The Journal of Chemical Physics

ISOKANN: Invariant subspaces of Koopman operators learned by a neural network

Cite as: J. Chem. Phys. **153**, 114109 (2020); https://doi.org/10.1063/5.0015132
Submitted: 26 May 2020 • Accepted: 30 August 2020 • Published Online: 16 September 2020

Robert Julian Rabben, Sourav Ray and Marcus Weber

AIP Publishing

The Journal of Chemical Physics ARTICLE scitation.org/journal/jcp

ISOKANN: Invariant subspaces of Koopman operators learned by a neural network

Cite as: J. Chem. Phys. 153, 114109 (2020); doi: 10.1063/5.0015132
Submitted: 26 May 2020 • Accepted: 30 August 2020 •
Published Online: 16 September 2020

Robert Julian Rabben,[1,2] Sourav Ray,[1] and Marcus Weber[1,a)]

AFFILIATIONS

[1] Zuse Institute Berlin (ZIB), Takustr. 7, 14195 Berlin, Germany
[2] Institute of Theoretical Physics, Technische Universität Berlin, Straße des 17. Juni 135, 10623 Berlin, Germany

Note: This paper is part of the JCP Special Topic on Machine Learning Meets Chemical Physics.
[a)] **Author to whom correspondence should be addressed:** weber@zib.de

ABSTRACT

The problem of determining the rate of rare events in dynamical systems is quite well-known but still difficult to solve. Recent attempts to overcome this problem exploit the fact that dynamic systems can be represented by a linear operator, such as the Koopman operator. Mathematically, the rare event problem comes down to the difficulty in finding invariant subspaces of these Koopman operators $\mathcal{K}$. In this article, we describe a method to learn basis functions of invariant subspaces using an artificial neural network.

I. INTRODUCTION

In molecular dynamics (MD) simulations, it is often difficult to determine the rates of rare events. Using the information from long-time simulations suffers from the statistical disadvantage that the time steps with which the (mechanical) movements of molecular building blocks are simulated are in the range of femtoseconds, while the transitions of interest (conformational changes, binding events, aggregation of building blocks, etc.) are in the range of milliseconds or seconds. The transitions themselves can be abrupt, but they are very rare. Therefore, simply counting the rare events is statistically inefficient.

In the last decades, a lot of approaches have been invented in order to overcome this rare event problem of molecular simulation. The main idea behind using short-term trajectories to compute the rate of rare events goes back to the method of milestoning.[1] Instead of tracing the steps of one single long-term trajectory, many short-term trajectories are "started," which represent intermediate steps of the analyzed molecular event of interest. In this approach, only transitions between "neighboring" intermediate steps are computed. Hence, these are not rare events anymore. This method applies a certain clustering (or projection) of system states into subsets of "different intermediate steps." In essence, it is clustering in a high-dimensional space. Another way to circumvent rare events statistics is to apply transition path theory.[2] In this approach, the problem of estimating transition rates is transformed into a function approximation problem in high-dimensional spaces. Committor functions are derived from a differential equation based on a linear partial-differential operator. These functions include all relevant information about the rareness of the transition events. Many physical processes are highly non-linear and therefore complicated to simulate and analyze. However, for any Markov process, as for any equation with a time-derivative of the first order, the description with a *linear* operator is always possible if the time-depending evolution of the probability density distribution of system states is considered. The transfer[3] operator $\mathcal{T}^\tau$ propagates a density function ϱ of system states for a given lag-time τ. This operator is the basis for the committor function computation as well as for the clustering approach in milestoning.

Instead of computing committor functions from linear operators, one can also regard the rare event problem as a (Galerkin) projection problem from an infinite-dimensional linear operator to a finite-dimensional matrix.[4] The density function is projected from a continuous state space to a finite discrete set of states, i.e., to a vector ψ, which can then be propagated from a simple transition matrix P to a (simplified) state $P^\tau\psi$ for the time interval τ. This projection commutes with propagation of states if and only if the projection function is an element of the invariant

subspace of $\mathcal{T}^\tau$ or (more precisely) its adjunct, the Koopman-operator $\mathcal{K}^\tau = (\mathcal{T}^\tau)^*$.

Our aim is to contribute to the coarsening strategy by presenting a method for the automated generation of invariant subspaces of Koopman operators while avoiding a linear algebraic discretization of the function space of ϱ. In a similar way, the VAMPnets approach[5] tries to circumvent the curse of dimensionality. In VAMPnets, two linear subspaces are learned: one for the "input" step of the molecular process and one for the "output" step. This corresponds, in principle, to a singular value decomposition approach of $\mathcal{K}^\tau$. The fitness function of this learning strategy is based on the variational principle of this type of decomposition. However, the trained spaces are not invariant with regard to $\mathcal{K}^\tau$, which is crucial in order to extract the correct time scales and to derive kinetics from the basis functions. In contrast to VAMP, our novel artificial neural network (ANN) approach invariant subspaces of Koopman operators learned by a neural network (ISOKANN) learns single basis functions of invariant subspaces of $\mathcal{K}^\tau$. This corresponds to a Schur-based decomposition of matrices and provides the intended kinetic information.[6]

Although our idea (ISOKANN) is mathematically straightforward, the benefits of learning invariant subspaces from Koopman operators efficiently can be very diverse. From the basis functions, the statistics of rare transitions can be derived, and even the reaction coordinates (non-zero directions of the gradients of the basis functions) can be extracted.[7]

II. MATHEMATICAL FRAMEWORK

The ISOKANN method is illustrated with a simple stochastic process: a microscopic particle moving in two-dimensional space with overdamped Langevin dynamics (OLD). Although our method for approximating the invariant subspace of Koopman operators is valid, in general, for all Markov processes (i.e., whenever the stochastic or deterministic differential equation describing the dynamics has only a first order derivative in time), we will use Langevin dynamics for illustrative reasons. Using this type of dynamics is advantageous, because not only the invariant subspace but also the characteristic time scales can be directly computed from the result of the trained neural network (see Sec. IV C).

A. Overdamped Langevin dynamics and MD

Langevin dynamics describes how a microscopic particle moves in a medium according to Brownian motion under the additional influence of a potential energy. Here, overdamped implies that the particle reaches the new velocity instantaneously (without acceleration phase) after each impact because the particle is considered to be massless and therefore has no inertia.

The Overdamped Langevin Dynamics (OLD) is given by the following equation:[8,9]

$$\gamma\dot{\boldsymbol{x}} = -\nabla V(\boldsymbol{x}) + \sqrt{2\beta^{-1}}\boldsymbol{R}(t) \quad \text{with} \quad \beta = (k_B T)^{-1}, \tag{1}$$

where $\boldsymbol{x} = \boldsymbol{x}(t) \in \mathbb{R}^n$ is the location of a particle, t is the time variable, $\dot{\boldsymbol{x}}$ is the time-derivative, $V : \mathbb{R}^n \to \mathbb{R}$ is the potential energy, k_B is the Boltzmann constant, and T is temperature. $\boldsymbol{R}(t)$ is a stationary Gauss process with an expectation value 0: $\langle \boldsymbol{R}(t) \rangle = 0$. Furthermore, $\langle \boldsymbol{R}(t)\boldsymbol{R}(t') \rangle = \delta(t - t')$ with the Dirac delta distribution δ, i.e., the probabilities at different points in time, are uncorrelated. Thus, OLD is a Markov process. For the sake of simplicity, we will set $\gamma = 1$.

Simulations of molecular processes can also be performed with the OLD approach. In the software package GROMACS,[10] this is denoted as "Stochastic Dynamics." In principle, any Markov process is suitable for ISOKANN. In order to turn deterministic Hamiltonian dynamics into a stochastic Markov process rigorously, one would need to project the phase space onto the spatial coordinates by treating the momentum variables as random variables.[3]

B. Invariant subspaces of Koopman operators

D denotes the state space of states $\boldsymbol{x} \in D$ of a system, and $\varrho(\boldsymbol{x}, t) : D \times \mathbb{R} \to W$ with $W := \{y \in \mathbb{R} \mid 0 \le y \le 1\}$ represents the temporal evolution of a density function over D according to an infinitesimal generator $\mathcal{L}$,

$$\frac{\partial \varrho(\boldsymbol{x}, t)}{\partial t} = (\mathcal{L}\varrho)(\boldsymbol{x}, t). \tag{2}$$

For the OLD investigated here, the $\mathcal{L}$ corresponds to the Fokker–Planck operator. The formal solution of (2) with a given initial density function $\varrho(\boldsymbol{x}, t)$ at time t is given by

$$\varrho(\boldsymbol{x}, t + \tau) = (\mathcal{T}^\tau \varrho)(\boldsymbol{x}, t), \tag{3}$$

where the transfer operator $\mathcal{T}^\tau$ can formally be written as

$$\mathcal{T}^\tau = \mathrm{e}^{\tau\mathcal{L}}. \tag{4}$$

The operators $\mathcal{T}^\tau$ and $\mathcal{K}^\tau$ are (as already stated) a pair of adjoint operators,

$$\mathcal{K}^\tau = (\mathcal{T}^\tau)^*. \tag{5}$$

Thus, the eigenvalues of $\mathcal{T}^\tau$ are also eigenvalues of $\mathcal{K}^\tau$, denoted as λ_i. Furthermore, (4) provides an approach for calculating the eigenvalues of $\mathcal{L}$ as $\ln \lambda_i$. Due to Perron–Frobenius theory, eigenvalues λ_i close to the highest possible absolute value 1 correspond to eigenvalues of $\mathcal{L}$ near zero because $\ln 1 = 0$. For the corresponding eigenfunctions, the approximation $\frac{\partial \varrho}{\partial t} \approx 0\varrho = 0$ holds. If it is possible to interpret such a function $\partial\varrho$ as a density function, then the following is applicable.

Eigenvalues close to zero correspond to probability densities that hardly change over time, and therefore, an initial state arranged according to this function will only show rare transitions to other parts of the system space. The higher the potential barrier that must be overcome for the system to assume a different state, the rarer or less likely the transitions of the system over those barriers. Precisely, these transitions are of particular interest in the biochemical context. This could be, e.g., a process in which a ligand (a drug molecule) binds to a target protein (a receptor). For this reason, an algorithm is chosen for this methodology, which provides basis functions χ_s of the space spanned by the eigenfunctions

close to the eigenvalues with the largest magnitude. Of primary interest is a finite-dimensional $\mathcal{K}^\tau$-invariant subspace U of a function space F,

$$\mathcal{K}^\tau(U) \subseteq U, \quad \text{where} \quad U \subseteq F. \tag{6}$$

The formal definition of a Koopman operator using the expectation value $\mathbb{E}$ of function values $f(\boldsymbol{x}(t)) \in F$ for a given lag-time τ is

$$(\mathcal{K}^\tau f)(\boldsymbol{x}) := \mathbb{E}(f(\boldsymbol{x}(\tau)); \boldsymbol{x}(0) = \boldsymbol{x}), \tag{7}$$

where $x(0)$ is an initial state of an ensemble of OLD simulations for lag-time τ.

C. Rare event analysis

As already indicated in the Introduction, the coarsening strategy uses a simplified (or coarser) representation $\boldsymbol{\psi} = \boldsymbol{\psi}(t) = (\psi_1, \psi_2, \ldots, \psi_s, \ldots, \psi_\sigma)^T$ of the system state. In contrast, the exact representation consists of the atomic positions $\boldsymbol{x}$ of the molecules at the time t or the representation of the probability density distribution $\varrho(\boldsymbol{x}, t)$ as an average over these positions. From the previous example, for the ligands placed in a solution containing receptors, at time $t = 0$, the ligands are not bound to the receptors. All ligands are in the "unbound" state [i.e., $\boldsymbol{\psi}(0) = (1; 0)^T$]. Over time, the proportion of bound ligands (possibly) increases [e.g., $\boldsymbol{\psi}(\tau) = (0.8; 0.2)^T$], if after time τ, one-fifth of the ligands are bound. In more general terms, the component $\psi_s(t)$ of this vector, which is mapped according to W, corresponds to the expected relative frequency of the sth preferred conformation of the molecule or the sth metastable state of a system at time t, where $\psi_1 + \psi_2 + \cdots + \psi_\sigma = 1$.

The procedure of rare event analysis is shown in Fig. 1, which is taken from Ref. 4 and extended by additional diagonal arrows. The projections are represented by vertical arrows, and the propagations are represented by horizontal arrows; diagonal arrows symbolize a combination of both. The probabilities of rare transitions of the system into other metastable states are of interest. However, from an algorithmic point of view, the way of first projecting and then propagating is more easy to represent in a computer program since a continuous operator $\mathcal{T}^\tau$ cannot be fully represented.

It can be assumed that the set of functions χ_s represented by $\boldsymbol{\chi} = \boldsymbol{\chi}(\boldsymbol{x}) = (\chi_1, \chi_2, \ldots, \chi_\sigma)^T$ is the basis of an invariant subspace U according to (6). Furthermore, χ_s are assumed to be non-negative functions with function values between 0 and 1. Then, they can be interpreted as membership functions of a subregion B_s of D. χ_s is also a metastable subregion of the process, if the invariant subspace U corresponds to eigenvalues of $\mathcal{K}^\tau$ with the highest absolute values.[11] The diagram in Fig. 1 commutes because of the invariant subspace condition of the projection.[4] Changing the order of propagation (with $\mathcal{T}^\tau$ or with P^τ) and projection (with $\boldsymbol{\chi}$) leads to the same result. This will be used now.

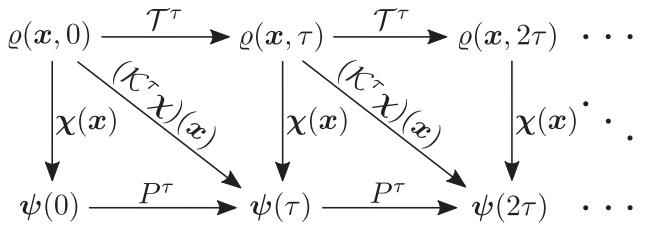

FIG. 1. Principle of the rare event analysis: Detailed state descriptions are positioned at the top of the diagram, and simplified descriptions are positioned at the bottom.

The projection of $\varrho(\boldsymbol{x}, t)$ onto $\boldsymbol{\psi}(t)$ is given by

$$\boldsymbol{\psi}(t) = \int_D \varrho(\boldsymbol{x}, t)\boldsymbol{\chi}(\boldsymbol{x})d\boldsymbol{x}. \tag{8}$$

Equivalently, the projection of $\varrho(\boldsymbol{x}, t + \tau)$ onto $\boldsymbol{\psi}(t + \tau)$ regarding (3) is

$$\boldsymbol{\psi}(t + \tau) = \int_D (\mathcal{T}^\tau \varrho)(\boldsymbol{x}, t)\boldsymbol{\chi}(\boldsymbol{x})d\boldsymbol{x}. \tag{9}$$

Finally, because of (5), Eq. (9) turns into

$$\boldsymbol{\psi}(t + \tau) = \int_D \varrho(\boldsymbol{x}, t)(\mathcal{K}^\tau \boldsymbol{\chi})(\boldsymbol{x})d\boldsymbol{x}. \tag{10}$$

Equation (10) corresponds to the diagonal arrows in the above diagram. Using the invariant subset definition (6), each element of U is a linear combination of basis elements of U, $\mathcal{K}^\tau \chi_s = \sum_{j=1}^{\sigma} p_{sj}^{(\tau)} \chi_j$. Therefore, the propagations between the simplified states can be expressed by a transition matrix $P^\tau \in \mathbb{R}^{\sigma \times \sigma}$ with entries $p_{sj}^{(\tau)}$. As an alternative to using expression (9) or (10), the propagated state $\boldsymbol{\psi}(t + \tau)$ can be directly represented by using $\boldsymbol{\psi}(t)$ to avoid the evaluation of the mapping $\mathcal{T}^\tau$,

$$\psi_s(t + \tau) = \sum_{j=1}^{\sigma} p_{sj}^{(\tau)} \psi_j(t). \tag{11}$$

Determining basis vectors χ_s is therefore the main step of rare event analysis. These functions can only be approximated numerically, which is the primary aim of this article. For this task, artificial neural networks (ANNs) are excellently suited (for each χ_s, its own network is trained).

III. ALGORITHMIC APPROACH

A methodology for the determination of an invariant subspace of a Koopman operator has been described, which avoids a discretization of the state space. ISOKANN is based on the power method (also known as *von Mises iteration*). It uses an algorithm that (under certain conditions) supplies the eigenvector for a given matrix at the eigenvalue with the largest magnitude.[12] The power method is applied to continuous functions that converge against the eigenfunctions connected to eigenvalues with the largest absolute value. The use of the power method is advantageous because only that part of the spectrum of eigenfunctions is sought that can be assigned to the eigenvalues with the largest absolute value.

A. Modified power method

The power method is an iterative procedure, where in each step the linear mapping, or here the Koopman operator $\mathcal{K}^\tau$, is applied to the current approximation f_i, with $i \in \mathbb{N}_0$, and then normalized,

$$f_{i+1} := \frac{\mathcal{K}^\tau f_i}{\|\mathcal{K}^\tau f_i\|}. \tag{12}$$

However, in our case, the scaling with the inverse of the norm $\|\mathcal{K}^\tau f_i\|^{-1}$ is supplemented by a shifting in such a way that the resulting function is χ_{i+1}: $D \to [0, 1]$. Therefore, the modified variant of the power method is

$$\chi_{i+1} := \frac{\mathcal{K}^\tau \chi_i - \min(\mathcal{K}^\tau \chi_i)}{\|\mathcal{K}^\tau \chi_i - \min(\mathcal{K}^\tau \chi_i)\|_{max}}. \tag{13}$$

Thus, convergence against the constant function, $f(x) \equiv 1$, is avoided, which is always the dominant eigenfunction of the Koopman operator. Instead, the procedure then converges against a *membership function* $\chi := \lim_{i\to\infty} \chi_i$, which will be explained later.

The requirement of an appropriate convergence criterion immediately arises for this iterative procedure. If χ is the linear combination of the constant function and a further dominant eigenfunction of $\mathcal{K}^\tau$, then a linear relation is expected,

$$\mathcal{K}^\tau \chi \sim \chi. \tag{14}$$

Thus, the pairs $(\chi_i(\boldsymbol{x}_n), \tilde{\chi}_{i+1}(\boldsymbol{x}_n))_n$, with $n \in \mathbb{N}$, are interpolated with a straight line, where $\tilde{\chi}_{i+1} := \mathcal{K}^\tau \chi_i$, i.e., the function values that were neither scaled nor shifted after the last iteration step. A suitable measure for the quality of a linear relationship is the correlation coefficient r.[13] In the context of the proposed algorithm, it is arbitrarily determined that the convergence criterion is met if $r \geq 0.999$ applies. For technical reasons, the correlation coefficient of the pairs just mentioned is not calculated, but that of the pairs $(\chi_i(\boldsymbol{x}_n), \chi_{i+1}(\boldsymbol{x}_n))_n$ is calculated. The calculated value of the correlation coefficient is the same for both sets of pairs. However, the linear single regression results in different gradients and axis intercepts of the straight line. As shown later, only the regression line to the pairs $(\chi_i(\boldsymbol{x}_n), \tilde{\chi}_{i+1}(\boldsymbol{x}_n))_n$ of the final iteration step is of interest.

In the illustrations, ISOKANN is applied to OLD (1) with the following potential $V(\boldsymbol{x}) : \mathbb{R}^2 \to \mathbb{R}$, which is taken from Ref. 14 and has already been applied in Refs. 4 and 7:

$$\begin{aligned} V = {} & 3e^{-{x_1}^2-\left(x_2-\frac{1}{3}\right)^2} - 3e^{-x_1^2-\left(x_2-\frac{5}{3}\right)^2} - 5e^{-(x_1-1)^2-{x_2}^2} \\ & - 5e^{-(x_1+1)^2-{x_2}^2} + 0.2{x_1}^4 + 0.2\left(x_2 - \frac{1}{3}\right)^4. \end{aligned} \tag{15}$$

The potential has three minima identified as metastabilities of the dynamics, which are approximately in the proximity of the following three points (see Fig. 2):

- Metastability 1: (0; 1.5),
- Metastability 2: (−1; 0),
- Metastability 3: (1; 0).

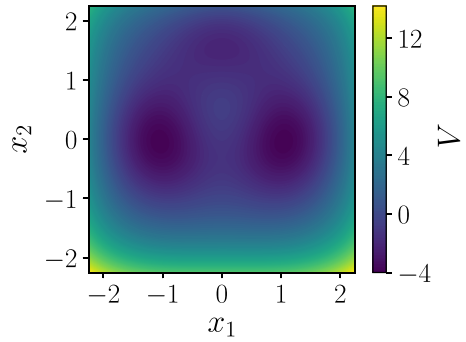

FIG. 2. Potential.

Thus, a set of three eigenfunctions with eigenvalues close to 1 can be expected. The next eigenfunctions have eigenvalues that are far away from 1. This clear separation of more and less dominant eigenfunctions in the spectrum of the Koopman operator makes the power method very efficient. The first eigenfunction is, as previously mentioned, the constant function $\mathbb{1}$. The next two eigenfunctions are e_2 and e_3. The χ membership functions are generally different from the eigenfunctions of the Koopman operator. However, we can find three membership functions that span the same invariant subspace like the eigenfunctions. Thus, we apply the power method (13) three times, each with different initial functions, in order to retrieve three linear independent functions. The three initial functions $\tilde{\chi}_{s_0} = \tilde{\chi}_{s_0}(x_1, x_2) : D \to \mathbb{R}$, where $s \in \{1, 2, 3\}$ are randomly chosen. In the OLD example,

$$\begin{aligned} \tilde{\chi}_{1_0} = {} & \sin(x_1 + 0.5x_2) + 0.5x_2 \cos(2{x_2}^2) + 0.01{x_1}^4 \\ & + 0.01(x_1 x_2)^3, \end{aligned} \tag{16}$$

$$\begin{aligned} \tilde{\chi}_{2_0} = {} & -2\sin\left(\frac{x_1}{3} + \frac{x_2}{6}\right) + 0.5x_2 \cos({x_2}^2) - 0.01{x_2}^4 \\ & + 0.005(-(x_1 - 0.5)(x_2 - 0.5))^3, \end{aligned} \tag{17}$$

$$\begin{aligned} \tilde{\chi}_{3_0} = {} & \sin(0.5x_1 + x_2) + 0.5\cos(0.1(x_2 - 3)^2) \\ & - 10^{-4}((x_2 - 6) - (0.7x_1)^2)^4. \end{aligned} \tag{18}$$

From the initial functions, we directly arrive at the scaled and shifted initial functions $\chi_{s_0} = \chi_{s_0}(x_1, x_2) : D \to [0, 1]$,

$$\chi_{s_0} := \frac{\tilde{\chi}_{s_0} - \min(\tilde{\chi}_{s_0})}{\|\tilde{\chi}_{s_0} - \min(\tilde{\chi}_{s_0})\|_{max}}. \tag{19}$$

The set of the three linear independent membership functions spans a three-dimensional space. After the convergence of the method, the three functions χ_1, χ_2, and χ_3 approximately span an invariant subspace of the Koopman operator, i.e., it is expected that

$$\lim_{i\to\infty} \chi_{s_i} =: \chi_s = a_s \mathbb{1} + b_s e_2 + c_s e_3, \quad \text{where } a_s, b_s, c_s \in \mathbb{R}, \tag{20}$$

because dominant eigenvalues "survive" the power method, whereas lower eigenvalues "vanish."

B. Training strategy

The initial functions χ_{s_0} and all approximations χ_{s_i} are obtained from a multilayer neural network. The ANN, therefore, represents an approximation of χ_{s_0} or χ_{s_i} at the starting point or after each individual iteration step of the learning phase. $\chi_{s_0} \in F$ can be chosen arbitrarily, but each of them has to contain a non-vanishing portion that is an element of a basis of the invariant subspace being searched for so that ISOKANN can converge on that portion. This condition should be fulfilled here because in this methodology, an approximation of the functions χ_{s_0} takes place and an exact zero fraction becomes improbable (Fig. 3).

Another possible approach would be to not select the initial functions manually but to directly generate the ANN with random weightings and threshold values. This would be advantageous for an application to problems that require a particularly large number of initial functions. Furthermore, initial functions generated from pseudo-random numbers (or even from real random numbers) are usually preferable to manually selected ones, in order to avoid any systematic error. However, in the illustrative OLD example, it is advantageous that the initial functions can be directly influenced so that the procedure can be tested with different start configurations. Therefore, it is possible to test, e.g., the shape of the membership function at the end if higher function values are to be found in the range of a certain metastability at the start time.

The algorithm ISOKANN for determining the invariant subspace of the Koopman operator is illustrated with a Python-style pseudocode (code 3). The source code, including the details of the system configuration, can be referred to in the supplementary material.

At the beginning, a large number of random test points $\boldsymbol{x}_{test_n} \widehat{=} \texttt{x\_test[n]}$ with $n \in \{1, 2, \ldots, \texttt{n\_points}\}$ are generated (pseudocode l. 1) such that $\boldsymbol{x}_{test_n} \in D$.

Next, the ANN `model` is initialized (see pseudocode l. 2). An ANN with 5 hidden layers and 50 artificial neurons per layer is created. After some test runs, these values turn out to be well-suited with regard to the highest possible accuracy on the one hand and also the overfitting to be avoided. In the last layer, a linear function is used as the activation function, and in all other layers, the ReLu function is used. Furthermore, the ANN is configured with the error function *mean square error* and with the optimizer *Adam*.[15]

This is followed by a loop that implements the power method (pseudocode ll. 4 ff.). Then, a multitude of random training points $\boldsymbol{x}_{train_k}$ with $k \in \{1, 2, \ldots, \texttt{n\_train\_points}\}$ is generated (pseudocode l. 5), for which $\boldsymbol{x}_{train_k} \in D$. It should be noted that the same test points are always used (even for different initial functions). However, new random training points are generated in each iteration step of the power method.

A new execution of the source code provides the same pseudo-random numbers used as test and training points for replicability. The trajectories of the dynamics will always be the same. However, this does not apply to the random numbers used by functions from the libraries *Keras*[16] and *TensorFlow*.[17] These two libraries are used in the program for implementing the ANN. Keras is the frontend, and TensorFlow is the backend. Thus, results can still be expected that differ slightly from each other for statistical reasons.

```
1  x_test = random_array(n_points)
2  model = initialize_neural_network()
3  previous_y_predicted = empty_array(n_points)
4  for i in range (n_iterations + 1):
5    x_train = random_array(n_train_points)
6    if i == 0:
7      y_train = initial_function_array(x_train)
8    else:
9      for k in range(n_train_points):
10       y_tmp = empty_array(n_trajectories_per_point)
11       for j in range(n_trajectories_per_point):
12         x_end = dynamic(x_train[k])
13         y_tmp[j] = model.predict(x_end)
14       y_train[k] = mean(y_tmp)
15   y_train = scale_and_shift(y_train)
16   model.fit(x_train, y_train)
17   y_predicted = model.predict(x_test)
18   if i >= 1:
19     pearson_r = calc_pearson_r(previous_y_predicted, y_predicted)
20   previous_y_predicted = y_predicted
21   if pearson_r >= 0.999:
22     break
```

FIG. 3. Pseudocode: ISOKANN for determining the invariant subspaces of the Koopman operator.

If the loop is run for the first time, the exact function values for the individual training points are calculated from the (current) initial function and stored in the array `y_train` (pseudocode ll. 6 f.). In the first loop pass, the power method is not yet executed. In the succeeding pass, two nested loops are executed (pseudocode ll. 8-14).

The first loop iterates over the number of training points and the second over the number of trajectories per point `n_trajectories_per_point`. Starting from the starting points `x_train`, trajectories are calculated according to dynamics (1), and the end points are stored in `x_end` (pseudocode l. 12). The duration of a trajectory is τ as the following applies: $\tau \mathrel{\widehat{=}} \texttt{timestep} \cdot \texttt{n\_steps}$. Then, calculation of the approximated function values `x_tmp` at the end points takes place with the ANN (pseudocode l. 13). The expected value is calculated from the set of all function values for a training point (pseudocode l. 14). The expected values y_{train_k} are scaled and shifted (pseudocode l. 15) so that the following applies: $y_{train_k} \in [0,1]$. As per (7), with $f = \chi_{s_i}$, the set of expected values then represents the training data `y_train`. In the next step, the ANN is trained with it (pseudocode l. 16). For the current iteration step, $i \mathrel{\widehat{=}}$ `i` would therefore apply at the end of the loop of the power method,

$$\chi_{s_i} \approx \chi_{s_{app_i}} \mathrel{\widehat{=}} \texttt{model}, \tag{21}$$

$$\chi_{s_{app_i}}(\boldsymbol{x}_{train_k}) = y_{train_k}. \tag{22}$$

Thereupon, the approximated function values at the test points can be calculated with the ANN and assigned to the array `y_predicted` (pseudocode l. 17).

If it is not the first loop pass (i.e., if $i \geq 1$), the correlation coefficient $r(y_{pred_i}, y_{pred_{i-1}})$ of the pairs `y_predicted` and `y_previous_predicted` is determined and assigned to the variable `pearson_r` (pseudocode ll. 18 f.). Afterward, the reference of `y_predicted` is assigned to `y_previous_predicted` (pseudocode l. 20), which has already been declared outside the scope of the loop such that the next iteration step is possible (pseudocode l. 3). If it is true that the correlation coefficient $r(y_{pred_i}, y_{pred_{i-1}}) \geq 0.999$, the convergence criterion is considered to be fulfilled and the power method is aborted (pseudocode ll. 21 f.).

In the last iteration step, the ANN must be trained with neither scaled nor shifted training data. This is exactly the difference between $\chi_{s_{app_i}}$ and $\tilde{\chi}_{s_{app_i}}$.

The algorithm described above has to be executed once for each of the three initial functions, where $s \mathrel{\widehat{=}}$ `initial_function_selection`. The same test points are always selected for comparison purposes. A separate (new) ANN is created for each initial function. On the other hand, the existing net is merely adjusted in each iteration step of the power method since it can be assumed that the function to be approximated changes only slightly during a single iteration step. This increases the accuracy of the approximation for the same computational effort for training the mesh.

Parallel to the calculation of membership functions using the power method and an ANN, the membership function ($\chi_{s_{calc_i}}$) is calculated directly for the sake of comparison by using long trajectories. The number of long trajectories generated for each test point is also `n_trajectories_per_point`. For the respective iteration step `i`, these consist of `i` multiplied by the number of steps of the trajectories used in the first mentioned calculation (i.e., the number `i` · `n_steps`). Hence, (7) is applied, with $i\tau$, instead of τ, and the arithmetic expectation value, instead of the exact one,

$$\begin{aligned}\chi_{s_{calc_i}} &:= \frac{\bar{\mathbb{E}} - \min(\bar{\mathbb{E}})}{\|\bar{\mathbb{E}} - \min(\bar{\mathbb{E}})\|_{max}} \\ &\approx \frac{(\mathcal{K}^{i\tau}\chi_{s_0})(\boldsymbol{x}) - \min((\mathcal{K}^{i\tau}\chi_{s_0})(\boldsymbol{x}))}{\|(\mathcal{K}^{i\tau}\chi_{s_0})(\boldsymbol{x}) - \min((\mathcal{K}^{i\tau}\chi_{s_0})(\boldsymbol{x}))\|_{max}},\end{aligned} \tag{23}$$

where $\bar{\mathbb{E}} = \bar{\mathbb{E}}(\chi_{s_0}(\boldsymbol{x}(i\tau)); \boldsymbol{x}(0) = \boldsymbol{x})$ does not represent the exact expected value $\mathbb{E}$, which is based on the theoretically expected frequency, but a numerical approximation of it inevitably results from a limited number of trajectories. This is because the formula using $\mathbb{E} = \mathbb{E}(\chi_{s_0}(\boldsymbol{x}(i\tau)); \boldsymbol{x}(0) = \boldsymbol{x})$ represents the following relationship:

$$\begin{aligned}\chi_{s_i} &= \frac{\mathbb{E} - \min(\mathbb{E})}{\|\mathbb{E} - \min(\mathbb{E})\|_{max}} \\ &= \frac{(\mathcal{K}^{i\tau}\chi_{s_0})(\boldsymbol{x}) - \min((\mathcal{K}^{i\tau}\chi_{s_0})(\boldsymbol{x}))}{\|(\mathcal{K}^{i\tau}\chi_{s_0})(\boldsymbol{x}) - \min((\mathcal{K}^{i\tau}\chi_{s_0})(\boldsymbol{x}))\|_{max}}.\end{aligned} \tag{24}$$

Nevertheless, the long trajectories are divided into several small ones during the direct computation for technical reasons. These trajectories are joined one after the other during the iteration steps of the loop. Thus, without redundant computational effort, a result from the direct calculation is obtained after each individual iteration step. The function values from the direct calculation are stored in the variable `y_calculated`. Both types of calculation suggest similar function values,

$$\chi_{s_i} \approx \chi_{s_{app_i}} \approx \chi_{s_{calc_i}}. \tag{25}$$

In an array `error` $\mathrel{\widehat{=}} \chi_{s_{err_i}}$, the difference between `y_calculated` and `y_predicted` (i.e., the errors between both arrays) are then stored and plotted.

IV. DISCUSSION OF THE INTRODUCTORY 2D-EXAMPLE

The results of ISOKANN presented earlier are described and discussed.

A. Rare event problem

To depict the motion of a particle according to OLD, a certain number (`n_saved_trajectories`) of trajectories are plotted. Figure 4 shows an example of a trajectory with 10 000 steps. The particle starts in the first metastability and finds its way to the third metastability after relatively few steps, where it remains until the end of the simulation. Repeating the simulation with the same starting point shows a similar picture. In most cases, the particle migrates into one of the other two metastabilities after a short time and is only rarely able to leave metastability 2 or 3. If it does, it usually migrates back into metastability 1 but rarely from the second metastability directly into the third or vice versa. The reason behind this is that the value of the potential at the location of metastabilities 2 and 3 is lower than at the location of metastability 1. In addition,

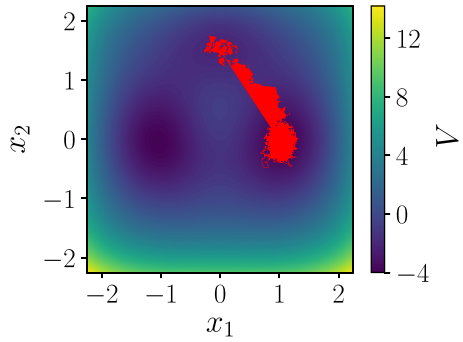

FIG. 4. Representation of a trajectory of a particle that starts at point (0; 1.5) and moves 10 000 steps according to the dynamics. The potential is shown in the background.

a higher potential barrier must be overcome directly between the lower two metastabilities than in the detour via the metastability on top. Increasing the temperature, which is indirectly achieved by changing the variable `BETA` $\widehat{=}\beta$, leads to larger steps because the stochastic part of the movement becomes larger. Therefore, at higher temperatures, a change of the particle into other metastable states is more often observed than at low temperatures.

However, in practical molecular simulations, it happens extremely rarely that the state changes from one metastability to another. Instead, the states remain in the same metastability all the time. The advantage of this approach is that instead of long trajectories leading from one metastability to another, only short trajectories are required, which can start at any location. An idea for further studies is to let the trajectories start at transient areas (i.e., areas with maximum potential values) in order to learn more about the transitions.

B. Application of ISOKANN

In order for the methodology to provide correct results, it is necessary for the ANN to be a sufficiently accurate approximation of the function to be learned after training. From Fig. 5, it can be seen that the ANN approximates the initial function well; therefore, the requirement is fulfilled. The exact function values at the test points are plotted on the left-hand side of the graph, which are calculated directly with (16). In the middle, the approximation of the function values at the test points is plotted by the ANN. The differences are negligible. The error between the exact and the approximated function values is shown on the right. The maximum error is approximately 0.060 for the first initial function, 0.011 for the second, and 0.023 for the third. A similar situation is observed when the script is executed again. This is due to the more complicated nature of the first initial function (e.g., shorter period length of trigonometric functions) compared to the other two. Nevertheless, each time the script is executed again, the distribution of the error is at least slightly different. The reason is that the initial setting (weightings and thresholds) of the ANN is randomly chosen, and the training also contains random components (these are pseudo-random numbers). This leads to always differently trained nets. In contrast, the predictions of the ANN are purely deterministic.

The number of steps of the trajectories used in the methodology is 10, i.e., significantly lesser than in the trajectory shown. The reason behind the low number of steps will be discussed later. Therefore, the ANN converges slowly against the membership function.

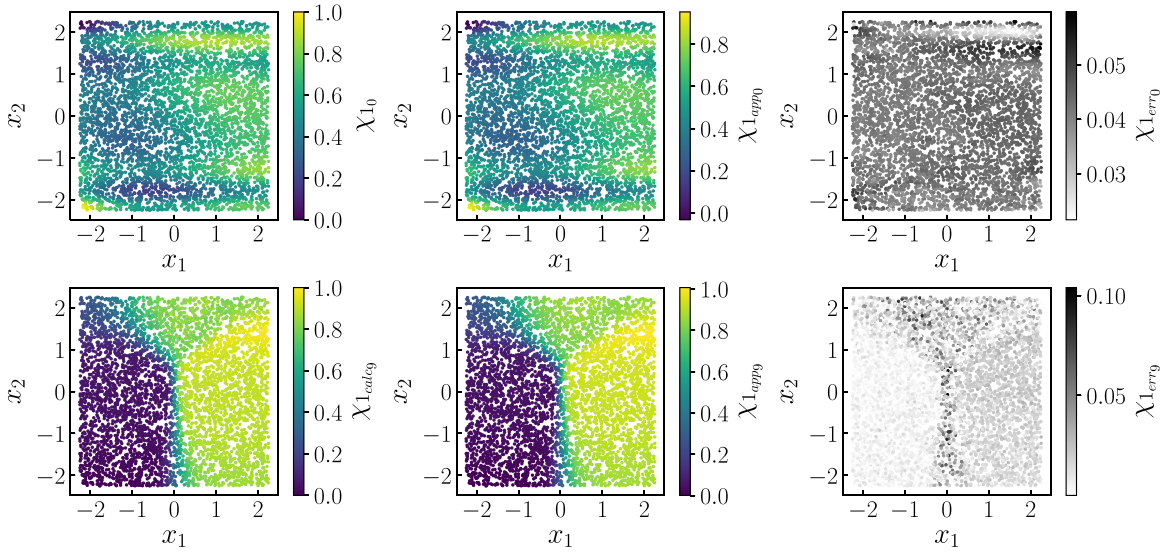

FIG. 5. Initial function 1: On the left side, the directly calculated function values are shown. In the middle, the approximation of the function values with the ANN is shown. The amount of the difference between the function values is shown on the right. The plots above show the function values before the power method is executed, and the plots below show the function values after the ninth iteration step is completed.

TABLE I. Median function values $\Omega_s^{(z)}$ in the region B_z of function χ_s.

	Metastability 1	Metastability 2	Metastability 3
Function 1	$\Omega_1^{(1)}$: 0.8	$\Omega_1^{(2)}$: 0.0	$\Omega_1^{(3)}$: 0.9
Function 2	$\Omega_2^{(1)}$: 0.1	$\Omega_2^{(2)}$: 1.0	$\Omega_2^{(3)}$: 0.3
Function 3	$\Omega_3^{(1)}$: 1.0	$\Omega_3^{(2)}$: 0.0	$\Omega_3^{(3)}$: 0.6

In the plots generated by the script after each single iteration step of the power method, the development of the ANN as an approximator from the initial function to the membership function can hence be traced exactly. The initial functions randomly selected at the beginning are iterated into the membership functions, which have three plateaus. The corresponding (mean) function values will be denoted as $\Omega_s^{(z)}$, where $z \in \{1, 2, 3\}$. A comparison with previous results from the literature (see Refs. 4 and 7), in which the same game example is treated, shows the similarity of the generated membership functions. Three ranges $\Omega_s^{(z)}$ can be seen per function since the dynamic also has three metastabilities. Each area of a plateau $\Omega_s^{(z)}$ contains the location of a metastability. The values merge continuously at the border of the areas. In Table I, the approximate average values of $\Omega_s^{(z)}$ and the metastabilities are listed. Due to the approximation, the values are not exact but lie approximately between 0 and 1.

In accordance with the fact that the ANN continuously converges against the membership functions, the plotted pairs also approach toward the regression line with every further iteration step. Correspondingly, the correlation coefficient also shows a value closer to 1 with every iteration step. The following list shows the total number of iteration steps until convergence is reached and the function obtained from the linear regression. The values are rounded to five decimals:

- Initial Function 1: After iteration step 9, $\tilde{\chi}1_{app_9} = 0.995\,30\chi 1_{app_8} + 0.006\,69$.
- Initial Function 2: After iteration step 7, $\tilde{\chi}2_{app_7} = 0.994\,67\chi 2_{app_6} + 0.005\,43$.
- Initial Function 3: After iteration step 12, $\tilde{\chi}3_{app_{12}} = 0.991\,44\chi 3_{app_{11}} + 0.001\,97$.

In Fig. 6 (bottom row), the functions are drawn as red straight lines. It should be noted that in the bottom row (in contrast to the upper row), the function values are plotted on the ordinate, which have neither been scaled nor shifted. Furthermore, Fig. 6 shows the strong dispersion of the pairs at the beginning, and the clear agreement of the pairs with the regression line after the convergence criterion has been reached (upper and lower rows). The corresponding values on the abscissa match the values from Table I quite well.

From Table I, it is apparent that there is a linear independence between the three membership functions. Together, they span a three-dimensional space U,

$$U := \operatorname{span}\{\tilde{\chi}1_{app_9}, \tilde{\chi}2_{app_7}, \tilde{\chi}3_{app_{12}}\}. \tag{26}$$

The $\mathcal{K}^\tau$ invariant subspace U is the central result of the von Mises iteration methodology.

C.Further calculations

The three trained functions span a three-dimensional invariant subspace of the Koopman operator $\mathcal{K}^\tau$. Due to the novel rescaling and shifting approach (23) used, the basis functions can also be seen as membership functions of fuzzy sets or regions B_s of the state space. In order to obtain a partition of unity, i.e., to get $\sum_{s=1}^{3} \tilde{\chi}_s(x) = 1$, a linear combination of these functions has to be determined by PCCA+,[18] which is beyond the scope of this article. Thus, continuing from the previous results, the next step would be to determine the exit rates (the rare event statistics) from the

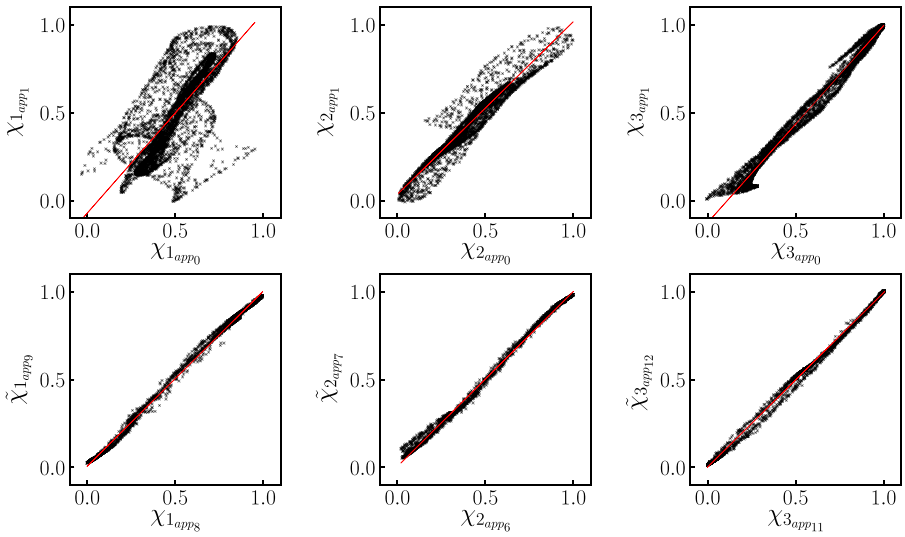

FIG. 6. Display of the function value dependency on the function values after the previous iteration step (from left to right) for the first, second, and third initial functions. Furthermore, a regression line is represented by each of these points. Upper row: representation of the pairs $(\chi_{s_{app_0}}(\boldsymbol{x}_n), \chi_{s_{app_1}}(\boldsymbol{x}_n))_n$ after completion of the first iteration step. Lower row (from left to right): representation of the pairs $(\chi_{s_{app_i}}(\boldsymbol{x}_n), \tilde{\chi}_{s_{app_{i+1}}}(\boldsymbol{x}_n))_n$ after completion of the ninth, seventh, and twelfth iteration steps.

linear functions listed above. This *exit rate* r_{exit_s} can be determined[7] for each region B_s. Furthermore, the *holding probability* p_s can be also calculated,

$$p_s(\boldsymbol{x}, \tau) = \tilde{\chi}_s(\boldsymbol{x}) e^{-r_{exit_s}\tau}. \tag{27}$$

The above equation implies that our learned functions are proportional to the holding probabilities (for fixed τ). The holding probability depends on the location of the particle (or the state of the system) and the time (the duration of the trajectory). It indicates that the probability of a particle being at $\boldsymbol{x}$ at the starting point will be in the region B_s for the entire duration τ of the observation. It decreases at a certain rate, namely, the exit rate r_{exit_s}. With increasing duration of the observation, the holding probability decreases. The membership function contains the spatial dependency. For example, if the particle is completely outside the region B_s from the outset, the holding probability is 0. However, if the particle starts in the middle of B_s, then p_s is higher. If the particle starts closer to the edge of B_s, then p_s is again smaller. With the algorithm described in Ref. 7, the exit rates can only be determined if the slope is less than 1 and the intercept is greater than 0. Both apply to the three determined regression lines. The exit rate is assumed to be the negative logarithm of the slope of the linear regressions divided by the lag-time of the simulation. The closer the slope is to 1, the rarer the transitions are between the different areas. Using proportionality (27), the role of $\nabla\tilde{\chi}$ becomes evident as the "spatial direction of the transition." It provides information about the reaction coordinates.

D. ISOKANN learns important transitions

Compared to the direct calculation, the ISOKANN method has the crucial advantage that with each new iteration step of the power method, the distribution of the starting points of the particles in space again corresponds to a statistically uniform distribution. The end points of the trajectories from the last iteration step are mostly located at places where the potential has low values. If the dynamics would be continued at these end points, then much more would be learned about the behavior of the particles at the minima and little about the behavior at places with high potential values (the transient regions). The behavior of the system is especially interesting in transient regions. For example, if a particle frequently leaves a certain area, then the ANN learns that, probably, there is a transient region to be found and how "quick" transitions take place. However, learning the behavior of the particles in the region of the minima hardly leads to a new insight because the particles move there mainly according to statistical noise and remain at the minimum. The computational effort required to simulate longer trajectories of molecular motion is not of any use if the trajectory arrived at minima since for the direct calculation approach, during most of the calculation time, only the dynamics at the minima are simulated and thus learned. This means that the structure of the dynamics in the entire space is not captured in the direct calculation, whereas the ANN method captures much more. Hence, the primary disadvantage of the direct calculation is that an even distribution occurs only once at the beginning. Accordingly, the result from the ANN method is closer to the theoretical exact membership function. Furthermore, optimal placements ($\chi \approx 0.5$) of starting points are possible within the iterative procedure of the power method based on the iteratively trained functions.

E. Influence of the initial ANN configuration

The reason behind configuring the ISOKANN methodology with a small number of steps `n_steps` is that the trajectories start more often from equally distributed points, and thus, the structure of the dynamics is learned more evenly and extensively. However, with a small number of steps applied, the number of iteration steps required increases proportionally. The configuration of the simulation with a higher number of steps `n_steps` leads to a transition from the ANN method to direct calculation, with the disadvantages already mentioned.

The indirect increase in the temperature in `BETA` leads to membership functions that show larger transitions between the ranges of constant values. This is due to the larger steps and due to the associated possibility of leaving metastable states more easily. Increasing `n_trajectories_per_point` results in the arithmetic expectation value being closer to the exact expectation value, and as a result, the membership function is also closer to the theoretical exact membership function. Similarly, increasing `n_train_points` leads to a larger amount of training data and thus also to a better result. Furthermore, increasing the number of hidden layers of the ANN `n_hidden_layers`, and the number of neurons per layer `n_neurons_per_layer`, leads to a more accurate approximation and therefore to a better result, provided that the amount of training data is also increased at the same time. Otherwise, there is a risk of overfitting and obtaining a worse result.

V. DISCUSSION OF A HIGH-DIMENSIONAL EXAMPLE

In contrast to the procedures in Refs. 4 and 7, ISOKANN avoids any discretization of the state space. The main challenge in the transition from a toy example to a problem of relevance, in practice, is the large computational cost involved. In the toy example shown here, the state space of the particle has only two dimensions. However, 1-ethylpiperidin, the central building block of the opioid fentanyl, consists of 23 atoms with (theoretically) three degrees of freedom each. Hence, the state space of this molecule has 69 dimensions. The application of methods that require the discretization of this high-dimensional space becomes too expensive due to the curse of dimensionality. Already, the rough discretization in ten subsets per dimension would lead, with the molecule mentioned before, to the enormous number of 10^{69} subsets of the state space. ISOKANN allows for the determination of the membership functions and the corresponding output rates, without the need to discretize the state space even once. Nevertheless, the generation of the training data and the training itself in a high-dimensional space involves a high computational cost. Therefore, only trajectories from a suitable subset of the state space should be generated. It is also possible that the number of degrees of freedom for molecules is reduced if the movement of some atoms is restricted due to the bonds. Then, the state of the whole molecule is effectively dependent only on the state of a certain number of atoms. This is another advantage of ISOKANN. Each dimension of the state space corresponds to an input of the first layer of ISOANN. During training, the weights of exactly those

inputs are automatically reduced, which correspond to degrees of freedom in which there is effectively no or only little movement. In such cases, the whole ANN can be designed smaller from the outset and can then be trained with less effort.

It would be advantageous to have software that calculates the optimal settings for a given problem, taking into account the desired accuracy. This includes, on the one hand, the number of training points, trajectories, and steps, and on the other hand, the number of hidden layers and neurons per layer. Suitable values of these parameters were determined only by manual frequent testing and estimation. Before training the network, the determination of the optimal number of epochs could also be done. Furthermore, alternative ANN learning methods could be used, which are more suitable for the underlying problem. Notably, with regard to the application to high-dimensional spaces, the approach from Ref. 19 appears promising. Since the generation of trajectories is particularly suitable for concurrent execution, parallel programming should be considered here. Although we have only shown our results for a small-dimensional example so far, we want to present in this section how high-dimensional molecular systems can be investigated with our proposed ISOKANN method.

A. A 243-dimensional model

Molecular dynamics (MD) simulations of the interaction of the μ-opioid receptor at pH 7, embedded in 1-palmitoyl-2-oleoyl-sn glycerol-3-phosphatidyl choline (POPC) bilayer,[20] with the protonated fentanyl was used to generate the data required to test the ISOKANN algorithm for a more complex system (Fig. 7). More specifically, the interaction of the opioid ligand with the crucial ASP 147 and HIS 297 residues of the receptor[21] was analyzed in detail in terms of three-dimensional coordinate positions of the corresponding 83 atoms (a 249-dimensional example). In order to get rid of three rotational and three translational degrees of freedom, the molecular structures have been aligned to the first frame of the simulation, which reduces the degrees of freedom from 249 to 243. These degrees of freedom are the input layer of the ANN. More details about the modeling and simulation aspect of this binding process can be found in the supplementary material.

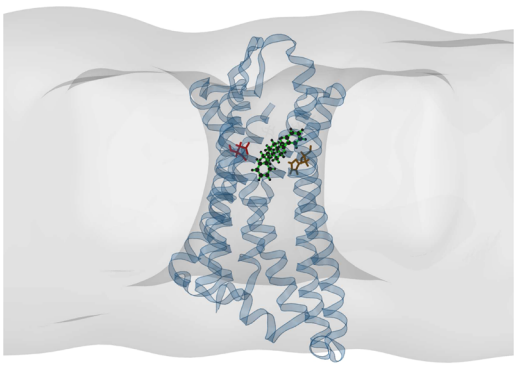

FIG. 7. Protonated fentanyl opioid in complex with the μ-opioid receptor at pH 7, embedded in a POPC bilayer, with ASP 147 and HIS 297 residues depicted in red and brown, respectively.

B. Algorithmic details for using long-term trajectories

The algorithm ISOKANN on p. 5 is based on two major ideas:

1. The training points, which are also the starting points of many short-term trajectories, should be distributed uniformly in the configurational space of the molecular system. Thus, trajectories that overcome free-energy barriers are not necessarily required, and the simulation can be started from high-energy levels of the barriers. The result of the first training cycle also provides information about the location of these barriers in space [for transient states x, $\chi(x) \approx 0.5$]. This idea is not compatible with long-term MD trajectories (having many integration steps). In general, straight forward MD-simulations depend on barrier-crossings, and they rarely reach the "top of these barriers."
2. In order to evaluate $(\mathcal{K}^\tau f)(x)$, many short-time simulations starting at x are needed. This is also not compatible with straight forward MD. In a long-term trajectory, only consecutive steps are sampled. Thus, only one point x_τ is generated out of $x_0 = x$. Using only this one point, the calculation of the expectation value would be based on very bad statistics.

Although it is recommended to exploit these two basic ideas, the described ISOKANN method can be applied to long-term trajectories. Point 1 can be solved by running extensive MD simulations, by restricting rate computations only to very localized events of molecular processes, or by running MD trajectories from many different starting points in space (including transition regions). For point 2, there exists a mathematical solution. The advantage of analyzing invariant subspaces of an operator $\mathcal{K}^\tau$ (instead of singular value decompositions) is that multiples and sums of multiples of $\mathcal{K}^\tau$, have the same invariant subspace. The operator $\overline{\mathcal{K}} = \frac{1}{w}\sum_{i=1}^{w} \mathcal{K}^{w\tau}$ has the same invariant subspaces like $\mathcal{K}^\tau$ whenever the Koopman operator $\mathcal{K}^\tau$ meets the Chapman–Kolmogorov property (which is the case if $\mathcal{K}^\tau$ has an infinitesimal generator, like in OLD or stochastic MD). Thus, instead of generating many trajectories from a starting point x, one can also generate one long-term trajectory and take w consecutive points for the mean value computation. It should be noted that the eigenvalues of $\overline{\mathcal{K}}$ are of the form $\frac{1}{w}\sum_{i=1}^{w} \lambda^w$ if λ is an eigenvalue of $\mathcal{K}^\tau$. This relation can be used to correlate the characteristic time scales of $\overline{\mathcal{K}}$ with those of $\mathcal{K}^\tau$.

C. Results of ISOKANN

In Fig. 8, it is shown how a trained function χ looks like for the high-dimensional example. After the first iteration of the power method, the correlation coefficient is 0.6847. A clear separation of molecular macrostates is not visible in the corresponding plot (top). After seven iterations (bottom), the convergence criterion is reached. The correlation coefficient is 0.9991. The piecewise constant-level pattern of membership functions is clearly visible. Rare transitions between these levels are also obvious. Transition regions can be identified (values "between the levels"). Furthermore, as expected, the χ-values show more fluctuations in less attracting or less stable regions than in stable regions. This illustrates again that the fluctuations of the χ-values include the relevant information for the exit rate computation. We also expect that the gradient of the

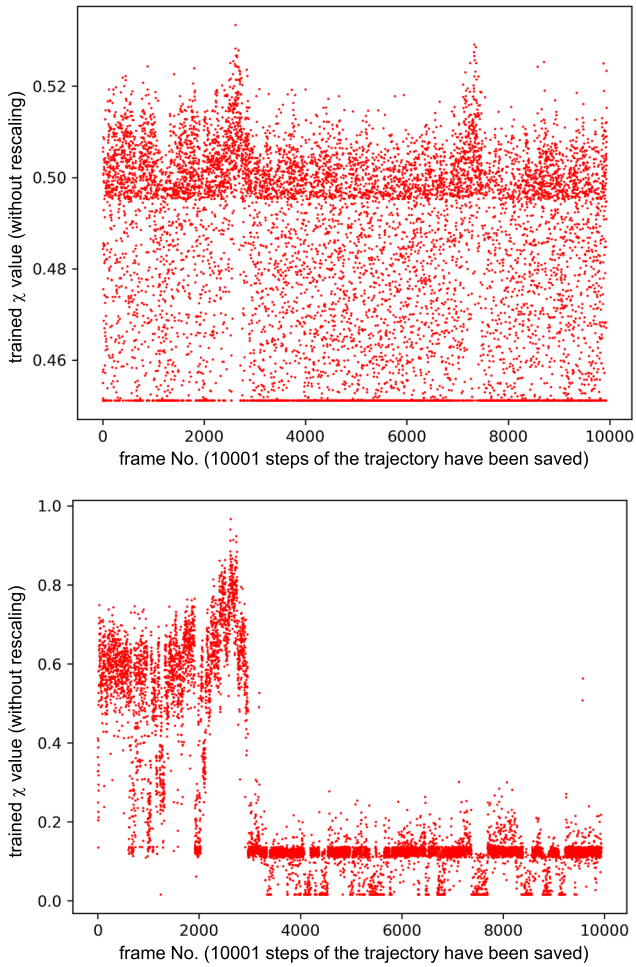

FIG. 8. Values of the ANN after the first iteration step (top, correlation coefficient is 0.6847) and after convergence in the seventh iteration step (bottom, correlation coefficient is 0.9991).

χ-function includes information about the reaction path of the binding event between the opioid molecule and the μ-opioid receptor.

VI. CONCLUSION

The novel ISOKANN method is able to automatically generate basis functions of an invariant subspace of the Koopman operator of an OLD. The power method reliably ensures that only the rare transitions between the system states are captured. The values of the correlation coefficients confirm that the method converges. Graphically, this is manifested by the fact that the plotted points are closer to the regression lines with each further step of the power method. The program also generates the slopes and intercepts of the lines as results. From this, the output rates and holding probabilities can be calculated. The methodology can be implemented in such a way that the problematic discretizations of high-dimensional spaces are always avoided. This approach is novel and offers the great advantage that invariant subspaces can be calculated efficiently even on high-dimensional state spaces. Furthermore, with this efficient method, the structure of the dynamics is better captured with further iteration steps by an optimal distribution of all starting points of the trajectory: Due to the high computational cost involved in MD simulations, any increase in efficiency is of great value. Up until now, MD simulations have not focused enough on questioning the benefit of each individual calculation. In contrast, this method shows how calculation steps that are not useful can be avoided (starting trajectories in regions with $\chi \approx 0.5$). This article provides a basis for the application of the ISOKANN method to practical problems.

SUPPLEMENTARY MATERIAL

See the supplementary material for the setup of the molecular dynamics simulations of fentanyl inside the opiod receptor binding pocket and also for a complete python script showing the implementation details of the ISOKANN algorithm.

DATA AVAILABILITY

The data that support the findings of this study are available from the corresponding author upon reasonable request. It can also be generated with the aid of the python-script present in our supplementary material.

REFERENCES

[1] A. K. Faradjian and R. Elber, "Computing time scales from reaction coordinates by ," J. Chem. Phys. **120**, 10880–10889 (2004).
[2] W. E and E. Vanden-Eijnden, "Towards a theory of transition paths," J. Stat. Phys. **123**, 503–523 (2006).
[3] W. Huisinga, "Metastability of Markovian systems," Ph.D. thesis, FU Berlin, Germany, 2001.
[4] M. Weber, "A subspace approach to molecular Markov state models via a new infinitesimal generator," Habilitation thesis, FU Berlin, 2011.
[5] A. Mardt, L. Pasquali, H. Wu, and F. Noé, "VAMPnets for deep learning of molecular kinetics," Nat. Commun. **9**, 4443 (2018).
[6] B. Reuter, K. Fackeldey, and M. Weber, "Generalized Markov modeling of nonreversible molecular kinetics," J. Chem. Phys. **150**, 174103 (2019).
[7] M. Weber and N. Ernst, "A fuzzy-set theoretical framework for computing exit rates of rare events in potential-driven diffusion processes," arXiv:1708.00679v1 (2017).
[8] T. Schlick, *Molecular Modeling and Simulation: An Interdisciplinary Guide/Tamar Schlick*, Interdisciplinary Applied Mathematics Vol. v. 21. Mathematical biology (Springer, New York; London, 2002).
[9] J. Quer, "Importance sampling for metastable dynamical systems in molecular dynamics," Ph.D. thesis, FU Berlin, 2018.
[10] H. J. C. Berendsen, D. van der Spoel, and R. van Drunen, "GROMACS: A message-passing parallel molecular dynamics implementation," Comput. Phys. Commun. **91**, 43–56 (1995).
[11] K. Fackeldey, A. Sikorski, and M. Weber, "Spectral clustering for non-reversible Markov chains," Comput. Appl. Math. **37**, 6376–6391 (2018).
[12] R. V. Mises and H. Pollaczek-Geiringer, "Praktische verfahren der gleichungsauflösung," Z. Angew. Math. Mech. **9**, 152–164 (1929).
[13] The SciPy Community, Scipy v1.4.1 reference guide: scipy.stats.pearsonr, https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.pearsonr.html, 2019; online accessed 10 November 2019.

[14]S. Park, M. K. Sener, D. Lu, and K. Schulten, "Reaction paths based on mean first-passage times," J. Chem. Phys. **119**, 1313–1319 (2003).
[15]D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," arXiv:1412.6980v9 (2014).
[16]F. Chollet *et al.*, Keras, https://github.com/keras-team/keras, 2019.
[17]Google Brain Team, TensorFlow, https://github.com/tensorflow/tensorflow, 2019.
[18]P. Deuflhard and M. Weber, "Robust Perron cluster analysis in conformation dynamics," Linear Algebra Appl. **398**, 161–184 (2005).
[19]G. M. Rotskoff and E. Vanden-Eijnden, "Trainability and accuracy of neural networks: An interacting particle system approach," arXiv:1805.00915v3 (2018).
[20]A. Koehl, H. Hu, S. Maeda, Y. Zhang, Q. Qu, J. M. Paggi, N. R. Latorraca, D. Hilger, R. Dawson, H. Matile, G. F. X. Schertler, S. Granier, W. I. Weis, R. O. Dror, A. Manglik, G. Skiniotis, and B. K. Kobilka, "Structure of the μ-opioid receptor–Gi protein complex," Nature **558**, 547–552 (2018).
[21]V. Spahn, G. Del Vecchio, D. Labuz, A. Rodriguez-Gaztelumendi, N. Massaly, J. Temp, V. Durmaz, P. Sabri, M. Reidelbach, H. Machelska, M. Weber, and C. Stein, "A nontoxic pain killer designed by modeling of pathological receptor conformations," Science **355**, 966–969 (2017).