How deep is your network? Deep vs. shallow learning of transfer operators

Mohammad Tabish
Maxwell Institute for Mathematical Sciences,
University of Edinburgh and Heriot–Watt University,
Edinburgh, UK
M.Tabish-1@sms.ed.ac.uk

Benedict Leimkuhler
School of Mathematics,
University of Edinburgh,
Edinburgh, UK
B.Leimkuhler@ed.ac.uk

Stefan Klus
School of Mathematical and Computer Sciences,
Heriot–Watt University,
Edinburgh, UK
S.Klus@hw.ac.uk

Abstract

We introduce RaNNDy, a randomized neural network framework for learning transfer operators and their spectral decompositions. By fixing the weights of the hidden layers and training only the output layer, RaNNDy achieves significant reductions in computational costs—often by two orders of magnitude—with similar accuracy compared to state-of-the-art deep learning methods. The method is less sensitive to hyperparameter tuning and provides a closed-form solution for the output layer weights that directly represent eigenfunctions of the learned operator. Moreover, it is possible to estimate uncertainties associated with the computed spectral properties via ensemble learning. We demonstrate the efficacy of the proposed approach using diverse benchmark problems, including stochastic dynamical systems and protein folding processes, highlighting the strengths but also weaknesses of RaNNDy.

1 Introduction

Data-driven approximations of operators describing, for example, the evolution of probability densities, observables, or wave functions are of key importance in applied mathematics and machine learning. Although such operators are typically infinite-dimensional, their finite-dimensional approximations can help us analyze systems that might be challenging to model or understand. Transfer operators such as the Koopman operator and the Perron–Frobenius operator, for instance, can be used to gain crucial insights into the evolution of complex dynamical systems, e.g., protein folding processes or fluid flows. Spectral decompositions of these operators allow us to analyze the global behavior of complex systems, e.g., metastable states in molecular dynamics or coherent sets in fluid dynamics, without requiring detailed mathematical models [1]. For an introduction to transfer operator theory, we refer to [2–5]. Data-driven algorithms for approximating transfer operators serve as a bridge between theoretical models and practical computations.

Related works: The most popular methods for approximating transfer operators are *Ulam's method* [6], *extended dynamic mode decomposition* (EDMD) [7, 5, 8], kernel-based variants such as *kernel EDMD* (kEDMD) [9, 10], as well as *generator EDMD* (gEDMD) [11]. Most of these techniques require a function space spanned by a set of fixed basis functions or a reproducing kernel

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

Hilbert space spanned by a given kernel to approximate the operator. The accuracy of the approximation depends strongly on the choice of this function space. Neural network-based dictionary learning methods have been successfully used for approximating transfer operators in combination with EDMD see, e.g., [12–15], and have been applied to various domains such as modeling and control of autonomous vehicles [15] and protein folding [16]. VAMPnets [16] use the output layer of the network as a set of basis functions. The network is trained by optimizing the VAMP-2 score [17], a loss function based on the variational principle for transfer operators [18]. Typically, training these neural networks requires backpropagation that iteratively updates the parameters (weights and biases) of the network. As the number of neurons and hidden layers increases, the computational complexity of the training algorithm grows significantly, leading to problems such as slow convergence, the existence of numerous local minima, exploding/vanishing gradients, and sensitivity to hyperparameters such as the learning rate [19, 20]. To address these challenges, more sophisticated approaches have been proposed, e.g., adaptive methods like Adam [21, 22] for faster training and escaping saddle points, and gradient clipping for exploding/vanishing gradients [23].

Neural networks with random weights [24, 25], on the other hand, aim to avoid or at least mitigate some of the aforementioned problems. In such networks, the weights of the hidden layers are randomly selected and kept fixed throughout the training and only the output layer of the network is trained with either a closed-form solution or iteratively, see [26–28] for an overview. This makes the architecture simple and easy to train. Single-layer feedforward neural network architectures with random weights, such as random vector functional link (RVFL) networks [29], extreme learning machines (ELM) [30], and broad learning systems (BLS) [31], have been studied and utilized for various classification and regression tasks and have also been shown to have the universal approximation property [32–34]. The above characteristics make them a powerful alternative for approximating operators. Random feature models have been used for partial differential equations (PDEs) [35] and developed further by designing function-valued random features. For further details on neural operator learning for PDEs, their connections to kernel methods, and computational and theoretical advantages of random feature models, see [36] and references therein. Moreover, [37] introduces a Koopman-informed method to construct the optimal hidden layer weights for recurrent neural networks without using gradient descent and presents applications to forecasting chaotic dynamical systems with comparable accuracy and reduced training time.

Contributions: We propose a novel data-driven framework, called RaNNDy, based on randomized neural networks and variational principles, to approximate different operators such as the Koopman operator and the Perron–Frobenius operator. We show how the randomization allows us to compute a closed-form solution for the output layer, representing the eigenfunctions of the operator using more interpretable basis functions. We demonstrate that one of the main advantages is that we can approximate the eigenfunctions of different operators more efficiently, while at the same time avoiding common problems associated with deep learning. Furthermore, we show that RaNNDy enables us to quantify the uncertainties in the data-driven approximation of transfer operators via ensemble learning. We illustrate the advantages of the framework with the aid of numerical examples ranging from the overdamped Langevin equation to high-dimensional protein folding processes.

Structure of the paper: We first discuss the data-driven approximation of self-adjoint and non-self-adjoint operators using variational principles as well as transfer operators for reversible and non-reversible systems in Section 2. In Section 3, we describe the proposed data-driven framework for the approximation of linear operators using randomized neural networks. We present numerical results for different operators and systems in Section 4 and highlight open problems and future research directions in Section 5.

2 Data-driven approximation of operators

2.1 Self-adjoint operators

In what follows, let H be a Hilbert space and $\mathcal{A}\colon H \to H$ a compact self-adjoint linear operator, i.e., $\langle \mathcal{A}f, g\rangle = \langle f, \mathcal{A}g\rangle$ for all $f, g \in \mathcal{D}(\mathcal{A})$, where $\mathcal{D}(\mathcal{A})$ is the domain of $\mathcal{A}$ and $\langle\cdot,\cdot\rangle$ the inner product associated with H.

Definition 2.1 (Rank-one operator)**.** *For $r, s \in H$, the* rank-one operator $s \otimes r\colon H \to H$ *is defined by* $(s \otimes r)f = \langle r, f\rangle s$.

It is well-known that there exists a spectral decomposition of $\mathcal{A}$ given by $\mathcal{A} = \sum_i \lambda_i(\varphi_i \otimes \varphi_i)$ where $\{\varphi_i\}_i$ are orthonormal eigenfunctions corresponding to the eigenvalues $\{\lambda_i\}_i \subset \mathbb{R}$. Assuming that the eigenvalues λ_i are sorted in non-increasing order, i.e., $\lambda_1 \geq \lambda_2 \geq \lambda_3 \geq \ldots$, we can compute eigenfunctions using variational formulations [38, 39].

Theorem 2.2. *Assume that $\hat{\varphi}_i$ is orthogonal to the previous $i-1$ eigenfunctions, then it holds that*

$$\max_{\hat{\varphi}_i} \frac{\langle \hat{\varphi}_i, \mathcal{A}\hat{\varphi}_i \rangle}{\langle \hat{\varphi}_i, \hat{\varphi}_i \rangle} = \lambda_i$$

and the Rayleigh quotient is maximized by the ith eigenfunction.

In Appendix I, we show how a matrix representation of the operator $\mathcal{A}$, restricted to finite-dimensional subspace, can be computed from data using the above variational principle.

Transfer operators for reversible systems: Transfer operators associated with reversible stochastic processes, which play an important role in molecular dynamics, are self-adjoint. Consider a stochastic process $\{\mathbf{X}_t\}$ defined on a state space $\mathbb{X} \subset \mathbb{R}^d$. Let $p_\tau \colon \mathbb{X} \times \mathbb{X} \to \mathbb{R}$ be the transition density function, i.e., $p_\tau(x, y)$ is the probability of $\mathbf{X}_{t+\tau} = y$ given $\mathbf{X}_t = x$, where $\tau > 0$ is a fixed lag time. Stochastic processes can be analyzed using transfer operators such as the Perron–Frobenius operator and the Koopman operator, which describe the evolution of probability densities and observables, respectively. In what follows, let $L^r(\mathbb{X})$ denote the space of r-Lebesgue integrable functions for $1 \leq r \leq \infty$. The *Perron–Frobenius operator* $\mathcal{P}^\tau$ is defined by

$$\mathcal{P}^\tau p(x) = \int_{\mathbb{X}} p_\tau(y, x)\, p(y)\, \mathrm{d}y.$$

The *Koopman operator* $\mathcal{K}^\tau$ is defined by

$$\mathcal{K}^\tau f(x) = \int_{\mathbb{X}} p_\tau(x, y) f(y)\, \mathrm{d}y = \mathbb{E}[f(\mathbf{X}_{t+\tau}) \mid \mathbf{X}_t = x].$$

The Perron–Frobenius operator determines the probability density, and its adjoint, the Koopman operator, the expected value of the observable after time τ. We are interested in eigenvalues λ_i and the corresponding eigenfunctions φ_i of these transfer operators since they help us identify, for example, metastable states in molecular dynamics and coherent sets in fluid flows [3, 40, 5, 41]. We focus in particular on the Koopman operator, i.e., $\mathcal{K}^\tau \varphi_i = \lambda_i(\tau)\, \varphi_i$.

Definition 2.3 (Invariant density)**.** *The density $\pi(x)$ is called stationary (or invariant) if it satisfies $\mathcal{P}\pi = \pi$, i.e., π is an eigenfunction of $\mathcal{P}$ corresponding to the eigenvalue $\lambda = 1$.*

If the invariant distribution π exists, the above operators are well-defined as $\mathcal{P}^\tau \colon L^2_{1/\pi}(\mathbb{X}) \to L^2_{1/\pi}(\mathbb{X})$ and $\mathcal{K}^\tau \colon L^2_\pi(\mathbb{X}) \to L^2_\pi(\mathbb{X})$. For details on the domains of the above operators, see [42].

Definition 2.4 (Detailed balance)**.** *The process $\mathbf{X}_t$ is called* reversible *if the detailed balance condition is satisfied, i.e., $\forall x, y \in \mathbb{X}$, we have $\pi(x)\, p_\tau(x, y) = \pi(y)\, p_\tau(y, x)$.*

It is important to note that detailed balance implies that the Koopman operator $\mathcal{K}^\tau$ and the Perron–Frobenius operator $\mathcal{P}^\tau$ are self-adjoint (with respect to suitably reweighted inner products). Hence, the eigenvalues λ_i of the Koopman operator are real-valued, and the eigenfunctions φ_i form an orthogonal basis with respect to the stationary density $\langle \cdot, \cdot \rangle_\pi$ [43]. Thus, we can expand a function $f \in L^2_\pi(\mathbb{X})$ in terms of eigenfunctions, i.e., $f = \sum_{i=1}^\infty \langle f, \varphi_i \rangle_\pi\, \varphi_i$ such that $\mathcal{K}^\tau f = \sum_{i=1}^\infty \lambda_i(\tau) \langle f, \varphi_i \rangle_\pi\, \varphi_i$. Furthermore, the eigenvalues decay exponentially as $\lambda_i(\tau) = e^{-\kappa_i \tau}$, where κ_i is the relaxation rate [18]. In particular, we have $1 = \lambda_1 > \lambda_2 \geq \lambda_3 \geq \ldots$, indicating that, for a sufficiently large τ, the first few dominant eigenvalues explain the dynamics well.

2.2 Non-self-adjoint operators

For a non-self-adjoint compact linear operator $\mathcal{A} \colon H_1 \to H_2$, where H_1 and H_2 are Hilbert spaces with associated inner products $\langle \cdot, \cdot \rangle_{H_1}$ and $\langle \cdot, \cdot \rangle_{H_2}$, there exists a singular value decomposition (SVD) of the form $\mathcal{A} = \sum_i \sigma_i(\chi_i \otimes \eta_i)$ Here, $\{\sigma_i\}_i \subset \mathbb{R}_{>0}$ are the nonzero singular values, sorted in non-increasing order so that $\sigma_1 \geq \sigma_2 \geq \sigma_3 \geq \ldots$, and the corresponding left singular functions $\{\chi_i\}_i \subset H_2$ and right singular functions $\{\eta_i\}_i \subset H_1$ are orthonormal [44]. If there are infinitely

many nonzero singular values, they form a null sequence. Assuming that f is orthogonal to the first $i-1$ right singular functions η_i and g is orthogonal to the first $i-1$ left singular functions χ_i, it holds that

$$\max_{f,g} \frac{\langle g, \mathcal{A}f\rangle_{H_2}}{\sqrt{\langle f, f\rangle_{H_1}\langle g, g\rangle_{H_2}}} = \sigma_i.$$

This is consistent with the variational principle for the eigenvalues and eigenfunctions and provides a way to formulate an optimization problem to approximate the singular values and singular functions of non-self-adjoint operators. In Appendix II, we derive matrix representations of finite-dimensional approximations of non-self-adjoint operators $\mathcal{A}$.

Transfer operators for non-reversible systems: For non-reversible dynamical systems, associated transfer operators are in general not self-adjoint. That is, their eigenvalues and eigenfunctions may be complex-valued. In this case, we approximate the singular values and singular functions of these operators to analyze the system. In fluid dynamics, these singular functions have been used to identify coherent sets [41]. For more details, we refer to [42, 45, 17]. The Perron–Frobenius operator reweighted with respect to a density μ, denoted by $\mathcal{T}^\tau : L^2_\mu \to L^2_\nu$, is defined as $\mathcal{T}^\tau u(x) = \frac{1}{\nu(x)} \int_{\mathbb{X}} p_\tau(y, x)u(y)\mu(y)\,\mathrm{d}y$, where $\nu = \mathcal{P}^\tau\mu$. It propagates densities w.r.t. the reference density μ and its adjoint the Koopman operator $\mathcal{K}^\tau : L^2_\nu \to L^2_\mu$ propagates the observables, where $\nu = \mathcal{P}\mu$.

3 Operator approximation using randomized neural networks

In randomized neural networks, the weights of the hidden layers are randomly selected and kept fixed throughout training. Only the output layer of the network is trained, either by computing a closed-form solution or using iterative methods, see [26–28] for an overview. This avoids backpropagation and makes the architecture simple and easy to train. The input data are transformed into a random feature space with hidden layers acting as a *random feature map* (RFM). The output layer then represents the output of the network. The model is a function $\mathbf{f} : \mathbb{R}^d \to \mathbb{R}^n$, defined by

$$\mathbf{f}(x) = W_o \cdot \sigma(Wx + b),$$

where σ is the activation function, $W \in \mathbb{R}^{N\times d}$ are the weights of the hidden layer, $b \in \mathbb{R}^N$ is the bias term, and $W_o \in \mathbb{R}^{n\times N}$ is the matrix containing the output weights. The RFM is $R(x) = \sigma(Wx + b)$.

3.1 Learning linear operators

We will now introduce the proposed framework for approximating a compact linear operator $\mathcal{A}$ and its spectral properties using randomized neural networks. Of particular interest is the approximation of the eigenvalues λ_i and the associated eigenfunctions φ_i satisfying $\mathcal{A}\varphi_i = \lambda_i\varphi_i$ and the approximation of singular values σ_i and associated left and right singular functions χ_i and η_i such that $\mathcal{A}\eta_i = \sigma_i\chi_i$ and $\mathcal{A}^*\chi_i = \sigma_i\eta_i$. Considering the strength of neural networks to approximate nonlinear functions from data, we design a randomized neural network called RaNNDy with two variants. In the first variant, the output layer represents the optimal basis functions. The network generates a high-dimensional random feature space using the random feature map (RFM) of the hidden layers. The data is transformed through the RFM and passed to the output layer that represents the basis functions. The training of the parameters in the output layer is done iteratively via loss functions formulated using the variational principle discussed in the previous section. In the second variant, taking advantage of randomization in the network, we formulate an optimization problem to directly approximate the eigenfunctions represented by the output layer of the network. In this case, the RFM acts as a randomized basis, see the figure in Appendix III.

3.2 Data and iterative training of the output layer

To approximate the operator $\mathcal{A}$, we need m training data points $\big(\psi(x_i), \mathcal{A}\psi(x_i)\big)_{i=1}^m$. For the Koopman operator, let $x_i, y_i \in \mathbb{X}$ denote the data points from a dynamical system with $x_i = \mathbf{X}_t$ and $y_i = \mathbf{X}_{t+\tau}$. Data can be obtained by simulating the dynamical system or by repeatedly measuring its state to obtain the matrices $X = [x_1, x_2, \ldots, x_m]$ and $Y = [y_1, y_2, \ldots, y_m] \in \mathbb{R}^{d\times m}$ containing m data points (or snapshots) of a d-dimensional system, where Y contains the states of the system after lag time τ. Hence, our training data is given by $\big(\psi(x_i), \mathcal{K}^\tau\psi(x_i)\big)_{i=1}^m = \big(\psi(x_i), \psi(y_i)\big)_{i=1}^m$.

Basis functions via RaNNDy: For the sake of simplicity, we derive the framework for networks with a single hidden layer. However, multiple hidden layers will follow similarly. The n neurons in the output layer represent the n basis functions $\psi_i \in \mathcal{D}(\mathcal{A})$, with $i = 1, 2, \ldots, n$, to approximate the operator $\mathcal{A}$. For the m data points contained in $X \in \mathbb{R}^{d\times m}$, the randomized features are represented by $R(X) \in \mathbb{R}^{N\times m}$. The output layer of the network will then be the map $\psi\colon \mathbb{R}^N \to \mathbb{R}^n$ given by $\psi(x, W_o) = \sigma(W_o R(x))$. Hence, for fixed network weights, after passing the data matrix X through the network, we get the transformed data matrix $\Psi_0 = \psi(X, W_o) \in \mathbb{R}^{n\times m}$. In order to train the output layer weights W_o of the network, we use the variational principle discussed above. For convenience, let us denote $\psi(\cdot, W_o)$ by ψ. Consider the initial output layer weights, $W_o \in \mathbb{R}^{n\times N}$, which define the basis $\psi = \sigma(W_o R(\cdot))$. Then, using the variational principle in Appendix I for a function $f = \sum_{i=1}^n w_i\psi_i$, we obtain the Rayleigh quotient

$$\max_f \frac{\langle f, \mathcal{A}f\rangle}{\langle f, f\rangle} = \max_w \frac{w^\top C_{01}(W_o) w}{w^\top C_{00}(W_o) w}.$$

Since the basis is represented by the output layer, the matrices $C_{01}(W_o) \in \mathbb{R}^{n\times n}$ and $C_{00}(W_o) \in \mathbb{R}^{n\times n}$ now depend on W_o. The above quotient leads to the eigenvalue problem $A(W_o)\, w = \lambda\, w$, where for a given output layer W_o, the matrix $A(W_o) = C_{00}^{-1}(W_o)\, C_{01}(W_o)$ represents the projected operator $\mathcal{A}_\psi$. The solutions of this eigenvalue problem are orthonormal vectors with respect to the inner product weighted by the matrix $C_{00}(W_o)$ [39]. The eigenvectors $w^{(i)}$ of the matrix $A(W_o)$ can be used to construct the eigenfunctions of the projected operator $\mathcal{A}_\psi$ as shown in Appendix I. Furthermore, using the variational principle, the sum of the eigenvalues, i.e., $\sum_{i=1}^n \lambda_i = \max_{\psi_1,\ldots,\psi_n} \sum_{i=1}^n \langle \psi_i, \mathcal{A}\psi_i\rangle$, s.t. $\langle \psi_i, \psi_j\rangle = \delta_{ij}$, can be optimized to get the best approximation of the eigenfunctions from data. Hence, the sum of eigenvalues of the estimated matrix $A(W_o)$ can be maximized (or minimized, depending on the operator) using, for example, gradient descent at each training epoch, which then gives the updated basis functions with the new weights, W_o, of the output layer. The data-driven approximations of $C_{00}(W_o)$ and $C_{01}(W_o)$ are

$$\widehat{C}_{00}(W_o) = \frac{1}{m}\sum_{i=1}^m \psi(x_i, W_o)\, \psi(x_i, W_o)^\top = \frac{1}{m}\Psi_0(W_o)\, \Psi_0(W_o)^\top,$$

$$\widehat{C}_{01}(W_o) = \frac{1}{m}\sum_{i=1}^m \psi(x_i, W_o)(\mathcal{A}\psi(x_i, W_o))^\top = \frac{1}{m}\Psi_0(W_o)\, \Psi_1(W_o)^\top,$$

where $\Psi_0(W_o) = \psi(X, W_o) \in \mathbb{R}^{n\times m}$ and $\Psi_1(W_o) = [(\mathcal{A}\psi)(x_1), (\mathcal{A}\psi)(x_2), \ldots, (\mathcal{A}\psi)(x_m)] \in \mathbb{R}^{n\times m}$. With $\widehat{A}(W_o) = \widehat{C}_{00}(W_o)^+ \widehat{C}_{01}(W_o)$, we thus get the loss function

$$\max_{W_o\in\mathbb{R}^{n\times N}} \operatorname{tr}\left(\widehat{A}(W_o)\right),$$

which can be used to iteratively train the network, see also [39, 16].

3.3 Can we directly approximate eigenfunctions?

The randomization in our network, i.e., defining the randomly sampled weights to be fixed, can lead to a straightforward optimization problem that allows us to compute a closed-form solution for the output layer. We leverage this fact to directly approximate the eigenfunctions of a self-adjoint operator $\mathcal{A}$, which will be represented by the output layer. In this case, the random features in the hidden layers will act as a randomized basis. Hence, we redefine the basis $\psi\colon \mathbb{R}^d \to \mathbb{R}^N$ by $\psi(x) = R(x) = \sigma(Wx + b)$. That is, we now obtain matrices $C_{00}, C_{01} \in \mathbb{R}^{N\times N}$. The eigenvalues of A correspond to the different orthonormal eigenvectors with respect to the inner product weighted by C_{00} and those eigenvectors lead to different orthonormal eigenfunctions. Let $W_o \in \mathbb{R}^{N\times n}$ be the matrix of the output layer weights for n outputs. In order to obtain the top n eigenvectors of the matrix A, we can optimize the trace of $W_o^\top C_{01} W_o$ with vectors in W_o being orthonormal, i.e., $W_o^\top C_{00} W_o = I$, where I is the identity matrix of size n. This results in the optimization problem

$$\begin{aligned} \max_{W_o\in\mathbb{R}^{N\times n}} \quad & \operatorname{tr}\left(W_o^\top C_{01}\, W_o\right), \\ \text{s.t.} \quad & W_o^\top C_{00} W_o = I. \end{aligned}$$

Defining the Lagrangian $\mathcal{L}(W_o, \Lambda) = \operatorname{tr}\left(W_o^\top C_{01} W_o\right) - \operatorname{tr}\left(\Lambda\left(W_o^\top C_{00} W_o - I\right)\right)$ and differentiating w.r.t. W_o, see [46] for matrix derivatives, yields $\nabla_{W_o}\mathcal{L}(W_o, \Lambda) = C_{10}W_o + C_{01}W_o - 2C_{00}W_o\Lambda$. Since the operator is self-adjoint, $C_{10} = C_{01}$, and using optimality conditions, we have $C_{01}W_o = C_{00}W_o\Lambda$. Let $\Psi_0 = R(X) \in \mathbb{R}^{N\times m}$ and $\Psi_1 = [(\mathcal{A}R)(x_1), (\mathcal{A}R)(x_2), \ldots, (\mathcal{A}R)(x_m)] \in \mathbb{R}^{N\times m}$, then $\widehat{C}_{00} = \Psi_0\Psi_0^\top$ and $\widehat{C}_{01} = \Psi_0\Psi_1^\top$. We finally obtain the eigenvalue problem

$$\widehat{C}_{01}W_o = \widehat{C}_{00}W_o\Lambda \implies \widehat{C}_{00}^{+}\widehat{C}_{01}W_o = W_o\Lambda.$$

Hence we have a closed-form solution for the output layer as W_o, representing the eigenfunctions of the approximated operator.

3.4 Non-self-adjoint operators

We now show how RaNNDy can be used to approximate singular values and functions of a non-self-adjoint operator $\mathcal{A}$. Using the same setting as above for the network with the randomized basis $R(x)$, we use the variational principle for singular values of $\mathcal{A}$. For the data-driven estimation, we start by considering the following ansatz functions for the singular functions $f(x) = W_o^\top R(x)$ and $g(y) = W_o'^\top R(y)$. Following the derivations in Appendix II, we get a constrained optimization problem to approximate the singular functions corresponding to the largest n singular values, i.e.,

$$\begin{aligned} \max_{W_o, W_o' \in \mathbb{R}^{N\times n}} \quad & \operatorname{tr}(W_o^\top C_{01} W_o'), \\ \text{s.t.} \quad & W_o^\top C_{00} W_o = I, \\ & W_o'^\top C_{11} W_o' = I, \end{aligned}$$

where the columns of $W_o' \in \mathbb{R}^{N\times n}$ and $W_o \in \mathbb{R}^{N\times n}$ denote the top n left and right singular vectors, respectively. Now, consider the Lagrangian

$$\mathcal{L}(W_o, W_o', \Lambda, \Lambda') = \operatorname{tr}(W_o^\top C_{01} W_o') - \operatorname{tr}\left(\frac{1}{2}\Lambda(W_o^\top C_{00} W_o - I)\right) - \operatorname{tr}\left(\frac{1}{2}\Lambda'(W_o'^\top C_{11} W_o' - I)\right).$$

Solving the above Lagrangian using the optimality conditions, we get

$$C_{00}^{+}C_{01}C_{11}^{+}C_{10}W_o = W_o\Lambda^2 \implies \widehat{C}_{00}^{+}\widehat{C}_{01}\widehat{C}_{11}^{+}\widehat{C}_{10}W_o = W_o\Lambda^2,$$

where $\widehat{C}_{01}, \widehat{C}_{00}, \widehat{C}_{11}$ are the data-driven estimates of the matrices defined in the previous section and $\widehat{C}_{10} = \widehat{C}_{01}^\top$. Hence, the output layer W_o represents the right singular functions of the operator $\mathcal{A}$, i.e., the eigenfunctions of the operator $\mathcal{A}^*\mathcal{A}$. The above procedure is summarized in Algorithm 1. Note that, in order to avoid ill-conditioned matrices, we can, for example, use Tikhonov regularization [47].

Algorithm 1 Eigenfunctions represented by the output layer.

Initialization:

- Given training data $\left(\psi(x_i), \mathcal{A}\psi(x_i)\right)_{i=1}^m$ for the operator $\mathcal{A}$ to be approximated.
- Randomly generate hidden layer weights $W_1, W_2, \ldots$, and bias weights, $b_1, b_2, \ldots$ from a given distribution to construct a random feature map $R(x)$.
- Select an activation function σ for the nonlinear transformation.

► Compute $\Psi_0 = [R(x_1), \ldots, R(x_m)]$ and $\Psi_1 = [(\mathcal{A}R)(x_1), \ldots, (\mathcal{A}R)(x_m)]$.

► Construct the matrices $\widehat{C}_{00} = \Psi_0\Psi_0^\top$, $\widehat{C}_{11} = \Psi_1\Psi_1^\top$, $\widehat{C}_{01} = \Psi_0\Psi_1^\top$ and $\widehat{C}_{10} = \Psi_1\Psi_0^\top$.

if $\mathcal{A}$ is self-adjoint **then**

► Solve the eigenvalue problem $\widehat{C}_{00}^{+}\widehat{C}_{01}W_o = W_o\Lambda$

else

► Solve the eigenvalue problem $\widehat{C}_{00}^{+}\widehat{C}_{01}\widehat{C}_{11}^{+}\widehat{C}_{10}W_o = W_o\Lambda^2$

end if

► Sort the eigenvectors in W_o according to the eigenvalues and select the top n eigenvectors to get the approximation of the dominant n eigenfunctions.

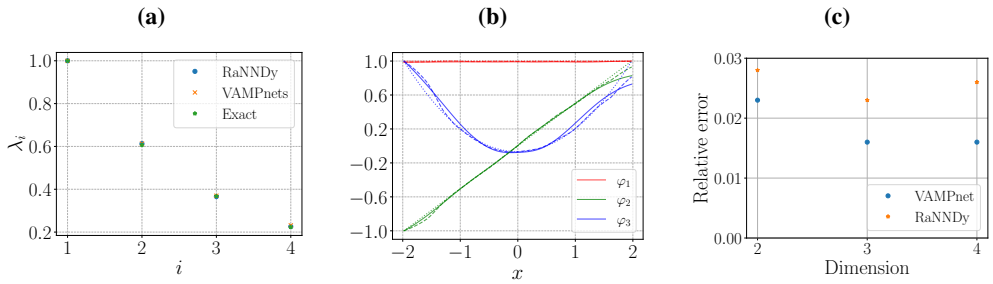


Figure 1: OU process results. (a) Four dominant eigenvalues showing a perfect match between RaNNDy, VAMPnets, and the exact eigenvalues. (b) Three dominant eigenfuntions. The solid lines represent RaNNDy approximations, dashed lines are VAMPnets, and dotted lines are the exact eigenfunctions. (c) Comparison of approximation errors for multi-dimensional OU processes.

4 Numerical experiments

We now apply the proposed techniques to different operators and various benchmark problems and also compare RaNNDy and VAMPnets in terms of the computational costs.

4.1 Overdamped Langevin SDE

For the one-dimensional Ornstein–Uhlenbeck (OU) process, we can compute eigenvalues and eigenfunctions of the Koopman operator analytically. The details on the OU process, governing SDE, and the experimental setup are provided in Appendix IV.1. Using the same setup, the numerical approximations computed by RaNNDy and VAMPnets for the eigenvalues and eigenfunctions are shown in Figures 1a and 1b, respectively, matching the analytical solutions. Since we create the data using a single long trajectory containing very few data points near the boundaries of the domain, the approximations are less accurate in these regions. Furthermore, for multi-dimensional OU processes, see [48, 49], we compute the analytical eigenvalues and compare them with their numerically computed counterparts, see Figure 1c. We can use a grid-search strategy to select optimal distributions for the initialization of the weights and biases of the randomized neural network using the variational score as a metric.

Further analysis: In Appendix IV.1, we provide a more detailed comparison between RaNNDy and VAMPnets regarding the accuracy and run times with different settings. These results support our claim that with the right initialization of the network, we can skip deep learning and avoid problems such as slow convergence, sensitivity to hyperparameters, loss of interpretability of basis functions, etc. Moreover, we show the approximation of singular value decomposition using RaNNDy for the case when the associated transfer operators are not self-adjoint using the Bickley jet system in Appendix IV.3.

4.2 Protein folding problems

We now show how RaNNDy can be applied to high-dimensional protein folding processes. We are in particular interested in identifying the folded and unfolded structures of molecules, see [50] for a detailed introduction to molecular dynamics. The goal is to gain a better understanding of biological processes, which is of significant importance in the development of new drugs or treatments [51]. We apply the Koopman analysis using the proposed framework to different data sets acquired from D.E. Shaw Research, see [52]. We apply RaNNDy, described in Algorithm 1, and VAMPnets with 100 epochs of the Adam optimizer [22] to train the network. In order to ensure the comparison is fair, we select the same network architecture consisting of three dense hidden layers comprising 256, 512, and 256 neurons and approximate the first few dominant eigenvalues and eigenfunctions of the Koopman operator.

Chignolin (CLN025): Chignolin is a mini-protein molecule consisting of 10 residues and 166 atoms. The trajectory we consider is $1.06 \cdot 10^8$ ps long. We subsample the trajectory to create the

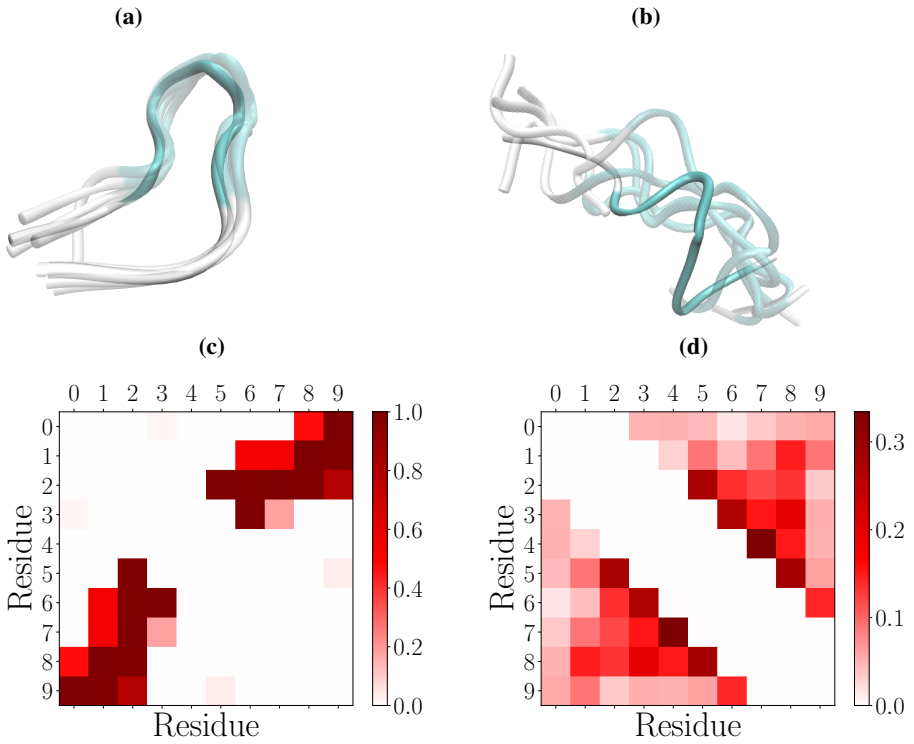


Figure 2: RaNNDy results for Chignolin. (a) & (b) Folded and unfolded states. (c) & (d) Frequency of contacts between different residue pairs over all the identified folded and unfolded states, respectively.

training data $\{x_i, y_i\}_{i=1}^m$ using the lag time $\tau = 9911.5$ps, i.e., the gap between x_i and y_i is 50 frames. The subsampled trajectory data is then transformed using contact maps, which measure the distances between different pairs of residues, resulting in $X, Y \in \mathbb{R}^{28\times 10\,694}$. A few folded and unfolded states, identified by RaNNDy, are shown in Figures 2a and 2b, respectively. Figures 2c and 2d show the frequency of contacts between different residues of the molecule for the sets of identified folded and unfolded states.

Protein NuG2: Protein G (NuG2) consists of 56 residues. The total trajectory is $3.680002 \cdot 10^8$ ps long. We consider a subset of the simulation trajectories and subsample the selected 929251 frames to create the training data $\{x_i, y_i\}_{i=1}^m$. The lag time is a gap of 50 frames between each x_i and y_i. We again compute the contact maps to get the data matrices $X, Y \in \mathbb{R}^{1431\times 18\,585}$. That is, the dynamics are really high-dimensional in this case. Figure 3 presents the folded and unfolded states and the contact frequencies of the NuG2 protein molecule identified using RaNNDy. Additionally, we provide the videos of the molecule's folded and unfolded states identified using RaNNDy in the supplementary material.

Comparison: Figure 4 shows the second eigenfunctions associated with the two protein molecules above identified using RaNNDy and VAMPnets (with the same network architecture). We can see that the behavior of the eigenfunctions computed by both methods is very similar. Furthermore, in Table 1, we compare RaNNDy and VAMPnets in terms of the computational costs and accuracy in identifying the two states of the molecules. The contact maps of the two states of the above molecules identified using RaNNDy again highlight its accuracy.

5 Conclusion

We proposed a novel data-driven framework, called RaNNDy, for the approximation of linear operators and their eigenvalues and eigenfunctions or singular values and singular functions. Built on randomized neural networks, RaNNDy generates randomized features through the hidden layers of

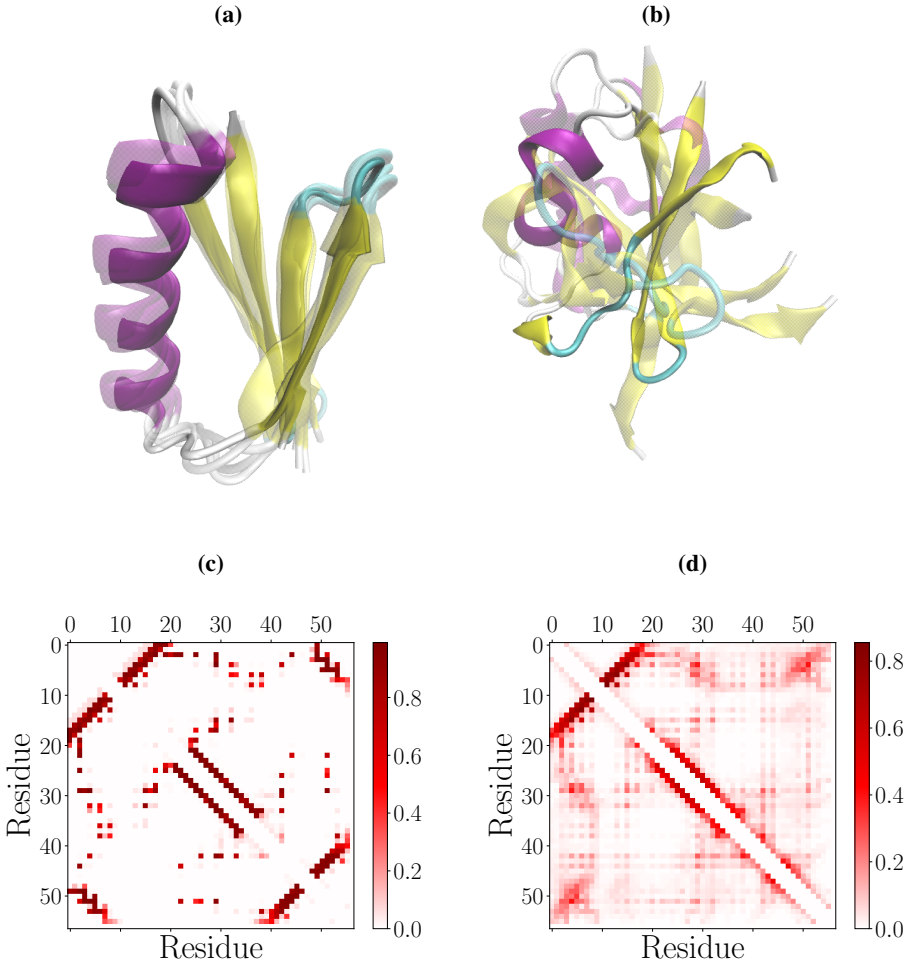


Figure 3: RaNNDy results for NuG2. (a) & (b) Some folded and unfolded states of the NuG2 protein. (c) & (d) Frequency of contacts between different residue pairs over all the identified folded and unfolded states, respectively.

Table 1: RaNNDy and VAMPnets run times for protein-folding problems.

	Computational time (in sec.)		Identified folded frames		Identified unfolded frames	
Protein	RaNNDy	VAMPnets	RaNNDy	VAMPnets	RaNNDy	VAMPnets
Chignolin	0.52	60.44	8282	8296	2412	2398
NuG2	1.34	139.98	11706	11728	6879	6857

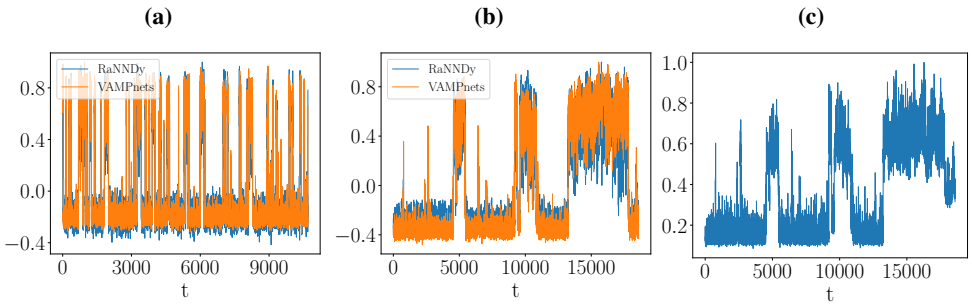


Figure 4: Protein folding results. Second dominant eigenfunction evaluated in the trajectory data using RaNNDy and VAMPnets for (a) the Chignolin mini-protein and (b) the NuG2 protein. (c) Uncertainty associated with the second eigenfunction of the NuG2 protein, indicating a higher uncertainty for the unfolded states. Unfolded states are more chaotic (as can be seen from the eigenfunction and also Figure 3b). Hence, we see a high uncertainty in that region.

the network, which are kept fixed. Only the output layer is trained to approximate the operator. We introduced loss functions to train RaNNDy using variational principles. Furthermore, we derived a closed-form solution for the output layer of RaNNDy that represents the eigenfunctions of the operator. This novel approach has the advantage that it significantly reduces the training time and computational costs, making the network simple and easy to implement, while mitigating the risk of common problems arising in the training of deep neural networks. The proposed framework furthermore enables us to use ensemble learning techniques to quantify uncertainties in the operator approximation. With a proper initialization of weights, we show that the numerical results of the proposed approach are efficient and comparable to state-of-the-art deep learning methods across different systems. The results imply that the randomly generated features are suitable basis functions and that, by using a grid-search type method for selecting proper distributions for initial weights, similar accuracy can be achieved. This also makes the algorithm more interpretable, as we know the structure of basis functions, i.e., the exact distribution of weights and biases that parametrize the basis.

Limitations and future work: RaNNDy can be viewed as a compromise between data-driven models that utilize fixed basis functions such as EDMD and fully optimizable neural networks such as VAMPnets. Although VAMPnets might in general be more flexible and have better approximation properties, RaNNDy is computationally efficient and provides comparable accuracy. There are, however, some open problems. The first and most important is to select the distribution of weights and biases for the randomized part of the network. Choosing distributions that make the matrix representations of the operators ill-conditioned will negatively affect the approximation. This analysis could be extremely important in making RaNNDy more robust and generalizable [53–55], similar to recent work on efficient and data-informed sampling of hidden layers' weights for deep and shallow networks [56]. We present detailed ablation studies in Appendix V, comparing different initializations of RaNNDy and illustrating that a grid search for finding the optimal distributions of weights still takes significantly less time than training the whole network. For a specific choice of activation function and distribution of weights, random Fourier feature models [57, 58] can be considered as a special case of RaNNDy. Hence, an interesting future research direction could be to explore the relationships of different activation functions and weight distributions with kernel-based techniques and random Fourier features. The code can be found at https://github.com/tabishhub/ranndy.

Acknowledgments

MT was supported by the EPSRC Centre for Doctoral Training in Mathematical Modeling, Analysis and Computation (MAC-MIGS) funded by the UK Engineering and Physical Sciences Research Council (grant EP/S023291/1), Heriot–Watt University and the University of Edinburgh. We would like to thank D.E. Shaw Research for providing the protein molecule data.

References

[1] S. Klus and N.D. Conrad. Dynamical systems and complex networks: A Koopman operator perspective. *Journal of Physics: Complexity*, 5(4):041001, 2024. doi: 10.1088/2632-072X/ad9e60.

[2] M. Dellnitz and O. Junge. On the approximation of complicated dynamical behavior. *SIAM Journal on Numerical Analysis*, 36(2):491–515, 1999. doi: 10.1137/S0036142996313002.

[3] I. Mezić. Spectral properties of dynamical systems, model reduction and decompositions. *Nonlinear Dynamics*, 41:309–325, 2005. doi: 10.1007/s11071-005-2824-x.

[4] A. Lasota and M.C. Mackey. *Chaos, fractals, and noise: Stochastic aspects of dynamics*, volume 97. Springer New York, NY, 2013. doi: 10.1007/978-1-4612-4286-4.

[5] S. Klus, P. Koltai, and C. Schütte. On the numerical approximation of the Perron–Frobenius and Koopman operator. *Journal of Computational Dynamics*, 3(1):51–79, 2016. doi: 10.3934/jcd.2016003.

[6] S. M. Ulam. *A Collection of Mathematical Problems*. Interscience Publisher NY, 1960.

[7] M.O. Williams, I.G. Kevrekidis, and C.W. Rowley. A data-driven approximation of the Koopman operator: Extending dynamic mode decomposition. *Journal of Nonlinear Science*, 25:1307–1346, 2015. doi: 10.1007/s00332-015-9258-5.

[8] M. Tabish, N.K. Chada, and S. Klus. Learning dynamical systems from data: Gradient-based dictionary optimization. *Physica D: Nonlinear Phenomena*, 481:134822, 2025. doi: 10.1016/j.physd.2025.134822.

[9] M. O. Williams, C. W. Rowley, and I. G. Kevrekidis. A kernel-based method for data-driven Koopman spectral analysis. *Journal of Computational Dynamics*, 2(2):247–265, 2015. doi: 10.3934/jcd.2015005.

[10] S. Klus, I. Schuster, and K. Muandet. Eigendecompositions of transfer operators in reproducing kernel Hilbert spaces. *Journal of Nonlinear Science*, 2020. doi: 10.1007/s00332-019-09574-z.

[11] S. Klus, F. Nüske, S. Peitz, J.-H. Niemann, C. Clementi, and C. Schütte. Data-driven approximation of the Koopman generator: Model reduction, system identification, and control. *Physica D: Nonlinear Phenomena*, 406:132416, 2020. doi: 10.1016/j.physd.2020.132416.

[12] Q. Li, F. Dietrich, E. M. Bollt, and I. G. Kevrekidis. Extended dynamic mode decomposition with dictionary learning: A data-driven adaptive spectral decomposition of the Koopman operator. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 27(10), 2017. doi: 10.1063/1.4993854.

[13] E. Yeung, S. Kundu, and N. Hodas. Learning deep neural network representations for Koopman operators of nonlinear dynamical systems. In *2019 American Control Conference (ACC)*, pages 4832–4839, 2019. doi: 10.23919/ACC.2019.8815339.

[14] M. Gulina and A. Mauroy. Two methods to approximate the Koopman operator with a reservoir computer. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 31(2), 2021. doi: 10.1063/5.0026380.

[15] Y. Xiao, X. Zhang, X. Xu, X. Liu, and J. Liu. Deep neural networks with Koopman operators for modeling and control of autonomous vehicles. *IEEE Transactions on Intelligent Vehicles*, 8 (1):135–146, 2022.

[16] A. Mardt, L. Pasquali, H. Wu, and F. Noé. VAMPnets for deep learning of molecular kinetics. *Nature communications*, 9(1):5, 2018. doi: 10.1038/s41467-017-02388-1.

[17] H. Wu and F. Noé. Variational approach for learning Markov processes from time series data. *Journal of Nonlinear Science*, 30(1):23–66, 2020. doi: 10.1007/s00332-019-09567-y.

[18] F. Noé and F. Nüske. A variational approach to modeling slow processes in stochastic dynamical systems. *Multiscale Modeling & Simulation*, 11(2):635–655, 2013. doi: 10.1137/110858616.

[19] M. Gori, A. Tesi, et al. On the problem of local minima in backpropagation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(1):76–86, 1992. doi: 10.1109/34.107014.

[20] H.A.B. Te Braake and G. Van Straten. Random activation weight neural net (RAWN) for fast non-iterative training. *Engineering Applications of Artificial Intelligence*, 8(1):71–80, 1995. doi: 10.1016/0952-1976(94)00056-S.

[21] M. Staib, S. Reddi, S. Kale, S. Kumar, and S. Sra. Escaping saddle points with adaptive gradient methods. In *International Conference on Machine Learning*, pages 5956–5965. PMLR, 2019.

[22] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*, 2015. URL https://arxiv.org/abs/1412.6980.

[23] R. Pascanu, T. Mikolov, and Y. Bengio. On the difficulty of training recurrent neural networks. In *International Conference on Machine Learning*, pages 1310–1318. PMLR, 2013.

[24] L. Zhang and P.N. Suganthan. A survey of randomized algorithms for training neural networks. *Information Sciences*, 364:146–155, 2016. doi: 10.1016/j.ins.2016.01.039.

[25] P.N. Suganthan and R. Katuwal. On the origins of randomization-based feedforward neural networks. *Applied Soft Computing*, 105:107239, 2021. doi: 10.1016/j.asoc.2021.107239.

[26] A.K. Malik, R. Gao, M.A. Ganaie, M. Tanveer, and P.N. Suganthan. Random vector functional link network: recent developments, applications, and future directions. *Applied Soft Computing*, 143:110377, 2023. doi: 10.1016/j.asoc.2023.110377.

[27] C. Weipeng, W Xizhao, M. Zhong, and G. Jinzhu. A review on neural networks with random weights. *Neurocomputing*, 275:278–287, 2018. ISSN 0925-2312. doi: https://doi.org/10.1016/j.neucom.2017.08.040.

[28] L. Zhang and P.N. Suganthan. A survey of randomized algorithms for training neural networks. *Information Sciences*, 364-365:146–155, 2016. ISSN 0020-0255. doi: https://doi.org/10.1016/j.ins.2016.01.039.

[29] Y.-H. Pao and Y. Takefuji. Functional-link net computing: theory, system architecture, and functionalities. *Computer*, 25(5):76–79, 1992.

[30] G.-B. Huang, Q.-Y. Zhu, and C.-K. Siew. Extreme learning machine: Theory and applications. *Neurocomputing*, 70(1):489–501, 2006. ISSN 0925-2312. doi: https://doi.org/10.1016/j.neucom.2005.12.126. Neural Networks.

[31] C.L.P. Chen and Z. Liu. Broad learning system: A new learning paradigm and system without going deep. In *2017 32nd youth academic annual conference of Chinese association of automation (YAC)*, pages 1271–1276. IEEE, 2017. doi: 10.1109/YAC.2017.7967609.

[32] J. Park and I.W. Sandberg. Universal approximation using radial-basis-function networks. *Neural Computation*, 3(2):246–257, 1991. doi: 10.1162/neco.1991.3.2.246.

[33] F. Scarselli and A.C. Tsoi. Universal approximation using feedforward neural networks: A survey of some existing methods, and some new results. *Neural Networks*, 11(1):15–37, 1998. doi: 10.1016/S0893-6080(97)00097-X.

[34] C.L.P. Chen, Z. Liu, and S. Feng. Universal approximation capability of broad learning system and its structural variations. *IEEE transactions on neural networks and learning systems*, 30(4):1191–1204, 2018. doi: 10.1109/TNNLS.2018.2866622.

[35] N. H. Nelsen and A. M. Stuart. The random feature model for input-output maps between banach spaces. *SIAM Journal on Scientific Computing*, 43(5):A3212–A3243, 2021. doi: 10.1137/20M133957X.

[36] N. H. Nelsen and A. M. Stuart. Operator learning using random features: A tool for scientific computing. *SIAM Review*, 66(3):535–571, 2024. doi: 10.1137/24M1648703.

[37] E. L. Bolager, A. Čukarska, I. Burak, Z. Monfared, and F. Dietrich. Koopman-informed recurrent neural networks. *arXiv preprint arXiv:2410.23467*, 2024.

[38] D. Eschwé and M. Langer. Variational principles for eigenvalues of self-adjoint operator functions. *Integral Equations and Operator Theory*, 49(3):287–321, 2004. doi: 10.1007/s00020-002-1209-5.

[39] F. Nüske, B.G. Keller, G. Pérez-Hernández, A.S.J.S. Mey, and F. Noé. Variational approach to molecular kinetics. *Journal of chemical theory and computation*, 10(4):1739–1752, 2014. doi: 10.1021/ct4009156.

[40] C. Schütte and M. Sarich. *Metastability and Markov state models in molecular dynamics*, volume 24. American Mathematical Soc., 2013. URL https://bookstore.ams.org/cln-24.

[41] R. Banisch and P. Koltai. Understanding the geometry of transport: Diffusion maps for Lagrangian trajectory data unravel coherent sets. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 27(3), 2017. doi: 10.1063/1.4971788.

[42] P. Koltai, H. Wu, F. Noé, and C. Schütte. Optimal data-driven estimation of generalized Markov state models for non-equilibrium dynamics. *Computation*, 6(1):22, 2018. doi: 10.3390/computation6010022.

[43] S. Klus, F. Nüske, P. Koltai, H. Wu, I. Kevrekidis, C. Schütte, and F. Noé. Data-driven model reduction and transfer operator approximation. *Journal of Nonlinear Science*, 28(3):985–1010, 2018. doi: 10.1007/s00332-017-9437-7.

[44] M. Mollenhauer, I. Schuster, S. Klus, and C. Schütte. Singular value decomposition of operators on reproducing kernel Hilbert spaces. In *Advances in Dynamics, Optimization and Computation*, pages 109–131, Cham, 2020. Springer. doi: 10.1007/978-3-030-51264-4_5.

[45] S. Klus, B.E. Husic, M. Mollenhauer, and F. Noé. Kernel methods for detecting coherent structures in dynamical data. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 29(12), 2019. doi: 10.1063/1.5100267.

[46] K. B. Petersen et al. The matrix cookbook. *Technical University of Denmark*, 7(15):510, 2012. URL http://www2.imm.dtu.dk/pubdb/p.php?3274.

[47] G. H. Golub, P. C. Hansen, and D. P. O'Leary. Tikhonov regularization and total least squares. *SIAM journal on matrix analysis and applications*, 21(1):185–194, 1999.

[48] B. J. Zhang, T. Sahai, and Y. M. Marzouk. Computing eigenfunctions of the multidimensional Ornstein–Uhlenbeck operator. *arXiv preprint arXiv:2110.09229*, 2021.

[49] M. Tabish. *Scientific machine learning for the analysis of complex dynamical systems*. PhD thesis, Heriot–Watt University and University of Edinburgh, 2026.

[50] C. Schütte, S. Klus, and C. Hartmann. Overcoming the timescale barrier in molecular dynamics: Transfer operators, variational principles and machine learning. *Acta Numerica*, 32:517–673, 2023. doi: 10.1017/S0962492923000016.

[51] K. A. Dill, S. B. Ozkan, M. S. Shell, and T. R. Weikl. The protein folding problem. *Annu. Rev. Biophys.*, 37:289–316, 2008. doi: 10.1146/annurev.biophys.37.092707.153558.

[52] K. Lindorff-Larsen, S. Piana, R. O. Dror, and D. E. Shaw. How fast-folding proteins fold. *Science*, 334(6055):517–520, 2011. doi: 10.1126/science.1208351.

[53] D. Arpit, V. Campos, and Y. Bengio. How to initialize your network? robust initialization for WeightNorm & ResNets. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL https://proceedings.neurips.cc/paper_files/paper/2019/file/e520f70ac3930490458892665cda6620-Paper.pdf.

[54] D. Mishkin and J. Matas. All you need is a good init. *CoRR*, abs/1511.06422, 2015. URL https://api.semanticscholar.org/CorpusID:2780493.

[55] L. Datta. A survey on activation functions and their relation with Xavier and He normal initialization. *arXiv preprint arXiv:2004.06632*, 2020.

[56] E. L. Bolager, I. Burak, C. Datar, Q. Sun, and F. Dietrich. Sampling weights of deep neural networks. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine, editors, *Advances in Neural Information Processing Systems*, volume 36, pages 63075–63116. Curran Associates, Inc., 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/c7201deff8d507a8fe2e86d34094e154-Paper-Conference.pdf.

[57] A. Rahimi and B. Recht. Random features for large-scale kernel machines. *Advances in Neural Information Processing Systems*, 20, 2007. URL https://proceedings.neurips.cc/paper_files/paper/2007/file/013a006f03dbc5392effeb8f18fda755-Paper.pdf.

[58] F. Nüske and S. Klus. Efficient approximation of molecular kinetics using random Fourier features. *The Journal of Chemical Physics*, 159(7), 2023. doi: 10.1063/5.0162619.

[59] R. Fletcher. *Practical methods of optimization*. John Wiley & Sons, 2000.

[60] H. Hotelling. Relations between two sets of variates. *Biometrika*, 28(3-4):321–377, 12 1936. doi: 10.1093/biomet/28.3-4.321.

[61] J. Bradbury, R. Frostig, P. Hawkins, M. J. Johnson, C. Leary, D. Maclaurin, G. Necula, A. Paszke, J. VanderPlas, S. Wanderman-Milne, and Q. Zhang. JAX: composable transformations of Python+NumPy programs, 2018. URL http://github.com/google/jax.

[62] J. Heek, A. Levskaya, A. Oliver, M. Ritter, B. Rondepierre, A. Steiner, and M. van Zee. Flax: A neural network library and ecosystem for JAX, 2024. URL http://github.com/google/flax.

[63] A. Bittracher, P. Koltai, S. Klus, R. Banisch, M. Dellnitz, and C. Schütte. Transition manifolds of complex metastable systems: Theory and data-driven computation of effective dynamics. *Journal of nonlinear science*, 28(2):471–512, 2018. doi: 10.1007/s00332-017-9415-0.

[64] G. Froyland. An analytic framework for identifying finite-time coherent sets in time-dependent dynamical systems. *Physica D: Nonlinear Phenomena*, 250:1–19, 2013. doi: 10.1016/j.physd.2013.01.013.

[65] I. I. Rypina, M. G. Brown, F. J. Beron-Vera, H. Koçak, M. J. Olascoaga, and I. A. Udovydchenkov. On the Lagrangian dynamics of atmospheric zonal jets and the permeability of the stratospheric polar vortex. *Journal of the Atmospheric Sciences*, 64(10):3595–3610, 2007. doi: 10.1175/JAS4036.1.

[66] M. Hoffmann, M. Scherer, T. Hempel, A. Mardt, B. de Silva, B.E. Husic, S. Klus, H. Wu, N. Kutz, S.L. Brunton, et al. Deeptime: a Python library for machine learning dynamical models from time series data. *Machine Learning: Science and Technology*, 3(1):015009, 2021. doi: 10.1088/2632-2153/ac3de0.

[67] P. I. Frazier. A tutorial on bayesian optimization, 2018. URL https://arxiv.org/abs/1807.02811.

I Data-driven approximation of self-adjoint operators

We will use the variational principle to compute eigenfunctions of the linear operator $\mathcal{A}$. Consider a set of n fixed linearly independent basis functions $\psi_i \in \mathcal{D}(\mathcal{A})$, with $i = 1, 2, \ldots, n$, and let $V = \operatorname{span}\{\psi_1, \ldots, \psi_n\}$ so that V is an n-dimensional subspace of $\mathcal{D}(\mathcal{A})$. We are interested in computing the Galerkin projection of $\mathcal{A}$ onto V. A typical choice of basis functions is to consider monomials, radial basis functions, or indicator functions. However, the optimal choice of basis functions depends on the operator. Finding suitable dictionaries is in general an open problem. An arbitrary function $f \in V$ can be written as a linear combination of the form

$$f(x) = \sum_{i=1}^{n} w_i \psi_i(x) = w^\top \psi(x),$$

where $w = [w_1, \ldots, w_n]^\top$ and $\psi \colon \mathbb{R}^d \to \mathbb{R}^n$ is defined by $\psi(x) = [\psi_1(x), \psi_2(x), \ldots, \psi_n(x)]^\top$. We can use the Rayleigh variational principle to obtain approximations of the eigenvalues λ_i and eigenfunctions φ_i of $\mathcal{A}$. For an arbitrary function f, we have

$$\langle f, \mathcal{A}f \rangle = \left\langle \sum_{i=1}^{n} w_i \psi_i, \sum_{j=1}^{n} w_j \mathcal{A}\psi_j \right\rangle = \sum_{i=1}^{n} \sum_{j=1}^{n} w_i w_j \langle \psi_i, \mathcal{A}\psi_j \rangle = w^\top C_{01} w,$$

where $[C_{01}]_{ij} = \langle \psi_i, \mathcal{A}\psi_j \rangle$. Similarly, $\langle f, f \rangle = w^\top C_{00} w$, with $[C_{00}]_{ij} = \langle \psi_i, \psi_j \rangle$. That is, we obtain

$$\max_f \frac{\langle f, \mathcal{A}f \rangle}{\langle f, f \rangle} = \max_{w \in \mathbb{R}^n} \frac{w^\top C_{01} w}{w^\top C_{00} w}.$$

If, on the other hand, we want to compute the smallest eigenvalues and corresponding eigenfunctions, we have to turn the maximization problem into a minimization problem. Since the above quotient is invariant under the scaling of w by some scalar $\gamma \in \mathbb{R}$, we can write this as a constrained optimization problem of the form

$$\max_{w \in \mathbb{R}^n} \quad w^\top C_{01} w \quad \text{s.t.} \quad w^\top C_{00} w = 1.$$

We can now use the method of Lagrange multipliers [59] and define the Lagrangian

$$\mathcal{L}(w, \lambda) = w^\top C_{01}\, w - \lambda w^\top C_{00}\, w.$$

As $\mathcal{A}$ is a self-adjoint operator, C_{01} is symmetric so that

$$\nabla_w \mathcal{L}(w, \lambda) = (C_{01} + C_{01}^\top)\, w - 2\lambda C_{00}\, w = 2 C_{01}\, w - 2\lambda C_{00}\, w.$$

We thus obtain a generalized eigenvalue problem of the form

$$C_{01}\, w = \lambda C_{00}\, w.$$

The cost function is maximized by the eigenvector corresponding to the largest eigenvalue of this generalized eigenvalue problem. In fact, the matrix $A = C_{00}^{-1} C_{01}$ is a representation of the operator $\mathcal{A}$ projected onto the subspace V, denoted by $\mathcal{A}_\psi$, which is defined by

$$\mathcal{A}_\psi f(x) := (Ac)^\top \psi(x),$$

see, e.g., [5]. The eigenfunctions of $\mathcal{A}_\psi$ can then be computed using the eigenvectors of A. Let $w^{(i)}$ be the ith eigenvector of A corresponding to the eigenvalue λ_i, i.e., $A\, w^{(i)} = \lambda_i\, w^{(i)}$, then defining $\varphi_i(x) = (w^{(i)})^\top \psi(x)$, we have

$$\mathcal{A}_\psi \varphi_i(x) = (A w^{(i)})^\top \psi(x) = \lambda_i (w^{(i)})^\top \psi(x) = \lambda_i\, \varphi_i(x). \tag{1}$$

Hence, φ_i is an eigenfunction of the projected operator. Now suppose we have training data of the form $\big\{(\psi(x_i), \mathcal{A}\psi(x_i))\big\}_{i=1}^m$. A data-driven approximation of the matrices C_{00} and C_{01} is then given by

$$\widehat{C}_{00} = \frac{1}{m} \sum_{i=1}^{m} \psi(x_i) \psi(x_i)^\top = \frac{1}{m} \Psi_0 \Psi_0^\top,$$

$$\widehat{C}_{01} = \frac{1}{m} \sum_{i=1}^{m} \psi(x_i) (\mathcal{A}\psi(x_i))^\top = \frac{1}{m} \Psi_0 \Psi_1^\top,$$

where

$$\begin{aligned}\Psi_0 &= [\psi(x_1) \quad \psi(x_2) \quad \dots \quad \psi(x_m)] \in \mathbb{R}^{n\times m},\\ \Psi_1 &= [\mathcal{A}\psi(x_1) \quad \mathcal{A}\psi(x_2) \quad \dots \quad \mathcal{A}\psi(x_m)] \in \mathbb{R}^{n\times m}.\end{aligned} \tag{2}$$

Hence, the final data-driven approximation of the matrix representation of the projected operator $\mathcal{A}_\psi$ is given by $\widehat{A} = \widehat{C}_{00}^{-1}\widehat{C}_{01}$. In the transfer operator context, this approach is called *extended dynamic mode decomposition* (EDMD) [7, 5], which, for reversible systems, is equivalent to the *variational approach of conformation dynamics* (VAC) [39] as shown in [43]. However, the approach can also be used to compute the ground state and excited states of quantum systems by computing eigenfunctions of the time-independent Schödinger equation.

II Data-driven approximation of non-self-adjoint operators

For non-self-adjoint operators $\mathcal{A}$, we define two sets of n linearly independent basis functions $\{\psi_i\}_{i=1}^n \subset H_1$ and $\{\psi_i'\}_{i=1}^n \subset H_2$ and consider

$$f(x) = \sum_{i=1}^n w_i\,\psi_i(x) = w^\top\psi(x) \quad \text{and} \quad g(x) = \sum_{i=1}^n w_i'\,\psi_i'(x) = w'^\top\psi'(x),$$

where the vectors $w = [w_1, \dots, w_n]^\top$ and $w' = [w_1', \dots, w_n']^\top$ contain the coefficients and ψ and ψ' are vector-valued functions defined by the dictionaries. This results in

$$\max_{f,g}\frac{\langle g, \mathcal{A}f\rangle_{H_2}}{\sqrt{\langle f,f\rangle_{H_1}\langle g,g\rangle_{H_2}}} = \max_{w,w'\in\mathbb{R}^n}\frac{w^\top C_{01}w'}{\sqrt{(w^\top C_{00}w)(w'^\top C_{11}w')}},$$

where $[C_{01}]_{ij} = \langle\psi_j', \mathcal{A}\psi_i\rangle_{H_2}$, $[C_{00}]_{ij} = \langle\psi_i,\psi_j\rangle_{H_1}$, and $[C_{11}]_{ij} = \langle\psi_i',\psi_j'\rangle_{H_2}$. Thus, we obtain the constrained optimization problem

$$\begin{aligned}\max_{w,w'\in\mathbb{R}^n} \quad & w^\top C_{01}\,w',\\ \text{s.t.} \quad & w^\top C_{00}\,w = 1,\\ & w'^\top C_{11}\,w' = 1.\end{aligned}$$

The Lagrangian is given by

$$\mathcal{L}(w,w',\lambda,\lambda') = (w^\top C_{01}w') - \frac{\lambda}{2}(w^\top C_{00}\,w - 1) - \frac{\lambda'}{2}(w'^\top C_{11}\,w' - 1).$$

The gradients of $\mathcal{L}$ w.r.t. w and w' are

$$\begin{aligned}\nabla_w\mathcal{L} &= C_{01}\,w' - \lambda C_{00}\,w,\\ \nabla_{w'}\mathcal{L} &= C_{10}\,w - \lambda' C_{11}\,w',\end{aligned}$$

where $C_{10} = C_{01}^\top$. Using the optimality conditions, we get

$$C_{01}\,w' = \lambda C_{00}\,w, \tag{3}$$

$$C_{10}\,w = \lambda' C_{11}\,w'. \tag{4}$$

Multiplying (3) from the left by w and (4) by w' implies that $\lambda = \lambda'$. Thus, solving (4) for w' and plugging the expression into (3), we have

$$C_{00}^{-1}C_{01}C_{11}^{-1}C_{10}\,w = \lambda^2 w.$$

In practice, we again estimate the matrices from data, i.e., we use the empirical estimates $\widehat{C}_{00}^{-1}$, $\widehat{C}_{01}$, $\widehat{C}_{11}^{-1}$, and $\widehat{C}_{10}$ for the approximation of singular values and singular functions. The derived problem formulation is closely related to *canonical correlation analysis* (CCA) and also VAMPnets, see [16, 45].

Remark II.1. *A class of loss functions for computing singular values and functions of the Koopman operator, known as VAMP-r scores, has been derived in [17]. It was shown in [45] that the loss function for $r = 2$ is closely related to* canonical correlation analysis *(CCA) [60] and its kernel and deep learning counterparts.*

III RaNNDy variants

In Figure 5, the two variants of RaNNDy are shown. In the first variant, the output layer represents the optimal basis functions, whereas the output layer in the second variant directly approximates the eigenfunctions using RFM as the basis.

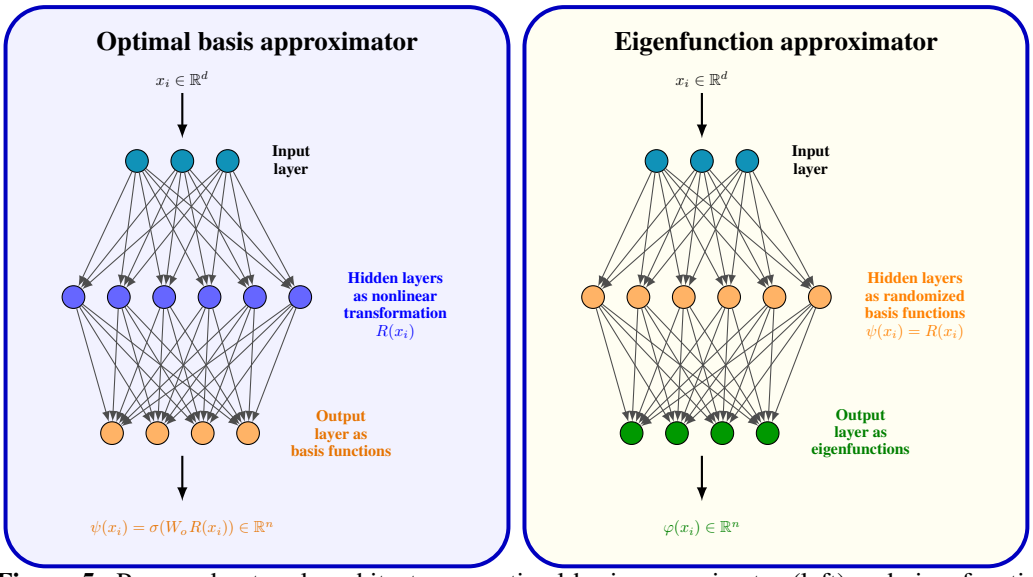


Figure 5: Proposed network architectures: optimal basis approximator (left) and eigenfunction approximator (right).

IV Further numerical results

IV.1 Overdamped Langevin equation

The experiments are run on the CPU of a laptop with 12th Gen Intel Core i7 and 32 GB RAM. We use the Python libraries JAX/Flax [61, 62] for the implementation of the neural networks. Here, we will show the approximation of the eigenvalues and eigenfunctions of Koopman operators associated with simple stochastic differential equations. Consider the overdamped Langevin equation, given by

$$\mathrm{d}\mathbf{X}_t = -\nabla V(\mathbf{X}_t)\,\mathrm{d}t + \sigma(\mathbf{X}_t)\,\mathrm{d}W_t, \tag{5}$$

where $V(x)$ is the potential of the system, $\sigma(X) = \sqrt{2\beta^{-1}}$ the diffusion term, $\beta > 0$ the inverse temperature, and $\{W_t\}_{t\geq 0}$ a Wiener process. To generate the required trajectory data $\{x_i, y_i\}_{i=1}^m$, we use the Euler–Maruyama scheme

$$\mathbf{X}_{k+1} = \mathbf{X}_k - h\nabla V(\mathbf{X}_k) + 2\beta^{-1}\Delta W_k, \tag{6}$$

where h is the step size and $\Delta W_k = W_k - W_{k-1} \sim \mathcal{N}(0, h)$, i.e., normally distributed with mean 0 and variance h. We apply RaNNDy, described in Algorithm 1, and VAMPnets using 100 epochs. In order to ensure the comparison is fair, we select the same network architecture consisting of three dense hidden layers comprising 256, 512, and 256 neurons.

The OU process is a one-dimensional system with potential $V(x) = \frac{\alpha}{2}x^2$, where α is the friction coefficient. We choose $\alpha = 1$ and $\beta = 4$ and generate $m = 20000$ data points with lag time $\tau = 0.5$ by simulating the SDE (5) using (6). The analytically computed eigenvalues and eigenfunctions of the Koopman operator are given by

$$\lambda_i = e^{-\alpha(i-1)\tau}, \quad \varphi_i(x) = \frac{1}{\sqrt{(i-1)!}} H_{i-1}\big(\sqrt{\alpha\beta}x\big), \quad i = 1, 2, 3, \ldots,$$

where H_i denotes the ith probabilists' Hermite polynomial.

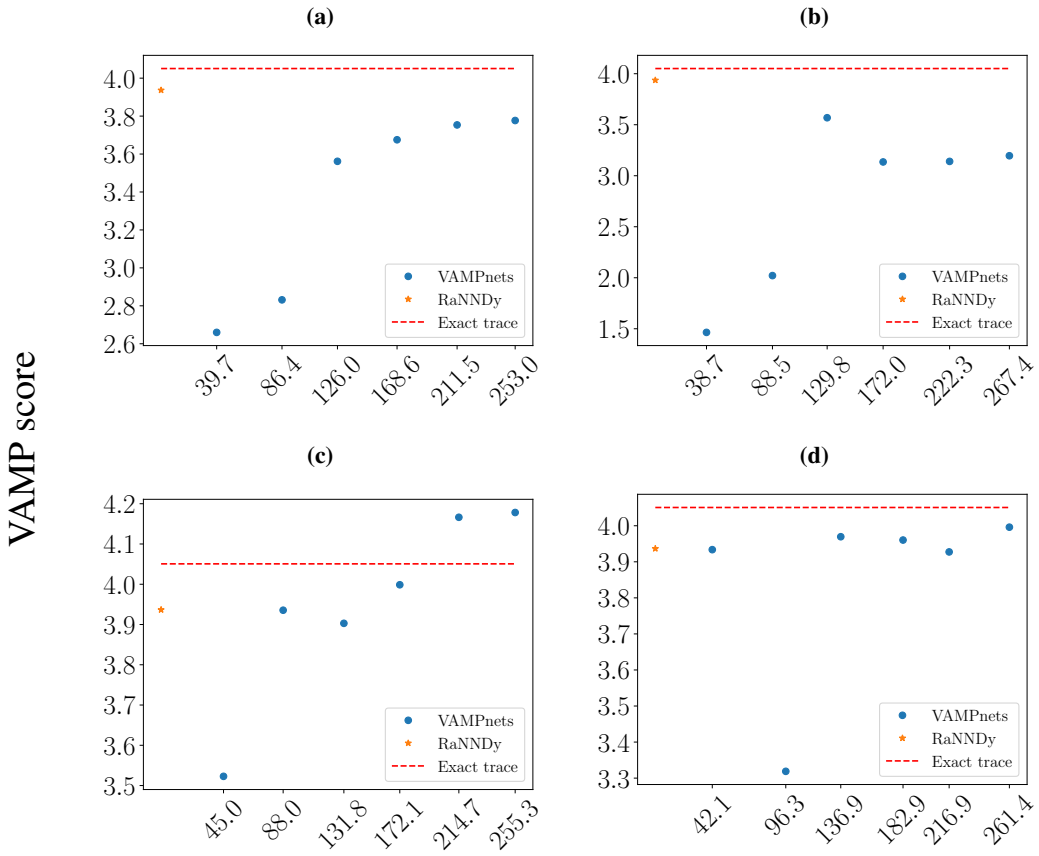


Figure 6: RaNNDy and VAMPnet comparison for the approximation of the sum of five dominant eigenvalues of a 2D OU process. The run times for VAMPnet, on the x-axis, correspond to 20, 40, 60, 80, 100, and 120 epochs, respectively. The weights in VAMPnets are initialized using (a) a normal distribution, (b) a uniform distribution, (c) the LeCun normal distribution (which is also the default initialization in Flax). The overestimation of the true VAMP score in (c) might be due to ill-conditioned matrices or spectral pollution. (d) Sometimes a poor initialization of the weights leads to fluctuations during training.

In Figure 6, we provide further analysis illustrating the robustness of RaNNDy for the approximation of transfer operators. The eigenvalues and eigenfunctions associated with multi-dimensional OU processes can also be computed analytically [48, 49]. We compute the sum of the five largest eigenvalues using RaNNDy and VAMPnets for a 2D OU process. The results show that VAMPnets sometimes converge slowly, whereas RaNNDy typically performs well, provided that the weights are initialized properly. It is important to note that RaNNDy also gives poor approximations for poor initializations of weights.

IV.2 Two-dimensional potentials

Uncertainty quantification: One of the advantages of our framework is that it can be used to quantify the uncertainty in the approximation of the eigenfunctions or singular functions. The performance of a single randomized neural network can be unstable due to the random nature of features in the hidden layers. A standard way to make these models more robust is to utilize an ensemble learning approach [26]. In ensemble learning, multiple models (base learners) are created to solve a specific task. The base learners are then combined to produce an output of the aggregated model. The base learners can be created by selecting multiple random features from the data and averaging the output of each model to return the final output. For the eigenfunction approximation, we create the base learners in Algorithm 1 by sampling multiple weights and biases (W, b) from

Table 2: RaNNDy and VAMPnets run times for overdamped Langevin equations.

	Computational time (in sec.)	
System	RaNNDy	VAMPnets (100 epochs)
OU process	0.32	121.35
Lemon-slice	0.33	117.65
Triple-well	0.65	125.70

a distribution. Each random feature set gives us a random basis via hidden layers to approximate the eigenfunctions. Hence, we can approximate the operator's eigenfunctions using Algorithm 1 by creating multiple models and calculating the average eigenfunction for all models and quantify uncertainty by calculating the variances across the models. We will show in the numerical results section that, as expected, the uncertainty is high in regions containing only a few data points and low in densely sampled regions.

We now consider the two-dimensional lemon-slice potential

$$V(x,y) = \cos\left(q \cdot \operatorname{atan2}(y,x)\right) + 10\left(\sqrt{x^2+y^2} - 1\right)^2,$$

which comprises q wells arranged on a circle [63]. The potential for $q = 5$ is visualized in Figure 7a. We also consider the triple-well potential

$$\begin{aligned} V(x,y) = {} & 3\exp(-x^2 - (y-1/3)^2) - 3\exp(-x^2 - (y-5/3)^2) \\ & - 5\exp(-(x-1)^2 - y^2) - 5\exp(-(x+1)^2 - y^2) \\ & + \frac{2}{10}x^4 + \frac{2}{10}(y-1/3)^4. \end{aligned}$$

We generate $m = 20000$ data points for the above potentials and then approximate the five and three dominant eigenfunctions of the Koopman operator for the lemon-slice and triple-well potentials, respectively. For the lemon-slice potential, the clustering of the dominant eigenfunctions into five clusters is presented in Figure 7b, clearly indicating the five wells of the system. Furthermore, we apply the proposed ensemble learning RaNNDy to quantify the uncertainties in the RaNNDy approximation of the triple-well eigenfunctions by generating 100 models. Figure 7d represents the uncertainty in the approximated eigenvalues, showing that the uncertainty increases for smaller eigenvalues, i.e., faster timescales. In Figures 7e and 7f, we show the average approximation (for all models) of the second and third dominant eigenfunctions colored according to the uncertainty in the approximation in different regions of the domain. The results illustrate that, as expected, regions with more data points have less uncertainty.

Comparison: Using the same network architecture, our approach requires significantly less computational time compared to VAMPnets, see Table 2. This is due to the fact that RaNNDy does not optimize the weights of all the hidden layers. The obtained accuracy for the benchmark problems, however, is comparable, which implies that the randomly generated features are suitable basis functions. It is important to note that in cases where a randomized basis is not optimal, the VAMPnets approach might provide better approximations because of its flexibility in optimizing the basis.

IV.3 Bickley jet

We will now illustrate how the proposed framework can be used to detect coherent structures [64, 41]. As a simple benchmark problem, we consider the well-known Bickley jet, which is an example of an idealized stratospheric flow [65]. The dynamics are governed by a system of non-autonomous ordinary differential equations (ODEs).

$$\begin{bmatrix} \dot{x} \\ \dot{y} \end{bmatrix} = \begin{bmatrix} -\dfrac{\partial \Phi}{\partial y} \\ \dfrac{\partial \Phi}{\partial x} \end{bmatrix},$$

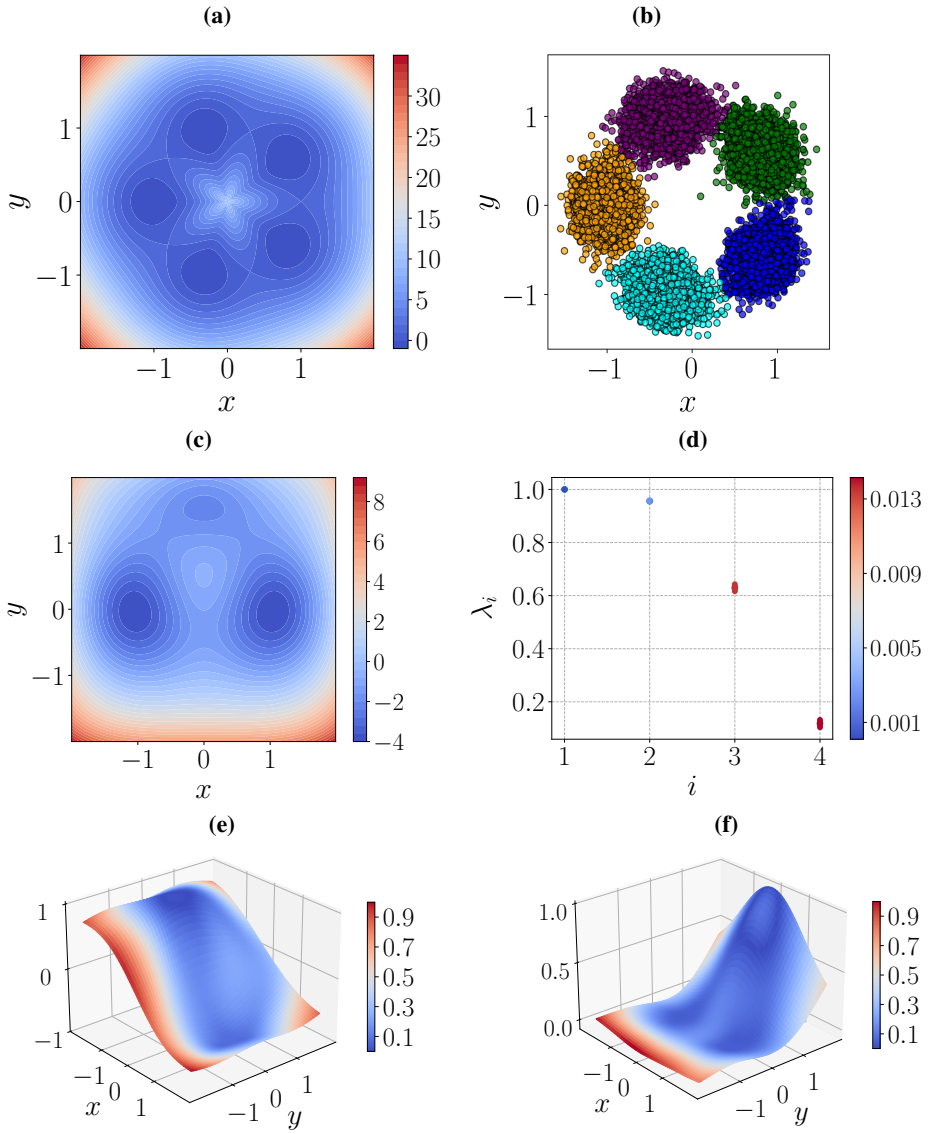


Figure 7: RaNNDy results for 2D potentials. (a) The lemon-slice potential function with $n = 5$ wells. (b) Clustering of the five dominant eigenfunctions into $n = 5$ clusters. (c) The triple-well 2D potential. (d) Uncertainty in the dominant eigenvalues. (e) Second eigenfunction of the Koopman operator colored according to the uncertainties in approximation in different regions. (f) Third dominant eigenfunction of the Koopman operator colored according to the approximation uncertainties.

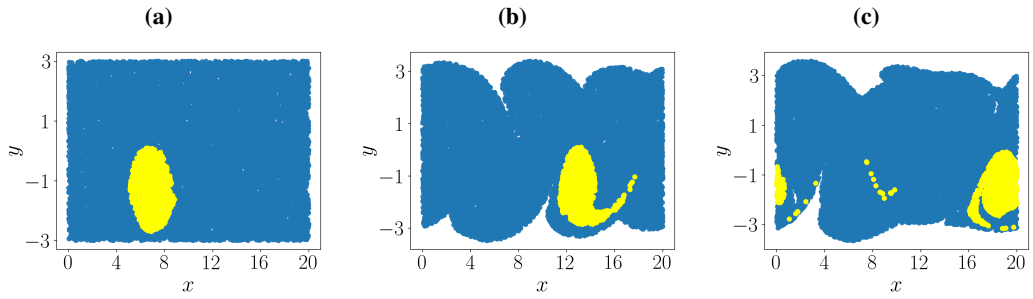


Figure 8: Bickley jet flow at time (a) $t = 0$, (b) $t = 25$, and (c) $t = 50$. The particles in yellow stay in close proximity during the flow, forming a coherent set.

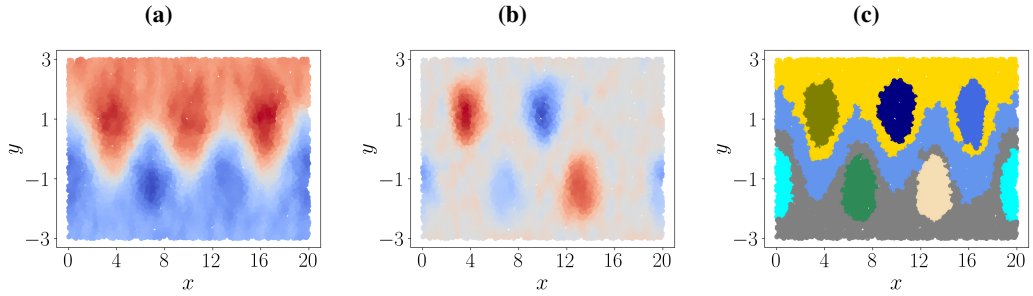


Figure 9: RaNNDy results for the Bickley jet. (a) First and (b) second dominant singular functions. (c) Clustering of the dominant singular functions into nine clusters.

with the stream function

$$\begin{aligned}\Phi(x, y, t) = c_3 y - U_0 L \tanh\left(\frac{y}{L}\right) + A_3 U_0 L \operatorname{sech}^2\left(\frac{y}{L}\right) \cos(k_1 x) \\ + A_2 U_0 L \operatorname{sech}^2\left(\frac{y}{L}\right) \cos(k_2 x - \sigma_2 t) \\ + A_1 U_0 L \operatorname{sech}^2\left(\frac{y}{L}\right) \cos(k_1 x - \sigma_1 t).\end{aligned}$$

The data is generated using the methods and parameters from the Python library *deeptime* [66]. Trajectories of $m = 15000$ uniformly sampled data points within the domain $[0, 20] \times [-4, 4]$ are simulated by integrating the points from $t_0 = 0$ to $t_1 = 40$. A few snapshots of the flow at different times t are visualized in Figure 8. We approximate the dominant nine right singular functions of the operator $\mathcal{T}^\tau$ and use them to decompose the state space into nine different clusters. The clusters represent the coherent sets in the flow. Two dominant singular functions are shown in Figures 9a and 9b and the resulting clusters in Figure 9c. Additionally, we provide the video of the coherent sets in the flow identified using RaNNDy in the supplementary material.

V Ablation studies

V.1 Grid search using the VAMP score

The initialization of a randomized neural network plays an important role in determining the quality and numerical stability of the approximated Koopman operator. As discussed in Section 3, the randomized feature map

$$R(x) = \sigma(Wx + b)$$

generates a finite-dimensional feature space for the approximation. Consequently, the distributions used to sample the weights W and biases b, as well as their associated parameters, can influence the richness of the resulting feature space, the conditioning of the matrices involved in the approximation, and the quality of the estimated operator. An unsuitable initialization may produce features with

insufficient variability or excessive redundancy, potentially leading to poorly conditioned covariance matrices and less reliable estimates. It is therefore useful to investigate the sensitivity of the approximation to the choice of initialization.

We employ a grid-search strategy based on the VAMP score to compare different initialization schemes. Specifically, we consider candidate distributions for the weights and biases and apply Algorithm 1 to each combination. We then evaluate the resulting approximation using the VAMP score, defined here as the sum of the dominant eigenvalues of the approximated operator matrix as discussed in Section 3. The combination yielding the highest score is selected as the preferred initialization among the candidates considered. This procedure provides a systematic and computationally straightforward means of assessing how different initialization distributions affect the resulting approximation.

In addition to the choice of distribution, we investigate the influence of distributional parameters. For example, when Gaussian distributions are used, we denote the weight and bias scales by ω_W and ω_b, respectively. The feature map can then be written as

$$R(x;\omega) = \sigma\big(W(\omega_W)x + b(\omega_b)\big), \qquad \omega = (\omega_W, \omega_b),$$

where $W(\omega_W)$ and $b(\omega_b)$ denote random variables sampled from distributions parameterized by the corresponding scales. A grid search over candidate values of ω can be used to identify parameter ranges that yield favorable approximation performance. For larger or more complex parameter spaces, Bayesian optimization [67] might provide a more suitable potential alternative.

Tables 3 and 4 report the results of these experiments for the Ornstein–Uhlenbeck (OU) process and the Chignolin molecule, respectively. The distributional comparison indicates that the choice of initialization can have a noticeable effect on the VAMP score. In the OU experiment, Gaussian weight and bias initialization achieves a score of 2.1902, whereas uniform initialization for both weights and biases yields a substantially lower score of 0.4332. These results suggest that the distribution used for initialization can influence the quality of the resulting finite-dimensional approximation. However, the optimal distribution may depend on the underlying dynamical system.

Table 3: Grid search over weight and bias initialization distributions for the OU process using the VAMP score.

Weight Distribution	Bias Distribution	Time (s)	VAMP Score
normal	normal	0.3186	2.1902
normal	uniform	0.2529	2.1782
lecun_normal	normal	0.7938	2.1018
lecun_normal	uniform	0.8749	2.0938
uniform	normal	0.3097	0.4270
uniform	uniform	0.3197	0.4332
glorot_uniform	normal	0.3796	1.6128
glorot_uniform	uniform	0.2670	1.6128

In Table 5, the scale-sensitivity experiment further illustrates that the effect of initialization parameters is not uniform across the parameter space. For example, when both the weight and bias scales are set to 0.001, the resulting VAMP score is 1.0000, compared with 2.2060 when both scales are set to 0.5. In contrast, several configurations with moderate or larger weight scales and non-negligible bias scales produce scores close to 2.2.

In practice, standard normal initialization provides a simple and effective default for the examples considered here. Nevertheless, the grid-search results demonstrate that this choice should not be regarded as universally optimal. In particular, the VAMP score provides a useful criterion for comparing candidate initializations, but it should ideally be complemented by diagnostics based on matrix conditioning, etc. For the examples considered in this paper, a standard grid search over different distributions takes around 5–10 seconds depending on the problem size and implementation, see Table 6 for protein folding results.

Table 4: Grid search over weight and bias initialization distributions for the Chignolin molecule using the VAMP score.

Weight Distribution	Bias Distribution	Time (s)	VAMP Score
normal	normal	1.5467	2.7475
normal	uniform	0.1705	2.7879
normal	zeros	0.6861	2.7802
lecun_normal	normal	0.2509	2.0616
lecun_normal	uniform	0.1862	2.0510
lecun_normal	zeros	0.1554	2.0708
uniform	normal	0.1793	1.0000
uniform	uniform	0.1479	1.0000
uniform	zeros	0.1929	1.0000
glorot_uniform	normal	0.2629	1.9793
glorot_uniform	uniform	0.2048	2.0316
glorot_uniform	zeros	0.1518	2.0524

Table 5: Grid search over weight and bias initialization scales for the OU process.

Weight Scale (ω_w)	Bias Scale (ω_b)	Time (s)	VAMP Score
0.001	0.001	0.3179	1.0000
0.001	0.5	0.2962	1.0000
0.001	3.0	0.2718	1.0000
0.001	10.0	0.2725	1.0000
0.05	0.001	0.4051	1.6128
0.05	0.5	0.4017	1.6128
0.05	3.0	0.2847	1.0000
0.05	10.0	0.3175	1.0000
0.5	0.001	0.2899	1.3881
0.5	0.5	0.3310	2.2060
0.5	3.0	0.2564	2.2009
0.5	10.0	0.3167	2.1764
1.0	0.001	0.3165	1.3013
1.0	0.5	0.3169	2.1973
1.0	3.0	0.3159	2.1955
1.0	10.0	0.6499	2.1883
5.0	0.001	0.3242	0.9650
5.0	0.5	0.2887	2.1786
5.0	3.0	0.4262	2.1931
5.0	10.0	0.2694	2.1972
10.0	0.001	0.9138	0.9430
10.0	0.5	0.2512	2.1611
10.0	3.0	0.3093	2.1820
10.0	10.0	0.8377	2.1812

V.2 Robustness to different random seeds

Having investigated the sensitivity of the approximation to the choice of the distribution used for initialization and its parameters, we next examine the variability arising from different random realizations drawn from a fixed distribution. Although the chosen initialization distribution determines the richness of the randomized features, individual realizations may still lead to different finite-dimensional feature spaces and, consequently, different approximations of the Koopman operator. Assessing this variability is important for determining whether the observed approximation quality is representative of the chosen initialization scheme or depends strongly on a particular random seed.

To quantify this effect, we generate an ensemble of randomized neural networks using independent random seeds while keeping the initialization distribution and the remaining experimental settings

Table 6: Comparison of initialization distribution for protein molecules

System	Avg. Search Time (s)	Best Initialization	Best VAMP Score
Chignolin	5–6	normal weights + normal biases	2.57
NuG2	9–10	LeCun normal weights + Zero biases	4.61

fixed. For each realization, we apply Algorithm 1 and compute the four dominant eigenvalues for the OU process. We then calculate the ensemble mean and standard deviation of each eigenvalue, as well as the corresponding VAMP score. This procedure allows us to assess both the variability of the individual spectral estimates and the stability of their aggregate score across independent random initializations.

For the OU process, the ensemble-averaged leading eigenvalues are

$$\bar{\lambda} = [0.999533,\ 0.612335,\ 0.352190,\ 0.218048].$$

The corresponding mean-plus-minus-standard-deviation intervals are

$$\begin{aligned}\lambda_1 &: [0.999297,\ 0.999768],\\ \lambda_2 &: [0.612077,\ 0.612592],\\ \lambda_3 &: [0.347320,\ 0.357061],\\ \lambda_4 &: [0.213200,\ 0.222877].\end{aligned}$$

The first two eigenvalues exhibit particularly small variability across the ensemble, whereas the third and fourth exhibit somewhat larger, but still comparatively limited variation. These results suggest that the leading spectral estimates are reasonably stable under changes in the random seed used for the initialization scheme and experimental configuration considered here.

We further investigate the convergence of the ensemble-averaged VAMP score as the number of realizations increases. Table 7 reports the sum of the ensemble-averaged leading four eigenvalues for ensemble sizes ranging from 50 to 600. The resulting scores remain close to 2.182 throughout the experiment, with no systematic increase or decrease as the ensemble size grows. In particular, the score changes from 2.182460 for an ensemble of 50 realizations to 2.182425 for an ensemble of 600 realizations. The small variation across these ensemble sizes suggests that the sum of estimated eigenvalues is relatively insensitive to the particular collection of random seeds.

These observations thus indicate that, once an appropriate initialization scheme has been selected, the resulting Koopman approximation exhibits limited sensitivity to individual random realizations. However, the degree of robustness may depend on the dynamical system, the feature dimension, the initialization distribution, and the numerical conditioning of the matrices involved.

Table 7: Convergence of the ensemble-averaged VAMP score with increasing ensemble size. The reported score is the sum of the average of the leading four eigenvalues over all models in the ensemble.

Ensemble Size	Sum of Average Eigenvalues (VAMP Score)
50	2.182460
100	2.182810
200	2.182152
400	2.182144
600	2.182425

V.3 Comparison with random Fourier features (RFF)

Random Fourier features (RFFs) [57] provide a randomized approach to constructing finite-dimensional feature representations. They can be viewed as a special case of RaNNDy, corresponding to a single hidden layer with a cosine activation function and randomly sampled frequency vectors. RFFs have already been successfully used for learning Koopman operators [58]. It has in particular been shown that dominant eigenfunctions can be efficiently represented using Fourier features.

However, the RaNNDy framework provides a broader family of randomized feature maps, allowing for different choices of activation functions and weight and bias distributions. This flexibility could be particularly relevant when approximating non-self-adjoint Koopman operators, for which the dominant singular functions might not be well represented by a fixed Fourier basis. Thus, although RFFs can be effective for certain classes of dynamical systems, their suitability may depend on the structure of the underlying dynamics.

To investigate this, we use the Bickley jet as a benchmark problem and compare RaNNDy with RFFs, focusing on the approximation of the dominant singular functions of the Koopman operator. For higher bandwidths of the Gaussian kernel and across random feature dimensions of 2048 and 4096, the RFF approach achieves low VAMP scores of 3.98 and 3.50, respectively, which is not sufficient for the Bickley jet; however, with smaller bandwidth, i.e., high scaling of the normal distribution of weights σ, the VAMP score increases and the accuracy of RFFs and RaNNDy are similar, see Table 8. It is important to note that for a high number of random features in RFF, the size of the covariance matrices increases and makes the computations time-consuming, whereas RaNNDy attains a similar accuracy faster using a small number of neurons in the last hidden layer to reduce the size of the approximated matrices. The observed VAMP score is consistent with the expected VAMP score for the Bickley jet, which exhibits nine dominant singular patterns (see the accompanying video and, for example, [64, 66]). Furthermore, the greater flexibility in allowing different activation functions and distributions of randomized weights and biases may lead to better representations of more complicated structures.

Table 8 highlights the advantage of the RaNNDy approach, as we use a narrower final hidden layer (architecture [4096, 128]) before solving the variational problem. This reduces the dimensionality of the feature space N and, consequently, the size of the covariance matrices involved in the generalized eigenvalue problem, leading to significantly lower computational costs, while maintaining a similar VAMP score compared to the corresponding random Fourier feature models. The computational complexity of solving the generalized eigenvalue problems in RaNNDy scales cubically with the number of neurons in the last hidden layer of the network N and typically requires $\mathcal{O}(N^3)$ operations. Storing the covariance and cross-covariance matrices C_{00} and C_{01} requires $\mathcal{O}(N^2)$ memory.

Table 8: Comparison of RFFs and RaNNDy for the non-self-adjoint Koopman operator associated with the non-reversible Bickley jet. The feature configuration for RaNNDy specifies the number of neurons in its hidden layers.

Method	Feature Configuration	Scale (σ)	VAMP Score	Time (s)
RFF	2048	1.0	3.98	6.47
RFF	4096	1.0	3.50	84.47
RFF	2048	2.0	7.25	7.72
RFF	4096	2.0	7.11	72.47
RFF	8192	2.0	6.59	422.49
RFF	2048	3.0	8.17	7.28
RFF	4096	3.0	8.12	69.72
RaNNDy (best initialization)	[4096, 256]	0.5	7.72	1.63
RaNNDy (scales search)	[4096, 256]	0.5	7.72	13.63

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: We have provided theoretical proofs and a variety of numerical experiments for different benchmark problems corroborating the results.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: We have discussed the limitations of the algorithms in the conclusion section as well as in the numerical results section.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [Yes]

Justification: All assumptions are clearly stated.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: Yes, the code is provided, and all the experiments can be reproduced. For the protein folding experiments, the data sets have to be acquired from D.E. Shaw Research; we have provided the link in the paper.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes].

Justification: See above.

Guidelines:

- The answer [N/A] means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (https://neurips.cc/public/guides/CodeSubmissionPolicy) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (https://neurips.cc/public/guides/CodeSubmissionPolicy) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes].

   Justification: The details are mentioned in the numerical results section.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: We provide error metrics for the problems for which it is possible to find analytical solutions. For the protein folding problems, it is in general not possible to compute analytical eigenvalues and eigenfunctions.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).

- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: Yes, we have provided the details of the experimental setup and configuration of the machine used for the experiments.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics https://neurips.cc/public/EthicsGuidelines?

   Answer: [Yes]

   Justification: Yes, we have checked the code of ethics, and the research abides by them.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes].

    Justification: The developed methodologies can be used to analyze biomolecules such as proteins, RNA, and DNA more efficiently and thoroughly, which will help researchers in discovering new drugs and understanding biological processes and in particular disease mechanisms. The same methods that can be used to design life-saving drugs can potentially also be misused to design pathogens. This project is mostly concerned with the mathematical foundations of machine learning and operator learning and has applications beyond molecular dynamics.

Guidelines:

- The answer [N/A] means that there is no societal impact of the work performed.
- If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A].

    Justification: The paper does not pose any such risk.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: We have given proper citations for the protein folding data provided by D.E. Shaw research.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.

- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [N/A].

    Justification: The code is publicly available and properly documented.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A].

    Justification: The research does not involve any crowdsourcing or research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A].

    Justification: The research does not involve any crowdsourcing.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.

- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [N/A].

    Justification: This research work is our original contribution.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.