 1$, we directly include this function in the basis as a non-trainable function, i.e., $\varphi_\theta^1(y, z) = 1$. To initialize $\tilde{\varphi}_1^a$ for each $a > 1$, we choose a standard Gaussian vector $(Y^a, Z^a) \in \mathbb{R}^2$, and set $\tilde{\varphi}_1^a(y, z) = y\,Y^a + z\,Z^a$. This ensures that the initial basis vectors are well-separated and the first QR step is numerically stable. Here and in all subsequent Müller–Brown tests, batches of trajectories are drawn from the entire dataset with replacement. Other hyperparameters are listed in Table I.

Figure 3 shows that we obtain good agreement between the estimate produced by the inexact subspace iteration in Algorithm 1 and

**TABLE I.** Parameter choices used in this work.

| | Spectral estimation | | Committor | | | MFPT |
|---|---|---|---|---|---|---|
| Hyperparameter | Müller-Brown | AIB$_9$ | Müller-Brown | Modified Müller-Brown | AIB$_9$ | AIB$_9$ |
| Subspace dimension $k$ | 3 | 5 | 1 | 2, 1[a] | 1 | 5 |
| Input dimensionality | 2 | 174 | 2 | 2 | 174 | 174 |
| Hidden layers | 6 | 6 | 6 | 6 | 6 | 6 |
| Hidden layer width | 64 | 128 | 64 | 64 | 150 | 150 |
| Hidden layer nonlinearity | CeLU | CeLU | ReLU | ReLU | ReLU | ReLU |
| Output layer nonlinearity | None | tanh | Sigmoid/none | None | None | None |
| Outer iterations $S$ | 10 | 100 | 100 | 4 + 10[a] | 100 | 300 |
| Inner iterations $M$ | 5000 | 2000 | 200 | 5000 | 2000 | 1000 |
| $\sigma$ | 2 | 50 | 50 | 0 | 50 | 0 |
| Batch size $B$ | 2000 | 1024 | 5000 | 2000 | 1024 | 2000 |
| Learning rate $\eta$ | 0.001 | 0.0001 | 0.001 | 0.001 | 0.001 | 0.001 |
| $\gamma_1$ | 0.15 | 0.001 | $\cdots$ | $\cdots$ | $\cdots$ | 0.1 |
| $\gamma_2$ | 0.01 | 0.01 | $\cdots$ | $\cdots$ | $\cdots$ | 10 |
| Loss for $\varphi_\theta^1$ | $L_\mu^2$ | $L_\mu^2$ | $L_\mu^2$/softplus | Softplus | Softplus | $L_\mu^2$ |
| Loss for $\varphi_\theta^a$ for $a > 1$ | $L_\mu^2$ | $L_\mu^2$ | $\cdots$ | $L_\mu^2$ | $\cdots$ | $L_\mu^2$ |

[a]Four subspace iterations with $k = 2$ followed by ten iterations with $k = 1$.

reference eigenfunctions. Figure 4 (upper panels) shows how the corresponding eigenvalues vary with lag time; again there is good agreement with the reference. Furthermore, there is a significant gap between $\lambda_3$ and $\lambda_4$, indicating that a three-dimensional subspace captures the dynamics of interest for this system.

We compare the subspace that we obtain from our method with that from an MSM constructed from the same amount of data by using $k$-means to cluster the configurations into 400 states and counting the transitions between clusters. This is a very fine discretization for this system, and the MSM is sufficiently expressive to yield eigenfunctions in good agreement with the reference. The relative error of $1 - \lambda_2$ is comparable for the two methods (Fig. 4, lower left). To compare two finite dimensional subspaces, $\mathcal{U}$ and $\mathcal{V}$, we define the subspace distance as[4]

$$d(\mathcal{U}, \mathcal{V}) = \|(I - P_{\mathcal{U}})P_{\mathcal{V}}\|_F, \tag{34}$$

where $P_{\mathcal{U}}$ and $P_{\mathcal{V}}$ denote the orthogonal projections onto $\mathcal{U}$ and $\mathcal{V}$, respectively, and $\|\cdot\|_F$ is the Frobenius norm. Figure 4 (lower right) shows the subspace distances from the reference as functions of lag time. We see that the inexact subspace iteration better approximates the three-dimensional dominant eigenspace for moderate to long lag times, even though the eigenvalues are comparable.

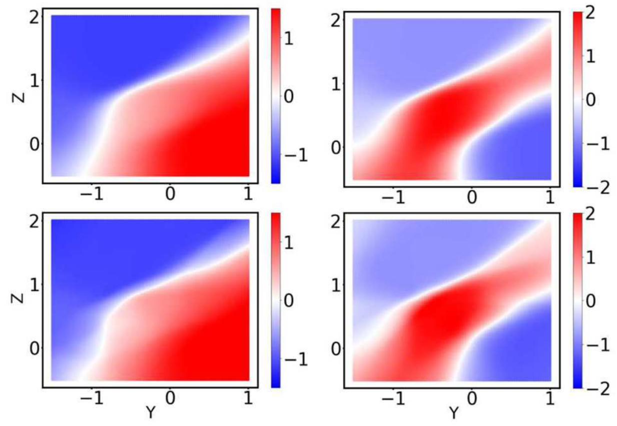

**FIG. 3.** First two non-trivial eigenfunctions of the Müller–Brown model. (Top) Grid-based reference. (Bottom) Neural network subspace after ten subspace iteration steps, computed with $\tau$ = 300 steps (i.e., 0.3 $\delta_t^{-1}$).

## B. AIB$_9$

For the molecular test system, we compute the dominant five-dimensional subspace. We compare the inexact subspace iteration in Algorithm 1 with MSMs constructed on dihedral angles ("dihedral MSM") and on Cartesian coordinates ("Cartesian MSM"). We expect the dihedral MSM to be accurate given that the dynamics of AIB$_9$ are well-described by the backbone dihedral angles,[49,50] and we thus use it as a reference. It is constructed by clustering the sine and cosine of each of the backbone dihedral angles ($\phi$ and $\psi$) for the nine residues (for a total of $2 \times 2 \times 9 = 36$ input features) into 1000 clusters using $k$-means and counting transitions between clusters. The Cartesian MSM is constructed by similarly counting transitions between 1000 clusters from the $k$-means algorithm, but the clusters are based on the Cartesian coordinates of all non-hydrogen atoms after aligning the backbone atoms of the trajectories, for a total of 174 input features. Because of the difficulty of clustering high-dimensional data, we expect the Cartesian MSM basis to give poor results. The neural network for the inexact subspace iteration receives the same 174 Cartesian coordinates as input features. We choose to use Cartesian coordinates rather than dihedral angles as inputs because it requires the network to identify nontrivial representations for describing the dynamics.

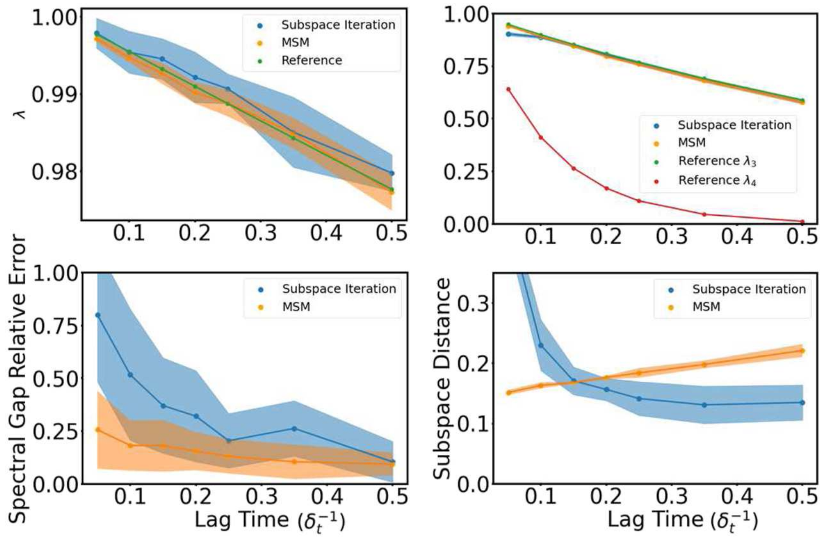

**FIG. 4.** Spectral estimation as a function of lag time (in units of $\delta_t^{-1}$) for the Müller–Brown model. (Top left) Second eigenvalue. (Top right) Third and fourth eigenvalues; only the reference fourth eigenvalue is shown to illustrate the spectral gap. (Bottom left) Relative error in the first spectral gap (i.e., $1 - \lambda_2$). (Bottom right) Subspace distance between estimated and reference three-dimensional invariant subspaces.

As in the Müller–Brown example, we use $\varphi_\theta^1 = 1$ and a random linear combination of coordinate functions to initialize $\tilde{\varphi}_1^a$ for $a > 1$. Other hyperparameters are listed in Table I. With these choices, the neural-network training typically requires about 20 minutes on a single NVIDIA A40 GPU; this is much longer than the time required for diagonalization of the $1000 \times 1000$ MSM transition matrix, which is nearly instantaneous. However, the time for neural-network training is negligible compared with the time to generate the dataset, which is the same for both approaches.

Taking the dihedral MSM as a reference, the Cartesian MSM systematically underestimates the eigenvalues (Fig. 5). The subspace iteration is very accurate for the first four eigenvalues, but the estimates for the fifth are low and vary considerably from run to run. A very small gap between $\lambda_4$ and $\lambda_5$ may contribute to the difficulty in estimating $\lambda_5$. In Fig. 6, we plot the first two non-trivial eigenfunctions ($v_2$ and $v_3$), which align with the axes of the dPC projection. The eigenfunction $v_2$ corresponds to the transition between the left- and right-handed helices; the eigenfunction $v_3$ is nearly orthogonal to $v_2$ and corresponds to transitions between intermediate states. It is challenging to visualize the remaining two eigenfunctions by projecting onto the first two dPCs because $v_4$ and $v_5$ are orthogonal to $v_2$ and $v_3$. The estimates for $v_2$ are in qualitative agreement for all lag times tested (Fig. 6 shows results for $\tau$ corresponding to 40 ps), but the subspace iteration results are less noisy for the shortest lag times. Moreover, the estimate for $v_3$ from subspace iteration agrees more closely with that from the dihedral MSM than does the estimate for $v_3$ from the Cartesian MSM. The subspace distance for $v_2$ and $v_3$ between the subspace iteration and the dihedral MSM is 0.947, compared with a value of 0.969 for the subspace distance between the two MSMs. Together, our results indicate that

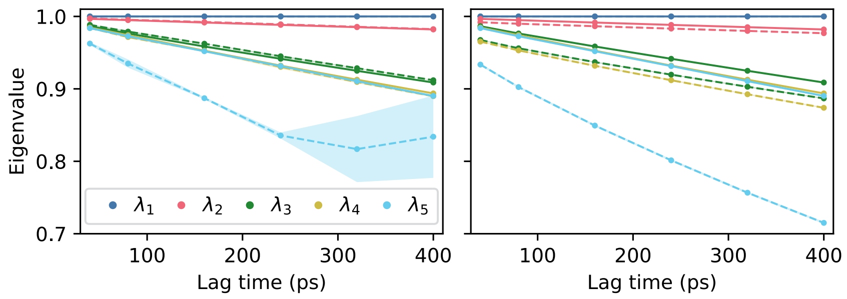

**FIG. 5.** First five eigenvalues of the transition operator for AIB$_9$ as a function of lag time. (Left) Comparison between eigenvalues computed using the dihedral MSM with 1000 clusters (solid lines) and the inexact subspace iteration (dashed lines). The shading indicates standard deviations over five trained networks for the subspace iteration. (Right) Comparison between a dihedral MSM (solid lines) and Cartesian MSMs with 1000 clusters (dashed lines). The standard deviations for the Cartesian MSMs over five random seeds for $k$-means clustering are too narrow to be seen.

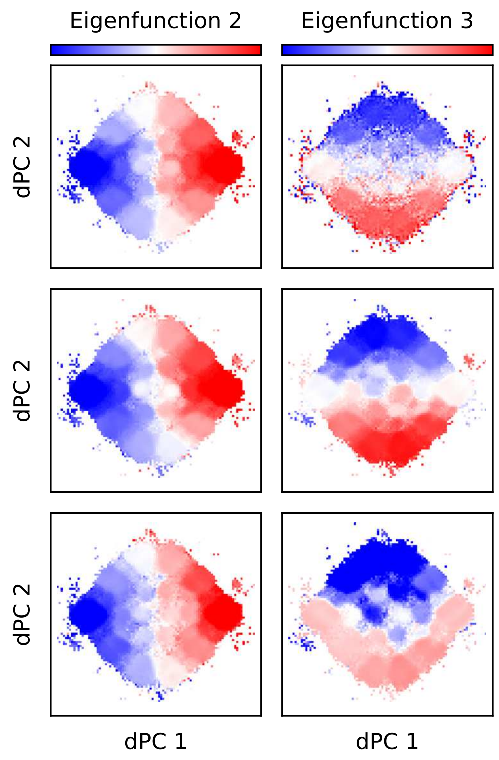

**FIG. 6.** First two non-trivial eigenfunctions of $\mathrm{AIB}_9$ projected onto the first two dPCs (i.e., averaged for bins in the two-dimensional space shown). (Top) MSM constructed on sine and cosine of dihedral angles with 1000 clusters and lag time corresponding to 40 ps. (Middle) Inexact subspace iteration using Cartesian coordinates and the same lag time. (Bottom) MSM constructed on Cartesian coordinates with 1000 clusters and the same lag time.

the neural networks are able to learn the leading eigenfunctions and eigenvalues of the transition operator (dynamical modes) of this system despite being presented with coordinates that are not the natural ones for describing the dynamics.

## VIII. PREDICTION

Inexact subspace iteration for $\mathcal{A}$ in (14) is equivalent to performing the inexact Richardson iteration in (12) on the first basis function $\varphi_\theta^1$ and then performing an inexact subspace iteration for the operator $\mathcal{S}^\tau$ on the rest of the basis functions. The iteration requires unbiased estimators of the forms

$$\langle f, \mathcal{S}^\tau g\rangle_\mu \approx \frac{1}{n}\sum_{j=1}^{n} f(X_j^0) g(X_j^{\tau\wedge T_j}) \tag{35}$$

and

$$\langle f, r\rangle_\mu \approx \frac{1}{n}\sum_{j=1}^{n} f(X_j^0) \sum_{t=0}^{(\tau\wedge T_j)-1} \Gamma(X_j^t), \tag{36}$$

where $T_j$ is the first time $X_j^t$ enters $D^{\mathrm{C}}$ and $r$ is the right-hand side of (5), as previously.

The Richardson iterate, $\varphi_\theta^1$, must satisfy the boundary condition $\varphi_\theta^1(x) = \Psi(x)$ for $x \notin D$. The other basis functions should satisfy $\varphi_\theta^a(x) = 0$ for $x \notin D$. In practice, we enforce the boundary conditions by explicitly setting $\varphi_\theta^1(x) = \Psi(x)$ and $\varphi_\theta^a(x) = 0$ for $a > 1$ when $x \notin D$.

When the boundary condition is zero, as for the MFPT, we find an approximate solution of the form

$$u_\theta = \sum_{a=1}^{k} w_a \varphi_\theta^a \tag{37}$$

by solving the $k$-dimensional linear system

$$\left(C^0 - C^\tau\right) w = p, \tag{38}$$

where, for $a, b \geq 1$,

$$C_{ab}^t = \left\langle \varphi_\theta^a, \mathcal{S}^t \varphi_\theta^b \right\rangle_\mu \tag{39}$$

for $t = 0, \tau$, and

$$p_a = \left\langle \varphi_\theta^a, \mathbb{E}_x[\rho(X)]\right\rangle_\mu. \tag{40}$$

In (40), we introduce the notation

$$\rho(X) = \sum_{t=0}^{(\tau\wedge T)-1} \Gamma(X^t) \tag{41}$$

for use in Algorithm 2.

When the boundary condition is non-zero, as for the committor, we restrict (38) to a $(k-1)$-dimensional linear system by excluding the indices $a = 1$ and $b = 1$ in (39) and (40) and setting

$$\rho(X) = \varphi_\theta^1(X^{\tau\wedge T}) - \varphi_\theta^1(X^0) + \sum_{t=0}^{(\tau\wedge T)-1} \Gamma(X^t). \tag{42}$$

In this case, the corresponding approximate solution is

$$u_\theta = \varphi_\theta^1 + \sum_{a=2}^{k} w_a \varphi_\theta^a. \tag{43}$$

This approximate solution corresponds to the one given by dynamical Galerkin approximation[10,11] with the basis $\{\varphi_\theta^a\}_{a=2}^k$ and a "guess" function of $\varphi_\theta^1$.

When the boundary conditions are zero, the orthogonalization procedure and the matrix $K$ are applied to all basis functions as in Sec. VII. When the boundary conditions are non-zero, the orthogonalization procedure is only applied to the basis functions $\{\varphi_\theta^a\}_{a=2}^k$, and $K_{a1} = I_{a1}$ the $a1$ element of the identity matrix. We summarize our procedure for prediction in Algorithm 2.

### A. Müller–Brown committor

In this section, we demonstrate the use of our method for prediction by estimating the committor for the Müller–Brown model with a shallow intermediate basin at $(-0.25, 0.5)$ (Fig. 1). Here, the sets $A$ and $B$ are defined as in Eq. (24) and $T$ is the time of first entrance to $D^{\mathrm{C}} = A \cup B$. In this case, a one-dimensional subspace iteration (i.e., $k = 1$ in Algorithm 2) appears sufficient to accurately

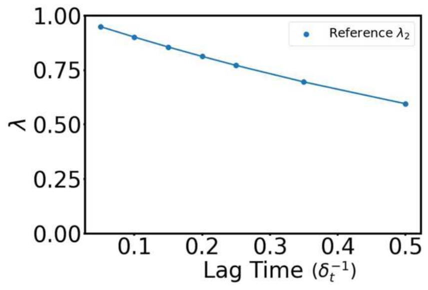

**FIG. 7.** First eigenvalue of $\mathcal{S}^\tau$ [second of $\mathcal{A}$ in (14)] for the Müller–Brown model as a function of lag time (in units of $\delta_t^{-1}$). The gap between this eigenvalue and the dominant eigenvalue, which is one, determines the rate of convergence of the Richardson iteration.

solve the prediction problem. Figure 7 shows the largest eigenvalue of the stopped transition operator $\mathcal{S}^\tau$ [the second largest of $\mathcal{A}$ in (14)] computed from our grid-based reference scheme (Sec. VI A). Richardson iteration should converge geometrically in this eigenvalue,[33] and so, for moderate lag times, we can expect our method to converge in a few dozen iterations. To initialize the algorithm we choose $\tilde{\varphi}_1^1 = \mathbb{1}_B$. All other hyperparameters are listed in Table I.

We compare the estimate of the committor from our approach with that from an MSM constructed from the same amount of data by using $k$-means to cluster the configurations outside $A$ and $B$ into 400 states and counting the transitions between clusters. In addition to the root mean square error (RMSE) for the committor itself, we show the RMSE of

$$\text{logit}_\varepsilon(q) = \log\left(\frac{\varepsilon + q}{1 + \varepsilon - q}\right) \tag{44}$$

for points outside $A$ and $B$. This function amplifies the importance of values close to zero and one. We include $\varepsilon$ because we want to assign only a finite penalty if the procedure estimates $q$ to be exactly zero or one; we use $\varepsilon = e^{-20}$.

Results as a function of lag time are shown in Fig. 8. We see that the Richardson iterate is more accurate than the MSM for all but the shortest lag times. When using the $L^2_\mu$ loss, the results are comparable, whereas the softplus loss allows the Richardson iterate to improve the RMSE of the logit function in (44) with no decrease in performance with respect to the RMSE of the committor. Results as a function of the size of the dataset are shown in Fig. 9 for a fixed lag time of $\tau = 0.1\delta_t^{-1}$. The Richardson iterate generally does as well or better than the MSM. Again, the differences are more apparent in the RMSE of the logit function in (44). By that measure, the Richardson iterate obtained with both loss functions is significantly more accurate than the MSM for small numbers of trajectories. The softplus loss maintains an advantage even for large numbers of trajectories.

## B. Accelerating convergence by incorporating eigenfunctions

As discussed in Sec. III, we expect Richardson iteration to converge slowly when the largest eigenvalue of $\mathcal{S}^\tau$, $\lambda_1$, is close to 1. More precisely, the number of iterations required to reach convergence should scale with $-1/\log\lambda_1 = \mathbb{E}[T]/\tau$, the mean escape time from the quasi-stationary distribution to the boundary of $D$ divided by the lag time. With this in mind, we can expect inexact Richardson iteration for the Müller–Brown to perform poorly if we deepen the intermediate basin at $(-0.25, 0.5)$ as in Fig. 10 (top left). Again, the sets $A$ and $B$ are defined as in (24), and $T$ is the time of first entrance to $D^C = A \cup B$. In this case, $-1/\log\lambda_1$ is on the order of 100 for the lag times we consider and, as expected, inexact Richardson iteration converges slowly (Fig. 10, bottom left). Estimates of the committor

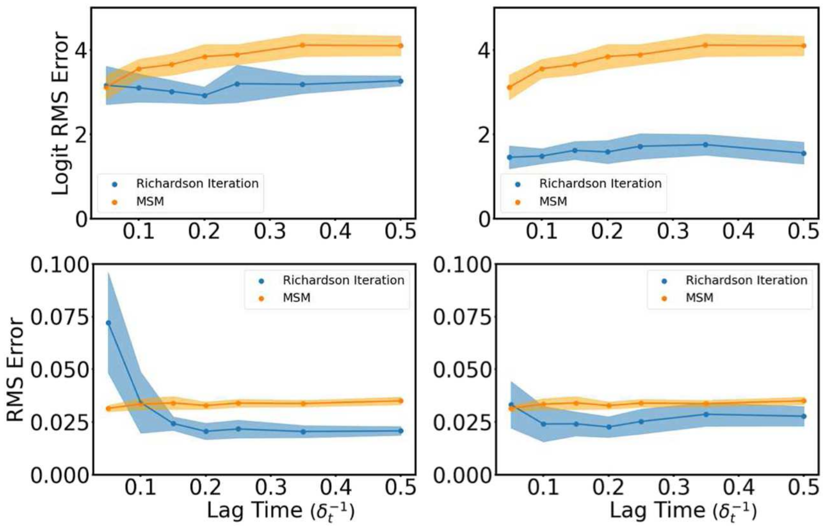

**FIG. 8.** Committor prediction for the Müller–Brown system as a function of lag time (in units of $\delta_t^{-1}$). (Left) Comparison of the inexact Richardson iteration using the $L^2_\mu$ loss and an MSM with 400 states. (Right) Same comparison using the softplus loss in place of the $L^2_\mu$ loss.

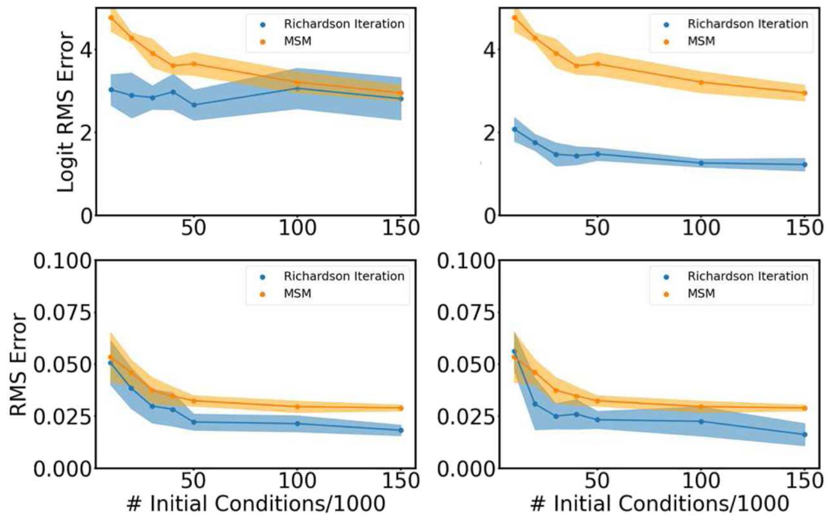

**FIG. 9.** Committor prediction for the Müller–Brown as a function of number of initial conditions for a fixed lag time of $\tau = 0.1\delta_t^{-1}$. (Left) Comparison of inexact Richardson iteration using the $L^2_\mu$ loss and an MSM with 400 states. (Right) Same comparison using the softplus loss in place of the $L^2_\mu$ loss.

by inexact Richardson iteration do not reach the correct values even after hundreds of iterations (Fig. 10, bottom right).

We now show that convergence can be accelerated dramatically by incorporating additional eigenfunctions of $\mathcal{S}^\tau$ (i.e., $k > 1$ in Algorithm 2). For the Müller–Brown model with a deepened intermediate basin, the second eigenvalue of $\mathcal{S}^\tau$ is of order $10^{-4}$ for a lag time of $\tau = 1000$ steps or $1\,\delta_t^{-1}$ (while the first is near one as discussed above). We therefore choose $k = 2$ with $\tilde{\varphi}_1^2$ initialized as a random linear combination of coordinate functions as in previous examples. We run the subspace iteration for four iterations, compute the committor as a linear combination of the resulting functions, and then refine this result with a further ten Richardson iterations (i.e., $k = 1$ with the starting vector as the output of the $k = 2$ subspace iteration). To combine the functions, we use a linear solve which incorporates memory (Algorithm 3).[58,59] We find that the use of memory improves the data-efficiency substantially for poorly conditioned problems. For our tests here, we use three memory kernels, corresponding to $\tau_{\text{mem}} = \lfloor \tau/4 \rfloor$.

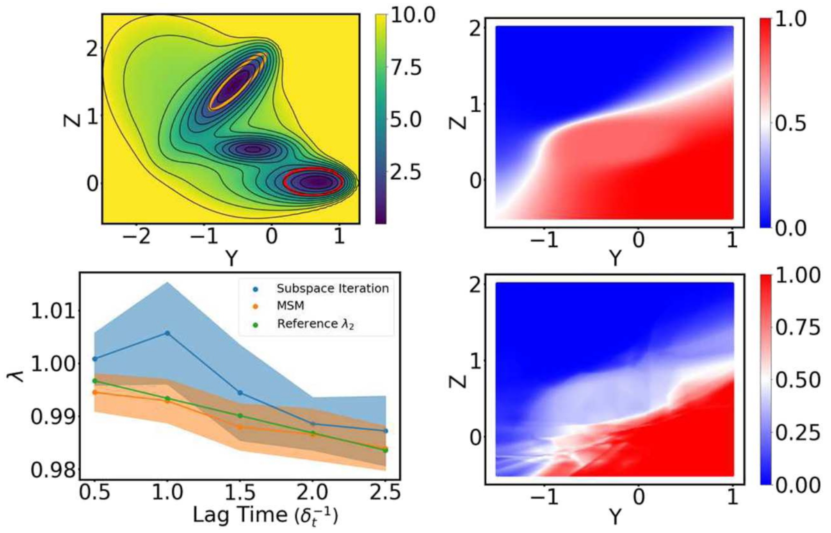

**FIG. 10.** Richardson iteration for the committor converges slowly for a Müller–Brown potential with a deepened intermediate. (Top left) Potential energy surface, with states $A$ and $B$ indicated. Contour lines are drawn every $1\,\beta^{-1}$. (Top right) Reference committor. (Bottom left) Dominant eigenvalue as a function of lag time (in units of $\delta_t^{-1}$) from an MSM with 400 states, subspace iteration, and the grid-based reference. (Bottom right) Example of the Richardson iteration after 400 iterations. Note the overfitting artifacts and lack of convergence near the intermediate state.

**ALGORITHM 3.** Memory-corrected linear solve for predictions.

```
Require: Stopped transition data $\left\{X_j^0, X_j^{1\wedge T_j}, \ldots, X_j^{\tau\wedge T_j}\right\}_{j=1}^n$, guess function $h$, reward data $\{\rho(X_j)\}_{j=1}^n$, basis set $\{f^a\}_{a=1}^k$, lag between memory kernels $\tau_{\text{mem}}$.
1:  for $s = 0 \ldots (\tau/\tau_{\text{mem}})$ do
2:      Initialize the matrix $C^s$ with zeros        ▷ $C^s \in \mathbb{R}^{(k+1)\times(k+1)}$
3:      $C_{11}^s \leftarrow 1$
4:      for $a = 2 \ldots k$ do
5:          $C_{a1}^s \leftarrow \frac{1}{n}\sum_{j=1}^n f^a(X_j^0)\rho(X_j^{s\tau_{\text{mem}}\wedge T_j})$
6:          for $b = 2 \ldots k$ do
7:              $C_{ab}^s \leftarrow \frac{1}{n}\sum_{j=1}^n f^a(X_j^0) f^b(X_j^{s\tau_{\text{mem}}\wedge T_j})$
8:          end for
9:      end for
10: end for
11: $A \leftarrow C^1 - C^0$
12: for $s = 0 \ldots (\tau/\tau_{\text{mem}}) - 2$ do
13:     $M^s \leftarrow C^{s+2} - C^{s+1} - A(C^0)^{-1}C^{s+1} - \sum_{j=0}^{s} M^j (C^0)^{-1} C^{s-j}$
14: end for
15: $A_{\text{mem}} \leftarrow A + \sum_{s=0}^{(\tau/\tau_{\text{mem}})-2} M^s$
16: Solve $A_{\text{mem}} w = w$
17: return $u = h + \sum_{a=2}^{k} w_a f^a$
```

The bottom row of Fig. 11 illustrates the idea of the subspace iteration. The second eigenfunction (Fig. 11, center) is peaked at the intermediate. As a result, the two neural-network functions linearly combined by the Galerkin approach with memory can yield a good result for the committor (Fig. 11, bottom right). Figure 12 compares the RMSE for the committor and the RMSE for the logit in (44) for Algorithm 2 with $k = 1$ (pure Richardson iteration) and $k = 2$ (incorporating the first non-trivial eigenfunction), and an MSM with 400 states. We see that the Richardson iteration suffers large errors at all lag times; as noted previously, this error is mainly in the vicinity of the intermediate. The MSM cannot accurately compute the small probabilities, but does as well as the subspace iteration in terms of RMSE.

## C. AIB$_9$ prediction results

As an example of prediction in a high-dimensional system, we compute the committor for the transition between the left- and

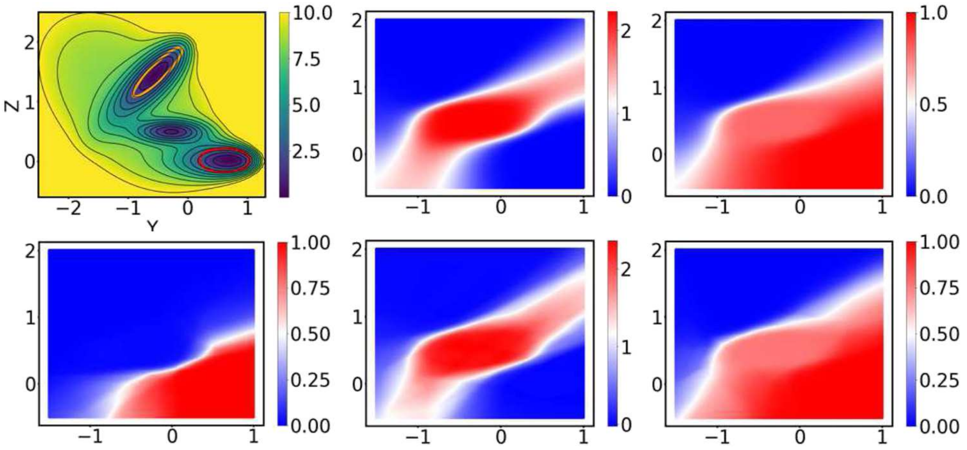

**FIG. 11.** Illustration of the subspace iteration for the Müller–Brown committor. (Top left) Modified Müller–Brown potential. (Top center) Reference second eigenfunction. (Top right) Reference committor. (Bottom left) Neural-network Richardson iterate after four iterations. (Bottom center) First non-dominant eigenfunction obtained from the neural network after four iterations. (Bottom right) Committor resulting from linear combination of the Richardson iterate and second eigenfunction. Results shown are for $\tau = 1000$ steps (i.e., $1\ \delta_t^{-1}$).

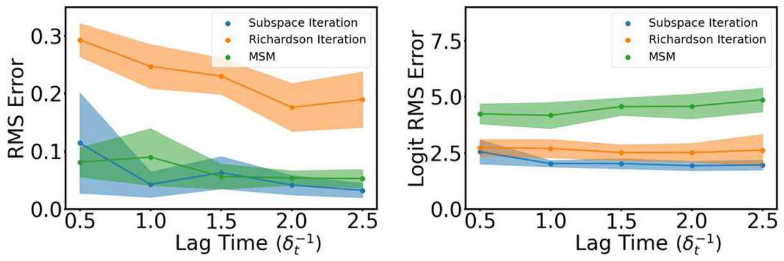

FIG. 12. Committor for the Müller–Brown potential with deepened intermediate as a function of lag time (in units of $\delta_t{}^{-1}$). (Left) Comparison of RMSE for subspace iteration as described above, Richardson iteration (as in Sec. VIII A but instead with 500 subspace iterations), and an MSM with 400 states. (Right) RMSE of the logit function in (44).

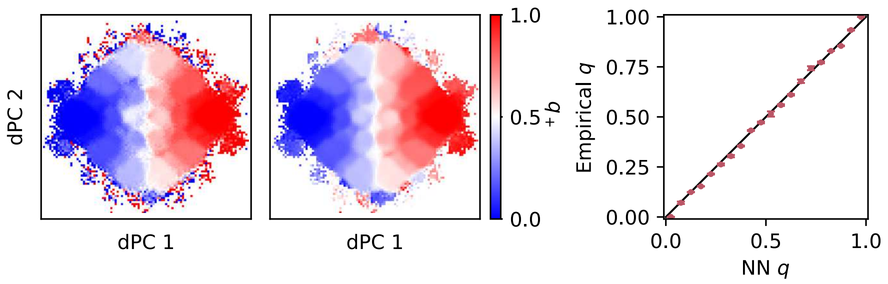

FIG. 13. AIB$_9$ committor for the transition between left- and right-handed helices. (Left) Averages of $\mathbb{1}_B(X^T)$ for initial conditions in bins in the first two dPCs computed from 20 long (5 $\mu$s) trajectories. (Middle) Averages of representative neural-network committors trained on the dataset of 6910 short (20 ns) trajectories; $\tau$ corresponds to 400 ps. (Right) Comparison between empirical committors [as defined in (45)] and the neural-network committors (trained as for the middle panel). Error bars indicate standard deviations over ten different initializations of the neural-network parameters.

right-handed helices of AIB$_9$ using the inexact Richardson iteration scheme ($k = 1$ in Algorithm 2) with the softplus loss function. Specifically, for this committor calculation, $T$ is the time of first entrance to $D^{\mathrm{C}} = A \cup B$ with $A$ and $B$ defined in Sec. VI B. As before, we initialize $\tilde{\varphi}_1^1 = \mathbb{1}_B$.

To validate our results, we use the 5 $\mu$s reference trajectories to compute an empirical committor as a function of the neural network outputs, binned into intervals

$$\bar{q}(s) = \mathbb{P}\Big[X^T \in B \,\Big|\, u_\theta(X^0) \in [s, s+\Delta s]\Big] \tag{45}$$

for $s \in [0, 1-\Delta s]$. Here, we use $\Delta s = 0.05$. The overall error in the committor estimate is defined as

$$q \text{ error} = \left(\Delta s \sum_{n=0}^{1/\Delta s - 1} \left[\bar{q}(n\Delta s) - n\Delta s\right]^2\right)^{1/2}. \tag{46}$$

While this measure of error can only be used when the dataset contains trajectories of long enough duration to reach $D^{\mathrm{C}}$, it has the advantage that it does not depend on the choice of projection that we use to visualize the results.

Results for the full dataset with $\tau$ corresponding to 400 ps are shown in Fig. 13. The projection on the principal components is consistent with the symmetry of the system, and the predictions show good agreement with the empirical committors. As $\tau$ decreases, the results become less accurate (Fig. 14, top left); at shorter lag times we would expect further increases in the error. We also examine

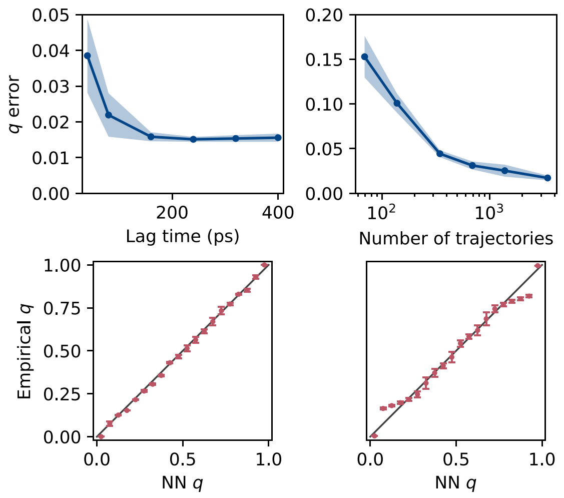

FIG. 14. AIB$_9$ committor for the transition between left- and right-handed helices, as functions of lag time (in ps) and number of initial conditions. (Top left) Error in the committor as a function of lag time (in ps). Shading indicates the standard deviation over ten different initializations of the neural-network parameters. (Top right) Error in the committor as a function of the number of initial conditions with $\tau$ corresponding to 160 ps. Shading indicates the standard deviation over ten different random samples of the trajectories. (Bottom) Comparison between empirical committors and neural-network committors trained on datasets with (left) 1/2 and (right) 1/20 of the short trajectories. Error bars indicate standard deviations over ten random samples of the trajectories.

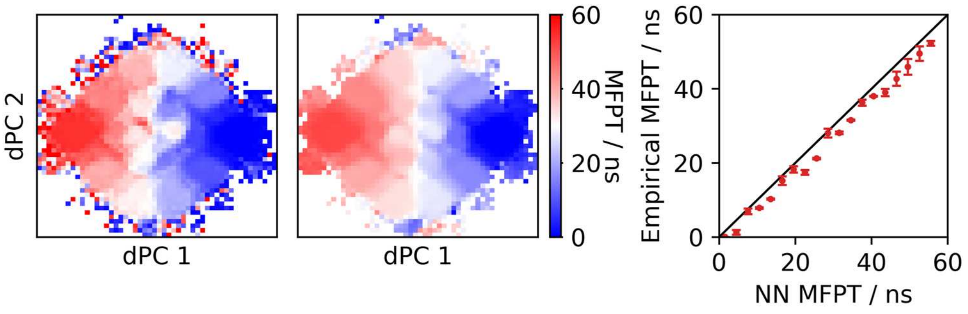

**FIG. 15.** $AIB_9$ MFPT to the right-handed helix. (Left) Averages of the time to next reach $B$ for initial conditions in bins in the first two dPCs computed from 20 long ($5\ \mu$s) trajectories. (Middle) Averages of representative neural-network committors trained on the dataset of 6910 short (20 ns) trajectories; $\tau$ corresponds to 400 ps. (Right) Comparison between empirical committors [as defined in (47)] and the neural-network committors (trained as for the middle panel). Error bars indicate standard deviations over ten different initializations of the neural-network parameters.

the dependence of the results on the size of the dataset by subsampling the short trajectories and then training neural networks on the reduced set of trajectories (Fig. 14, top right). We find that the performance steadily drops as the number of trajectories is reduced and degrades rapidly for the datasets subsampled more than 20-fold (Fig. 14, bottom right), corresponding to about 7 $\mu$s of total sampling.

Finally, we compute the MFPT to reach the right-handed helix using the same dataset. For the MFPT calculation $T$ is the time of first entrance to $D^C = B$. Note that the time of first entrance to $B$ includes long dwell times in $A$ and is expected to be much larger than the time of first entrance to $A \cup B$.

We compare against an empirical estimate of the MFPT defined by

$$\bar{m}(s) = \mathbb{E}\Big[T\,\Big|\,u_\theta(X^0) \in [s, s + \Delta s]\Big] \tag{47}$$

for $s \in [0, m_{\max} - \Delta s]$, where $\Delta s = 3$ and $m_{\max} = 57$ ns. Overall error is defined analogously to Eq. (46).

In Fig. 15, we show the MFPT obtained from Algorithm 2 with $k = 5$ and the $L^2_\mu$ loss function. Initially, we set $\tilde{\varphi}^1_1$ equal to an arbitrary positive function (we use $5\mathbb{1}_A$) and $\tilde{\varphi}^a_s$ for $a > 1$ to a random linear combination of coordinate functions. In Fig. 16, we examine the convergence of the MFPT from the left-handed helix to

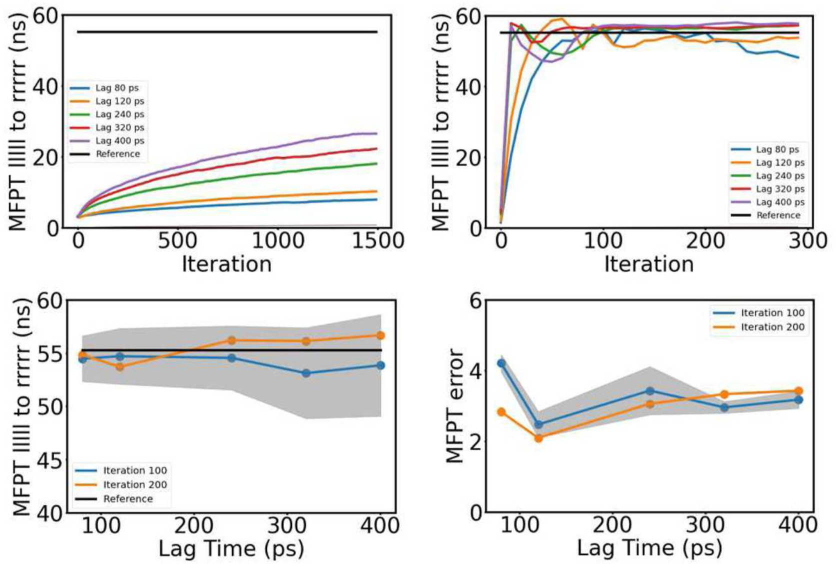

**FIG. 16.** MFPT between left- and right-handed helices for the $AIB_9$ system. (Top left) Convergence of Richardson iteration. (Top right) Convergence of a five-dimensional subspace iteration. (Bottom left) MFPT after 100 and 200 subspace iterations as a function of lag time. Shading indicates standard deviations over ten different initializations of the neural-network parameters. (Bottom right) Overall error in MFPT. To obtain the results shown in this figure, we first use the short-trajectory dataset to train neural networks to predict the MFPT; we then use these networks with fixed parameters to evaluate the MFPT for all structures in the long reference trajectories and average the results for those structures in the left-handed helix state. The horizontal lines in the top panels are obtained from averaging the time to the right-handed helix for the same structures.

the right-handed helix for the MFPT computed with $k = 1$ (pure Richardson iteration) and $k = 5$. To obtain the results shown, we train neural networks on the short-trajectory dataset and then average their predictions for structures in the left-handed helix state in the long reference trajectories. The horizontal line in Fig. 16 indicates a MFPT of about 56 ns estimated from the long reference trajectories. We see that the algorithm with $k = 5$ converges much faster (note the differing scales of the horizontal axes) and yields accurate results at all lag times other than the shortest shown. The need to choose $k > 1$ for this MFPT calculation is again consistent with theoretical convergence behavior of exact subspace iteration. Because the typical time of first entrance to $B$ from points in $A$ is very large, we expect the dominant eigenvalue of $\mathcal{S}^\tau$ to be very near to one when $D = B^C$. In contrast, the committor calculation benefits from the fact that the time of first entrance to $A \cup B$ is much shorter, implying a smaller dominant eigenvalue of $\mathcal{S}^\tau$ when $D = (A \cup B)^C$.

## IX. CONCLUSIONS

In this work, we have presented a method for spectral estimation and rare-event prediction from short-trajectory data. The key idea is that we use the data as the basis for an inexact subspace iteration. For the test systems that we considered, the method not only outperforms high-resolution MSMs, but it can be tuned through the choice of loss function to compute committor probabilities accurately near the reactants, transition states, and products. Other than the Markov assumption, our method requires no knowledge of the underlying model and puts no restrictions on its dynamics.

As discussed in prior neural-network based prediction work,[24,30] our method is sensitive to the quality and distribution of the initial sampling data. However, our work shares with Ref. 24 the major advantage of allowing the use of arbitrary inner products. This enables adaptive sampling of the state space[24,60] and—together with the features described above—the application to observational and experimental data, for which the stationary distribution is generally unavailable.

In the present work, we focused on statistics of transition operators, but our method can readily be extended to solve problems involving their adjoint operators as well. By this means, we can obtain the stationary distribution as well as the backward committor. The combination of forward and backward predictions allows the analysis of transition paths using transition path theory without needing to generate full transition paths[37,38,48] and has been used to understand rare transition events in molecular dynamics[10,13,16,17,61,62] and geophysical flows.[63–67] We leave these extensions to future work.

In cases in which trajectories that reach the reactant and product states are available, it would be interesting to compare our inexact iterative schemes against schemes for committor approximation based on logistic regression and related approaches.[35,41–46,68] These schemes are closely related to what is called "Monte Carlo" approximation in reinforcement learning,[31] and also to the inexact Richardson iteration that we propose here with $\tau \to \infty$.

We have seen that temporal difference (TD) learning, a workhorse for prediction in reinforcement learning, is closely related to an inexact form of Richardson iteration. Variants like TD($\lambda$), have similar relationships to inexact iterative schemes. As we showed, subspace iteration is a natural way of addressing slow convergence. We thus anticipate that our results have implications for reinforcement learning, particularly in scenarios in which the value function depends on the occurrence of some rare-event. Finally, we note that our framework can be extended to the wider range of iterative numerical linear algebra algorithms. In particular, Krylov or block Krylov subspace methods may offer further acceleration. In fact, very recently an approach along these lines was introduced for value function estimation in reinforcement learning.[69]

## ACKNOWLEDGMENTS

We are grateful to Arnaud Doucet for pointing our attention to TD methods and the inexact power iteration in Ref. 32, both of which were key motivations for this work. We also thank Joan Bruna for helpful conversations about reinforcement learning. This work was supported by the National Institutes of Health (Award No. R35 GM136381) and the National Science Foundation (Award No. DMS-2054306). S.C.G. acknowledges support by the National Science Foundation Graduate Research Fellowship under Grant No. 2140001. J.W. acknowledges support from the Advanced Scientific Computing Research Program within the DOE Office of Science through Award No. DE-SC0020427. This work was completed in part with resources provided by the University of Chicago Research Computing Center, and we are grateful for their assistance with the calculations. "Beagle-3: A Shared GPU Cluster for Biomolecular Sciences" was supported by the National Institutes of Health (NIH) under the High-End Instrumentation (HEI) Grant Program Award No. 1S10OD028655-0.

## AUTHOR DECLARATIONS

### Conflict of Interest

The authors have no conflicts to disclose.

### Author Contributions

J.S., S.C.G., A.R.D., and J.W. conceptualized research. J.S. developed the method. C.L. adapted the linear solve used for prediction from Refs. 58 and 59. J.S. and S.C.G. performed the numerical tests. A.R.D. and J.W. supervised the research. All authors wrote and edited the manuscript.

**John Strahan**: Conceptualization (equal); Investigation (equal); Writing – original draft (equal); Writing – review & editing (equal). **Spencer C. Guo**: Conceptualization (equal); Investigation (equal); Writing – original draft (equal); Writing – review & editing (equal). **Chatipat Lorpaiboon**: Methodology (equal); Writing – review & editing (equal). **Aaron R. Dinner**: Conceptualization (equal); Funding acquisition (equal); Supervision (equal); Writing – original draft (equal); Writing – review & editing (equal). **Jonathan Weare**: Conceptualization (equal); Funding acquisition (equal); Supervision (equal); Writing – original draft (equal); Writing – review & editing (equal).

## DATA AVAILABILITY

The data that support the findings of this study are available within the article. Code implementing the algorithms and a Jupyter notebook illustrating use of the method on the Müller–Brown example are available at https://github.com/dinner-group/inexact-subspace-iteration.

## REFERENCES

[1] F. Noé and F. Nüske, "A variational approach to modeling slow processes in stochastic dynamical systems," Multiscale Model. Simul. **11**(2), 635–655 (2013).
[2] F. Nüske, B. G. Keller, G. Pérez-Hernández, A. S. J. S. Mey, and F. Noé, "Variational approach to molecular kinetics," J. Chem. Theory Comput. **10**(4), 1739–1752 (2014).
[3] S. Klus, F. Nüske, P. Koltai, H. Wu, I. Kevrekidis, C. Schütte, and F. Noé, "Data-driven model reduction and transfer operator approximation," J. Nonlinear Sci. **28**, 985–1010 (2018).
[4] R. J. Webber, E. H. Thiede, D. Dow, A. R. Dinner, and J. Weare, "Error bounds for dynamical spectral estimation," SIAM J. Math. Data Sci. **3**(1), 225–252 (2021).
[5] C. Lorpaiboon, E. H. Thiede, R. J. Webber, J. Weare, and A. R. Dinner, "Integrated variational approach to conformational dynamics: A robust strategy for identifying eigenfunctions of dynamical operators," J. Phys. Chem. B **124**(42), 9354–9364 (2020).
[6] R. T. McGibbon, B. E. Husic, and V. S. Pande, "Identification of simple reaction coordinates from complex dynamics," J. Chem. Phys. **146**(4), 044109 (2017).
[7] L. Busto-Moner, C.-J. Feng, A. Antoszewski, A. Tokmakoff, and A. R. Dinner, "Structural ensemble of the insulin monomer," Biochemistry **60**(42), 3125–3136 (2021).
[8] G. Pérez-Hernández, F. Paul, T. Giorgino, G. De Fabritiis, and F. Noé, "Identification of slow molecular order parameters for Markov model construction," J. Chem. Phys. **139**(1), 015102 (2013).
[9] C. R. Schwantes and V. S. Pande, "Improvements in Markov State Model construction reveal many non-native interactions in the folding of NTL9," J. Chem. Theory Comput. **9**(4), 2000–2009 (2013).
[10] J. Strahan, A. Antoszewski, C. Lorpaiboon, B. P. Vani, J. Weare, and A. R. Dinner, "Long-time-scale predictions from short-trajectory data: A benchmark analysis of the trp-cage miniprotein," J. Chem. Theory Comput. **17**(5), 2948–2963 (2021).
[11] E. H. Thiede, D. Giannakis, A. R. Dinner, and J. Weare, "Galerkin approximation of dynamical quantities using trajectory data," J. Chem. Phys. **150**(24), 244111 (2019).
[12] W. C. Swope, J. W. Pitera, and F. Suits, "Describing protein folding kinetics by molecular dynamics simulations. 1. Theory," J. Phys. Chem. B **108**(21), 6571–6581 (2004).
[13] N. Frank, C. Schütte, E. Vanden-Eijnden, L. Reich, and T. R. Weik, "Constructing the equilibrium ensemble of folding pathways from short off-equilibrium simulations," Proc. Natl. Acad. Sci. U. S. A. **106**(45), 19011–19016 (2009).
[14] *An Introduction to Markov State Models and Their Application to Long Timescale Molecular Simulation*, edited by G. R. Bowman, V. S. Pande, and N. Frank, *Volume 797 of Advances in Experimental Medicine and Biology* (Springer Netherlands, Dordrecht, 2014).
[15] J. Finkel, R. J. Webber, D. S. Abbot, E. P. Gerber, and J. Weare, "Learning forecasts of rare stratospheric transitions from short simulations," Mon. Weather Rev. **149**(11), 3647–3669 (2021).
[16] A. Antoszewski, C. Lorpaiboon, J. Strahan, and A. R. Dinner, "Kinetics of phenol escape from the insulin $R_6$ hexamer," J. Phys. Chem. B **125**(42), 11637–11649 (2021).
[17] S. C. Guo, R. Shen, B. Roux, and A. R. Dinner, "Dynamics of activation in the voltage-sensing domain of Ci-VSP," bioRxiv:10.1101/2022.12.19.521128 (2022).
[18] G. Andrew, R. Arora, J. Bilmes, and K. Livescu, "Deep canonical correlation analysis," in *Proceedings of the 30th International Conference on Machine Learning* (PMLR, 2013), pp. 1247–1255.
[19] A. Mardt, L. Pasquali, H. Wu, and N. Frank, "VAMPnets for deep learning of molecular kinetics," Nat. Commun. **9**(1), 5 (2018).
[20] C. Wehmeyer and F. Noé, "Time-lagged autoencoders: Deep learning of slow collective variables for molecular kinetics," J. Chem. Phys. **148**(24), 241703 (2018).
[21] B. Lusch, J. N. Kutz, and S. L. Brunton, "Deep learning for universal linear embeddings of nonlinear dynamics," Nat. Commun. **9**(1), 4950 (2018).
[22] W. Chen, H. Sidky, and A. L. Ferguson, "Nonlinear discovery of slow molecular modes using state-free reversible VAMPnets," J. Chem. Phys. **150**(21), 214114 (2019).
[23] A. Glielmo, B. E. Husic, A. Rodriguez, C. Clementi, N. Frank, and A. Laio, "Unsupervised learning methods for molecular simulation data," Chem. Rev. **121**, 9722 (2021).
[24] J. Strahan, J. Finkel, A. R. Dinner, and J. Weare, "Predicting rare events using neural networks and short-trajectory data," J. Comput. Phys. **488**, 112152 (2023).
[25] H. Li, Y. Khoo, Y. Ren, and L. Ying, "A semigroup method for high dimensional committor functions based on neural network," in *Proceedings of the 2nd Mathematical and Scientific Machine Learning Conference* (PMLR, 2022), pp. 598–618.
[26] Y. Khoo, J. Lu, and L. Ying, "Solving for high-dimensional committor functions using artificial neural networks," Res. Math. Sci. **6**(1), 1 (2018).
[27] Q. Li, B. Lin, and W. Ren, "Computing committor functions for the study of rare events using deep learning," J. Chem. Phys. **151**(5), 054112 (2019).
[28] B. Roux, "String method with swarms-of-trajectories, mean drifts, lag time, and committor," J. Phys. Chem. A **125**(34), 7558–7571 (2021).
[29] B. Roux, "Transition rate theory, spectral analysis, and reactive paths," J. Chem. Phys. **156**(13), 134111 (2022).
[30] G. M. Rotskoff, A. R. Mitchell, and E. Vanden-Eijnden, "Active importance sampling for variational objectives dominated by rare events: Consequences for optimization and generalization," in *Proceedings of the 2nd Mathematical and Scientific Machine Learning Conference* (PMLR, 2022), pp. 757–780.
[31] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*, 2nd ed.(The MIT Press, Cambridge, MA, 2018).
[32] J. Wen, B. Dai, L. Li, and S. Dale, "Batch stationary distribution estimation," in *Proceedings of the 37th International Conference on Machine Learning, ICML'20* (JMLR.org, 2020), pp. 10203–10213.
[33] G. H. Golub and C. F. Van Loan, *Matrix Computations*, 3rd ed. (The Johns Hopkins University Press, 1996).
[34] R. Du, V. S. Pande, A. Y. Grosberg, T. Tanaka, and E. S. Shakhnovich, "On the transition coordinate for protein folding," J. Chem. Phys. **108**(1), 334–350 (1998).
[35] A. Ma and A. R. Dinner, "Automatic method for identifying reaction coordinates in complex systems," J. Phys. Chem. B **109**(14), 6769–6779 (2005).
[36] S. V. Krivov, "On reaction coordinate optimality," J. Chem. Theory Comput. **9**(1), 135–146 (2013).
[37] W. E and E. Vanden-Eijnden, "Transition-path theory and path-finding algorithms for the study of rare events," Annu. Rev. Phys. Chem. **61**, 391–420 (2010).
[38] E. Vanden-Eijnden, "Transition path theory," in *Computer Simulations in Condensed Matter Systems: From Materials to Chemical Biology* (Springer, 2006), Vol. 1, pp. 453–493.
[39] P. Collett, S. Martinez, and J. San Martin, "Quasi-stationary distributions," in *Probability and its Applications* (Springer Berlin Heidelberg, 2012).
[40] X. Nguyen, M. J. Wainwright, and M. I. Jordan, "Estimating divergence functionals and the likelihood ratio by convex risk minimization," IEEE Trans. Inf. Theory **56**(11), 5847–5861 (2010).
[41] B. Peters and B. L. Trout, "Obtaining reaction coordinates by likelihood maximization," J. Chem. Phys. **125**(5), 054108 (2006).
[42] B. Peters, G. T. Beckham, and B. L. Trout, "Extensions to the likelihood maximization approach for finding reaction coordinates," J. Chem. Phys. **127**(3), 034109 (2007).
[43] H. Jung, R. Covino, and G. Hummer, "Artificial intelligence assists discovery of reaction coordinates and mechanisms from molecular dynamics simulations," arXiv:1901.04595 (2019).
[44] A. Chattopadhyay, E. Nabizadeh, and P. Hassanzadeh, "Analog forecasting of extreme-causing weather patterns using deep learning," J. Adv. Model. Earth Syst. **12**(2), e2019MS001958 (2020).

[45]H. Jung, R. Covino, A. Arjun, C. Leitold, C. Dellago, P. G. Bolhuis, and G. Hummer, "Machine-guided path sampling to discover mechanisms of molecular self-organization," Nat. Comput. Sci. **3**(4), 334–345 (2023).
[46]M. George, B. Cozian, P. Abry, P. Borgnat, and F. Bouchet, "Probabilistic forecasts of extreme heatwaves using convolutional neural networks in a regime of lack of data," Phys. Rev. Fluids **8**(4), 040501 (2023).
[47]K. Müller and L. D. Brown, "Location of saddle points and minimum energy paths by a constrained simplex optimization procedure," Theor. Chim. Acta **53**(1), 75–93 (1979).
[48]C. Lorpaiboon, J. Weare, and A. R. Dinner, "Augmented transition path theory for sequences of events," J. Chem. Phys. **157**(9), 094115 (2022).
[49]S. Buchenberg, N. Schaudinnus, and G. Gerhard Stock, "Hierarchical biomolecular dynamics: Picosecond hydrogen bonding regulates microsecond conformational transitions," J. Chem. Theory Comput. **11**(3), 1330–1336 (2015).
[50]A. Perez, F. Sittel, G. Stock, and K. Dill, "MELD-path efficiently computes conformational transitions, including multiple and diverse paths," J. Chem. Theory Comput. **14**(4), 2109–2116 (2018).
[51]F. Sittel, T. Filk, and G. Stock, "Principal component analysis on a torus: Theory and application to protein dynamics," J. Chem. Phys. **147**(24), 244101 (2017).
[52]C. W. Hopkins, R. C. Walker, S. Le Grand, and A. E. Roitberg, "Long-Time-step molecular dynamics through hydrogen mass repartitioning," J. Chem. Theory Comput. **11**(4), 1864–1874 (2015).
[53]G. A. Khoury, J. Smadbeck, P. Tamamis, A. C. Vandris, C. A. Kieslich, and C. A. Floudas, "Ab Initio charge parameters to aid in the discovery and design of therapeutic proteins and peptides with unnatural amino acids and their application to complement inhibitors of the compstatin family," ACS Synth. Biol. **3**(12), 855–869 (2014).
[54]H. Nguyen, D. R. Roe, and C. Simmerling, "Improved generalized Born solvent model parameters for protein simulations," J. Chem. Theory Comput. **9**(4), 2020–2034 (2013).
[55]P. Eastman, J. Swails, J. D. Chodera, R. T. McGibbon, Y. Zhao, K. A. Beauchamp, L.-P. Wang, A. C. Simmonett, M. P. Harrigan, C. D. Stern, R. P. Wiewiora, B. R. Brooks, and V. S. Pande, "OpenMM 7: Rapid development of high performance algorithms for molecular dynamics," PLoS Comput. Biol. **13**(7), e1005659 (2017).
[56]P. Metzner, C. Schütte, and E. Vanden-Eijnden, "Illustration of transition path theory on a collection of simple examples," J. Chem. Phys. **125**(8), 084110 (2006).
[57]D. P. Kingma and B. Jimmy, "Adam: A method for stochastic optimization," arXiv:1412.6980 (2014).
[58]E. Darve, J. Solomon, and A. Kia, "Computing generalized Langevin equations and generalized Fokker-Planck equations," Proc. Natl. Acad. Sci. U. S. A. **106**(27), 10884–10889 (2009).
[59]S. Cao, A. Montoya-Castillo, W. Wang, T. E. Markland, and X. Huang, "On the advantages of exploiting memory in Markov state models for biomolecular dynamics," J. Chem. Phys. **153**(1), 014105 (2020).
[60]D. Lucente, J. Rolland, C. Herbert, and F. Bouchet, "Coupling rare event algorithms with data-based learned committor functions using the analogue Markov chain," J. Stat. Mech.: Theory Exp. **2022**(8), 083201 (2022).
[61]Y. Meng, D. Shukla, V. S. Pande, and B. Roux, "Transition path theory analysis of c-Src kinase activation," Proc. Natl. Acad. Sci. U. S. A. **113**(33), 9193–9198 (2016).
[62]B. P. Vani, J. Weare, and A. R. Dinner, "Computing transition path theory quantities with trajectory stratification," J. Chem. Phys. **157**(3), 034106 (2022).
[63]J. Finkel, D. S. Abbot, and J. Weare, "Path properties of atmospheric transitions: Illustration with a low-order sudden stratospheric warming model," J. Atmos. Sci. **77**(7), 2327–2347 (2020).
[64]P. Miron, F. J. Beron-Vera, L. Helfmann, and P. Koltai, "Transition paths of marine debris and the stability of the garbage patches," Chaos **31**(3), 033101 (2021).
[65]D. Lucente, C. Herbert, and F. Bouchet, "Committor functions for climate phenomena at the predictability margin: The example of El Niño-Southern Oscillation in the Jin and Timmermann model," J. Atmos. Sci. **79**(9), 2387–2400 (2022).
[66]J. Finkel, E. P. Gerber, S. A. Dorian, and J. Weare, "Revealing the statistics of extreme events hidden in short weather forecast data," AGU Adv. **4**(2), e2023AV000881 (2023).
[67]J. Finkel, R. J. Webber, E. P. Gerber, D. S. Abbot, and J. Weare, "Data-driven transition path analysis yields a statistical understanding of sudden stratospheric warming events in an idealized model," J. Atmos. Sci. **80**(2), 519–534 (2023).
[68]J. Hu, A. Ma, and A. R. Dinner, "A two-step nucleotide-flipping mechanism enables kinetic discrimination of DNA lesions by AGT," Proc. Natl. Acad. Sci. U. S. A. **105**(12), 4615–4620 (2008).
[69]E. Xia and M. Wainwright, "Krylov-Bellman boosting: Super-linear policy evaluation in general state spaces," in *Proceedings of The 26th International Conference on Artificial Intelligence and Statistics, Volume 206 of Proceedings of Machine Learning Research*, edited by F. Ruiz, D. Jennifer, and J.-W. van de Meent (PMLR, 2023), pp. 9137–9166.