Learning Koopman eigenfunctions of stochastic diffusions with optimal importance sampling and ISOKANN

View Online   Export Citation   CrossMark

A. Sikorski,[1,2,a] [iD] E. Ribera Borrell,[1,2] [iD] and M. Weber[1] [iD]

AFFILIATIONS

[1] Zuse Institute Berlin, 14195 Berlin, Germany
[2] Institute of Mathematics, Freie Universität Berlin, 14195 Berlin, Germany

Note: Paper published as part of the Special Topic on Cascades of Scales: Applications and Mathematical Methodologies.
[a] Author to whom correspondence should be addressed: sikorski@zib.de

ABSTRACT

The dominant eigenfunctions of the Koopman operator characterize the metastabilities and slow-timescale dynamics of stochastic diffusion processes. In the context of molecular dynamics and Markov state modeling, they allow for a description of the location and frequencies of rare transitions, which are hard to obtain by direct simulation alone. In this article, we reformulate the eigenproblem in terms of the ISOKANN framework, an iterative algorithm that learns the eigenfunctions by alternating between short burst simulations and a mixture of machine learning and classical numerics, which naturally leads to a proof of convergence. We furthermore show how the intermediate iterates can be used to reduce the sampling variance by importance sampling and optimal control (enhanced sampling), as well as to select locations for further training (adaptive sampling). We demonstrate the usage of our proposed method in experiments, increasing the approximation accuracy by several orders of magnitude.

I. INTRODUCTION

Many real-world stochastic processes contain rare events, for example, folding and binding events in molecular systems. The analysis of the frequency and mechanism of these events often takes operators associated with the process and their dominant invariant subspaces into account.[1–5] Usually, this type of analysis leads to a kind of chicken and egg problem. In order to compute the dominant invariant subspace of the Koopman operator of the process, one has to somehow also "sample those events that are usually rare." However, in order to know how to generate a bias on the process for observing these events, one would need information from the dominant invariant subspace of the Koopman operator.

The key idea of this article is to take an iterative algorithm that approximates the dominant eigenfunctions of the operator and to use the intermediate approximations for sampling along the reaction paths and generating an optimal bias to observe the relevant events. The algorithm for approximating eigenfunctions is ISOKANN[4] (Invariant subspaces of Koopman operators with artificial neural networks), and the bias is computed according to the theory of optimal importance sampling and the change of path measures.[6]

Briefly speaking, ISOKANN can be thought of as an Arnoldi-like method using neural networks as function representations and replacing the subspace projections by a transformation suitable to its application on neural networks. Instead of computing the eigenfunctions themselves, it stores linear combinations of them, the so-called χ-functions. These span an invariant subspace of the Koopman operator and can be used to reconstruct the eigenfunctions. Furthermore the χ-functions themselves allow for an interpretation as reaction coordinates indicating the locations of rare events and their reaction paths.[7,8]

02 February 2024 08:34:30

ALGORITHM 1. ISOKANN.

Input: N the number of power iterations
   M the number of x-samples
   K the number of Koopman Monte-Carlo samples
   χ_0 initial neural network
Output: χ_N approximates a χ function [cf. Eq. (5)]
1: algpx@indentEndPage-0 **for** $n = 0$ to $N - 1$ **do**
2:   algpx@indentEndPage-1 **for** $m = 1$ to M **do**
3:     $x_m \leftarrow \text{SAMPLE}X_0()$ ▷ sample training points
4:     algpx@indentEndPage-2 **for** $k = 1$ **to** K **do**
5:       $y_k \leftarrow \text{SAMPLE}X_T(x_m)$ ▷ simulate trajectories
6:       $\kappa_m \leftarrow \frac{1}{K}\sum_k \chi_n(y_k)$ ▷ Koopman approximation
7:     $s \leftarrow S(\kappa)$ ▷ transformed target data
8:     $\Delta \leftarrow \nabla_{\theta_n}\sum_m \left(\chi_n(x_m) - s_m\right)^2$ ▷ compute loss gradient
9:     $\chi_{n+1} \leftarrow \text{OPTIM}(\chi_n, \Delta)$ ▷ train the neural network
10: **return** χ_N
Subroutines:
   SAMPLEX_0: Subroutine sampling the starting points x_m - either uniform
   or χ-*stratified* (Sec. II F).
   SAMPLEX_T: SDE solver, e.g., Euler-Maruyama - either uncontrolled
   or controlled (Sec. III D).
   S: empirical shift-scale or PCCA$^+$ (Sec. II D or Sec. II C).
   OPTIM: Gradient based optimization of the neural network (e.g., 100 SGD steps)

Using this interpretation, the χ-functions obtained during previous iterations can be used to adapt the sample locations for further iterations. For example, this is done in Sec. II F by χ-*stratified sampling*, which provides better global coverage and facilitates exploration. The theory of optimal importance sampling on the other hand allows to exploit the information locally by decreasing the variance of the samples required to approximate the action of the Koopman operator. This in turn allows ISOKANN to arrive at results either more quickly or more precisely.

It is due to this feedback loop, coupling local and global information, together with its representation of the χ-functions as neural networks that we expect ISOKANN to perform well even for complex high-dimensional systems. While these general ideas are far from being formalized yet, in this article we will try to construct the basic building blocks in a way amenable for their future analysis.

In the following, we start by recalling basic knowledge about the eigenfunctions of Koopman operators (Sec. II A) and provide an abstract formulation of the ISOKANN problem in terms of χ-functions (Sec. II B) encoding the invariant subspaces of the Koopman operator. This abstract formulation is incomplete without the choice of an adequate transformation, which we discuss in Sec. II C. In particular, we show how the explicit solution to PCCA$^+$ with two eigenfunctions leads to 1D-ISOKANN which corresponds to the classical ISOKANN algorithm in 4. After proving convergence of 1D-ISOKANN we show how to reconstruct the eigenfunctions of the Koopman operator from the χ-functions in Sec. II E. We then conclude Sec. II by providing an algorithmic description in form of Algorithm 1 and discussing the actual sampling and computation procedure (Sec. II F). Section III starts with an introduction to the theory of optimal importance sampling of classical random variables (Sec. III A) before reciting the result for the optimal sampling of path observables for diffusion processes with the Girsanov re-weighting in Sec. III B. After showing how to apply this result to obtain a zero variance sampler for the evaluation of the Koopman operator (Sec. III C) we discuss how to integrate the result into the ISOKANN framework (Sec. III D). In the end (Sec. III E) we apply the 1D-ISOKANN algorithm to a one-dimensional double-well potential and observe the improvement of the accuracy of the controlled vs the uncontrolled case.

II. ISOKANN THEORY

In this chapter we recall the basics of Koopman operator theory and summarize the ISOKANN method while also complementing it by some new results. In particular, we provide a new dimension-agnostic formulation (6) which naturally leads to a possible extension of ISOKANN to higher dimensions. We then show how the classical 1D-ISOKANN can be seen as a special case of this formulation and prove convergence of the algorithm to a χ function (Theorem 1). After showing how to reconstruct the dominant eigenfunctions from the ISOKANN result (Proposition 1) we conclude this section with a discussion of the actual implementation of ISOKANN, suggesting a new adaptive sampling scheme. In the next chapter, we will then introduce the key idea of this paper, namely optimal importance sampling in order to estimate the Koopman expectations with less variance.

A. The Koopman operator

Although our approach can be generalized to non-reversible stochastic processes (by shifting the focus from eigenfunctions to invariant subspaces), for simplicity we will restrict our explanations to the reversible case. More precisely, we will investigate potential-driven diffusion processes $X = (X_t)_{t\geq0}$ of the form

$$\mathrm{d}X_t = b(X_t)\mathrm{d}t + \sigma\mathrm{d}B_t, \tag{1}$$

taking values in the state space $\mathbf{X} = \mathbb{R}^n$ with constant diffusion term $\sigma \in \mathbb{R}^{n\times n}$ and a force-field $b : \mathbf{X} \to \mathbb{R}^n$ that (w.l.o.g.) is given by the gradient of a smooth potential $b = -\nabla U$, $U : \mathbf{X} \to \mathbb{R}$.

The Koopman operator for lag time T, $\mathbf{K}^T : L^\infty(\mathbf{X}) \to L^\infty(\mathbf{X})$, applied to a function $f \in L^\infty(\mathbf{X})$ is defined by its pointwise evaluation via

$$\left(\mathbf{K}^T f\right)(x) = \mathbf{E}[f(X_T)|X_0 = x] \tag{2}$$

i.e. the expectation value of f at time T when starting the system in $X_0 = x$. Recall that since the process is time-homogeneous the Koopman operator just depends on the lag time for any start time $t \geq 0$

$$\mathbf{E}[f(X_{t+T})|X_t = x] = \left(\mathbf{K}^T f\right)(x).$$

The eigenfunctions $v_i \in L^\infty(\mathbf{X})$ of $\mathbf{K}^T$ satisfy for all lag times $T \geq 0$

$$\mathbf{K}^T v_i = \lambda_i(T)\, v_i, \qquad\qquad i = 1, 2, \ldots \tag{3}$$

$$\lambda_i(T) = \exp\left(T q_i\right), \qquad\qquad 0 = q_1 > q_2 \geq \ldots \tag{4}$$

with time-dependent eigenvalues $\lambda_i(T)$, exponential in time with rates q_i (which in turn are the eigenvalues of the corresponding infinitesimal generator of $\mathbf{K}^T$). In the following we will refer to $v_1, \ldots, v_d$ as the *d dominant eigenfunctions* and call $v_1 \equiv 1$ the *trivial eigenfunction*. When clear from the context or of no importance we will omit the lag time T and simply speak of the Koopman operator $\mathbf{K}$.

The dominant eigenfunctions are of particular interest as they decay the slowest and hence dominate the long time behavior of the system. The number d of eigenfunctions of interest depends on the time scales of the system and is usually chosen up to a spectral gap.

There exist different approaches to estimate the eigenfunctions. Many depend on the discretization of the state-space into cells leading to a matrix representation of $\mathbf{K}$. A classical method is starting trajectories in each such cell and counting how many of them end up in a certain cell. This sample-driven method can be interpreted as an approximate Ulam/Galerkin discretization onto indicator functions of the cells.[9] The eigenfunctions of $\mathbf{K}$ are then approximated by the eigenfunctions of its (dense) matrix approximation. Unfortunately this scheme breaks down in high dimensions as the number of cells in a structured grid increases exponentially.

One possible remedy to this problem is posed by the Square-Root-Approximation method (SQRA).[10] It approximates the infinitesimal generator of the Koopman operator by a finite volume approximation where the volumes are implicitly defined by the Voronoi tesselation induced by some sample points. Using only evaluations of the potential U at these samples it can be understood as a semi-parametric method resulting in a sparse matrix representation, which in turn can be used for the computation of the eigenfunctions.

All these classical approaches however depend on a discretization *before* solving the eigenproblem. As an alternative we will now summarize ISOKANN, a recent matrix-free approach learning a linear combination of the eigenfunctions by neural networks.

B. ISOKANN– computing the dominant eigenspace

The ISOKANN algorithm[4] uses a nonparametric representation in the form of a neural network in order to learn the dominant invariant subspace by interleaving an Arnoldi-like power iteration with the approximation of the Koopman operator by Monte-Carlo simulations.

To this end it will be useful to reformulate the eigenproblem in terms of the so-called χ-function $\chi = (\chi_i)_{i=1}^d : \mathbf{X} \to \mathbb{R}^d$ with $0 \leq \chi_i \leq 1$, $\sum_i \chi_i = 1$ satisfying the χ-equation,

$$\chi = S\mathbf{K}\chi, \tag{5}$$

for an appropriately chosen matrix $S : \mathbb{R}^d \to \mathbb{R}^d$, which will be specified in Secs. II C and II D. The core idea here is that the components χ_i of χ span an invariant subspace of $\mathbf{K}$, or more specifically they consists of linear combinations of eigenfunctions of $\mathbf{K}$. Furthermore the action of $\mathbf{K}$ on this space is then explicitly given by the matrix S^{-1}. This definition of χ functions does not only allow us to reconstruct the eigendecomposition of $\mathbf{K}$ (Proposition 1) but also allows for an interpretation as macrostates in the form of *fuzzy memberships* to d different sets[11] and leads to a direct characterization of rare transitions in form of their holding times, reaction rates, exit paths etc.[7,8]

Formally ISOKANN then approximates the χ-function representing the dominant eigenspace by an iterative sequence of approximations $\chi_n : \mathbf{X} \to \mathbb{R}^d$ satisfying the ISOKANN *equation*,

$$\chi_{n+1}(x) = S_n\mathbf{K}^T\chi_n(x) = S_n\mathbf{E}[\chi_n(X_T)|X_0 = x], \tag{6}$$

with $S_n = S_n(\mathbf{K}^T \chi_n)$ being a linear map depending on the previous iteration and constructed in such a way as to provide convergence to a desired form of S in (5).

Similar to the Arnoldi-iteration, the iterative application of the Koopman operator leads to a decay of the eigenfunctions that is exponential in their eigenvalue. In Secs. II C and II D we will discuss how to construct S_n as to compensate this decay just so that the dominant components will prevail whilst the non-dominant ones vanish. In the limit, χ_n spans the dominant subspace (Theorem 1)

$$\text{span}\{(\chi_n)_1, \ldots, (\chi_n)_d\} \xrightarrow{n \to \infty} \text{span}\{v_1, \ldots, v_d\}, \tag{7}$$

with v_i denoting the dominant eigenfunctions. This in turn allows us to learn the linear action of $\mathbf{K}$ on that subspace leading to

$$\chi_n \to \chi \text{ and } S_n \to S. \tag{8}$$

Let us note that this presentation of ISOKANN differs from the original variant[4] in that the iteration in the form of (6) allows general d-dimensional valued χ functions. However we will see that for the case of $d = 2$ this is equivalent to the original variant, which by slight abuse of notation we will refer to as 1D-ISOKANN.

1. Illustrative example

To illustrate this better, let us consider the following example given by the Stochastic Differential Equations (SDE) (1) with a double well potential

$$U(x) = (x^2 - 1)^2. \tag{9}$$

The double well potential with two minima is shown in Fig. 1, the two dominant eigenfunctions are given in Fig. 2. In Fig. 3, we show the resulting two χ-functions, that are a linear combination of the eigenfunctions, but allow us to grasp the slow time-scale dynamics better. In particular, the two χ-functions can be interpreted as membership functions of the two wells of the potential. Their exit paths are charachterized by the gradients $-\nabla\chi$ and their holding probabilties or exit rates can be computed from the associated matrix S in (5) (see Refs. 7 and 8).
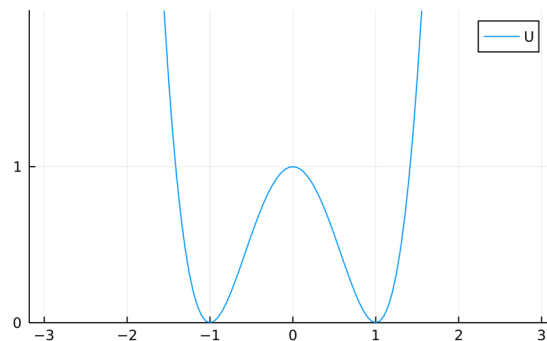

FIG. 1. Potential function U of the double well with two metastable regions separated by a potential barrier.
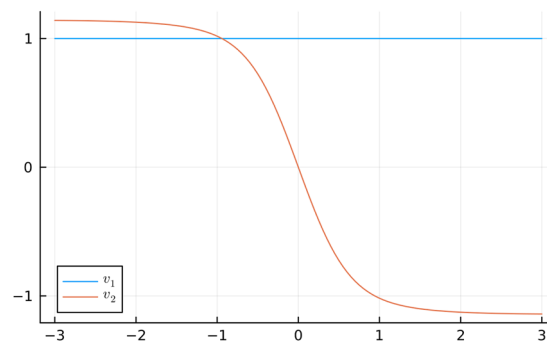

FIG. 2. The two dominant eigenfunctions v_1, v_2 of the Koopman operator for the double well potential.
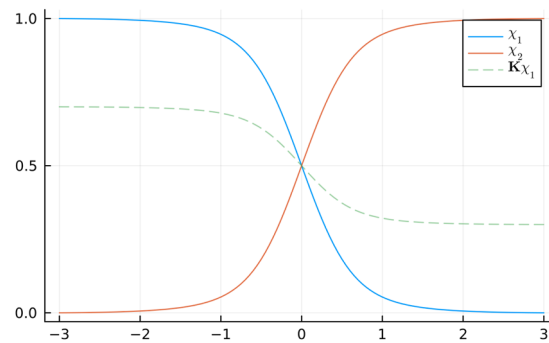
FIG. 3. Both components of the two-dimensional χ-function, which are linear combinations of the eigenfunctions (Fig. 2). In this case the application of K, which in general is linear in χ, corresponds to a shift-scale (see Sec. II D).

C. Choice of the transformation S

In the previous description we were talking about an "appropriately chosen" linear transformation S or a sequence of S_n. We will now explain the role that S plays in ISOKANN and suggest a way of determining suitable S.

In principle the map S can be chosen arbitrarily such that (5) has a solution in χ. However to obtain a useful result and for the ISOKANN iterations to converge with the corresponding choice of S_n we require some specific properties.

In order to understand the role of S let us for now think about the classical *Arnoldi method* to find the dominant eigenfunctions of a matrix. In principle the Arnoldi method also takes the form of (1), where the application of S_n corresponds to a *Gram-Schmidt orthonormalization*. Here the orthogonalization ensures that the leading eigenvectors are projected out from the subsequent ones and the following normalization step then ensures that the eigenvectors do not decay over multiple iterations.

Note here that while the Gram-Schmidt orthonormalization on its own is a nonlinear procedure, the resulting action on a given set of input vectors can be expressed as a linear map, i.e. if ω is the orthonormalization procedure, for each input matrix k we can find a matrix $\Omega(k)$ (depending on k) such that

$$\omega(k) = \Omega(k)\, k. \tag{10}$$

It is in exactly this way that we understand S as a linear map computed non-linearly on the data in Eq. (6).

For ISOKANN we want S to fulfill the same role: The goal of each linear transformation S_n is to counteract the decay of the dominant eigenfunction components contained in χ_n after application of the Koopman operator $\mathbf{K}$.

However, we are interested in linear combinations of the eigenfunctions over a *continuous space* represented by a neural network. In this setting orthonormalization is a hard problem, involving integration over the whole state space. In principle one might resort to restricting the χ functions to a finite number of fixed points and orthonormalize wrt. the resulting vectors. We expect this to work fine as long as one includes those points where the eigenfunction differ strongly, i.e. the individual metastabilites, which are usually not known a *priori*.

On the other hand, orthonormality is a strong assumption which is not necessarily required. If we manage to choose S such that it amplifies the first d eigenfunctions such that they stay bounded away from zero, we will obtain a representation of the dominant subspace, which in sequence allows for the reconstruction of eigenfunctions (c.f. Sec. II E).

We now motivate a heuristic approach, based on the PCCA$^+$ algorithm,[11] to construct a transformation S and will prove its convergence in the 1D-setting.

Let us shortly summarize the idea of the PCCA$^+$ methodology. In the context of metastable systems the metastabilities are those regions where one of the dominant eigenfunctions becomes approximately constant as well as extremal. In practice, plotting the image of the eigenfunction's components v_i the resulting set often resembles a simplex in which the metastable regions accumulate at its vertices. PCCA$^+$ can be understood as a method to identify this simplex structure and construct the linear transformation (in the space of eigenfunction components) mapping this set into the unit-simplex (see also Fig. 4), such that the result becomes "as big as possible" (specified by an optimization problem).

We can similarly imagine this picture for the χ functions: After sufficient time propagation (or power iterations), the χ functions are mainly composed of the dominant eigenfunctions. The image of χ thus will be close to the image of v modulo a linear transformation/a change of basis. We can thus apply PCCA$^+$ onto our intermediate $\mathbf{K}\chi_n$ in order to find a transformation such that the resulting χ_{n+1} fills this unit-simplex. This inhibits the exponential decay of the non-trivial eigenfunctions by maintaining a set of d linearly independent components. Since the dominant eigenfunctions decay slower then the following non-dominant ones, they will dominate the behavior and prevail for $n \to \infty$.

Even though preliminary results have shown that using PCCA$^+$ to construct S in higher dimensions works fine (with a suitable stabilization against permutations), the focus of this paper is on the optimal control so we will reserve a more detailed report to future work and hence give a proof only for the simpler case of computing the first non-trivial eigenfunction.
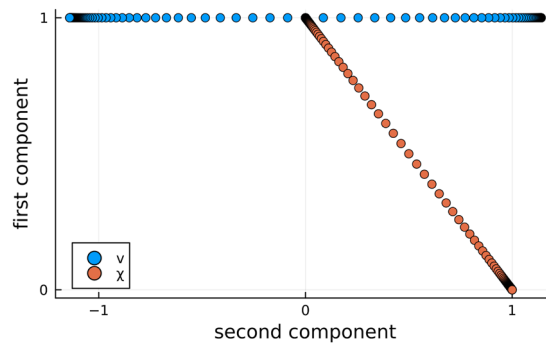
02 February 2024 08:34:30

FIG. 4. Scatter plot depicting v_1 against v_2 and χ_1 against χ_2 on a uniform grid over $\mathbf{X}$. PCCA$^+$ constructs the linear map S mapping v onto the unit-simplex χ.

D. 1D-ISOKANN with explicit PCCA$^+$

In the case where one is interested merely in the first non-trivial eigenfunction, v_2, the PCCA$^+$ solution can be computed explicitly, which in turn allows us to prove convergence of ISOKANN for $d = 2$ and explicitly specify S.

Note that because of $\chi_1 + \chi_2 = 1$, the desired solution $\chi : \mathbf{X} \to \mathbb{R}^2$ is fully determined by its first component $\bar{\chi} := \chi_1 : \mathbf{X} \to \mathbb{R}$ alone,

$$\chi = \begin{pmatrix} \bar{\chi} \\ 1 - \bar{\chi} \end{pmatrix}. \tag{11}$$

This representation allows us to solve the two dimensional ISOKANN problem with just one scalar function, approximated by a series of scalar neural networks $\bar{\chi}_n : \mathbf{X} \to \mathbb{R}$. This scalar representation is the approach taken in 4 and in the following we will refer to it as 1D-ISOKANN (since ISOKANN for $d = 1$ would otherwise only denote the trivial solution $\chi \equiv 1$).

Let us start by constructing the explicit PCCA$^+$ solution. Recall that $v_1 \equiv 1$ and note that $v_2(\mathbf{X}) = [\min(v_2), \max(v_2)]$. Thus the image of $\mathbf{X}$ forms a line segment, i.e. a 1-dimensional simplex, in the v_1-v_2-plane. PCCA$^+$ then constructs the unique map S which maps this simplex onto the unit simplex $\{(x, y) | x + y = 1, x > 0, y > 0\}$ (see Fig. 4).

To this end, let us introduce the map $\bar{S}$ for bounded continuous functions $\kappa \in C(\mathbf{X})$

$$\bar{S}(\kappa) = \frac{\kappa - \min(\kappa)}{\max(\kappa) - \min(\kappa)} \tag{12}$$

such that $\bar{S}(\kappa) : \mathbf{X} \twoheadrightarrow [0, 1]$ maps surjectively onto the unit interval. Even though $\bar{S}$ (consisting of a shift and a scale) is only affine-linear in its argument, it can be seen as a linear map on the 1D subspaces $\{(1, \kappa) | \kappa \in C(\mathbf{X})\}$ or similarly $\{(\kappa, 1 - \kappa) | \kappa \in C(\mathbf{X})\}$. Indeed, it is the action of the PCCA$^+$ solution on a single component, i.e., there exists a matrix $S \in \mathbb{R}^{2 \times 2}$ such that it satisfies

$$S \cdot \begin{pmatrix} \kappa \\ 1 - \kappa \end{pmatrix} = \begin{pmatrix} \bar{S}(\kappa) \\ 1 - \bar{S}(1 - \kappa) \end{pmatrix}. \tag{13}$$

So S determined by PCCA$^+$ is indeed a linear map depending non-linearly on the input k, just as in the case of the orthonormalization procedure (10), and the action on the first component is equivalently given by the affine-linear map $\bar{S}$.

Using $\bar{S}$ to learn only the first component of the χ function, we now can formulate the following explicit iterative 1D-ISOKANN procedure:

Theorem 1. *Let χ, $\bar{S}$ and S be chosen as above in Eqs. (11)–(13). For generic $\bar{\chi}_0 : \mathbf{X} \to \mathbb{R}$, i.e. containing components of v_1 and v_2, the* 1D-ISOKANN *iteration*

$$\bar{\chi}_{n+1} = \bar{S}(\mathbf{K}\bar{\chi}_n) \tag{14}$$

converges to the χ-function

$$\bar{\chi} = \lim_{n \to \infty} \bar{\chi}_n = \alpha v_1 + \beta v_2, \tag{15}$$

for some $\alpha, \beta \in \mathbb{R}$, which in turn solves the ISOKANN *problem*

$$\bar{\chi} = S\mathbf{K}\bar{\chi}. \tag{16}$$

Proof. Noting that $\bar{S}$ is merely a shift-scale, i.e. affine linear, and $\mathbf{K}\mathbf{1} = \mathbf{1}$ one obtains

$$(\bar{S} \circ \mathbf{K})^n = \bar{S} \circ \mathbf{K}^n. \tag{17}$$

Looking at the eigendecomposition of $\bar{\chi}_0 = \sum_{i=1}^{\infty} a_i v_i$, we have

$$\mathbf{K}^n \bar{\chi}_0 = \sum_{i=1}^{\infty} a_i \lambda_i^n v_i = a_1 \mathbf{1} + \sum_{i=2}^{\infty} a_i \lambda_i^n v_i \tag{18}$$

For large n, the contribution of the faster eigenfunctions $v_i, i > 2$ decays exponentially faster than the contribution of v_1, v_2. Hence the shift-scale is dominated by these slow eigenfunctions and with (17) we have for some $\alpha, \beta \in \mathbb{R}$

$$\bar{\chi} = \lim_{n \to \infty} (\bar{S} \circ \mathbf{K})^n \bar{\chi}_0 = \lim_{n \to \infty} \bar{S} \circ \mathbf{K}^n \bar{\chi}_0 = \alpha v_1 + \beta v_2, \tag{19}$$

which proves (15). Taking into account that $\bar{\chi}(\mathbf{X}) = [0, 1]$ and $\mathbf{K}$ acts as a shift-scale, $\bar{S}$ then subsequently rescales $\mathbf{K}\bar{\chi}$ back to the range [0,1]. This gives us $\bar{S}(\mathbf{K}\bar{\chi}) = \bar{\chi}$, which, because S is implicitly constructed from $\bar{S}$ in Eq. (13), gives us the fixed-point result in (16). □

To summarize, we have shown how the representation of ISOKANN for $d = 2$ as a scalar problem reproduces the classical 1D-ISOKANN procedure[4] where the role of the linear map S (determined by PCCA$^+$) gets replaced by the (explicitly given) affine-linear $\bar{S}$. This simpler representation allowed us to show convergence to a χ-function and thus solving the ISOKANN problem.

E. Restoring the eigenfunctions

At the beginning of the article we intended to compute the eigenfunctions of $\mathbf{K}$. Whereas the χ functions only span the corresponding invariant subspace, we now show how we can restore the eigenfunctions from the solution χ to the ISOKANN problem $S\mathbf{K}\chi = \chi$. This is equivalent to

$$\mathbf{K}\chi = S^{-1}\chi \tag{20}$$

which means that the action of $\mathbf{K}$ on the subspace χ is given by the matrix S^{-1}. By means of a basis transformation (making use of the *Moore–Penrose pseudoinverse* χ^+) we can recover the eigenfunctions of $\mathbf{K}$ from S and χ.

Proposition 1. Let $\chi = (\chi_i)_{i=1}^d$ be a column vector of χ_i component functions and S be a full rank matrix such that

$$S\mathbf{K}\chi = \chi \tag{21}$$

If (X, Λ) is an eigendecomposition of $Q := \chi^+ S^{-1} \chi$, i.e.,

$$QX = X\Lambda, \tag{22}$$

then $E := \chi X$ are the eigenvectors of $\mathbf{K}$ with eigenvalues Λ.

Proof. By assumption we have $\mathbf{K}\chi = S^{-1}\chi$. Inserting this into (22), multiplying with χ from the left and noting that $\chi^+\chi = \mathrm{Id}$ by definition, we arrive at

$$\mathbf{K}\chi X = \chi X \Lambda \tag{23}$$

which gives the desired result. □

F. Computational procedure

With these theoretical considerations, let us now discuss how to apply the algorithm in practice, i.e. using neural networks as function approximators and Monte Carlo (MC) simulations for the Koopman evaluations.

To this end let us recall the main formula for the iterative update (6):

$$\chi_{n+1}(x_m) \leftarrow S_n \mathbf{K}^T \chi_n(x_m) = S_n \mathbf{E}[\chi_n(X_T)|X_0 = x_m]. \tag{24}$$

Here we replaced the equality by a $\leftarrow$ which indicates classical supervised learning (with the common mean squared error loss) along multiple (possibly random) training points $x_m, m = 1, \ldots, M$. For a procedural description see Algorithm 1. The main challenges posed by this iterative scheme consist of (a) a representation of the function(s) χ_n and (b) the evaluation of the right hand side, i.e. the computation of S_n and the evaluation of the Koopman operator.

For (a), the representation of the χ_n, we choose neural networks as they promise good approximation properties in high dimensions and their differentiability will prove crucial for the following optimal control part. In general any feed-forward architecture should be suitable

and whilst specific architectures, such as symmetry invariant networks, could certainly improve the results, in the example we will confine ourselves to a fully connected architecture for simplicity. In any case, the update step for χ_{n+1} consists of a classical supervised learning routine with the labeled data

$$D_n = \{(x_m, s_m)\}, \quad s_m = S_n \mathbf{K} \chi_n(x_m) \tag{25}$$

generated by evaluation of the current χ_n. Because the χ_n are not expected to be changing a lot between the iterations it makes sense to initialize χ_{n+1} with the weights from χ_n as to transfer the already learned structure and speed up the learning. Note here that whilst we talk about different networks χ_n for each iteration n to emphasize the iterative nature of (24), in practice we can update a single instance of the network.

In this view, the learning procedure can be seen as iterative supervised batch learning, where the whole data batch D_n is generated along a set of points $\{x_m\}$ using the current representation χ_n. The update step itself can be performed using any stochastic optimizer such as classical stochastic gradient descent or Adam to minimize the empirical L^2 error

$$\min_{\chi_{n+1}}! \sum_{(x_m, s_m) \in D_n} (\chi_{n+1}(x_m) - s_m)^2. \tag{26}$$

Considering (b), i.e., the evaluation of the right hand side in Eq. (24), let us start with the approximation of the Koopman operator. For a given training point x_m, we use its representation as an expectation value and approximate the action of the Koopman operator by a Monte-Carlo sum. Each simulation consists of starting $K > 0$ trajectories at the point x_m and propagating them according to the SDE (1) using an SDE integrator, such as the Euler-Maruyama scheme, for the lag time T and storing their end points $y_{k,m}, k = 1, \ldots, K$. The action of the Koopman operator at x_m is then approximated by the empirical average

$$\kappa_m := \mathbf{K} \chi_n(x_m) = \mathbf{E}[\chi_n(X_T) | X_0 = x_m]$$
$$\approx \frac{1}{K} \sum_{k=1}^{K} \chi_n(y_{k,m}). \tag{27}$$

To summarize, for each x_m we average the evaluation of χ_n at K propagated positions obtained by SDE simulations.

What now remains is the application of the shift-scale S_n. In case of 1D-ISOKANN the action of S_n is determined by the shift-scale $\bar{S}$ from (12) which depends on the (global) extrema of the input function k. In practice we thus use the empirical extrema over the observed data $\{\kappa_m\}$ to directly compute $s_m = (\bar{S}(\kappa))_m$ without explicitly constructing S_n. In higher dimensions we compute the matrix S_n using PCCA$^+$ to find the transformation that maps the columns of the matrix $K = [\kappa_1 \ldots \kappa_m]$ into the unit simplex. Note that the use of the empirical extrema requires that the training points x_m indeed cover the areas where $\mathbf{K}\chi_n$ becomes (approximately) extremal.

This brings us to the choice of the M training points x_m. In principle ISOKANN can be applied to find the χ functions of a system based on a fixed set of precomputed or assimilated trajectories (replacing the SDE integration). However its iterative nature makes it especially useful in the synthetic data regime where the trajectories are computed on-line as it allows adapting the training points x_m, and as we will see the trajectory simulations too, to the information obtained so far.

Since the χ-functions can be interpreted as reaction coordinates[7] we suggest "χ-*stratified*" *sampling* of the x_m, i.e. such that $\chi(x_m)$ is approximately uniform in $[0,1]$. In practice we achieve this by subsampling from the pool of start and end points of the previous simulations,

$$P_n = \left(\bigcup_m x_m^{(n-1)} \right) \cup \left(\bigcup_{m,k} y_{k,m}^{(n-1)} \right). \tag{28}$$

We then draw M stratified uniform samples $u_i \in [0,1]$ by sampling uniformly from each of M equally sized partitions of the interval $[0,1]$. Finally we choose those $p_i \in P_n$ such that $\chi_n(p_i)$ is the closest to one of the u_i. We furthermore retain those samples which were extremal in χ_n to facilitate good approximation of the extrema by $\bar{S}$ or PCCA$^+$.

Heuristically speaking we obtain samples x_m which are uniform in χ and hence provide good coverage or "bridges" along the transition region, thus facilitating an efficient "flow of information" during the power-iteration process. Furthermore the regions with a higher variation of χ, i.e. those that are "harder to learn," will also obtain more samples which in turn is also beneficial for the training of the neural network itself.

Last but not least the samples obtained this way followed the system's ergodic dynamics and will therefore approximate the stationary distribution (restricted on each level set of χ). This allows us to evade the curse of dimensionality by restricting the sampling to physically meaningful samples along the reaction paths.

To summarize, χ-*stratified sampling* allows us to sample uniform along χ and stationary conditioned on it without much additional cost and adapted to the learning process.

The main ISOKANN routine can be summarized by three loops (N power iterations, M training points, K trajectories) which can be loosely tied to the three main ingredients of ISOKANN:

02 February 2024 08:34:30

(1) The power iteration learning the dominant subspace.
(2) The neural network approximation of the next χ iterate.
(3) The Monte-Carlo simulation of the Koopman evaluation.

In the outermost loop (1) we perform the power iteration χ_n to χ_{n+1}. Whereas in theory we have convergence for $n \to \infty$, we choose to terminate after a fixed number of iterations N. This could be replaced by classical convergence criteria such as relative and absolute tolerances. Note that the rate of convergence and hence the required number of power iterates N depends on the eigenvalues of the Koopman operator, where a bigger spectral gap implies faster decay of the non-dominant spectrum and hence faster convergence.

When training the neural network (2) we loop over a batch of labeled training data at the M training points x_m which in turn (3) require K individual trajectory simulations. Both the number of training points M as well as trajectories per point K depend on the step sizes chosen for the neural network optimizer. Since in practice the evaluation of the Koopman operator is rather expensive compared to the neural network update, it may be efficient to perform multiple update steps on the same batch of data before proceeding to the next iteration.

Note that the variance of the training data, scaling with K^{-1}, is particularly high for metastable systems due to the impact of rare transitions. Whereas above we proposed the use of χ-*stratified* subsampling as a heuristic to deal with sampling in $\mathbf{X}$ space, we will now address the problem of variance in the "$\mathbf{K}\chi_n$-direction" using the techniques of optimal control and importance sampling.

III. OPTIMAL SAMPLING OF KOOPMAN EIGENFUNCTIONS AND χ-FUNCTIONS

In this chapter we first recall importance sampling, before showing how the theory allows to better sample eigenfunctions of the Koopman operator and ISOKANNχ-functions. We conclude this chapter with a numerical example.

A. Importance sampling for random variables

Importance sampling allows to express the expectation value of an observable $f > 0$ with respect to some distribution p by an expectation value with respect to some other distribution q (with $p \ll q$) by the formula

$$Z := \mathbf{E}_p[f] = \mathbf{E}_q\left[f\frac{\mathrm{d}p}{\mathrm{d}q}\right], \tag{29}$$

where the observable f is reweighted by the Radon-Nikodym derivative $\frac{\mathrm{d}p}{\mathrm{d}q}$. It is easy to see that by choosing q^* such that $\frac{\mathrm{d}p}{\mathrm{d}q^*} = \frac{Z}{f}$ (i.e. $q^* = \frac{f}{Z}p$), we have

$$Z = \mathbf{E}_{q^*}\left[f\frac{Z}{f}\right] = \mathbf{E}_{q^*}[Z]. \tag{30}$$

Now, since Z is a constant, it can be computed with a single (reweighted) f-sample from q^*. We therefore refer to this importance sampler with sampling distribution q^* as *zero-variance-sampler*, or *optimal-importance-sampler*. Note however that we needed to know the (*a priori* unknown) result Z in order to define the optimal sampling distribution q^*.

B. Optimal importance sampling for diffusion processes

Since we are working with diffusion processes, importance sampling is further complicated in that the measures are path measures and admit no probability density function. However, importance sampling can still be generalized to stochastic processes. Let us first consider the diffusion process of Eq. (1). In general, one is interested in computing the expectation of path-dependent quantities. Let us define the *work* along a trajectory over $[t, T]$ by the accumulation of a *running cost* f and a *terminal cost* g as:

$$W_{t,T}(X) := \int_t^T f(X_s, s)\mathrm{d}s + g(X_T). \tag{31}$$

One then is interested in estimating expectation values of the form

$$\begin{aligned}\psi(x,t) &:= \mathbf{E}_{X_t=x}\left[\exp\left(-W_{t,T}(X)\right)\right] \\ &= \mathbf{E}_{X_t=x}\left[\exp\left(-\int_t^T f(X_s, s)\mathrm{d}s - g(X_T)\right)\right].\end{aligned} \tag{32}$$

Girsanov's theorem builds the bridge from importance sampling to diffusion processes by allowing to sample from another diffusion processes. In particularly, it allows us to compute the change of measure in terms of the Radon-Nikodym derivative. To this end let us introduce the controlled process $X^u = (X_t^u)_{t \geq 0}$

$$\mathrm{d}X_t^u = (b(X_t^u) + \sigma u(X_t^u, t))\mathrm{d}t + \sigma\mathrm{d}B_t, \tag{33}$$

with an admissible control term u acting as an external forcing to the original dynamics. Note that with zero control $u = 0$ one recovers the original dynamics $X = X^{u=0}$. Let $\mathcal{P}$ denote the path measure induced by X and $\mathcal{Q}$ denote the measure induced by X^u. According to Girsanov's theorem the change of measure from $\mathcal{Q}$ to $\mathcal{P}$ (analogous to $\frac{dp}{dq}$ above) along a given controlled trajectory X^u is then given by

$$
\begin{aligned}
G_{t,T}(X^u) &:= \frac{d\mathcal{P}}{d\mathcal{Q}}\Big|_{[t,T]}(X^u) \\
&= \exp\left(-\int_t^T u(X_s^u, s) \cdot dB_s - \frac{1}{2}\int_t^T |u(X_s^u, s)|^2 ds\right)
\end{aligned}
\tag{34}
$$

which in turn provides an unbiased estimator of ψ in terms of the controlled process:

$$
\begin{aligned}
\psi(x, 0) &= \mathbf{E}_{X_0 = x}[\exp(-W_{0,T}(X))] \\
&= \mathbf{E}_{X_0 = x}[\exp(-W_{0,T}(X^u))G_{0,T}(X^u)].
\end{aligned}
\tag{35}
$$

Note that even though the expected value of this estimator is the same for any control u its variance will vary. Analogous to the case above there exists an optimal measure corresponding to an optimal control u^* for which the controlled estimator exhibits zero variance:[12,13]

Theorem 2. *The optimal control u^* is given by*

$$
u^*(x, t) = \sigma^\top \nabla_x \log \psi(x, t)
\tag{36}
$$

and leads to the zero variance estimator for ψ

$$
\begin{aligned}
\psi(x, 0) &= \mathbf{E}_{X_0 = x}[\exp(-W_{0,T}(X))] \\
&\stackrel{a.s.}{=} G_{0,T}(X^{u^*}) \exp(-W_{0,T}(X^{u^*}))
\end{aligned}
\tag{37}
$$

with $X_0^{u^*} = x$.

C. Optimal sampling of eigenfunctions of the Koopman operator

We will now show how this optimal control theorem can be used to evaluate the Koopman operator.

Corollary 2.1. Let $h \in L^\infty(\mathbf{X})$ be a function. A single realization of the controlled process X^u starting in $X_0^u = x$ with control

$$
u(x, t) = \sigma^\top \nabla_x \log(\mathbf{K}^{T-t}h)(x)
\tag{38}
$$

then gives the evaluation of $\mathbf{K}^T h$ at that point x:

$$
(\mathbf{K}^T h)(x) \stackrel{a.s.}{=} G_{0,T}(X^u)h(X_T^u).
\tag{39}
$$

Proof. With $f(x) = 0$ and $g(x) = -\log h(x)$ Eq. (32) becomes

$$
\begin{aligned}
\psi(x, t) &= \mathbf{E}_{X_t = x}[h(X_T)] \\
&= \mathbf{E}_{X_0 = x}[h(X_{T-t})] = (\mathbf{K}^{T-t}h)(x).
\end{aligned}
\tag{40}
$$

Application of Theorem 2 then leads to the desired result. ☐

This result shows us that in order to compute the optimal control for evaluating $\mathbf{K}h$ we need to have access to the derivatives of $\mathbf{K}h$. This conundrum is in line with the general optimal importance result and comes at no surprise. In Sec. III D we will argue how this result can still be of use for ISOKANN where the convergence of the χ_n provides us with an approximate description which we will use to compute the control.

Let us for now consider the case where the observable of interest h is an eigenfunction of $\mathbf{K}^T$, i.e. $h = v_i$ with eigenvalue $\lambda_i(T)$. In this case we can replace the action of the Koopman operator by its eigenvalue, which in turns cancels out after application of $\nabla \log$ and results in a time-independent control:

$$
\begin{aligned}
u^*(x, t) &= \sigma^\top \nabla_x \log(K^{T-t}v_i)(x) \\
&= \sigma^\top \nabla_x \log(\lambda_i(T-t)v_i(x)) = \sigma^\top \frac{\nabla_x v_i(x)}{v_i(x)}.
\end{aligned}
\tag{41}
$$

This simple example provides a good point to get a feeling for how optimal importance sampling works. We can see that the control pushes the system in the direction of (relative) maximal ascent. If the system follows the forcing its expected evaluation increases, but the path taken also has an increased probability, resulting in a decreasing Girsanov weight. Both increments happen on a commensurate "relative scale": the expectation value gets pushed by an amount relative to its current value $\left(\frac{\nabla v}{v}\right)$ whereas the reweighting is adjusted relatively in magnitude due to the exponentiated integral (which may be seen as an infinite product over all time-points) in (34). In this way the increase of the observable is balanced with the decreasing weight exactly so that no matter the path taken these always equalize and one obtains a zero-variance sampler.

Unfortunately however, the control becomes singular whenever $v_i(x) = 0$. According to the Perron-Frobenius theorem, every non-trivial eigenfunction is unsigned, i.e. crosses the 0 at some point, so we have to find a way around that problem. We can alleviate this problem by shifting and [anticipating the form of $\tilde{S}$ in (12)] also rescaling the eigenfunction.

Denote the shift-scaled eigenfunction by

$$\chi := \alpha v_i + \beta \mathbf{1}, \quad \alpha, \beta \in \mathbb{R}, \quad \chi > 0. \tag{42}$$

Due to linearity of $\mathbf{K}^T$, we obtain

$$(\mathbf{K}^T \chi)(x) = (\mathbf{K}^T \alpha v_i)(x) + \beta = \alpha \lambda_i(T) v_i(x) + \beta \tag{43}$$

and thus after application of Corollary 2.1 the control in terms of χ is

$$u^*(x,t) = \sigma^\top \nabla_x \log(\mathbf{K}^{T-t}\chi)(x) = \sigma^\top \frac{\nabla \chi(x)}{\chi(x) + \frac{\beta}{\lambda_i(T-t)} - \beta}. \tag{44}$$

In this case we see that control is time dependent (which makes sense as the relative contributions of the dominant eigenfunctions to the expectation value change over time). From $\lambda_i(t) \leq \lambda_i(0) = 1$ we can conclude that $u^*(\cdot, t) = \gamma \sigma^\top \frac{\nabla \chi}{\chi}$ with function γ monotonically increasing with $\gamma(1) = 1$, i.e. the control is pointing in the same direction as for the pure eigenfunction case, starting weaker and increasing until hitting the full magnitude at $t = T$.

Note that the requirement for χ functions to satisfy $0 \leq \chi \leq 1$, which so far was merely motivated by their interpretation as macrostates, now also facilitates their optimally controlled importance sampling.

D. Application to ISOKANN

In Sec. III C we have shown how to obtain a zero-variance sampler for the Koopman operator in terms of the gradient of its solution, either for general observables h or (shift-scaled) eigenfunctions $v_i(\chi)$. In either case the solution has to be known *a priori* as to compute the control. We will now argue how to integrate this result into the ISOKANN procedure.

The main idea of using optimal importance sampling in ISOKANN is to use the intermediate results χ_n and S_n to compute a *pseudo-optimal control* as to lower the sampling variance. We therefore have to assume that using an approximation to the optimal control indeed leads to a variance reduction. Whereas we do not know of any proof to this statement it was shown that the objective of the associated optimal control problem is indeed convex in the control[14] which leads us to conjecture that the variance should be well-behaving for approximate optimal controls as well.

Note that the importance sampler (35) is unbiased for any control. Thus, as long as the variance does not become unbounded ISOKANN will still converge for a bad control (under the usual conditions for stochastic descent convergence, i.e. decaying learn rate). Note that the control can always be bound artificially by clipping, bounding the Girsanov reweighting term and in turn also the overall sampling variance.

Let us recall the equation for the control (38):

$$u(x,t) = \sigma^\top \nabla_x \log(\mathbf{K}^{T-t}h)(x) \tag{45}$$

In order to compute the differential of $\mathbf{K}^{T-t}$ at χ_n we assume sufficient convergence of ISOKANN together with (6),

$$\chi_n \approx \chi_{n-1}, \quad \chi_n = S_{n-1}\mathbf{K}^T\chi_{n-1}, \tag{46}$$

to approximate the action of $\mathbf{K}^T$ by $(S_{n-1})^{-1}$:

$$(S_{n-1})^{-1}\chi_n = \mathbf{K}^T\chi_{n-1} \approx \mathbf{K}^T\chi_n. \tag{47}$$

Using the semi-group property of $\mathbf{K}$ and the matrix logarithm we can extend this to other lag times $T - t$ to obtain the matrix approximation $\widetilde{\mathbf{K}}^{T-t}$

$$\mathbf{K}^{T-t}\chi_n \approx \widetilde{\mathbf{K}}^{T-t}\chi_n := \exp\left(\frac{T-t}{T}\log\left((S_{n-1})^{-1}\right)\right)\chi_n. \tag{48}$$

Note that in the general d-dimensional case the expectation values in the ISOKANN iterations $\mathbf{K}^T\chi_n$ are vector valued. Optimal importance sampling however works only in the scalar case. Therefore we have to compute an individual control for sampling each component $(\mathbf{K}^T\chi_n)_i$ individually.

Thus using the optimal control Corollary 2.1 together with the matrix approximation (48) we can compute the pseudo-optimal control for the i-th component of $\mathbf{K}^T\chi_n$ explicitly

$$u_i^*(x,t) = \sigma^\top \nabla_x \log\left(\sum_j \widetilde{\mathbf{K}}_{ij}^{T-t}\chi_j(x)\right) = \sigma^\top \frac{\sum_j \widetilde{\mathbf{K}}_{ij}^{T-t}\nabla_x\chi_j(x)}{\sum_j \widetilde{\mathbf{K}}_{ij}^{T-t}\chi_j(x)} \tag{49}$$

In the case of 1D-ISOKANN the action of the Koopman operator on χ_n converges to a shift-scale as in (43) and we can therefore estimate the parameters α, β and λ_2 from the extrema of $\mathbf{K}\chi_{n-1}$ as to apply the explicit control for the shift-scaled eigenfunction (44).

Now that we know how to compute the control we can modify the algorithm (Line 5) by sampling the trajectories according to the controlled SDE (33). In order to compute the reweighting (34) we have to either save the trajectory and noise, or integrating it on the fly in an addition SDE component with

$$G_T = \exp\left(-g_T\right) \quad \mathrm{d}g_t = \frac{1}{2}u(X_t^u,t)^2\mathrm{d}t + u(X_t^u,t)\cdot\mathrm{d}B_t \quad g_0 = 0. \tag{50}$$

Finally, for the Koopman Monte Carlo approximation (Line 6) we average over the χ evaluations at the endpoints of K independent trajectories starting in x_m weighted with their respective weights G_m:

$$\mathbf{K}\chi_n(x_m) \approx \frac{1}{K}\sum_k \chi_n(X_{T,m}^u)G_m. \tag{51}$$

In this way (and with the above assumption) we obtain a feedback loop where better approximation of the χ functions results in in a better approximation of the action of $\mathbf{K}$ and hence in a better approximation of the optimal control. This pseudo-optimal control in turn decreases the sampling variance which facilitates better approximation of the power iterates, i.e. the χ function.

As a proof of concept we will now illustrate the reduction of variance at the hand of the classic double-well potential.

E. Example: Controlled 1D-ISOKANN for the double well

Let us consider the controlled process as stated in (33) for the double-well potential (9) which leads to the simplest problem exhibiting metastable behavior and hence a challenging sample variance. In our experiments we compare the training performance of the ISOKANN algorithm, both, with and without the control (44).

We start with a randomly initialized fully connected network $\chi_0 : \mathbb{R} \to \mathbb{R}$ with sigmoidal activation functions and 2 hidden layers, each of size 5 (i.e. with layer sizes $1 \times 5 \times 5 \times 1$). For each network generation n, we compute Monte Carlo approximations of the Koopman expectation at $M = 30$ positions. These are initially drawn uniformly from the interval $[-2,2]$ and subsequently obtained by χ-stratified subsampling as described in Sec. II F. From each starting position we then simulate $K = 20$ trajectories using the SROCK2 SDE integrator of strong order 1 with step-size $\Delta t = 0.001$. The next generation $n + 1$ is trained against these M training points by $L = 500$ stochastic gradient descent steps using the ADAM optimizer (with learning rate $\eta = 0.001$). We repeat this evaluation-training procedure (corresponding to a single power iteration) for a total of $N = 50$ iterations.

For each experiment we monitor the root of the training loss (26), i.e. the root mean squared error,

$$\mathrm{RMSE} := \sqrt{\frac{1}{M}\sum_m \left(\chi_n(x_m) - s_m\right)^2} \tag{52}$$

and the mean standard deviation of the MC estimator (27)

$$\mathrm{MSTD} := \frac{1}{M}\sum_m \sqrt{\frac{1}{K}\sum_k \left(\chi_n(y_{k,m}) - \mu_m\right)^2}, \tag{53}$$

where

$$\mu_m = \frac{1}{K}\sum_k \left(\chi_n(y_{k,m})\right), \tag{54}$$

over the training phase of $N = 50$ iterations with $L = 500$ training steps each.

Let us now compare the uncontrolled with the controlled experiment. Figure 5 shows the square root of our training loss together with the standard deviation of the Monte Carlo estimator for the two cases of study. In 5(a) we observe that the uncontrolled system quickly (after 3 iterations) approaches its plateau at an error of about 10^{-1} but afterwards the training loss does not decrease any further. This comes at no surprise since the training data exhibits noise of the same magnitude, and we cannot expect the average loss to be lower than the noise
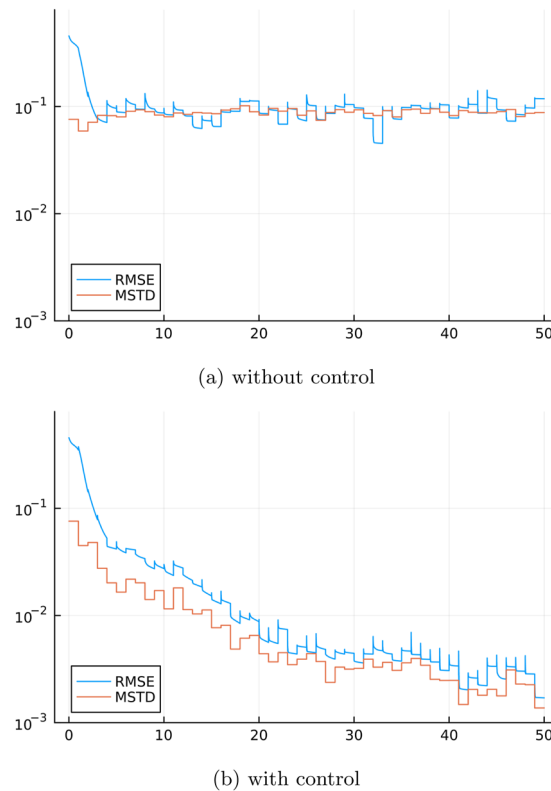
(a) without control

(b) with control

FIG. 5. Training performance over 50 power iterations/batches with 500 ADAM steps each. The blue line shows the training loss and the red line shows the standard deviation of the Monte-Carlo samples in the training data.

in the data. Note, however, that even though the loss itself seems to have leveled off, this does not necessarily mean that the solution does not improve. Whilst the empirical loss will necessarily remain at the level of the noise, the solution could still converge due to the inherent averaging of that noise in the SGD method.

Looking at the controlled experiment, we observe in 5(b) that for the loss behaves similar to the uncontrolled experiment for the first 3 iterations, reaching a value of 10^{-1}. From there on however the loss decreases further getting close to 10^{-3} after 50 power iterations and still not having hit a plateau. Notice that the training noise, in strong contrast to the uncontrolled case, decreases rapidly from the beginning of the training. It is furthermore interesting to see that the training loss seems to be following the noise level closely, indicating that the sampling variance is indeed of high importance.

Looking more closely at these performance plots we can furthermore identify the individual training batches: The mean standard deviation (MSTD) is piecewise constant along each such batch, since the training data (and hence its standard deviation) is updated only in-between the neural network training loops. Whereas the root-mean-square error (RMSE) plateaus continuously decrease during each batch, we can observe how they jump up slightly at the beginning of each new batch. These jumps are caused by overfitting the previous batch (the plateau) without generalization to the following batch (the flank). We suspect that the performance of the training could be improved by taking the plateaus as indicators for the generation of new training data.

Let us finally look at Fig. 6, which shows the different learned χ functions after the application of the ISOKANN algorithm together with the evaluations of $SK\chi_{n-1}(x_m)$ at the M random locations x_m. The error bars represent the standard deviation of the individual Monte-Carlo estimators, i.e. the noise in the training data. We see that both learned χ functions qualitatively match the expectation. The uncontrolled case however has problems reaching 0 resp. 1 at the boundaries, which can be understood as a result of the noise and the subsequent noisy estimation of the empirical shift-scale. Last we notice that the Monte-Carlo standard deviation for the Koopman evaluation at the χ-sampled positions is considerably lower by the controlled approach.
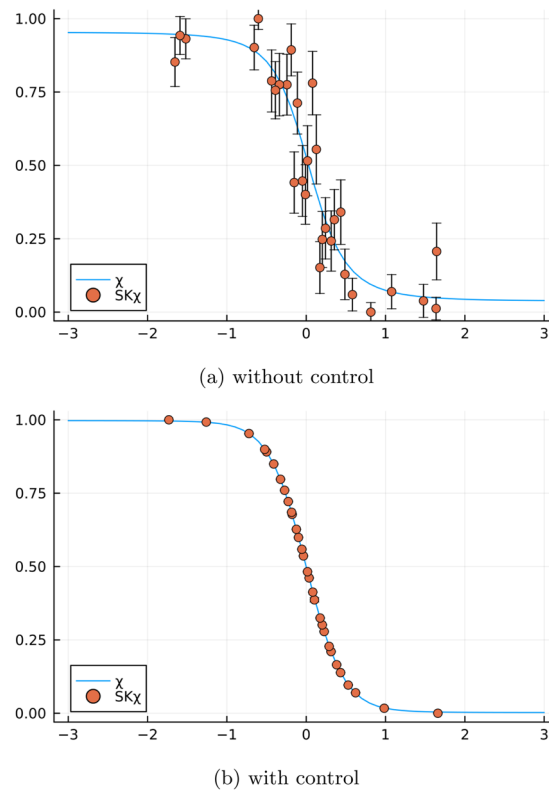
02 February 2024 08:34:30

(a) without control

(b) with control

FIG. 6. The blue line shows the learned χ function at the end of training. The red dots show the train target, i.e. the Koopman evaluations at the x_m, with the Monte-Carlo standard deviation as error bars.

IV. CONCLUSION

In this article we started by enhancing ISOKANN with new theoretical results that prove the strengths of ISOKANN, namely a convergence proof (Theorem 1) and a method for reconstructing eigenfunctions from χ-functions (Proposition 1). We also proposed a new adaptive sampling strategy called χ-stratified sampling, which complements ISOKANN well and deserves further investigation. By formulating ISOKANN in terms of the transformation S (6), we paved the way for higher-dimensional χ-functions while generalizing the original 1D-ISOKANN. However, whereas we argued for using PCCA$^+$ for the construction of S for $d > 1$ a more detailed study and a proof of convergence for this case remain open for future work.

The second main contribution in this article is the introduction of importance sampling into ISOKANN. Whereas we know that the resulting estimator is unbiased, we argue only heuristically why the variance should not explode. A proof of the convexity of the variance in the control, or even better, a proof of the convergence of the control in ISOKANN is still missing. Note, furthermore, that the concept of optimal importance sampling may be useful for the iterative solution of Koopman evaluations in general (cf. Corollary 2.1).

An important next step would be to apply controlled ISOKANN to an actual molecular dynamics (MD) system to test how well the introduced techniques fare with the complexities of real-world problems. This, however, requires a way to run many trajectories with different start locations and low-overhead as well as to inject the optimal control into the MD simulations. We hope that once these interfaces are implemented, ISOKANN will enhance the research on molecular systems.

ACKNOWLEDGMENTS

02 February 2024 08:34:30

AUTHOR DECLARATIONS

Conflict of Interest

The authors have no conflicts to disclose.

Author Contributions

A. Sikorski: Conceptualization (lead); Formal analysis (lead); Methodology (lead); Software (lead); Visualization (lead); Writing – original draft (lead); Writing – review & editing (lead). **E. Ribera Borrell**: Conceptualization (supporting); Formal analysis (supporting); Methodology (supporting); Writing – original draft (supporting); Writing – review & editing (supporting). **M. Weber**: Conceptualization (supporting); Funding acquisition (lead); Methodology (supporting); Supervision (lead); Writing – review & editing (supporting).

DATA AVAILABILITY

The data that support the findings of this study are openly available in GitHub at https://github.com/axsk/OptImpSampling.jl/tree/jmp.[15]

REFERENCES

[1] V. Gayrard, A. Bovier, M. Eckhoff, and M. Klein, "Metastability in reversible diffusion processes I: Sharp asymptotics for capacities and exit times," J. Eur. Math. Soc. **6**, 399–424 (2004).

[2] W. Huisinga, "Metastability of Markovian systems a transfer operator based approach in application to molecular dynamics," Ph.D. thesis, Fachbereich Mathematik und Informatik, FU Berlin, 2001.

[3] A. Nielsen, "The Monte Carlo computation error of transition probabilities," Stat. Probab. Lett. **118**, 163–170 (2016).

[4] MR. J. Rabben, S. Ray, and M. Weber, "ISOKANN: Invariant subspaces of Koopman operators learned by a neural network," J. Chem. Phys. **153**, 114109 (2020).

[5] B. J. Zhang, T. Sahai, and Y. M. Marzouk, "A koopman framework for rare event simulation in stochastic differential equations," J. Comput. Phys. **456**, 111025 (2022).

[6] C. Hartmann, C. Schütte, M. Weber, and W. Zhang, "Importance sampling in path space for diffusion processes with slow-fast variables," Probab. Theory Relat. Fields **170**, 177 (2018).

[7] N. Ernst, K. Fackeldey, A. Volkamer, O. Opatz, and M. Weber, "Computation of temperature-dependent dissociation rates of metastable protein-ligand complexes," Mol. Simul. **45**, 904–911 (2019).

[8] C. Schütte, S. Klus, and C. Hartmann, "Overcoming the timescale barrier in molecular dynamics: Transfer operators, variational principles, and machine learning," Technical Report No. 22-25, ZIB, Berlin, 2022.

[9] C. Schütte, P. Koltai, and S. Klus, "On the numerical approximation of the Perron-Frobenius and Koopman operator," J. Comput. Dyn. **3**, 51–79 (2016).

[10] L. Donati, M. Weber, and B. G. Keller, "Markov models from the square root approximation of the Fokker–Planck equation: Calculating the grid-dependent flux," J. Phys.: Condens. Matter **33**, 115902 (2021).

[11] S. Röblitz and M. Weber, "Fuzzy spectral clustering by PCCA+: Application to Markov state models and data classification," Adv. Data Anal. Classif. **7**, 147–179 (2013).

[12] C. Hartmann, L. Richter, C. Schütte, and W. Zhang, "Variational characterization of free energy: Theory and algorithms," Entropy **19**, 626 (2017).

[13] N. Nüsken and L. Richter, "Solving high-dimensional Hamilton–Jacobi–Bellman PDEs using neural networks: Perspectives from the theory of controlled diffusions and measures on path space," Partial Differ. Equations Appl. **2**, 48 (2021).

[14] H. C. Lie, "Convexity of a stochastic control functional related to importance sampling of Itô diffusions," arXiv:1603.05900 (2016).

[15] A. Sikorski, "OptImpSampling.jl Julia package, tag #jmp," https://github.com/axsk/OptImpSampling.jl/tree/jmp, 2023.

02 February 2024 08:34:30