Physica D 481 (2025) 134822

Contents lists available at ScienceDirect

Physica D

journal homepage: www.elsevier.com/locate/physd

Learning dynamical systems from data: Gradient-based dictionary optimization

Mohammad Tabish [a,*], Neil K. Chada [b], Stefan Klus [c]

[a] Maxwell Institute for Mathematical Sciences, University of Edinburgh and Heriot–Watt University, Edinburgh, UK
[b] Department of Mathematics, City University of Hong Kong, Hong Kong Special Administrative Region
[c] School of Mathematical and Computer Sciences, Heriot–Watt University, Edinburgh, UK

ARTICLE INFO

Communicated by Dmitry Pelinovsky

Keywords:
Koopman operator
System identification
Dictionary learning
Gradient descent

ABSTRACT

The Koopman operator plays a crucial role in analyzing the global behavior of dynamical systems. Existing data-driven methods for approximating the Koopman operator or discovering the governing equations of the underlying system typically require a fixed set of basis functions, also called *dictionary*. The optimal choice of basis functions is highly problem-dependent and often requires domain knowledge. We present a novel gradient descent-based optimization framework for learning suitable and interpretable basis functions from data and show how it can be used in combination with EDMD, SINDy, and PDE-FIND. We illustrate the efficacy of the proposed approach with the aid of various benchmark problems such as the Ornstein–Uhlenbeck process, Chua's circuit, a nonlinear heat equation, as well as protein-folding data.

1. Introduction

Dynamical systems can be used to describe the motion of atoms, fluids, and planets as well as biological and chemical processes to name just a few examples. Deriving mathematical models for such complex problems can be challenging. Even if we do have mathematical models, the resulting dynamical systems will often be high-dimensional and highly nonlinear, which makes their analysis difficult or sometimes impossible. The goal of data-driven modeling approaches is to learn the governing equations or transfer operators associated with the system from measurement data. Instead of analyzing individual trajectories of the system, transfer operators such as the Koopman operator and Perron–Frobenius operator describe the evolution of observables and probability densities [1–4]. Data-driven methods allow us to study the global behavior of the system without requiring detailed mathematical models, see [5] for an overview of different applications. Of particular interest are the eigenvalues and eigenfunctions of transfer operators since they contain important information about timescales and slowly evolving spatiotemporal patterns of the systems. Often only a few dominant eigenfunctions and modes can help us understand the dynamics. Among the most frequently used techniques for numerically approximating transfer operators are *Ulam's method* [6,7], *extended dynamic mode decomposition* (EDMD) [4,8], and various extensions such as *kernel EDMD* (kEDMD) [9,10] and *generator EDMD* (gEDMD) [11]. While standard EDMD requires choosing a finite set of basis functions, kernel EDMD maps the data to a potentially infinite-dimensional feature space implicitly defined by the kernel. In the infinite-data limit, EDMD converges to a Galerkin projection of the Koopman operator onto the space spanned by the set of basis functions. Ulam's method can be regarded as a special where the dictionary contains indicator functions for a decomposition of the domain into disjoint sets [4].

Although the Koopman operator and its generator can also be used for system identification and forecasting, a method that directly identifies the governing equations, called *sparse identification of nonlinear dynamics* (SINDy), was proposed in [12]. Relationships between the Koopman generator and system identification are discussed in more detail in [11,13]. Just like the methods for approximating transfer operators, SINDy also requires a set of basis functions. The approach can also be extended to learn partial differential equations from data. The resulting method is called *PDE functional identification of nonlinear dynamics* (PDE-FIND) [14].

The accuracy of the learned transfer operators or governing equations depends strongly on the dictionary. Poorly chosen basis functions can result in ill-conditioned matrices, spectral pollution, and incorrect predictions. Selecting suitable basis functions is an open problem. In practice, often dictionaries comprising, for instance, monomials, trigonometric functions, or radial-basis functions are used. Our goal is to develop a more flexible framework for data-driven modeling approaches that not only learns the dynamics but also, at the same time,

* Corresponding author.
E-mail address: M.Tabish-1@sms.ed.ac.uk (M. Tabish).

https://doi.org/10.1016/j.physd.2025.134822
Received 8 November 2024; Received in revised form 5 May 2025; Accepted 25 June 2025
Available online 7 July 2025

the basis functions. Although neural network-based dictionary learning methods have been successfully used for approximating transfer operators in combination with EDMD and to identify governing equations, see, e.g., [15–19], where the output layer of the network represents the basis functions, we lose the interpretability of the dictionary and cannot immediately identify the governing equations. We propose a dictionary learning approach that is based on well-known gradient-based optimization techniques such as stochastic gradient descent (SGD), Nesterov's method, or Adam, see [20–22], and allows us to optimize the parameters of basis functions (e.g., the centers and bandwidths of Gaussians). The key advantage of our method is that it is a generalized framework that encapsulates different data-driven algorithms for learning dynamical systems and results in an interpretable representation. It can be regarded as a compromise between the flexibility of fully optimizable basis functions generated by a neural network and the interpretability and expressivity of pre-selected dictionaries. Alternating optimization algorithms to learn the Koopman operator have also been proposed in [19,23] using the reconstruction error as a loss function. However, we present a general framework for alternating optimization to learn the dynamics and optimal and interpretable dictionaries using different loss functions. The main contributions of this work are:

1. We propose a novel framework for learning dynamical systems from data using gradient-based optimization methods, which allow us to simultaneously learn the dynamics and suitable basis functions.
2. We show how this approach can be used to approximate the Koopman operator and to learn governing equations and illustrate its advantages over existing techniques.
3. We demonstrate the results with the aid of several different applications ranging from chaotic dynamical systems and protein-folding problems to a nonlinear heat equation.

The remainder of the paper is structured as follows: We first introduce the required concepts including transfer operators, EDMD, and SINDy as well as gradient-based optimization algorithms in Section 2. In Section 3, we discuss the proposed optimization framework for learning dynamical systems from data. We present numerical results for various deterministic and stochastic dynamical systems in Section 4. Open problems and a discussion of future work can be found in Section 5.

2. Background

In this section, we provide the necessary background material required for the derivation of our optimization framework, including the Koopman operator, data-driven algorithms for learning dynamical systems, as well as gradient-based optimization techniques.

2.1. Koopman operator

Let $\mathbb{X} \subset \mathbb{R}^d$ be the state space and $S^t : \mathbb{X} \to \mathbb{X}$ the flow map associated with a given dynamical system, i.e., for an initial condition $x(0) = x_0$, it holds that $S^t(x_0) = x(t)$. We are in particular interested in stochastic processes $\{X_t\}_{t\geq 0}$ governed by *stochastic differential equations* (SDE) of the form

$$\mathrm{d}X_t = b(X_t)\,\mathrm{d}t + \sigma(X_t)\,\mathrm{d}W_t,$$

where $b : \mathbb{R}^d \to \mathbb{R}^d$ is the drift term, $\sigma : \mathbb{R}^d \to \mathbb{R}^{d\times d}$ is the diffusion term, and W_t is a d-dimensional Wiener process. Let $\mathbb{E}[\cdot]$ denote the expectation of a random variable. The semigroup of Koopman operators $\{\mathcal{K}^t\}_{t\geq 0}$, with $\mathcal{K}^t : \mathcal{L}^\infty \to \mathcal{L}^\infty$, is defined by

$$(\mathcal{K}^t f)(x) = \mathbb{E}[f(S^t(x))].$$

The Koopman operator describes the evolution of observables rather than the evolution of the state of the system, see [1,3,4,24] for more details.

2.2. Extended dynamic mode decomposition

A data-driven algorithm for approximating the Koopman operator, called *extended dynamic mode decomposition* (EDMD), was proposed in [8]. EDMD requires trajectory data $\{(x_i, y_i)\}_{i=1}^m$, where $y_i = S^\tau(x_i)$ for a fixed lag time τ. We select a set of basis functions $D = \{\psi_1, \psi_2, \dots, \psi_n\}$, also known as *dictionary*. These basis functions can be, for instance, monomials, indicator functions, or radial basis functions. We define the vector-valued function $\psi : \mathbb{R}^d \to \mathbb{R}^n$ by

$$\psi(x) = [\psi_1(x), \psi_2(x), \dots, \psi_n(x)]^\top.$$

EDMD aims to find the best approximation of the Koopman operator $\mathcal{K}^\tau$ projected onto the space spanned by the selected basis functions. In what follows, let $\|\cdot\|_F$ denote the Frobenius norm. In order to obtain an approximation of the Koopman operator, we first map the training data to the feature space by defining

$$\Psi_x = [\psi(x_1), \psi(x_2), \dots, \psi(x_m)] \in \mathbb{R}^{n\times m} \quad \text{and}$$
$$\Psi_y = [\psi(y_1), \psi(y_2), \dots, \psi(y_m)] \in \mathbb{R}^{n\times m}$$

and then minimize the loss function

$$\mathcal{F}(K) = \left\|\Psi_y - K^\top \Psi_x\right\|_F.$$

An optimal solution of the regression problem is given by

$$K^\top = \Psi_y \Psi_x^+ = \left(\Psi_y \Psi_x^\top\right)\left(\Psi_x \Psi_x^\top\right)^+ = C_{yx} C_{xx}^+,$$

where $^+$ denotes the pseudoinverse and C_{xx} and $C_{xy} = C_{yx}^\top$ are the (uncentered) covariance and cross-covariance matrices, respectively. The matrix $K \in \mathbb{R}^{n\times n}$ is the matrix representation of the projected Koopman operator $\mathcal{K}^\tau$. We can then compute the eigenvalues λ_i and the eigenvectors v_i of the matrix K to obtain eigenfunctions

$$\varphi_i(x) = v_i^\top \psi(x)$$

of the projected operator.

2.3. Sparse identification of nonlinear dynamics

Instead of learning transfer operators associated with a given dynamical system, we can also directly approximate the governing equations from data using *sparse identification of nonlinear dynamics* (SINDy) [12]. Assuming the governing equations comprise only a few simple terms, SINDy solves a sparse regression problem to discover parsimonious models. Although SINDy has been extended to stochastic differential equations, see [11,25], we will restrict ourselves here to autonomous ordinary differential equations

$$\dot{x}(t) = b(x(t)).$$

Given training data of the form $\{(x_i, \dot{x}_i)\}_{i=1}^m$, which can, for example, be extracted from one long trajectory, we construct the data matrix

$$\dot{X} = [\dot{x}_1, \dot{x}_2, \dots, \dot{x}_m] \in \mathbb{R}^{d\times m}.$$

The matrix $\dot{X}$ contains the time-derivatives at the training data points, which can also be approximated numerically using finite differences. In order to find an approximation of the governing equations, we minimize the loss function

$$\mathcal{F}(\Xi) = \left\|\dot{X} - \Xi^\top \Psi_x\right\|_F,$$

where $\Xi \in \mathbb{R}^{n\times d}$. The least squares solution to the above problem is

$$\Xi^\top = \dot{X}\Psi_x^+ = \left(\dot{X}\Psi_x^\top\right)\left(\Psi_x \Psi_x^\top\right)^+.$$

The data-driven approximation of the dynamical system is then defined by

$$\dot{x} \approx \Xi^\top \psi(x).$$

2.4. SINDy for PDEs

PDE-FIND [14] is an extension of the SINDy approach to partial differential equations. The algorithm discovers the governing equations from time-series data by solving a sparse regression problem. The general form of a PDE that the algorithm recovers is

$$u_t = N(x, u, u_x, u_{xx}, \dots),$$

where the subscripts denote the partial derivatives with respect to t and x. Suppose that we have data from a solution of the PDE on an $n \times m$ grid, i.e., for m spatial and n time points. We start by constructing a discretized version of the right-hand side of the above equation and express $u(x,t)$ and its derivatives using the matrix $\Theta(U)$. Each column of the matrix $\Theta(U)$ represents a candidate term that can be present in the right-hand side of the PDE. That is, if we have d candidate terms, then there will be d columns. Each column represents the value of a particular candidate function on all mn space–time points. We can write the discretized version of the PDE as

$$U_t = \Theta(U)\xi,$$

where $\Theta(U) \in \mathbb{R}^{mn \times d}$. Each nonzero entry in ξ corresponds to a term in the PDE selected from the dictionary of candidate terms. To obtain a sparse vector, the algorithm minimizes the loss function

$$\mathcal{F}(\xi) = \left\| U_t - \Theta(U)\xi \right\|_2^2 + \epsilon \kappa(\Theta(U)) \left\| \xi \right\|_0,$$

where the second term is added to avoid overfitting. Here, $\epsilon > 0$ is a regularization parameter, $\kappa(\Theta(U))$ is the condition number of the matrix $\Theta(U)$, and $\|\xi\|_0$ is the number of nonzero entries of ξ.

2.5. Gradient descent algorithms

We now turn our attention to gradient descent algorithms, which aim to solve minimization problems of the form

$$x^* = \operatorname*{argmin}_{x \in \mathbb{R}^n} Q(x),$$

where Q is called *loss function*, *cost function*, or *objective function*. There exist many different gradient descent algorithms to find a minimizer x^* of the above unconstrained optimization problem. We will briefly review some of these methods, but refer the reader to [21,22,26,27] for more details.

2.5.1. Gradient descent

Gradient descent (GD) aims to find a minimizer by following the gradient "downhill". Given an initial guess x_0, GD is defined by

$$x_{t+1} = x_t - h \cdot \nabla_x Q(x_t),$$

where $h > 0$ is the so-called *learning rate* or step size. However, computing the full gradient of the loss function can be computationally expensive.

2.5.2. Stochastic gradient descent

Stochastic gradient descent (SGD) is a stochastic variant of the gradient descent algorithm. First, we rewrite the loss function as

$$Q(x) = \sum_{i=1}^{m} Q_i(x).$$

We then approximate the actual gradient $\nabla_x Q(x)$ by the gradient $\nabla_x Q_i(x)$ w.r.t. a single data point and compute

$$x_{t+1} = x_t - h \cdot \nabla_x Q_i(x_t).$$

For optimization problems involving a large number of data points, SGD reduces the computational costs considerably. However, the convergence can be very slow due to the approximation of the true gradient. A compromise between computational efficiency and accuracy is to compute the gradient with respect to a random subset or so-called *batch* of data points. This is known as mini-batch SGD.

2.5.3. Nesterov's method

Nesterov's method [22,28] is an accelerated version of gradient descent that attains faster convergence. The main difference is that the updates are based on a combination of the current step and the previous step, scaled using a parameter β that is also updated with each iteration. Typically, β_t is computed using

$$p_{t+1} = \frac{1}{2}\left(1 + \sqrt{1 + 4p_t^2}\right),$$
$$\beta_t = \frac{p_t - 1}{p_{t+1}},$$

where $p_0 = 0$. We then define

$$x_{t+1} = x_t + \beta_t(x_t - x_{t-1}) - h \cdot \nabla_x Q(x_t + \beta_t(x_t - x_{t-1})).$$

2.5.4. Adam

The last optimization method that we will consider is Adam [20], which is an extension of SGD. It is among the most popular methods for training neural networks as it is computationally efficient for large-scale optimization problems, has reasonable memory requirements, and is suitable for noisy gradients as well. The algorithm is based on the adaptive estimation of first- and second-order moments and involves the calculation of moving averages of the gradients while controlling the decay of these averages with the parameters β_1 and β_2. For a theoretical analysis of the convergence of Adam, see, e.g., [29,30]. The values of the hyperparameters are typically set to the default values $\beta_1 = 0.9$, $\beta_2 = 0.999$, and $\epsilon = 1 \times 10^{-8}$. The main steps for the algorithm are:

- Update biased first moment: $m_{t+1} = \beta_1 m_t + (1 - \beta_1) \cdot \nabla_x Q(x_t)$.
- Update biased second raw moment estimate: $v_{t+1} = \beta_2 v_t + (1 - \beta_2)(\nabla_x Q(x_t))^2$.
- Compute bias-corrected first moment estimate: $\hat{m}_{t+1} = m_{t+1}/(1 - \beta_1^{t+1})$.
- Compute bias-corrected second raw moment estimate: $\hat{v}_{t+1} = v_{t+1}/(1 - \beta_2^{t+1})$.
- Compute next iterate: $x_{t+1} = x_t - h \cdot \hat{m}_{t+1}/(\sqrt{\hat{v}_{t+1}} + \epsilon)$.

3. Proposed optimization framework

We now derive the proposed optimization framework for learning dynamical systems from data. From the above discussion on different data-driven algorithms, we have seen that they require a set of basis functions to learn the dynamics. The selection of appropriate basis functions is not straightforward since it is problem-dependent. We use gradient descent-based algorithms to learn the dynamics and the parameters of the basis functions from data in an alternating optimization fashion.

3.1. Parametric EDMD

To illustrate our approach, we use the example of learning the Koopman operator with a set of n parametric basis functions

$$\mathcal{D} = \{\psi_1(x, w_1), \psi_2(x, w_2), \dots, \psi_n(x, w_n)\}, \tag{1}$$

where each w_i is a p_i-dimensional vector, i.e., each function has p_i unknown parameters. We define the parameter-dependent vector-valued function

$$\psi(x, w) = [\psi_1(x, w_1), \psi_2(x, w_2), \dots, \psi_n(x, w_n)]^\top. \tag{2}$$

For a fixed parameter vector w, we can transform the training data $\{x_i, y_i\}_{i=1}^m$ using (2) to get $\Psi_x(w), \Psi_y(w) \in \mathbb{R}^{n \times m}$. We then minimize the reconstruction error, given by

$$\mathcal{F}(K, w) = \left\| \Psi_y(w) - K^\top \Psi_x(w) \right\|_F, \tag{3}$$

using gradient descent algorithms to find K. However, the reconstruction error might not be a good choice for optimizing the parameters of the basis functions. Suppose, for example, we choose the basis functions to be Gaussians with variable centers and bandwidths. In that case, the bandwidths of the Gaussian functions might tend to infinity so that $\Psi_x(w)$ and $\Psi_y(w)$ become almost constant and choosing $K = I$ minimizes the loss function. To avoid this problem, we utilize the *variational approach for Markov processes* (VAMP) score [15]. The VAMP-2 score is defined by

$$\hat{\mathcal{R}}_2(K, w) = \left\| C_{xx}^{-1/2}(w) C_{xy}(w) C_{yy}^{-1/2}(w) \right\|_F^2, \tag{4}$$

where the covariance and cross-covariance matrices are given by

$$C_{xx}(w) = \Psi_x(w)\Psi_x^\top(w), \quad C_{xy}(w) = \Psi_x(w)\Psi_y^\top(w), \quad \text{and}$$
$$C_{yy}(w) = \Psi_y(w)\Psi_y^\top(w).$$

It was shown in [31] that maximizing the VAMP-2 score results in basis functions associated with the slow dynamics of the system. We use the VAMP-2 score to optimize the parameters w.

Proposition 3.1. *For the loss function (3), it holds that*

$$\nabla_K \mathcal{F}(K, w) = 2(\Psi_x(w)\Psi_x^\top(w)K - \Psi_x(w)\Psi_y^\top(w)).$$

Proof. We have

$$\begin{aligned}\mathcal{F}(K, w) &= \operatorname{tr}\left((\Psi_y(w) - K^\top\Psi_x(w))^\top(\Psi_y(w) - K^\top\Psi_x(w))\right) \\ &= \operatorname{tr}\left(\Psi_y^\top(w)\Psi_y(w)\right) - 2\operatorname{tr}\left(\Psi_y^\top(w)K^\top\Psi_x(w)\right) \\ &\quad + \operatorname{tr}\left(\Psi_x^\top(w)KK^\top\Psi_x(w)\right).\end{aligned}$$

Computing the derivative with respect to K, we have

$$\begin{aligned}\nabla_K \mathcal{F}(K, w) &= \left(\Psi_x(w)\Psi_x^\top(w)K + \Psi_x(w)\Psi_x^\top(w)K\right) - 2\Psi_x(w)\Psi_y^\top(w) \\ &= 2\Psi_x(w)\Psi_x^\top(w)K - 2\Psi_x(w)\Psi_y^\top(w),\end{aligned}$$

see, e.g., [32]. □

We use the Python library JAX [33] to compute the gradient of the VAMP-2 score (4) with respect to the parameters w. In order to be able to apply the stochastic variant of gradient descent algorithms, we need to decompose the loss function into a sum of loss functions.

Proposition 3.2. *For the loss function* $\mathcal{F}(K, w) = \left\|\Psi_y(w) - K^\top\Psi_x(w)\right\|_F^2$, *we write*

$$\mathcal{F}(K, w) = \sum_{i=1}^{m} \mathcal{F}_i(K, w), \quad \text{with} \quad \mathcal{F}_i(K, w) = \left\|\psi(y_i, w) - K^\top\psi(x_i, w)\right\|_2^2,$$

where m is the number of data points.

Proof. This follows immediately from the property $\|A\|_F^2 = \sum_i \|a_i\|_2^2$, where a_i denotes the ith column of A. □

Based on the above splitting of the cost function, we can write the gradient as

$$\nabla_K \mathcal{F}(K, w) = \sum_{i=1}^{m} \nabla_K \mathcal{F}_i(K, w)$$

and then select a random subset of data points to approximate the gradient.

Remark 3.3. Using the definition of the Frobenius norm, we can further show that the optimization problem (3) can also be divided into subproblems for each column of the matrix K as

$$\mathcal{F}(K, w) = \left\|\Psi_y(w) - K^\top\Psi_x(w)\right\|_F^2 = \sum_{i=1}^{n} \left\|[\Psi_y^\top(w)]_i - \Psi_x^\top(w)[K]_i\right\|_2^2,$$

where $[\Psi_y^\top(w)]_i$ is the ith row of $\Psi_y(w)$ and $[K]_i$ is the ith column of K. For each i, the above regression problem is of the form

$$\min_{x\in\mathbb{R}^n} \left\|Ax - b\right\|_2^2,$$

with $A = \Psi_x^\top(w) \in \mathbb{R}^{m\times n}$ and $b = [\Psi_y^\top(w)]_i \in \mathbb{R}^m$. The application of gradient-descent techniques to regression problems of this form has been studied, for example, in [22]. It has been shown that a good choice for the step size is the inverse of the largest eigenvalue of the matrix $A^\top A$. We thus choose the step size $h = \frac{1}{\lambda_{max}}$, where λ_{max} is the largest eigenvalue of $\Psi_x(w)\Psi_x^\top(w)$.

3.2. Parametric SINDy for a system of ODEs

The optimization framework discussed above can in the same way be applied to system identification problems. We aim to discover autonomous ordinary differential equations

$$\dot{x}(t) = b(x(t), w),$$

where $x(t) \in \mathbb{R}^d$ represents the state of the system at time t and w is a vector of parameters. The standard SINDy algorithm cannot detect the values of parameters w since it requires a fixed set of basis functions. Assume, for instance, that the right-hand side contains $\sin(\alpha x_i)$ or $e^{\alpha x_i}$, then SINDy would only be able to identify the governing equations if $\sin(\alpha x_i)$ with the correct value of α is contained in the dictionary. To address this issue, we select a set of parametric basis functions (1) and compute the matrix $\Psi_x(w) \in \mathbb{R}^{n\times m}$ using (2). We then minimize the reconstruction error

$$\mathcal{F}(\Xi, w) = \left\|\dot{X} - \Xi^\top\Psi_x(w)\right\|_F. \tag{5}$$

This allows us to optimize Ξ and w at the same time. We again use gradient descent to minimize the loss function in an alternating fashion. The approximation of the dynamical system is then given by

$$\dot{x} \approx \Xi^\top\psi(x, w).$$

Proposition 3.4. *For the loss function (5), the derivative of $\mathcal{F}(\Xi, w)$ with respect to Ξ is given by*

$$\nabla_\Xi \mathcal{F}(\Xi, w) = 2(\Psi_x(w)\Psi_x^\top(w)\Xi - \dot{X}\Psi_x^\top(w)).$$

Proof. The proof is similar to the EDMD counterpart. □

For computing the gradient of (5) with respect to w, we again use JAX.

Proposition 3.5. *The cost function* $\mathcal{F}(\Xi, w) = \left\|\dot{X} - \Xi^\top\Psi_x(w)\right\|_F^2$ *can be written as*

$$\mathcal{F}(\Xi, w) = \sum_{i=1}^{m} \mathcal{F}_i(\Xi, w), \quad \text{with} \quad \mathcal{F}_i(\Xi, w) = \left\|\dot{x}_i - \Xi^\top\psi(x_i, w)\right\|_2^2,$$

where m is the number of data points.

Proof. The proof again follows from the properties of the Frobenius norm. □

3.3. Parametric SINDy for PDEs

We can also extend the framework to learn PDEs from data. Assume the PDE we aim to recover is of the form

$$u_t = N(x, u, u_x, u_{xx}, f(u, w_1), \dots),$$

where $N(\cdot)$ is a nonlinear function of $u(x, t)$ and its partial derivatives with respect to x and t and w is the vector of parameters of the PDE. We create the matrix $\Theta(U(w)) \in \mathbb{R}^{mn\times d}$ of the basis functions as discussed in Section 2. We then minimize the loss function

$$\mathcal{F}(\xi, w) = \left\|U_t - \Theta(U(w))\xi\right\|_2^2 \tag{6}$$

using the proposed framework. We omit the $\|\cdot\|_0$ penalty term due to differentiability issues.

Proposition 3.6. *For fixed w, the derivative of the loss function (6) with respect to ξ is given by*

$$\frac{\partial}{\partial \xi}\mathcal{F}(\xi, w) = 2\Theta(U(w))^\top \Theta(U(w))\xi - 2\Theta(U(w))^\top U_t.$$

Proof. This follows from the proof of Proposition 3.1. □

For computing the gradient of (6) with respect to w, we again use JAX.

Remark 3.7. In order to avoid overfitting, it would also be possible to add regularization terms to the various loss functions. For example, SINDy is based on the assumption that there are only a few terms governing the dynamics. We can hence include an L^1 regularization of the matrix Ξ to obtain a sparse solution. For the PDE identification approach, a regularization based on the condition number of the matrix $\Theta(U(w))$ might be beneficial as shown in [14].

3.4. Proposed algorithm

We now illustrate the proposed optimization framework. Let $\mathcal{L}_1(A, w)$ and $\mathcal{L}_2(A, w)$ be two arbitrary loss functions. We define an alternating Adam algorithm, which can be used for both parametric EDMD and SINDy. Other gradient-based optimization methods can be implemented in a similar way.

Algorithm 1 Alternating Adam

Initialization:

- Select $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 1 \times 10^{-8}$ (default) & a step size $h = 0.01$
- Select initial matrices w_1 for parameters in basis and A_0 for dynamics matrix.
- $m_0^a = 0$ (first moment for A) & $v_0^a = 0$ (second moment for A)
- $m_1^w = 0$ (first moment for w) & $v_1^w = 0$ (second moment for w)

while not converged **do**

$$\begin{cases} m_{t+1}^a = \beta_1 m_t^a + (1-\beta_1)\cdot \nabla_A \mathcal{L}_1(A_t, w_{t+1}) \\ m_{t+1}^w = \beta_1 m_t^w + (1-\beta_1)\cdot \nabla_w \mathcal{L}_2(A_t, w_t) \end{cases}$$ ▷ Update biased first moments

$$\begin{cases} v_{t+1}^a = \beta_2 v_t^a + (1-\beta_2)(\nabla_A \mathcal{L}_1(A_t, w_{t+1}))^2 \\ v_{t+1}^w = \beta_2 v_t^w + (1-\beta_2)(\nabla_w \mathcal{L}_2(A_t, w_t))^2 \end{cases}$$ ▷ Update biased second moments

$$\begin{cases} \hat{m}_{t+1}^a = m_{t+1}^a/(1-\beta_1^{t+1}) \\ \hat{m}_{t+1}^w = m_{t+1}^w/(1-\beta_1^{t+1}) \end{cases}$$ ▷ Update bias-corrected first moments

$$\begin{cases} \hat{v}_{t+1}^a = v_{t+1}^a/(1-\beta_2^{t+1}) \\ \hat{v}_{t+1}^w = v_{t+1}^w/(1-\beta_2^{t+1}) \end{cases}$$ ▷ Update bias-corrected second moments

$$w_{t+1} = w_t - h\cdot \hat{m}_{t+1}^w/(\sqrt{\hat{v}_{t+1}^w} + \epsilon)$$ ▷ Basis parameters update

$$A_{t+1} = A_t - h\cdot \hat{m}_{t+1}^a/(\sqrt{\hat{v}_{t+1}^a} + \epsilon)$$ ▷ Dynamics matrix update

end while

To apply Algorithm 1 to parametric EDMD, we use the loss functions $\mathcal{L}_1 = \mathcal{F}$ (reconstruction error (3)) and $\mathcal{L}_2 = \hat{\mathcal{R}}_2$ (VAMP-2 score (4)) for the optimization of the matrix $A = K$ and the parameters w, respectively. For parametric SINDy, $\mathcal{L}_1 = \mathcal{L}_2 = \mathcal{F}$ are given by the reconstruction error (5), which optimizes both the matrix $A = \Xi$ and the parameters w. Similarly, for PDE-FIND, the loss function for the optimization of the vector $A = \xi$ and the parameters w is the reconstruction error (6).

4. Numerical results

We now apply the proposed framework to various benchmark problems.

4.1. Parametric EDMD

4.1.1. Protein folding

Analyzing the structure of proteins, especially transitions between folded and unfolded states, helps us understand biological processes and develop new drugs or treatments [34]. We consider the protein Chignolin (CLN025) consisting of 10 residues and 166 atoms. The simulation data was generated by D.E. Shaw Research, see [35] for more details. The trajectory is $1.06 \cdot 10^8$ ps long. We subsample the trajectory to create the training data $\{x_i, y_i\}_{i=1}^m$ using the lag time $\tau = 9911.5$ ps. We then transform the trajectory data using contact maps, which measure the distances between different pairs of residues, resulting in $\Psi_x, \Psi_y \in \mathbb{R}^{28\times 10{,}694}$. Since the basis functions are in this case the contact distances, there are no parameters to optimize.

To approximate the Koopman operator, we apply standard EDMD and the proposed optimization framework. The relative approximation error for different gradient descent-based algorithms compared to the exact EDMD solution are shown in Fig. 1(a). Eigenvalues of the approximated Koopman operator are given in Fig. 1(b). There is a spectral gap between the second and third eigenvalue, which indicates that there exist two metastable states. Furthermore, the values of the two dominant eigenfunctions, shown in Fig. 1(c), can be clustered into two sets representing the folded and unfolded states of the molecule. Examples of folded and unfolded states are shown in Fig. 2(a) and 2(b), respectively. The frequency of contacts between different residues of the molecule for the set of folded and unfolded states is presented in Fig. 2(c) and 2(d), respectively. We can see that Chignolin is mostly in its folded (native) state. The example shows that the gradient descent algorithms converge to the EDMD approximation and that Adam eventually outperforms the other gradient descent approaches.

4.1.2. Ornstein–Uhlenbeck process

We simulate a one-dimensional Ornstein–Uhlenbeck process, given by

$$dX_t = -\nabla V(X_t)\,dt + \sigma(X_t)\,dW_t, \tag{7}$$

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

$$X_{k+1} = X_k - \eta\nabla V(X_k) + 2\beta^{-1}\Delta W_k,$$

where η is the step size and $\Delta W_k = W_k - W_{k-1} \sim \mathcal{N}(0, \eta)$, i.e., normally distributed with mean 0 and variance η. We choose $\alpha = 1$ and $\beta = 4$ and generate $m = 5000$ data points with a lag time $\tau = 0.5$. The exact eigenvalues and eigenfunctions of the Koopman operator are

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

where H_i denotes the ith probabilists' Hermite polynomial. We use the basis functions

$$\mathcal{D} = \left\{ \psi_i(x, w_i) = \exp\left(-\frac{(x-c_i)^2}{2\sigma_i^2}\right),\ i = 1, 2, \ldots, n \right\},$$

i.e., Gaussian functions with variable centers c_i and bandwidths σ_i. That is, $w_i = [c_i, \sigma_i]^\top$ and $w \in \mathbb{R}^{2n}$ contains the centers and bandwidths of all basis functions. We select the $n = 14$ basis functions shown in Fig. 3(a) to approximate the Koopman operator. Note that most of the centers of the Gaussian functions are initially in the left half of the domain. As a result, the approximation of the eigenfunctions will be poor in the right half. Applying the alternating optimization algorithm allows us to find more suitable basis functions. The optimized basis

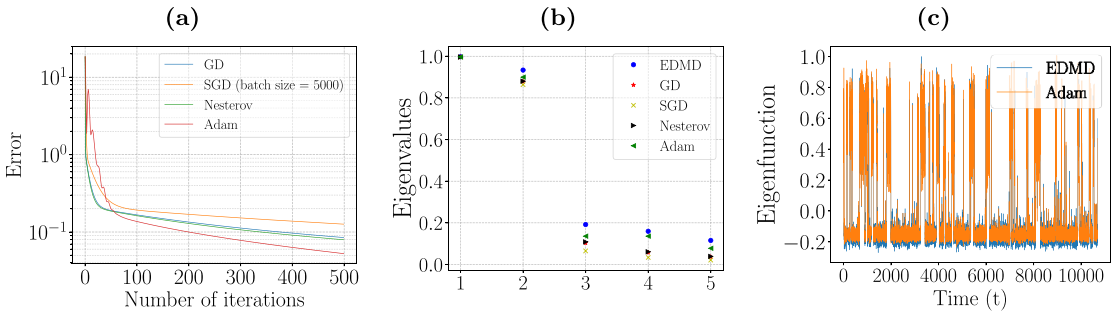

Fig. 1. Koopman operator approximation for the Chignolin protein. (a) Convergence of the relative approximation error of different algorithms for the matrix K. (b) Eigenvalues of the approximated Koopman operator. (c) Values of the second eigenfunction of the approximated Koopman operator.

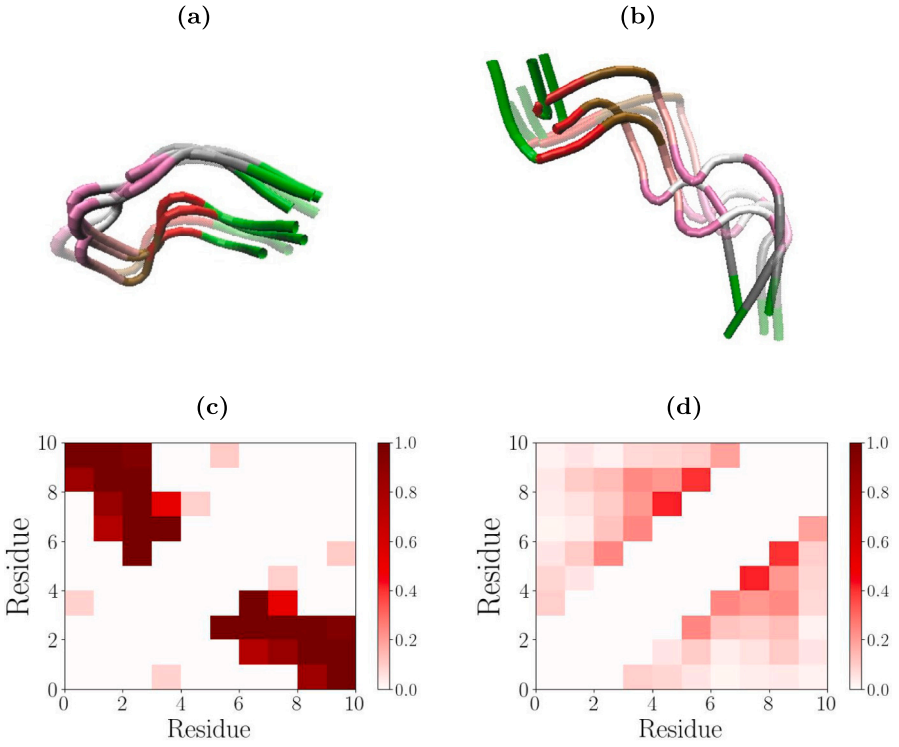

Fig. 2. (a) & (b) Folded and unfolded states of the Chignolin protein. (c) & (d) Frequency of contacts between different residue pairs over all the identified folded and unfolded states, respectively.

functions are shown in Fig. 3(b). The centers are now evenly distributed and lead to a better approximation of the Koopman operator. The convergence of the reconstruction error (3) is presented in Fig. 3(c) and the convergence of the VAMP-2 score in Fig. 3(d). The eigenvalues and eigenfunctions of the approximated Koopman operator are presented in Fig. 3(e) and 3(f), respectively. The numerically computed eigenvalues and eigenfunctions match the theoretical results.

4.1.3. Triple-well 2D

We now consider a two-dimensional triple-well problem [36], given by

$$V(x_1,x_2) = 3\exp(-x_1^2-(x_2-1/3)^2) - 3\exp(-x_1^2-(x_2-5/3)^2) - 5\exp(-(x_1-1)^2-x_2^2) - 5\exp(-(x_1+1)^2-x_2^2) + \frac{2}{10}x_1^4 + \frac{2}{10}(x_2-1/3)^4.$$

The potential is visualized in Fig. 4(a). We set $\beta = 1.68$ and uniformly sample $m = 100000$ training data points in the domain $[-2,2]\times[-2,2]$ and integrate the SDE (7) using the Euler–Maruyama method. To approximate the Koopman operator, we use $n = 25$ Gaussian functions in the domain $[0,2]\times[-2,2]$ as shown in Fig. 4(b). That is, the centers of all the basis functions are initially in the right half of the domain. The optimized basis functions are shown in Fig. 4(c). The centers moved and also the bandwidths changed significantly, leading to a more accurate approximation of the Koopman operator. The second eigenfunction of the approximated Koopman operator, separating the two deeper wells, is shown in Fig. 4(d).

4.2. Parametric SINDy

4.2.1. Chua's circuit

We consider the modified Chua circuit [37] given by

$$\dot{x}_1 = \alpha[x_2 - f(x)],$$
$$\dot{x}_2 = x_1 - x_2 + x_3,$$
$$\dot{x}_3 = -\beta x_2,$$

where $f(x) = -b\sin\left(\frac{\pi x_1(t)}{a} + d\right)$ describes the electrical response of the resistor, see [38,39]. We generate training data using $a = 2.6, b = 0.11, d = 0, \alpha = 10.2, \beta = 14.286$. To illustrate how the parameters affect

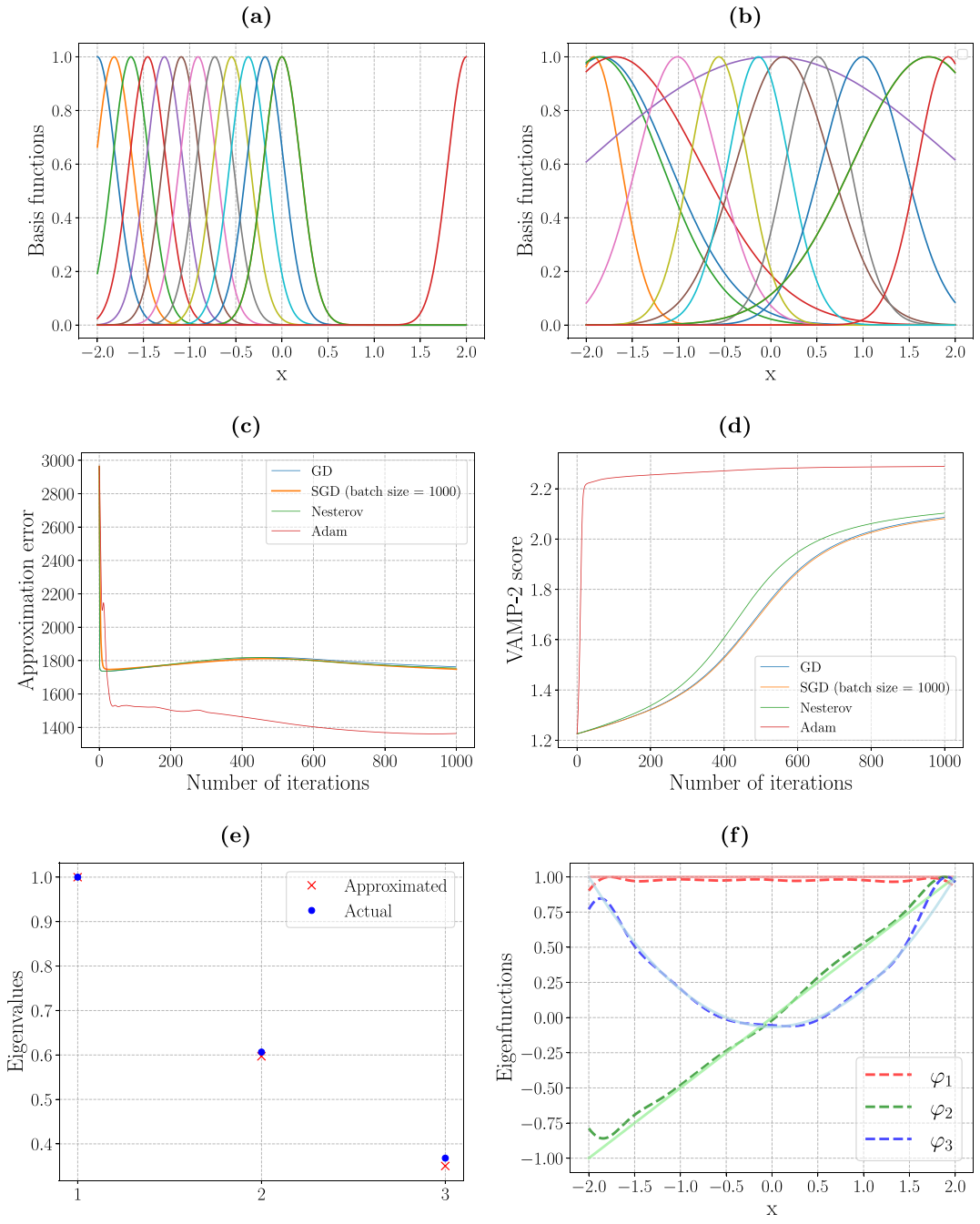

Fig. 3. Koopman operator approximation for the Ornstein–Uhlenbeck process using parametric Gaussian functions. (a) Initial basis functions. (b) Optimized basis functions. (c) Convergence of the reconstruction error. (d) Convergence of the VAMP-2 score for optimizing the parameters of the basis functions. (e) Three dominant eigenvalues of the Koopman operator. (f) Corresponding eigenfunctions of the Koopman operator, where the solid lines represent the true eigenfunctions.

the dynamics, we plot the reconstruction error for varying values of w_1 using the basis functions

$$D = \left\{x_1, x_2, x_3, x_1x_3, x_1x_2, x_2^2, \sin(w_1x_1), \cos(w_2x_2)\right\}.$$

The results are shown in Fig. 5(a). The minimum reconstruction error is obtained for $w_1 \approx 1.208$, i.e., $a \approx 2.6$, which is the correct value of the parameter. We see that there are two regions in the energy landscape. Selecting the initial value for w_1 in Region 1, we obtain the global minimum, i.e., zero reconstruction error, whereas choosing the initial value to be in Region 2, the algorithm converges to the local minimum. The governing equations cannot be identified using standard SINDy, unless we explicitly choose the basis function $\sin(\frac{\pi}{2.6}x_1)$. We use the proposed framework with the above set of parametric basis functions D to find the optimal matrix Ξ and parameters w. The convergence of the algorithms is illustrated in Fig. 5(b) and 5(c), respectively. The dynamics of the discovered equations shown in Fig. 5(d) illustrate that the algorithm determines the correct matrix Ξ and the correct value of the parameter w_1, provided that we start in Region 1.

4.2.2. Nonlinear heat PDE

As a final example, we use the proposed framework to identify a PDE from simulation data. Consider the nonlinear heat equation

$$\rho c_p u_t = \frac{\partial}{\partial x}(\kappa(u)u_x) = \frac{\partial}{\partial u}\kappa(u)\left(\frac{\partial u}{\partial x}\right)^2 + \kappa(u)\frac{\partial^2 u}{\partial x^2},$$

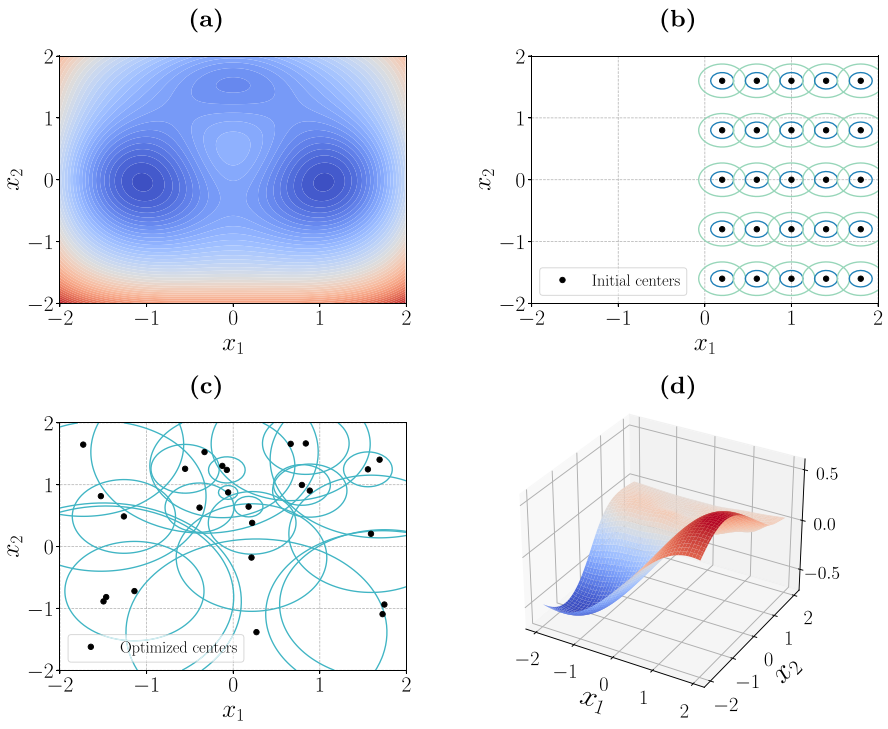

Fig. 4. Koopman operator approximation for the triple-well problem using parametric Gaussian basis functions. (a) Potential V. (b) Initial basis functions. (c) Optimized basis functions. (d) Second eigenfunction of the Koopman operator.

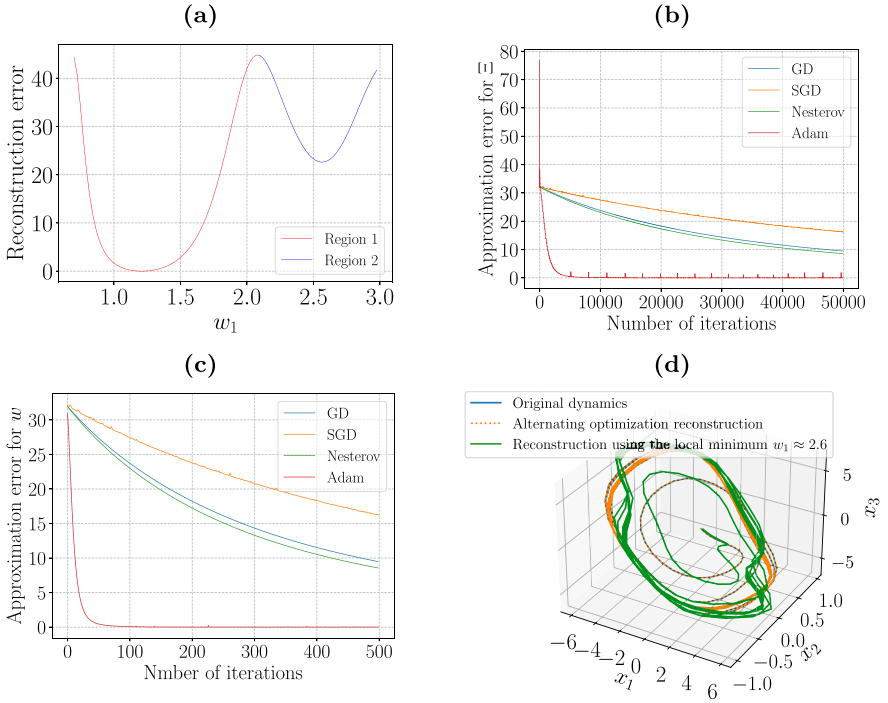

Fig. 5. Results for Chua's circuit. (a) SINDy reconstruction error as a function of w_1. (b) & (c) Convergence of the approximation errors for the loss functions for Ξ and w, respectively. (d) Reconstruction of the dynamics using the optimization algorithms. The orange graph completely overlaps the blue, representing a perfect reconstruction of the dynamics. The green trajectory shows the resulting dynamics using a different value of w_1.

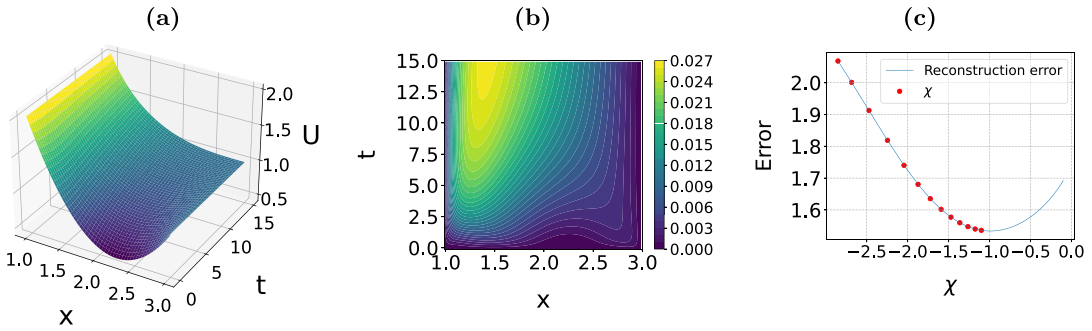

Fig. 6. Simulation of the nonlinear heat equation. (a) Surface plot of the solution of the PDE. (b) Contour plot of the error between the original and the discovered PDE. (c) Convergence of the algorithm to the correct value of χ.

taken from [40], where $u(x,t)$ is the temperature at position x and time t, c_p is the heat capacity, ρ is the density, and κ is the thermal conductivity that depends on the temperature u. For $\kappa(u) = \kappa_0 e^{\chi u}$, we get

$$\rho c_p u_t = \kappa_0 \chi e^{\chi u} u_x^2 + \kappa_0 e^{\chi u} u_{xx}.$$

We choose $\rho = c_p = 1$, $\kappa_0 = 0.1$, and $\chi = -1$ and solve the equation using a finite difference scheme. For more details, see [40]. The temperature at the boundary is kept constant, i.e.,

$$u(1,t) = 2, \quad u(3,t) = 1, \quad t > 0,$$

and the initial temperature is

$$u(x,0) = 2 - \frac{x-1}{2} + (x-1)(x-3),$$

for $x \in [1,3]$. We then construct the matrix $\Theta(U(\chi))$ as discussed in Section 2, which includes different candidate terms to discover the PDE. Since the parameter χ is present within the exponential term, it cannot be identified using the standard PDE-FIND algorithm. We build a library of parametrized basis functions

$$\Theta(U(\chi)) = \left[1, U, U_x, UU_x, U^2U_x, UU_{xx}, U^2U_{xx}, e^{\chi U}U_x^2, e^{\chi U}U_{xx}\right]$$

and apply the alternating optimization algorithm to the reconstruction error (6). The resulting vector ξ then contains the coefficients for the different terms. We obtain $\chi = -1.095$ and the PDE

$$u_t = -0.114\, e^{\chi u} u_x^2 + 0.105\, e^{\chi u} u_{xx}.$$

The small difference between the true PDE and the identified PDE is caused by the finite difference approximations of the derivatives. Nevertheless, the example shows that our method allows us to identify parameter-dependent PDEs from data (see Fig. 6).

5. Conclusion and future work

We proposed a novel data-driven alternating optimization framework for approximating the Koopman operator and discovering the governing equations of dynamical systems. The main goal was to learn optimal and interpretable dictionaries and the dynamics at the same time. We demonstrated the efficacy of the proposed framework using various benchmark problems such as the Ornstein–Uhlenbeck process, a triple-well potential, and protein folding data. Furthermore, we discovered the governing equations of Chua's circuit and a nonlinear heat equation with temperature-dependent thermal conductivity. The numerical results are promising and show that we can indeed learn more suitable dictionaries resulting in more accurate approximations of the Koopman operator or the dynamical system itself.

There are, however, open problems. One of the major challenges is selecting proper initial conditions and learning rates for the gradient descent algorithms. The algorithms might diverge or get stuck in local minima if the initial conditions or step sizes are chosen poorly. This leads to several interesting directions to extend this work in the future: A theoretical analysis of the optimization problems would help us understand the loss functions and the existence of local minima, which might allow us to choose suitable initial conditions and step sizes. An extension of this work would be to consider kernel-based variants of the data-driven algorithms, such as kernel EDMD [9,10]. The proposed framework could then be used to optimize the kernel parameters.

CRediT authorship contribution statement

Mohammad Tabish: Writing – original draft, Software, Conceptualization, Writing – review & editing, Investigation, Visualization, Methodology, Data curation. **Neil K. Chada:** Supervision, Writing – original draft, Writing – review & editing. **Stefan Klus:** Writing – original draft, Project administration, Validation, Investigation, Conceptualization, Writing – review & editing, Supervision, Methodology, Formal analysis.

Declaration of competing interest

The authors declare that they have no known competing personal relationships or financial interests that could have affected this study.

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. NKC is supported by an EPSRC-UKRI AI for Net Zero Grant: "Enabling CO2 Capture and Storage Projects Using AI", (Grant EP/Y006143/1). NKC is also supported by a City University of Hong Kong Start-up Grant, project number 7200809. We would like to thank D.E. Shaw Research for providing the Chignolin data.

Data availability

Data will be made available on request.

References

[1] A. Lasota, M.C. Mackey, Chaos, fractals, and noise: Stochastic aspects of dynamics, second ed., in: Applied Mathematical Sciences, vol. 97, Springer, New York, 1994.

[2] M. Dellnitz, O. Junge, On the approximation of complicated dynamical behavior, SIAM J. Numer. Anal. 36 (2) (1999) 491–515.

[3] I. Mezić, Spectral properties of dynamical systems, model reduction and decompositions, Nonlinear Dynam. 41 (1) (2005) 309–325, http://dx.doi.org/10.1007/s11071-005-2824-x.

[4] S. Klus, P. Koltai, C. Schütte, On the numerical approximation of the Perron–Frobenius and Koopman operator, J. Comput. Dyn. 3 (1) (2016) 51–79, http://dx.doi.org/10.3934/jcd.2016003.

[5] S. Klus, N. Djurdjevac Conrad, Dynamical systems and complex networks: A Koopman operator perspective, 2024, arXiv:2405.08940.
[6] S.M. Ulam, A Collection of Mathematical Problems, Interscience Publisher NY, 1960.
[7] G. Chen, T. Ueta, Chaos in Circuits and Systems, vol. 11, World Scientific, 2002.
[8] M.O. Williams, I.G. Kevrekidis, C.W. Rowley, A data-driven approximation of the Koopman operator: Extending dynamic mode decomposition, J. Nonlinear Sci. 25 (2015) 1307–1346, http://dx.doi.org/10.1007/s00332-015-9258-5.
[9] M.O. Williams, C.W. Rowley, I.G. Kevrekidis, A kernel-based method for data-driven Koopman spectral analysis, J. Comput. Dyn. 2 (2) (2015) 247–265, http://dx.doi.org/10.3934/jcd.2015005.
[10] S. Klus, I. Schuster, K. Muandet, Eigendecompositions of transfer operators in reproducing kernel Hilbert spaces, J. Nonlinear Sci. (2020) http://dx.doi.org/10.1007/s00332-019-09574-z.
[11] S. Klus, F. Nüske, S. Peitz, J.-H. Niemann, C. Clementi, C. Schütte, Data-driven approximation of the Koopman generator: Model reduction, system identification, and control, Phys. D: Nonlinear Phenom. 406 (2020) 132416, http://dx.doi.org/10.1016/j.physd.2020.132416.
[12] S.L. Brunton, J.L. Proctor, J.N. Kutz, Discovering governing equations from data by sparse identification of nonlinear dynamical systems, Proc. Natl. Acad. Sci. 113 (15) (2016) 3932–3937, http://dx.doi.org/10.1073/pnas.1517384113.
[13] A. Mauroy, J. Goncalves, Linear identification of nonlinear systems: A lifting technique based on the Koopman operator, in: 2016 IEEE 55th Conference on Decision and Control, CDC, 2016, pp. 6500–6505, http://dx.doi.org/10.1109/CDC.2016.7799269.
[14] S.H. Rudy, S.L. Brunton, J.L. Proctor, J.N. Kutz, Data-driven discovery of partial differential equations, Sci. Adv. 3 (4) (2017) e1602614, http://dx.doi.org/10.1126/sciadv.1602614.
[15] A. Mardt, L. Pasquali, H. Wu, F. Noé, VAMPnets for deep learning of molecular kinetics, Nat. Commun. 9 (1) (2018) 5, http://dx.doi.org/10.1038/s41467-017-02388-1.
[16] M. Gulina, A. Mauroy, Two methods to approximate the Koopman operator with a reservoir computer, Chaos: An Interdiscip. J. Nonlinear Sci. 31 (2) (2021) http://dx.doi.org/10.1063/5.0026380.
[17] Q. Li, F. Dietrich, E.M. Bollt, I.G. Kevrekidis, Extended dynamic mode decomposition with dictionary learning: A data-driven adaptive spectral decomposition of the Koopman operator, Chaos: An Interdiscip. J. Nonlinear Sci. 27 (10) (2017) http://dx.doi.org/10.1063/1.4993854.
[18] Y. Jin, L. Hou, S. Zhong, Extended dynamic mode decomposition with invertible dictionary learning, Neural Netw. 173 (2024) 106177, http://dx.doi.org/10.1016/j.neunet.2024.106177.
[19] E. Yeung, S. Kundu, N. Hodas, Learning deep neural network representations for Koopman operators of nonlinear dynamical systems, in: 2019 American Control Conference, ACC, 2019, pp. 4832–4839, http://dx.doi.org/10.23919/ACC.2019.8815339.
[20] D.P. Kingma, J. Ba, Adam: A method for stochastic optimization, 2014, http://dx.doi.org/10.48550/arXiv.1412.6980, arXiv preprint arXiv:1412.6980.
[21] J. Lu, Gradient descent, stochastic optimization, and other tales, 2022, http://dx.doi.org/10.48550/arXiv.2205.00832, arXiv preprint arXiv:2205.00832.
[22] Y. Nesterov, et al., Lectures on Convex Optimization, vol. 137, Springer Cham, 2018, http://dx.doi.org/10.1007/978-3-319-91578-4.
[23] Z. Liu, G. Ding, L. Chen, E. Yeung, Towards scalable Koopman operator learning: Convergence rates and a distributed learning algorithm, in: 2020 American Control Conference, ACC, IEEE, 2020, pp. 3983–3990, http://dx.doi.org/10.23919/ACC45564.2020.9147858.
[24] B.J. Hollingsworth, Stochastic Differential Equations: A Dynamical Systems Approach, Auburn University, 2008, URL http://hdl.handle.net/10415/5.
[25] L. Boninsegna, F. Nüske, C. Clementi, Sparse learning of stochastic dynamical equations, J. Chem. Phys. 148 (24) (2018) 241723, http://dx.doi.org/10.1063/1.5018409.
[26] H. Robbins, S. Monro, A stochastic approximation method, Ann. Math. Stat. (1951) 400–407, http://dx.doi.org/10.1214/aoms/1177729586.
[27] Q. Tran-Dinh, M. van Dijk, Gradient descent-type methods: Background and simple unified convergence analysis, in: Federated Learning, Elsevier, 2024, pp. 3–28, http://dx.doi.org/10.1016/B978-0-44-319037-7.00008-9.
[28] Y. Nesterov, A method for solving the convex programming problem with convergence rate O (1/k2), Dokl Akad Nauk. Sssr 269 (1983).
[29] A. Défossez, L. Bottou, F. Bach, N. Usunier, A simple convergence proof of Adam and Adagrad, Trans. Mach. Learn. Res. (2022).
[30] X. Chen, S. Liu, R. Sun, M. Hong, On the convergence of a class of Adam-type algorithms for non-convex optimization, in: International Conference on Learning Representations, 2019, URL https://openreview.net/forum?id=H1x-x309tm.
[31] H. Wu, F. Noé, Variational approach for learning Markov processes from time series data, J. Nonlinear Sci. 30 (1) (2020) 23–66, http://dx.doi.org/10.1007/s00332-019-09567-y.
[32] K.B. Petersen, et al., The matrix cookbook, Tech. Univ. Den. 7 (15) (2012) 510, URL http://www2.imm.dtu.dk/pubdb/p.php?3274.
[33] J. Bradbury, R. Frostig, P. Hawkins, M.J. Johnson, C. Leary, D. Maclaurin, G. Necula, A. Paszke, J. VanderPlas, S. Wanderman-Milne, Q. Zhang, JAX: composable transformations of python+numpy programs, 2018, URL http://github.com/google/jax.
[34] K.A. Dill, S.B. Ozkan, M.S. Shell, T.R. Weikl, The protein folding problem, Annu. Rev. Biophys. 37 (2008) 289–316, http://dx.doi.org/10.1146/annurev.biophys.37.092707.153558.
[35] K. Lindorff-Larsen, S. Piana, R.O. Dror, D.E. Shaw, How fast-folding proteins fold, Science 334 (6055) (2011) 517–520, http://dx.doi.org/10.1126/science.1208351.
[36] C. Schütte, M. Sarich, Metastability and Markov State Models in Molecular Dynamics, vol. 24, American Mathematical Soc., 2013, http://dx.doi.org/10.1090/cln/024.
[37] J. Mishra, Modified Chua chaotic attractor with differential operators with non-singular kernels, Chaos Solitons Fractals 125 (2019) 64–72, http://dx.doi.org/10.1016/j.chaos.2019.05.013.
[38] R.N. Madan, Chua's Circuit: a Paradigm for Chaos, vol. 1, World Scientific, 1993, http://dx.doi.org/10.1142/1997.
[39] R. Kiliç, A Practical Guide for Studying Chua's Circuits, vol. 71, World Scientific, 2010, http://dx.doi.org/10.1142/7538.
[40] S.M. Filipov, I. Faragó, Implicit Euler time discretization and FDM with Newton method in nonlinear heat transfer modeling, 2018, http://dx.doi.org/10.48550/arXiv.1811.06337, arXiv preprint arXiv:1811.06337.