

# Two methods to approximate the Koopman operator with a reservoir computer <sup>EP</sup>

Cite as: Chaos **31**, 023116 (2021); <https://doi.org/10.1063/5.0026380>

Submitted: 25 August 2020 • Accepted: 20 January 2021 • Published Online: 09 February 2021

 Marvyn Gulina and  Alexandre Mauroy

## COLLECTIONS

 This paper was selected as an Editor's Pick



View Online



Export Citation



CrossMark

## ARTICLES YOU MAY BE INTERESTED IN

On explaining the surprising success of reservoir computing forecaster of chaos? The universal machine learning dynamical system with contrast to VAR and DMD

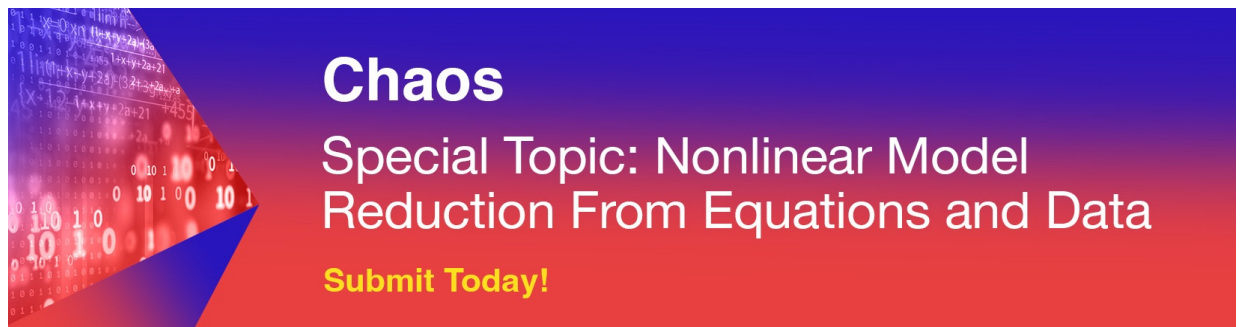
Chaos: An Interdisciplinary Journal of Nonlinear Science **31**, 013108 (2021); <https://doi.org/10.1063/5.0024890>

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**, 103111 (2017); <https://doi.org/10.1063/1.4993854>

Koopman operator and its approximations for systems with symmetries

Chaos: An Interdisciplinary Journal of Nonlinear Science **29**, 093128 (2019); <https://doi.org/10.1063/1.5099091>



**Chaos**  
Special Topic: Nonlinear Model  
Reduction From Equations and Data  
**Submit Today!**

# Two methods to approximate the Koopman operator with a reservoir computer

Cite as: Chaos 31, 023116 (2021); doi: 10.1063/5.0026380

Submitted: 25 August 2020 · Accepted: 20 January 2021 ·

Published Online: 9 February 2021



View Online



Export Citation



CrossMark

Marvyn Culina<sup>a)</sup>  and Alexandre Mauroy<sup>b)</sup> 

## AFFILIATIONS

Department of Mathematics and Namur Institute for Complex Systems (naXys), University of Namur, 5000 Namur, Belgium

<sup>a)</sup> Author to whom correspondence should be addressed: [marvyn.gulina@unamur.be](mailto:marvyn.gulina@unamur.be)

<sup>b)</sup> Electronic mail: [alexandre.mauroy@unamur.be](mailto:alexandre.mauroy@unamur.be)

## ABSTRACT

The Koopman operator provides a powerful framework for data-driven analysis of dynamical systems. In the last few years, a wealth of numerical methods providing finite-dimensional approximations of the operator have been proposed [e.g., extended dynamic mode decomposition (EDMD) and its variants]. While convergence results for EDMD require an infinite number of dictionary elements, recent studies have shown that only a few dictionary elements can yield an efficient approximation of the Koopman operator, provided that they are well-chosen through a proper training process. However, this training process typically relies on nonlinear optimization techniques. In this paper, we propose two novel methods based on a reservoir computer to train the dictionary. These methods rely solely on linear convex optimization. We illustrate the efficiency of the method with several numerical examples in the context of data reconstruction, prediction, and computation of the Koopman operator spectrum. These results pave the way for the use of the reservoir computer in the Koopman operator framework.

Published under license by AIP Publishing. <https://doi.org/10.1063/5.0026380>

The Koopman operator offers the possibility to turn nonlinear dynamical systems into linear ones. In this framework, dynamical systems can be studied with systematic linear techniques, and, in particular, they are amenable to spectral analysis. However, there is a price to pay. The Koopman operator is infinite-dimensional and must be approximated by a finite-rank operator (i.e., a matrix) as soon as numerical methods come into play. This approximation requires to choose a finite-dimensional subspace, a choice which is not necessarily appropriate since it is made *a priori*. Recent methods have been proposed using neural networks to “learn” the best finite-dimensional approximation subspace. The main drawback of these methods is that they rely on nonlinear optimization. In this paper, we propose to obtain a finite-dimensional approximation of the Koopman operator by using a reservoir computer. The reservoir computer is a specific recurrent neural network where only the weights of the nodes on the output layer are trained with the data, a training that can be performed with linear, convex optimization. Considering either the internal nodes or the output nodes of the reservoir computer to obtain the finite-dimensional approximation subspace, we derive two novel methods that compute a finite-dimensional approximation of the Koopman operator.

## I. INTRODUCTION

Dynamical system theory plays an important role in the context of data analysis. In fact, time-series can often be assumed to be generated by an underlying dynamical system and to be related to the system orbits through a given observation map. In contrast to this classical description, there exists an alternative description in terms of the observation maps themselves, also called *observables*. The dynamics, and, in particular, the time evolution of the observables, are then described through the so-called Koopman operator, which is a linear (but infinite-dimensional) operator. In this linear setting, it is natural to study the spectral properties of the operator and relate them to the dynamics of the nonlinear system.<sup>28</sup> The related notion of Koopman modes decomposition is also useful to study the systems in many contexts (e.g., fluids dynamics,<sup>32</sup> power grids,<sup>37</sup> epidemiology,<sup>31</sup> control<sup>27</sup>).

A noticeable fact is that the Koopman operator description is conducive to data analysis. In particular, there exist numerical techniques that can be used to compute a finite-dimensional approximation of the Koopman operator from data. Combined with the spectral analysis relying on Koopman modes, these data-driven techniques lead to the so-called (extended) dynamic mode decomposition [(E)DMD] method.<sup>6,9,33,42</sup> In practice, EDMD techniques

require choosing a specific approximation subspace, or equivalently a finite set of *dictionary functions*. This choice is crucial but has to be made *a priori* and is, therefore, not necessarily relevant to provide the best approximation of the operator.

Recently, *Dictionary learning* methods based on neural networks have been proposed to provide a relevant set of dictionary functions that are trained with the data and yield appropriate finite-dimensional representations of the Koopman operator.<sup>18,40</sup> Subsequent developments have also been made in the context of deep learning.<sup>5,21,29,43</sup> All these learning methods showed good performances, thereby demonstrating the effectiveness of dictionary learning in EDMD methods. However, these techniques require nonlinear optimization of the neural networks, while the classical EDMD method merely relies on linear optimization (i.e., linear least squares regression).

In this work, we propose a novel dictionary learning method for EDMD, which relies solely on linear optimization. Our key idea is to combine the EDMD method with a reservoir computer.<sup>11</sup> In more general contexts, reservoir computing competes with other algorithms on hard tasks such as channel equalization,<sup>34</sup> phoneme recognition,<sup>41</sup> and prediction,<sup>1,30</sup> among others (see the survey<sup>19,20</sup>). To our knowledge, we propose the first use of a reservoir computer for dictionary learning in the Koopman operator framework. Note that, very recently, the work<sup>3</sup> has emphasized a connection between the Koopman operator and the reservoir computer, though in a slightly different setting where the reservoir activation function is linear. This reinforces our claim that reservoir computing is relevant in the context of the Koopman operator. Interestingly, the recurrent neural network characterizing the reservoir allows training the dictionary with a dynamical network rather than with a static one. Hence, generated dictionary functions are nonlinear functions of time-delay coordinates, which are particularly relevant for time-delay systems and also bear some similarity to previous Koopman mode decomposition methods based on delayed coordinates (e.g., prony method,<sup>39</sup> Hankel DMD<sup>2</sup>). We derive two numerical schemes, where the dictionary functions are, respectively, the internal states and the outputs of the reservoir computer. We illustrate these two methods with several examples in the context of data reconstruction, prediction, and computation of the Koopman operator spectrum.

The rest of the paper is organized as follows. Section II provides an introduction to the Koopman operator framework, the EDMD method, and the reservoir computer. Section III presents our two methods obtained by combining the EDMD method with the reservoir computer. These two methods are illustrated in Sec. IV with numerical examples. Finally, concluding remarks are given in Sec. V.

## II. PRELIMINARIES

### A. The Koopman operator framework

The Koopman operator provides an alternative framework to describe the evolution of nonlinear systems in a purely linear fashion. Consider an autonomous dynamical system

$$\mathbf{x}(t+1) = F(\mathbf{x}(t)), \quad \mathbf{x} \in \mathcal{X}, \quad (1)$$

where  $\mathcal{X}$  is (an invariant subset of) the state space and  $F: \mathcal{X} \rightarrow \mathcal{X}$  is a nonlinear map. The Koopman operator is defined as the

composition<sup>16</sup>

$$\mathcal{K}f = f \circ F, \quad (2)$$

where  $f: \mathcal{X} \rightarrow \mathbb{C}$  is an *observable* that belongs to some function space  $\mathcal{F}$ . In the following, we will assume that  $\mathcal{F} = L^2(\mathcal{X})$  and that  $\mathcal{X}$  is a compact set. It is clear from (2) that the Koopman operator is linear. Also, while (1) describes the nonlinear dynamics of the state in the space  $\mathcal{X}$ , (2) equivalently describes the linear dynamics of the observables in  $\mathcal{F}$ . Roughly speaking, the system described in the space  $\mathcal{X}$  is *lifted* into the space  $\mathcal{F}$  when it is described in the Koopman framework.

The Koopman operator description can be used for several purposes. For instance, it can be used for prediction. Indeed, the (vector-valued) identity function  $\mathbf{g}(\mathbf{x}) = \mathbf{x}$ , also called projection maps, is characterized by the linear evolution

$$\mathbf{x}(t+1) = \mathcal{K}\mathbf{g}(\mathbf{x}(t)). \quad (3)$$

Provided that the Koopman operator associated with the system is known, (3) allows predicting future trajectories. Moreover, the spectral properties of the Koopman operator—namely, the eigenvalues  $\lambda \in \mathbb{C}$  and the associated eigenfunctions  $\phi \in \mathcal{F}$  satisfying  $\mathcal{K}\phi = \lambda\phi$ —provide meaningful information on the underlying dynamical system.<sup>25,28</sup> In particular, the eigenvalues are related to internal frequencies of the dynamics and the eigenfunctions reveal geometric properties in the state space. These spectral properties can also be used for control,<sup>10,13,17</sup> stability analysis,<sup>26,38</sup> time-series classification,<sup>36</sup> analysis and training of neural networks,<sup>5,23</sup> and network identification,<sup>24</sup> to list a few.

### B. Finite-dimensional approximation of the Koopman operator

Since the Koopman operator is infinite-dimensional, it is natural and often necessary to compute a finite-dimensional approximation. This approximation is given by the so-called *Koopman matrix*  $\mathbf{K}$ , which represents the projection  $\mathcal{P}$  of the operator onto a subspace  $\mathcal{F}_D$  spanned by the basis functions  $\psi_k \in \mathcal{F}$ ,  $k = 1, \dots, D$ , also called *dictionary*. More precisely, the  $i$ th row of  $\mathbf{K}$  is the coordinate vector of  $\mathcal{P}\mathcal{K}\psi_i$  in the dictionary. If one denotes  $\Psi(\mathbf{x}) = (\psi_1(\mathbf{x}), \dots, \psi_D(\mathbf{x}))^T$  and  $\mathcal{K}\Psi(\mathbf{x}) = (\mathcal{K}\psi_1(\mathbf{x}), \dots, \mathcal{K}\psi_D(\mathbf{x}))^T$ , one has

$$\mathcal{K}\Psi(\mathbf{x}) \approx \mathcal{P}\mathcal{K}\Psi(\mathbf{x}) = \mathbf{K}\Psi(\mathbf{x}),$$

so that one can obtain an approximation of the evolution of the dictionary functions under the action of the Koopman operator. In particular, if the identity belongs to the dictionary, it follows from (3) that an approximation of the system trajectories can be computed.

The finite-dimensional approximation of the Koopman operator can be obtained from data through the so-called extended dynamic mode decomposition (EDMD) method.<sup>42</sup> Given a set of snapshot pairs  $\{(\mathbf{x}_i, \mathbf{x}'_i = F(\mathbf{x}_i))\}_{i=1}^T$ , the Koopman matrix is given by

$$\mathbf{K} = \arg \min_{\tilde{\mathbf{K}} \in \mathbb{R}^{D \times D}} \sum_{i=1}^T \|\Psi(\mathbf{x}'_i) - \tilde{\mathbf{K}}\Psi(\mathbf{x}_i)\|^2 = \Psi' \Psi^+, \quad (4)$$

where  $^+$  denotes the pseudo-inverse,  $\|\cdot\|$  denotes the Euclidean norm, and where  $\Psi$  and  $\Psi' \in \mathbb{R}^{D \times T}$  denote the matrices whose

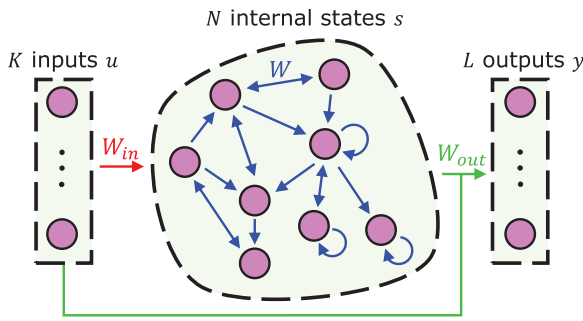


FIG. 1. Schematic of the reservoir computer (see the description in Sec. II C).

columns are  $\Psi(x_t)$  and  $\Psi(x'_t)$ , respectively, for  $t \in \{1, \dots, T\}$ . The Koopman matrix is the solution to a least squares problem and, therefore, represents the approximation of the operator obtained with a discrete orthogonal projection. Note that the specific dictionary  $\Psi(x) = x$  leads to the classical Dynamic Mode Decomposition (DMD) algorithm.<sup>9,32,33</sup> This statement justifies the term “extended” introduced above.

The finite-dimensional approximation of the operator depends on both the projection and the dictionary of basis functions. In a data-driven context, the discrete orthogonal projection used in the EDMD method is a natural and appropriate projection to use. However, the choice of the dictionary is somehow arbitrary but crucial since it affects the quality of the approximation. The original EDMD method<sup>42</sup> relies on a dictionary that is fixed and chosen *a priori* (e.g., polynomial functions, radial basis functions (RBF)). Recently, machine learning techniques have been used to guide the choice of the dictionary.<sup>18</sup> Building on this result, we propose to select the dictionary functions through a reservoir computer.

### C. Reservoir computer

The reservoir computer is a discrete-time neural network that consists of three layers: the inputs, the reservoir, and the outputs (Fig. 1). We denote the input signals by  $\mathbf{u} \in \mathbb{R}^K$ , the reservoir states by  $\mathbf{s} \in \mathbb{R}^N$ , and the output signals by  $\mathbf{y} \in \mathbb{R}^L$ . The reservoir states are updated according to the dynamics<sup>11</sup>

$$\mathbf{s}(t+1) = (1 - Ca)\mathbf{s}(t) + C\sigma[\mathbf{W}_{in}\mathbf{u}(t+1) + \mathbf{W}\mathbf{s}(t) + \mathbf{v}(t)], \quad (5)$$

where  $C$  is a timescale constant,  $\sigma$  is the activation function,  $a$  is the leaking rate,  $\mathbf{W} \in \mathbb{R}^{N \times N}$  and  $\mathbf{W}_{in} \in \mathbb{R}^{N \times K}$  are the matrices of internal connection weights and input weights, respectively, and  $\mathbf{v}$  is a noise term. The matrix  $\mathbf{W}$  is typically sparse and its density (i.e., the proportion of non-zero elements) is denoted by  $\gamma$ . The nonzero entries of  $\mathbf{W}_{in}$  and  $\mathbf{W}$  are uniformly randomly distributed over  $[-1, 1]$  and  $[-w, w]$ , respectively. The components of  $\mathbf{v}$  are uniformly distributed over  $[-\varepsilon, \varepsilon]$ . This noise term is added according to the work<sup>11</sup> as an alternative to Tikhonov regularization for the output weights training.

The reservoir can contain loops and is, therefore, a *recurrent neural network*. Furthermore, the gain  $w$  is chosen such that the

spectral radius  $\rho$  of  $\mathbf{W}$  satisfies the echo state property<sup>11,12</sup>

$$|1 - C(a - \rho)| < 1, \quad (6)$$

so that the reservoir “forgets” the initial condition  $\mathbf{s}(0)$ , which is uniformly distributed over  $[0, 1]$ . Hence the computer reservoir is an *echo state network*. In practice, we can discard the first (transient) states corresponding to the initialization of the reservoir.

Finally, the outputs are given by

$$\mathbf{y}(t) = \mathbf{W}_{out}\tilde{\mathbf{s}}(t), \quad (7)$$

where  $\tilde{\mathbf{s}}(t) = [\mathbf{s}(t); \mathbf{u}(t)] \in \mathbb{R}^{\tilde{N}}$  (with  $\tilde{N} = N + K$ ) is the vertical concatenation of internal states and inputs and where  $\mathbf{W}_{out} \in \mathbb{R}^{L \times \tilde{N}}$  is the matrix of output weights. It is noticeable that the outputs are obtained through *linear* combinations of the states and that *only* the output weights are trained. This is a computational advantage of the reservoir computer that we will leverage.

## III. NUMERICAL METHODS

In this section, we present our two methods, which combine the EDMD technique with the reservoir computer. The key idea is to use (linear combinations of) the internal states of the reservoir as dictionary functions. The first method uses all the states of the reservoir, while the second method selects a subset of these states.

### A. Method 1: EDMD using a reservoir computer

A straightforward method consists in using all the reservoir internal states as dictionary functions, i.e.,  $\Psi = \tilde{\mathbf{s}}$ . In this case, the optimization problem (4) becomes

$$\mathbf{K} = \arg \min_{\tilde{\mathbf{K}} \in \mathbb{R}^{N \times N}} \sum_{t=1}^{T-1} \|\tilde{\mathbf{s}}(t+1) - \tilde{\mathbf{K}}\tilde{\mathbf{s}}(t)\|^2 \quad (8)$$

and its solution is given by

$$\mathbf{K} = \mathbf{S}'\mathbf{S}^+, \quad (9)$$

where  $\mathbf{S}$  and  $\mathbf{S}' \in \mathbb{R}^{\tilde{N} \times (T-1)}$  denote the matrices whose columns are  $\tilde{\mathbf{s}}(t)$  for  $t \in \{1, \dots, T-1\}$  and  $t \in \{2, \dots, T\}$ , respectively. The internal states evolve according to the dynamics (5), where the input  $\mathbf{u}(t)$  is a trajectory  $\mathbf{x}(t)$  (equivalently,  $\tilde{\mathbf{s}} = [\mathbf{s}; \mathbf{x}]$ ). For this reason, the data points are generated from a single trajectory and not from a set of scattered data pairs.

**Remark 1.** The proposed dictionary can be interpreted as *non-linear functions* (resulting from several compositions of the activation function  $\sigma$ ) of *time-delay coordinates*. Indeed, since the internal states are solutions to (5), they depend on past values of the input  $\mathbf{u}(t)$ , or equivalently of the state  $\mathbf{x}(t)$ . Moreover, provided that the reservoir satisfies the echo state property and is initialized for a sufficiently long time before generating the data, the internal states do not depend on the (random) initial condition and are time-independent. Their evolution corresponds to the evolution of particular observables under the action of the Koopman operator associated with an extended dynamical system with time-delayed coordinates. The use of time-delay coordinates to construct the basis functions is reminiscent of previous works in the context of the Koopman operator.<sup>2,15,39</sup>

The proposed method is summarized in Algorithm 1.

---

**Algorithm 1** EDMD using a reservoir computer
 

---

**Input:** Sampled trajectory  $\mathbf{x}(t)$ ,  $t = 1, \dots, T$ ; parameters  $N, L, C, a, \gamma, \rho, w_{in}, \varepsilon$ .

**Output:** Koopman matrix  $K$ .

- 1: Initialize  $\mathbf{W}_{in}$ ,  $\mathbf{W}$  (see Sec. II C and set random initial reservoir states).
  - 2: Update the reservoir states according to (5) with the input  $\mathbf{u}(t) = \mathbf{x}(t)$ , for  $t = 1, \dots, T - 1$ .
  - 3: Compute the Koopman matrix  $K = \mathbf{S}'\mathbf{S}^+$  (9) (possibly discarding the first values of the states  $\tilde{\mathbf{s}}$ ).
- 

**B. Method 2: Dictionary learning for EDMD using a reservoir computer**

The method presented above yields a  $\tilde{N} \times \tilde{N}$  Koopman matrix, which is not so convenient since the number of internal states is typically large. In order to reduce the size of the Koopman matrix, we propose to define the dictionary functions as the output (7) of the reservoir, that is, we select  $L \ll N$  linear combinations of the states  $\tilde{\mathbf{s}} = [\mathbf{s}; \mathbf{x}]$  obtained through a dictionary learning step. The optimization problem is written as

$$\min_{\mathbf{W}_{out}, \mathbf{K}} \sum_{t=1}^{T-1} \|\mathbf{W}_{out} \tilde{\mathbf{s}}(t+1) - \mathbf{K} \mathbf{W}_{out} \tilde{\mathbf{s}}(t)\|^2. \quad (10)$$

It is clear that  $\mathbf{W}_{out} = \mathbf{0}$  is a trivial solution to the optimization problem. We, therefore, add the projection maps  $\mathbf{x}$  to the outputs, considering the augmented output weights matrix  $\mathbf{W}_{out} = [\mathbf{W}_1; \mathbf{W}_2] \in \mathbb{R}^{(L+K) \times \tilde{N}}$  with the output weights  $\mathbf{W}_1 \in \mathbb{R}^{L \times \tilde{N}}$  related to the main dictionary functions and the output weights  $\mathbf{W}_2 \in \mathbb{R}^{K \times \tilde{N}}$  related to the projection maps  $\mathbf{x}$ . The dictionary size is given by  $D = L + K$ . Since  $\mathbf{u}(t) = \mathbf{x}(t)$ , we have  $\mathbf{W}_2[\mathbf{s}; \mathbf{u}] = \mathbf{u}$ , so that the constraint  $\mathbf{W}_2 = [\mathbf{0}_{K,N}, \mathbf{I}_K]$  (where  $\mathbf{0}_{K,N}$  is the  $K \times N$  zero matrix and  $\mathbf{I}_K$  is the  $K \times K$  identity matrix) prevents the trivial zero solution. In this case, we only need to optimize over the weights  $\mathbf{W}_1$ .

This method presented can be interpreted as an intermediate method between the first method developed in Sec. III A and the DMD method. If we set  $L = N$ , the dictionary contains the maximal number of independent functions constructed with the internal states of the reservoir. These functions can be chosen as the internal states themselves and we recover the first method. In contrast, in the case  $L = 0$ , there is no optimization performed on the output weights and the optimization on  $\mathbf{K}$  associated with linear basis functions is equivalent to DMD. The proposed method can, therefore, be seen as a trade-off where an optimal subset of basis functions is obtained through the training process.

Similarly to Ref. 18, (10) can be solved with two alternating steps:

1. (Computation of the Koopman matrix) fix  $\mathbf{W}_1$  and optimize  $\mathbf{K}$  and
2. (Dictionary learning) fix  $\mathbf{K}$  and optimize  $\mathbf{W}_1$ .

A key advantage is that *both* optimization steps rely on *linear* optimization. It is clear that step 1 is a least squares problem, whose

solution is given by

$$\mathbf{K} = \mathbf{W}_{out} \mathbf{S}' (\mathbf{W}_{out} \mathbf{S})^+ = \mathbf{W}_{out} \mathbf{S}' \mathbf{S}^+ \mathbf{W}_{out}^+. \quad (11)$$

In fact, this is the standard EDMD problem  $\mathbf{K} = \Psi' \Psi^+$  in the case of our specific choice of dictionary functions  $\Psi = \mathbf{W}_{out} \mathbf{S}$ . We can also note that the matrix  $\mathbf{S}' \mathbf{S}^+$  in (11) is the Koopman matrix computed in the first method. The optimization over the output weight matrix  $\mathbf{W}_{out}$  somehow acts as a coupling in the optimization problem, where the columns of the Koopman matrix are optimized simultaneously in order to minimize the overall residue in (10).

Step 2 can also be cast into a least squares problem. Denoting

$$\mathbf{K} = \begin{pmatrix} \mathbf{K}_{11} & \mathbf{K}_{12} \\ \mathbf{K}_{21} & \mathbf{K}_{22} \end{pmatrix},$$

we can write problem (10) as the equation

$$\begin{aligned} \mathbf{W} \mathbf{S}' - \mathbf{K} \mathbf{W} \mathbf{S} &= \begin{pmatrix} \mathbf{W}_1 \mathbf{S}' \\ \mathbf{W}_2 \mathbf{S}' \end{pmatrix} - \begin{pmatrix} \mathbf{K}_{11} \mathbf{W}_1 \mathbf{S} + \mathbf{K}_{12} \mathbf{W}_2 \mathbf{S} \\ \mathbf{K}_{21} \mathbf{W}_1 \mathbf{S} + \mathbf{K}_{22} \mathbf{W}_2 \mathbf{S} \end{pmatrix} \\ &= \begin{pmatrix} \mathbf{W}_1 \mathbf{S}' \\ \mathbf{0}_{K,T-1} \end{pmatrix} - \begin{pmatrix} \mathbf{K}_{11} \\ \mathbf{K}_{21} \end{pmatrix} \mathbf{W}_1 \mathbf{S} - \begin{pmatrix} \mathbf{K}_{12} \mathbf{W}_2 \mathbf{S} \\ -\mathbf{W}_2 \mathbf{S}' + \mathbf{K}_{22} \mathbf{W}_2 \mathbf{S} \end{pmatrix} \\ &= \mathbf{0}_{D,T-1}. \end{aligned} \quad (12)$$

Denoting the independent terms  $\mathbf{C}_1 = \mathbf{K}_{12} \mathbf{W}_2 \mathbf{S}$  and  $\mathbf{C}_2 = -\mathbf{W}_2 \mathbf{S}' + \mathbf{K}_{22} \mathbf{W}_2 \mathbf{S}$ , we compute the optimal output weights  $\mathbf{W}_1$  as the least squares solution given by

$$\mathbf{W}_1(\cdot) = \begin{bmatrix} (\mathbf{S}')^T \otimes \mathbf{I}_L - \mathbf{S}^T \otimes \mathbf{K}_{11} \\ -\mathbf{S}^T \otimes \mathbf{K}_{21} \end{bmatrix}^+ \begin{bmatrix} \mathbf{C}_1(\cdot) \\ \mathbf{C}_2(\cdot) \end{bmatrix}, \quad (13)$$

where  $(\cdot)$  denotes the vectorization of matrices and  $\otimes$  is the Kronecker product.

A variant of the above numerical scheme can also be obtained. Instead of computing the least squares solution (13) in terms of output weights  $\mathbf{W}_1$ , we could replace them by the reservoir outputs  $\Psi_1 = \mathbf{W}_1 \mathbf{S}$  in (12). Using  $\mathbf{W}_1 = \Psi_1 \mathbf{S}^+$ , we obtain the Sylvester equations

$$\begin{aligned} \Psi_1 \mathbf{S}^+ \mathbf{S}' - \mathbf{K}_{11} \Psi_1 &= \mathbf{C}_1, \\ -\mathbf{K}_{21} \Psi_1 &= \mathbf{C}_2, \end{aligned}$$

whose least squares solution is

$$\Psi_1(\cdot) = \begin{bmatrix} (\mathbf{S}^+ \mathbf{S}')^T \otimes \mathbf{I}_L - \mathbf{I}_{T-1} \otimes \mathbf{K}_{11} \\ -\mathbf{I}_{T-1} \otimes \mathbf{K}_{21} \end{bmatrix}^+ \begin{bmatrix} \mathbf{C}_1(\cdot) \\ \mathbf{C}_2(\cdot) \end{bmatrix}. \quad (14)$$

In the case  $\tilde{N} \geq T - 1$ , it is noticeable that  $\mathbf{S}^+ \mathbf{S}'$  is a companion matrix so that only the last column has to be effectively computed. When  $\tilde{N} > T - 1$ , optimizing over  $\Psi_1 = \mathbf{W}_1 \mathbf{S}$  in the variant of the method makes the optimization problem more constrained, since the basis functions are not described anymore as linear combinations of the state reservoirs but through their values at a smaller number  $T - 1 < \tilde{N}$  of sample points. Conversely, the case  $\tilde{N} < T - 1$  is a relaxation of the optimization problem, which is less computationally efficient since  $\mathbf{S}^+ \mathbf{S}'$  is not sparse in this case. The intermediate setting  $\tilde{N} = T - 1$ , where  $\mathbf{S}$  is a square matrix and  $\Psi_1 = \mathbf{W}_1 \mathbf{S}$ , can be interpreted as a change of variable. It appears as an appropriate choice, but the size of the reservoir becomes forced by the size of the dataset.



Both Eqs. (13) and (14) provide an effective way to solve the second step of the alternating optimization, yielding two variants of our proposed second method. The first variant provides the output weights describing the dictionary functions, while the second variant provides the updated values of the dictionary functions, which can directly be used to compute the Koopman matrix  $K$ . Both variants require to invert a matrix of  $D(T-1) \times \tilde{N}L$  and  $D(T-1) \times (T-1)L$ , respectively. This matrix is not sparse for the first variant, but is sparse for the second variant (provided that  $\tilde{N} \geq T-1$ ). It follows that the second variant is more efficient from a computational point of view.

The method with its two variants is summarized in Algorithm 2.

**Algorithm 2** Dictionary learning for EDMD using a reservoir computer

- Input:** Sampled trajectory  $\mathbf{x}(t)$ ,  $t = 1, \dots, T$ ; parameters  $N, L, C, a, \gamma, \rho, w_{in}, \varepsilon$ .
- Output:** Koopman matrix  $K$ , output weights  $W_1$ .
- 1: Initialize  $W_{in}$ ,  $W$  (see Sec. II C) and set random initial reservoir states and output weights  $W_1$ .
  - 2: Update the reservoir states according to (5) with the input  $\mathbf{u}(t) = \mathbf{x}(t)$ , for  $t = 1, \dots, T-1$ .
  - 3: Compute the dictionary values  $\Psi = [\Psi_1; \Psi_2]$  with  $\Psi_1 = W_1 S$  and  $\Psi_2 = [0_{K,N}, I_K] S$  (possibly discarding the first values of the states  $\bar{s}$ ).
  - 4: **loop**
  - 5: Solve step 1: compute the Koopman matrix  $K = W_{out} S' S^+$  ( $W_{out}^+$  (variant 1) or  $K = \Psi' \Psi^+$  (variant 2)).
  - 6: Solve step 2: update the values  $W_1$  using (13) (variant 1) or  $\Psi_1$  using (14) (variant 2).
  - 7: Stop if some criterion is satisfied (e.g., upper bound on the number of iterations, lower bound on the least squares error).
  - 8: If variant 2 is used: compute the output weights  $W_1 = \Psi_1 S^+$ .

## C. Application to prediction and spectral properties

### 1. Reconstruction and prediction

The Koopman matrix  $K$  provided by Algorithm 1 or 2 is optimized so that

$$\forall t \in \{1, \dots, T-1\} : \Psi(t+1) \approx K\Psi(t).$$

It follows that we can iterate the matrix to recompute known dictionary values  $\hat{\Psi}(t+1) = K^t \Psi(1)$  (reconstruction) or predict new values  $\hat{\Psi}(t+T) = K^t \Psi(T)$  from the last data point (prediction). In particular, predicted states are obtained by considering the values of the dictionary functions related to the projection maps.

It should be noted that our use of the reservoir computer differs from the classic use in the context of prediction and is not aimed at this specific prediction objective. Here, the outputs are not optimized so that they provide the best predictions of the state. Instead, they provide the basis functions that yield the most accurate approximation of the Koopman operator, which can, in turn, be used for prediction as a by-product. This observation is particularly relevant to Method 2 and will be discussed with more details in Sec. IV.

In order to test the quality of the Koopman matrix approximation, we will first compute the optimization residue

$$\sum_{t=1}^T \|\Psi(\mathbf{x}_t') - \tilde{K}\Psi(\mathbf{x}_t)\|^2. \quad (15)$$

If the values of the residue are small, the dictionary functions span a subspace that is almost invariant under the action of the Koopman operator.

We will also consider the reconstruction error

$$E(t) = K^t \Psi(1) - \Psi(t+1) \quad (16)$$

and, in particular, the Normalized Root Mean Square Error (NRMSE)

$$\text{NRMSE} = \frac{\sqrt{\sum_{t=1}^{T-1} E(t)^2}}{\sqrt{\sum_{t=1}^{T-1} \left[ \Psi(t+1) - \frac{1}{T-1} \sum_{\tau=1}^{T-1} \Psi(\tau+1) \right]^2}},$$

where the square operations, the square roots, and the quotient are considered element-wise.

The NRMSE value can be interpreted as follows:  $\text{NRMSE} = 0$  means that the two series perfectly match and  $\text{NRMSE} = 1$  is the error obtained when the reconstructed time series is a constant value equal to the mean value of the other one. Similarly, we will denote by *nrmse* the error restricted to the projection maps.

## 2. Spectral properties

The Koopman matrix  $K$  can be used to compute spectral properties of the Koopman operator  $\mathcal{K}$ .<sup>42</sup> In particular, the eigenvalues of  $K$  provide an approximation of the Koopman operator spectrum. Its left eigenvectors  $\mathbf{w}$  provide the expansion of Koopman eigenfunctions  $\phi$  in the basis given by the dictionary functions, i.e.,  $\phi \approx \mathbf{w}^T \Psi$ . Note also that the right eigenvectors are related to the Koopman modes.

## IV. RESULTS

In this section, we illustrate the performance of our methods with several datasets in the context of trajectory reconstruction, prediction, and computation of spectral properties.

### A. Datasets and parameters

#### 1. Dynamical systems and data generation

We consider several systems, including chaotic dynamics.

*a. Duffing system.* The dynamics

$$\begin{aligned} \dot{x}_1 &= x_2, \\ \dot{x}_2 &= -\gamma x_2 - (\alpha x_1^2 + \beta)x_1, \end{aligned} \quad (17)$$

with  $\alpha = 1$ ,  $\beta = -1$ , and  $\gamma = 0.5$  admit a stable equilibrium at the origin. We have generated  $T = 501$  data points over the time interval  $[0, 20]$  (time step  $h = 0.04$ ) for the initial condition  $\mathbf{x}(0) = (-1.21, 0.81)$ .

*b. Mackey–Glass system.* We consider the following delayed equation:<sup>22</sup>

$$\dot{x} = \frac{\alpha x_\tau}{(1 + x_\tau^n)} - \beta x, \quad (18)$$

where  $x_\tau = x(t - \tau)$  with  $\tau = 17$  and  $\alpha = 0.2$ ,  $\beta = 0.1$ , and  $n = 10$ . We have generated  $T = 501$  data points over the time interval  $[0, 500]$  (time step  $h = 1$ ) for the initial condition  $x(t < 0) = 0.1$ . Note that the system is integrated using the Matlab function `dde23`. The largest Lyapunov exponent is equal to  $\lambda_L \approx 0.005$  ( $1/\lambda_L \approx 190$ ) and the Kaplan–Yorke dimension of the attractor is  $D_{KY} \approx 2.135$ . See Ref. 8 for more details on the system.

*c. Rössler system.* We have used the chaotic dynamics

$$\begin{aligned} \dot{x}_1 &= -x_2 - x_3, \\ \dot{x}_2 &= x_1 + \alpha x_2, \\ \dot{x}_3 &= \beta + (x_1 - \gamma)x_3, \end{aligned} \quad (19)$$

with  $\alpha = 0.1$ ,  $\beta = 0.1$ , and  $\gamma = 14$  to generate  $T = 601$  data points over the time interval  $[0, 300]$  (time step  $h = 0.5$ ) for the initial condition  $\mathbf{x}(0) = (2, 1, 5)$ . The largest Lyapunov exponent is  $\lambda_L \approx 0.06$  ( $1/\lambda_L \approx 17$ ) and the Kaplan–Yorke dimension of the attractor is  $D_{KY} \approx 2.005$ . See, e.g., Ref. 35 for more details on the system.

*d. Lorenz-63 system.* Using the chaotic dynamics

$$\begin{aligned} \dot{x}_1 &= s(x_2 - x_1), \\ \dot{x}_2 &= rx_1 - x_2 - x_1x_3, \\ \dot{x}_3 &= x_1x_2 - bx_3, \end{aligned} \quad (20)$$

with  $s = 10$ ,  $r = 28$ , and  $b = 8/3$ , we have generated  $T = 751$  data points over the time interval  $[0, 15]$  (time step  $h = 0.02$ ) for the initial condition  $\mathbf{x}(0) = (3, 3, 19)$ . The largest Lyapunov exponent is  $\lambda_L \approx 0.906$  ( $1/\lambda_L \approx 1.10$ ) and the Kaplan–Yorke dimension of the attractor is  $D_{KY} \approx 2.06$ . See, e.g., Ref. 35 for more details on the system.

*e. Lorenz-96 system.* We finally consider the  $n$ -dimensional chaotic system

$$\dot{x}_i = (x_{i+1} - x_{i-2})x_{i-1} - x_i + F, \quad (21)$$

where  $x_{-1} = x_{n-1}$ ,  $x_0 = x_n$ , and  $x_{n+1} = x_1$  and where  $F$  is a forcing constant. For each value of  $n \in \{5, 50, 100, 150\}$  and for  $F = 8$ , we have generated  $T = 501$  data points over the time interval  $[0, 10]$  (time step  $h = 0.02$ ) with the initial condition  $\mathbf{x}(0) = (F + 0.1, F, \dots, F) \in \mathbb{R}^n$ . We consider that only the first three states are measured and used with the reservoir computer (i.e.,  $K = 3$ ). The largest Lyapunov exponent is approximately equal to  $\lambda_L \approx 0.5$  ( $1/\lambda_L \approx 2$ ) for  $n = 5$  and  $\lambda_L \approx 1.8$  ( $1/\lambda_L \approx 0.56$ ) for  $n = 50, 100, 150$ . The Kaplan–Yorke dimension of the attractor is estimated to  $D_{KY} \approx 2.9$  for  $n = 5$ ,  $D_{KY} \approx 33.5$  for  $n = 50$ ,  $D_{KY} \approx 65$

for  $n = 100$ , and  $D_{KY} \approx 98$  for  $n = 150$ . See, e.g., Ref. 14 for more details on the system.

**Remark 2.** The data are scaled through a linear transformation that maps the minimum and maximum values of each state to  $-1$  and  $1$ , respectively. The values of the initial conditions are given before rescaling.

## 2. Parameter values

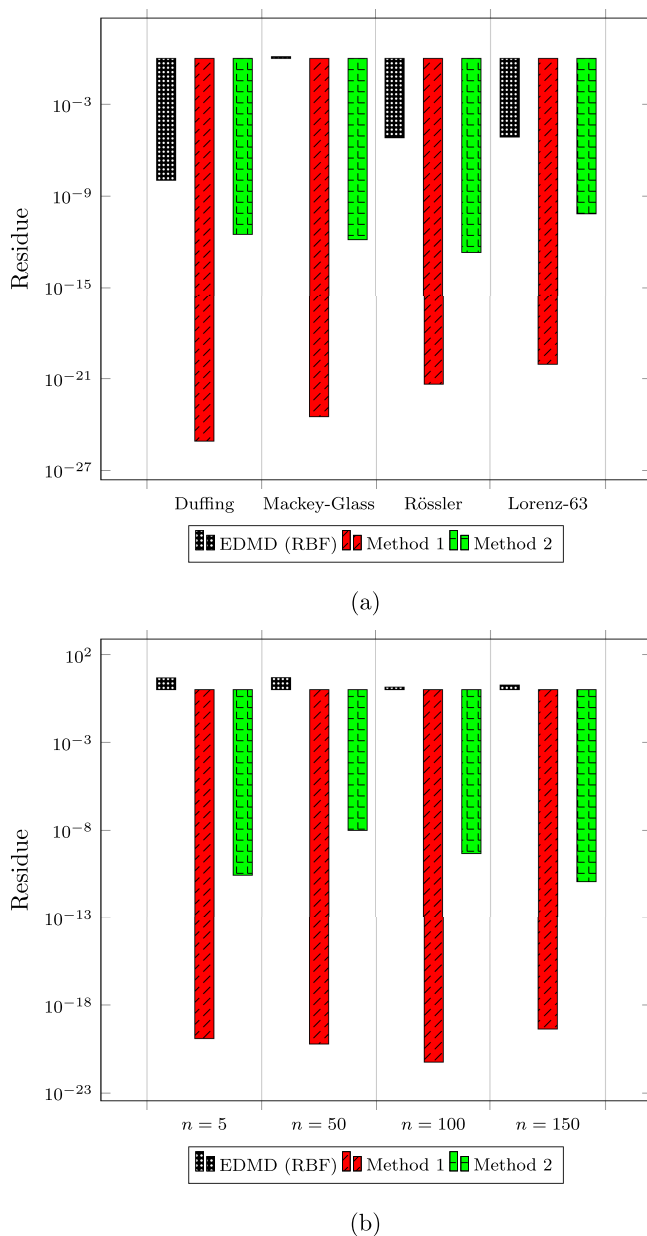
The number of basis functions is set to  $D = N + K = 1000$  and  $D = L + K = 15$  for the first and the second method, respectively. Note that the numbers  $N$  and  $L$  depend on the state dimension  $K$ . We note that  $\tilde{N} = 1000 > T - 1$  for all study cases. Although this choice yields a more constrained optimization problem (for the second variant of the second method, see Sec. III B), it yields the best results in terms of reconstruction, prediction and spectral properties while remaining computationally efficient.

For the second method, the second variant is used and the number of iterations is limited to 20. In the examples, the EDMD method is also used for comparison purpose with a dictionary of  $N$  Gaussian radial basis functions  $\psi_k(\mathbf{x}) = e^{-\eta \|\mathbf{x} - \hat{\mathbf{x}}_k\|^2}$ , where  $\hat{\mathbf{x}}_k$  is the center and  $\eta$  is a parameter. The number of basis functions is equal to the number  $N$  of reservoir states (with  $D = N + K = 1000$ ), except for the results shown in Fig. 3 where it is equal to  $L$  (with  $D = L + K = 15$ ). The value of the parameter  $\eta$  has been optimized by hand to allow a fair comparison to our methods (Duffing:  $\eta = 120$ ; Mackey–Glass:  $\eta = 10^{-2}$ ; Rössler:  $\eta = 120$ ; Lorenz-63:  $\eta = 10^{-3}$ ; Lorenz-96:  $\eta = 10^{-5}$ ).

In most cases, the reservoir parameters are kept constant for every system. The activation function  $\sigma$  is the hyperbolic tangent  $\sigma(x) = \tanh(x)$  for all systems except for the Lorenz-96 system where it is the ReLU function  $\sigma(x) = \max(0, x)$ . The spectral radius of the internal weights is set to  $\rho = 0.79$  for all cases. The leaking rate is set to  $a = 3$  for all systems except for the Mackey–Glass system where  $a = 1$  and for the Lorenz-96 system where  $a = 7$ . The noise level in the reservoir is  $\varepsilon = 10^{-4}$  except for the Lorenz-96 system where  $\varepsilon = 10^{-5}$ . The time constant is set to  $C = 0.45$  for the Duffing system and the Lorenz-96 and is set to  $C = 0.11$  for the Mackey–Glass system, the Rössler system, and the Lorenz-63 system.

## B. Reconstruction results

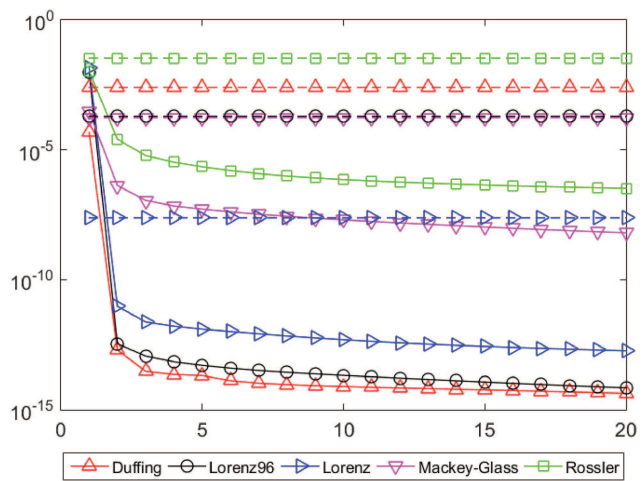
The Koopman matrix computed with the basis functions generated by the reservoir is efficient to reconstruct the trajectories. As shown in Fig. 2, small residues (15) are obtained with all the systems introduced above. Moreover, in each case, the proposed methods outperform the EDMD method using as many radial basis functions as there are internal states in the reservoir. We observe that the first method yields better results than the second method. This can be explained by the fact the second method uses a smaller number  $L < N$  of dictionary functions while the optimization problem (8) is underconstrained for the first method. We also note that the EDMD method is not able to reconstruct the trajectories of the Mackey–Glass system (residue larger than 1). This suggests that the classical EDMD method with Gaussian radial basis functions cannot capture time-delayed dynamics, in contrast to the reservoir whose



**FIG. 2.** The optimization residues of (4) are computed for several systems [Duffing, Mackey–Glass, Rössler, Lorenz-63 in panel (a) and Lorenz-96 with different values of the dimension  $n$  in panel (b)]. The results show that our methods 1 and 2 outperform EDMD. We also note that the values of the residues obtained with Method 1 can be interpreted as zero values, due to the fact that the least squares problem is underconstrained. (a) Duffing, Mackey–Glass, Rössler, and Lorenz-63 systems. (b) Lorenz-96 system.

internal states can be seen as functions depending on delayed input values (see Sec. III A).

The values of the residue also allow to compare different dictionaries as, for instance, they capture invariance properties of

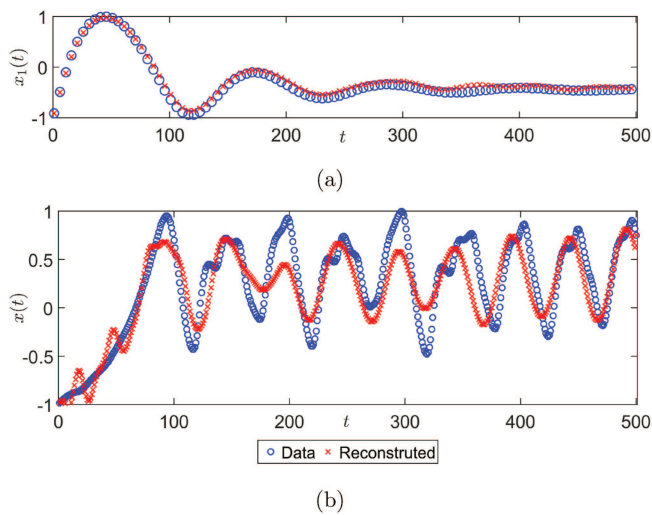


**FIG. 3.** The residues obtained with the second method (solid lines) rapidly decrease as the dictionary training proceeds. They are several orders of magnitude smaller than the residues obtained with the EDMD method (dashed lines) using the same number  $L$  of dictionary functions. Normalized residues are shown. They have been obtained by dividing (15) by the square of the Frobenius norm of  $\Psi$ .

the subspace spanned by the dictionary functions. For the first method, the residues obtained with the first method can be interpreted as zero values. But under the same conditions, the EDMD method yields nonzero values due to the fact that the basis functions (evaluated at the data points) are not linearly independent. If a large number of functions is used, the choice of a dictionary generated by the reservoir computer, therefore, appears to be more appropriate. For the second method, the evolution of the residue as the dictionary training proceeds is shown in Fig. 3. We observe a sharp decrease of several orders of magnitude during the first iterations so that the final values are much smaller than the residues obtained with the EDMD method using the same number of basis functions. However, we note that the residues obtained without training (i.e., iteration 1) are similar for both methods. In the case of a small dictionary with no training, the choice of Gaussian radial basis functions or functions generated by the reservoir computer seems equivalent.

The proposed methods provide a Koopman matrix that can be iterated to reconstruct the trajectories from the initial state. This is illustrated by the *nrmse* value shown in Table I. We note that a larger error is obtained with the second method for the Rössler, Lorenz-63, and Lorenz-96 systems, in which case reconstructed trajectories diverge after some time or converge to a constant. However, in all cases, the proposed methods outperform the EDMD method used with Gaussian radial basis functions. Finally, Fig. 4 illustrates the reconstruction performances of the second method for the Duffing system and the Mackey–Glass system. Note that the first method yields perfect reconstruction for each system (see Table I) and is, therefore, not shown in the figure.





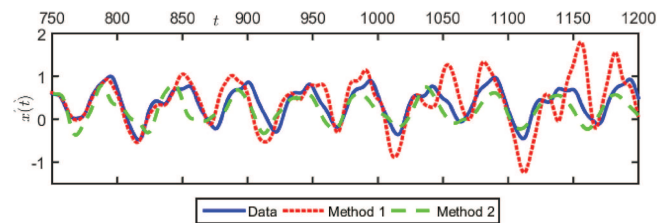
**FIG. 4.** The trajectories are reconstructed by iterating the Koopman matrix computed with the second method. Blue circles and red crosses denote the data and the reconstructed trajectory, respectively (note that all the data points are not shown so that the sampling period is smaller than it may seem on the figure). Panel (a) shows the first component of a (reconstructed) trajectory of the Duffing system. The second component is not shown but is similar. Panel (b) shows a (reconstructed) trajectory of the Mackey–Glass system.

### C. Prediction results

In this section, we briefly present prediction results for illustrative purposes. To do so, we consider the last data point of the training stage and iterate the Koopman matrix from this point. We focus on the Mackey–Glass system and on the Lorenz-96 system.

#### 1. Mackey–Glass system

Figure 5 shows that the proposed methods are also efficient to predict the trajectory of the chaotic Mackey–Glass dynamics. In the present setting, the first method provides more accurate results for short-term prediction, but the predicted trajectory eventually diverges. In contrast, the trajectory predicted with the second



**FIG. 5.** The two methods are efficient to predict the trajectory of the Mackey–Glass dynamics over some time horizon.

method converges to a constant that is close to the mean value of the data (see Fig. 6). This is due to the fact that the Koopman matrix does not have unstable eigenvalues and has only the eigenvalue  $\lambda = 1$  on the unit circle (see Sec. IV D below). It follows that the predicted trajectory converges to a steady state (associated with that eigenvalue  $\lambda = 1$ ) that corresponds to the average value of the identity observable on the attractor, computed with respect to the stationary density. This value is also equal to the time average of the identity observable along a trajectory, which is close to the mean value of the data.

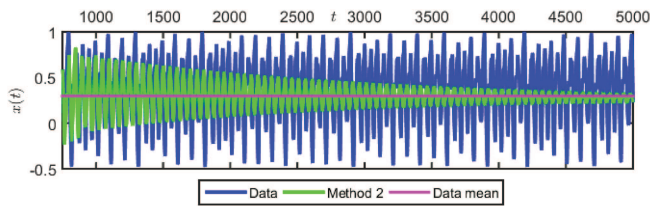
#### 2. Lorenz-96 system

We also use our second method to predict the trajectories of the Lorenz-96 system, for different numbers  $n$  of states. The results are shown in Fig. 7. The method yields an acceptable short-term prediction (with respect to the Lyapunov time scale), especially for the second and third state components. These results also suggest that the performance is quite independent of the number  $n$  of states. This might be due to our use of dictionary functions depending on time-delayed coordinates (see Remark 1), which allows the reconstruction of the attractor from partial measurements (i.e., Takens embedding theorem).

We finally recall that prediction is not the main goal of our methods but is used here as one specific test of the methods—reconstruction being another one. Although the prediction results are decent, successive iterations of the Koopman matrix may

**TABLE I.** The mean value of the *nrmse* vector components is shown for several systems. The error vectors are computed according to (16) for the first 100 reconstructed points. These results show that our two methods yield better performance.

System	Duffing	Mackey–Glass	Rössler	Lorenz-63
EDMD (RBF)	$9.88 \times 10^{-7}$	$4.57 \times 10^{-1}$	$8.51 \times 10^{-1}$	$3.08 \times 10^0$
Method 1	$5.32 \times 10^{-12}$	$5.36 \times 10^{-8}$	$2.80 \times 10^{-8}$	$3.40 \times 10^{-9}$
Method 2	$2.68 \times 10^{-1}$	$1.84 \times 10^{-1}$	$1.06 \times 10^0$	$1.17 \times 10^0$
Lorenz-96				
System	$n = 5$	$n = 50$	$n = 100$	$n = 150$
EDMD (RBF)	$9.64 \times 10^{-1}$	$1.27 \times 10^0$	$9.45 \times 10^{-1}$	$9.5 \times 10^{-1}$
Method 1	$2.80 \times 10^{-10}$	$6.55 \times 10^{-9}$	$2.10 \times 10^{-10}$	$1.40 \times 10^{-9}$
Method 2	$7.09 \times 10^{-1}$	$1.03 \times 10^0$	$8.41 \times 10^{-1}$	$8.83 \times 10^{-1}$



**FIG. 6.** The long-term prediction of the second method for the Mackey–Glass system converges to a constant value close to the mean value of the data.

lead to divergent prediction errors and could be avoided to improve the prediction results (see Sec. IV E).

### D. Spectral properties

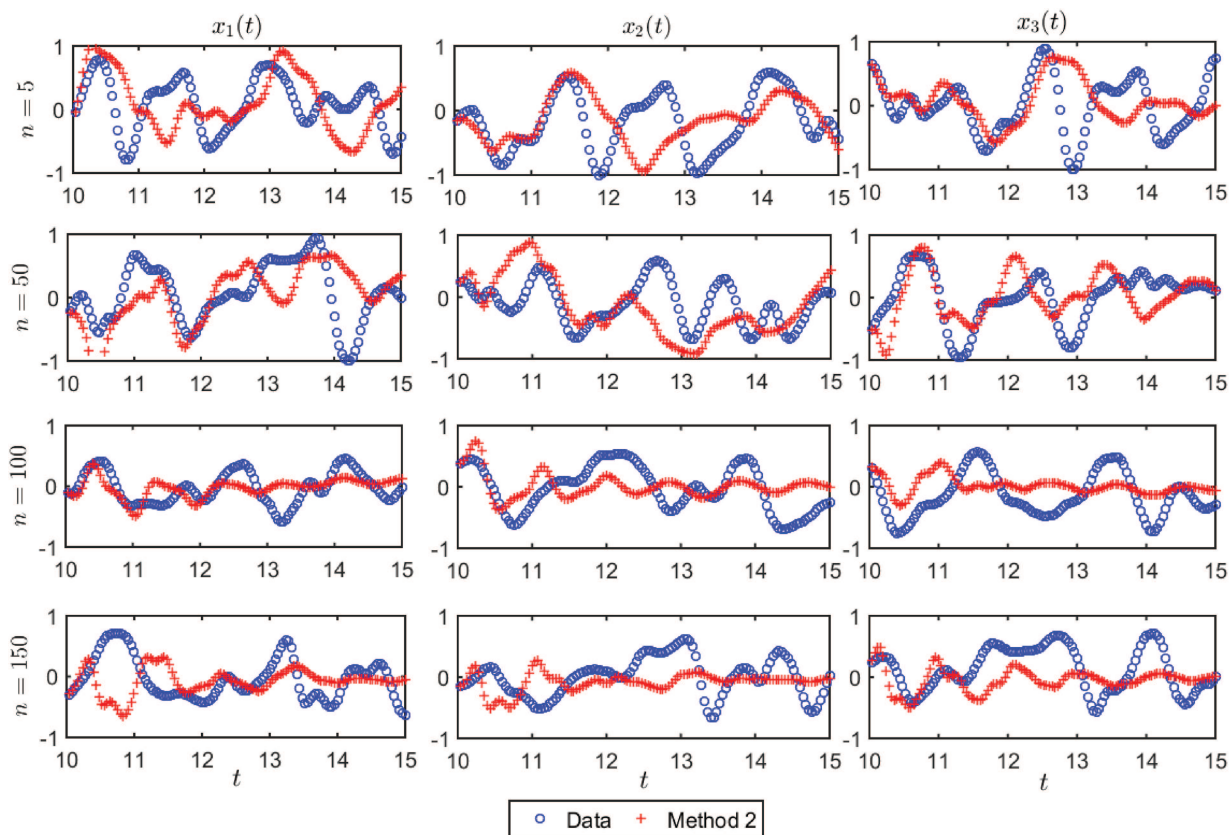
In this section, we compute an approximation of the spectrum of the Koopman operator for the Duffing system and the Rössler system. The results are shown in Figs. 8(a) and 8(b), respectively. In both cases, the EDMD method generates many eigenvalues at the

origin due to the rank-deficiency of the Koopman matrix. Our methods are characterized by less redundancy in the dictionary functions and, in particular, the second method provides a full-rank matrix. For the Rössler system, the EDMD method also generates spurious eigenvalues outside the unit circle (eigenvalues with module approximately equal to 10, not shown in the figure). This explains the fast divergence of the trajectories predicted with the EDMD method. In contrast, the first method recovers the whole unit circle, with a few additional eigenvalues inside the circle for both systems. The second method yields eigenvalues around 1 and inside the unit circle for the Duffing system. It yields more scattered eigenvalues inside the unit circle for the Rössler system. The inset in Fig. 8(a) shows that the eigenvalues of the Jacobian matrix at the stable fixed points are correctly recovered with the second method.

### E. Discussion

#### 1. Comparison of the two methods

The first method provides the best reconstruction results since it exploits all the internal states of the reservoir rather than a few linear combinations of them. It should, therefore, be preferred if one aims at obtaining the most accurate matrix approximation of the



**FIG. 7.** The second method is used to predict the first three components of trajectories of the Lorenz-96 system. Each row corresponds to a fixed number  $n$  of states. Short-term predictions are reasonably accurate for the states  $x_2$  and  $x_3$ , even for larger numbers  $n$  of states.

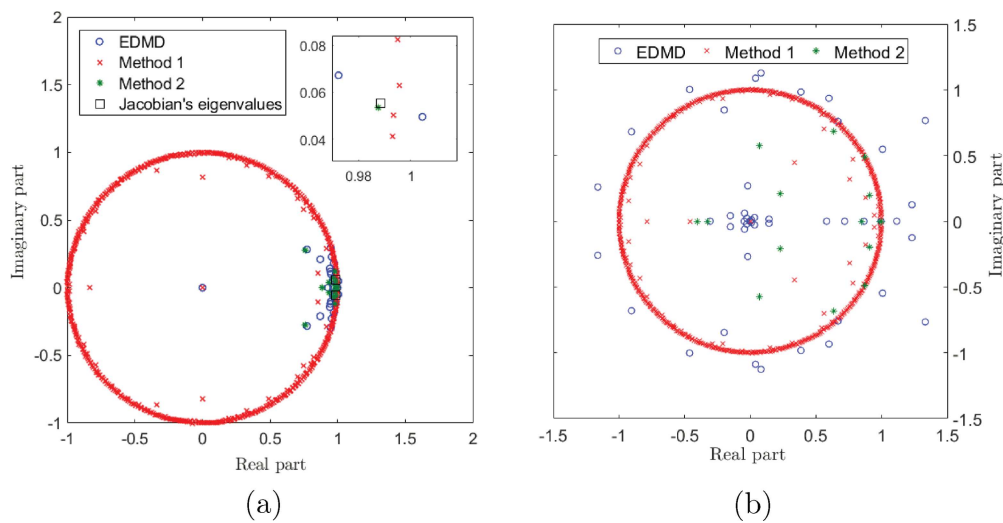


FIG. 8. Computation of the spectrum of the Koopman operator for the Duffing system (a) and the Rössler system (b).

Koopman operator. This method is also quite fast since there is no training process.

However, the Koopman matrix obtained with the first method may be very large since the number of internal states of the reservoir computer is typically large. The second method is motivated by a trade-off between the quality of the results and the size of the Koopman matrix at the cost of an additional computation time due to the dictionary training. It should be considered if one seeks a low-dimensional approximation of the Koopman operator. In the context of prediction, it also seems that the second method provides slightly better results. Also, the least squares residues sharply decrease during the training process. This implies that the second method is able to capture a subspace that is almost invariant through the action of the Koopman operator, thereby leading to an accurate finite-dimensional approximation of the operator. This is a clear advantage of the second method. Moreover, as the number of basis functions increases, the second method should produce results converging to those yielded by the first method. However, the computation time also drastically increases so that the first method appears to be more relevant and efficient in this case. Similarly, if a large dataset is available, the second method might be computationally demanding because of the huge matrices needed for the training.

## 2. Strengths and weaknesses

A main advantage of the proposed methods is that they rely solely on linear techniques thanks to the reservoir computer framework, in contrast to other Koopman operator-based learning techniques. Moreover, both reconstruction and prediction results obtained with these methods are improved with respect to the results obtained with the classical EDMD method. The Koopman spectrum computed with these methods is consistent and motivates the use of

a trained set of basis functions. Finally, we note that both methods require very little data to provide accurate results.

A main limitation of the second method is the computational cost of the dictionary training through the reservoir computer framework. We also note that both methods are not designed for prediction and cannot outperform state-of-the-art prediction methods. In particular, the method fails to predict trajectories from initial conditions that are not related to the training set.

## 3. Improving the prediction methods

Our proposed methods mainly aim at computing the Koopman matrix with appropriate dictionary functions to provide the best global linear approximation of the dynamics. In the context of prediction, however, better results could be obtained with nonlinear approximations of the dynamics. We refer to other works proposing an efficient use of the reservoir computer for prediction, i.e., Refs. 1 and 30.

In the context of prediction with nonlinear approximations, we can also note that the classical EDMD method could be used to compute a single iteration of the Koopman matrix, extract the updated value of the projection maps  $\mathbf{x}$ , and use them to evaluate the iterated values of all dictionary functions. This would allow us to project back the predicted trajectory onto the manifold containing the lifted states (see also a similar idea in the work<sup>4</sup>). In fact, this amounts to computing the least squares projection of the system map  $F$  in the span of the dictionary functions. Numerical simulations suggest that this method is very efficient in the context of prediction.

## V. CONCLUSION

We have proposed two novel methods for computing a finite-dimensional approximation of the Koopman operator. These methods combine classical EDMD with the use of a reservoir computer.

In the first method, the dictionary functions are chosen to be the internal states of the reservoir. In the second method, the reservoir computer is trained and the dictionary functions are optimized linear combinations of internal states. A key advantage of these two methods is that they rely on linear optimization techniques. The accuracy of the Koopman matrix approximation is assessed in the context of reconstruction, prediction, and computation of the Koopman operator spectrum. The results are encouraging and pave the way to the use of the reservoir computer in the Koopman operator framework.

Several research perspectives can be proposed. First, the method could be improved to achieve better predictive performances, although this is not our main goal in this paper (note also that other Koopman operator-based methods exist for this purpose, see, e.g., Refs. 7, 17, and 15). To do so, one could adapt the training of the reservoir according to this specific prediction objective, promote the computation of stable Koopman matrices, and use proper projections between the iterations of the Koopman matrix (see, e.g., Ref. 4). Our second method could also be complemented with convergence results for the alternating optimization scheme. Finally, the proposed methods could be used on real datasets, in the context of spectral analysis, network identification, time-series classification, event detection, and predictive control.

## ACKNOWLEDGMENTS

This work was supported by the Namur Institute for Complex Systems (naXys) at the University of Namur. M. Gulina is grateful to Adrien Fiorucci and Thomas Evrard for fruitful discussions and comments on the article.

## DATA AVAILABILITY

The data that support the findings of this study are available within the article.

## REFERENCES

- <sup>1</sup>P. Antonik, M. Gulina, J. Pauwels, and S. Massar, "Using a reservoir computer to learn chaotic attractors, with applications to chaos synchronization and cryptography," *Phys. Rev. E* **98**, 012215 (2018).
- <sup>2</sup>H. Arbabi and I. Mezić, "Ergodic theory, dynamic mode decomposition, and computation of spectral properties of the Koopman operator," *SIAM J. Appl. Dyn. Syst.* **16**, 2096–2126 (2017).
- <sup>3</sup>E. Bollt, "On explaining the surprising success of reservoir computing forecaster of chaos? The universal machine learning dynamical system with contrasts to VAR and DMD," *arXiv:2008.06530* (2020).
- <sup>4</sup>D. Bruder, B. Gillespie, C. D. Remy, and R. Vasudevan, "Modeling and control of soft robots using the Koopman operator and model predictive control," *arXiv:1902.02827* (2019).
- <sup>5</sup>A. S. Dogra and W. T. Redman, "Optimizing neural networks via Koopman operator theory," *arXiv:2006.02361* (2020).
- <sup>6</sup>Z. Drmač, I. Mezić, and R. Mohr, "Data driven modal decompositions: Analysis and enhancements," *SIAM J. Sci. Comput.* **40**, A2253 (2017).
- <sup>7</sup>D. Giannakis, S. Das, and J. Slawinska, "Reproducing kernel Hilbert space compactification of unitary evolution groups," *arXiv:1808.01515* (2018).
- <sup>8</sup>P. Grassberger and I. Procaccia, "Dimensions and entropies of strange attractors from a fluctuating dynamics approach," *Physica D* **13**, 34–54 (1984).
- <sup>9</sup>J. H. Tu, C. W. Rowley, D. M. Luchtenburg, S. L. Brunton, and J. Nathan Kutz, "On dynamic mode decomposition: Theory and applications," *J. Comput. Dyn.* **1**, 391–421 (2014).

- <sup>10</sup>B. Huang, X. Ma, and U. Vaidya, "Feedback stabilization using Koopman operator," in *2018 IEEE Conference on Decision and Control (CDC)* (IEEE, 2018), pp. 6434–6439.
- <sup>11</sup>H. Jaeger, "The 'echo state' approach to analysing and training recurrent neural networks," GMD Report No. 148 (German National Research Institute for Computer Science, 2001).
- <sup>12</sup>H. Jaeger and H. Haas, "Harnessing nonlinearity: Predicting chaotic systems and saving energy in wireless communication," *Science* **304**, 78–80 (2004).
- <sup>13</sup>E. Kaiser, J. N. Kutz, and S. L. Brunton, "Data-driven discovery of Koopman eigenfunctions for control," *arXiv:1707.01146* (2017).
- <sup>14</sup>A. Karimi and M. R. Paul, "Extensive chaos in the Lorenz-96 model," *Chaos* **20**, 043105 (2010).
- <sup>15</sup>M. A. Khodkar, P. Hassanzadeh, and A. Antoulas, "A Koopman-based framework for forecasting the spatiotemporal evolution of chaotic dynamics with nonlinearities modeled as exogenous forcings," *arXiv:1909.00076* (2019).
- <sup>16</sup>B. O. Koopman, "Hamiltonian systems and transformation in Hilbert space," *Proc. Natl. Acad. Sci. U.S.A.* **17**, 315–318 (1931).
- <sup>17</sup>M. Korda and I. Mezić, "Linear predictors for nonlinear dynamical systems: Koopman operator meets model predictive control," *Automatica* **93**, 149–160 (2018).
- <sup>18</sup>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* **27**, 103111 (2017).
- <sup>19</sup>M. Lukoševičius, H. Jaeger, and B. Schrauwen, "Reservoir computing trends," *Künstl. Intell.* **26**, 365–371 (2012).
- <sup>20</sup>M. Lukoševičius and H. Jaeger, "Survey: Reservoir computing approaches to recurrent neural network training," *Comput. Sci. Rev.* **3**, 127–149 (2009).
- <sup>21</sup>B. Lusch, J. N. Kutz, and S. L. Brunton, "Deep learning for universal linear embeddings of nonlinear dynamics," *Nat. Commun.* **9**, 4950 (2018).
- <sup>22</sup>M. Mackey and L. Glass, "Oscillation and chaos in physiological control systems," *Science* **197**, 287–289 (1977).
- <sup>23</sup>I. Manojlović, M. Fonoberova, R. Mohr, A. Andrejčuk, Z. Drmač, Y. Kevrekidis, and I. Mezić, "Applications of Koopman mode analysis to neural networks," *arXiv:2006.11765* (2020).
- <sup>24</sup>A. Mauroy and J. Hendrickx, "Spectral identification of networks using sparse measurements," *SIAM J. Appl. Dyn. Syst.* **16**, 479–513 (2017), Article.
- <sup>25</sup>A. Mauroy, I. Mezić, and J. Moehlis, "Isostables, isochrons, and Koopman spectrum for the action-angle representation of stable fixed point dynamics," *Physica D* **261**, 19–30 (2013).
- <sup>26</sup>A. Mauroy and I. Mezić, "Global stability analysis using the eigenfunctions of the Koopman operator," *IEEE Trans. Automat. Contr.* **61**, 3356–3369 (2016).
- <sup>27</sup>A. Mauroy, Y. Susuki, and I. Mezić, *The Koopman Operator in Systems and Control* (Springer, 2020).
- <sup>28</sup>I. Mezić, "Spectral properties of dynamical systems, model reduction and decompositions," *Nonlinear Dyn.* **41**, 309–325 (2005).
- <sup>29</sup>S. Pan and K. Duraisamy, "Physics-informed probabilistic learning of linear embeddings of nonlinear dynamics with guaranteed stability," *SIAM J. Appl. Dyn. Syst.* **19**, 480–509 (2020).
- <sup>30</sup>J. Pathak, A. Wikner, R. Fussell, S. Chandra, B. R. Hunt, M. Girvan, and E. Ott, "Hybrid forecasting of chaotic processes: Using machine learning in conjunction with a knowledge-based model," *Chaos* **28**, 041101 (2018).
- <sup>31</sup>J. L. Proctor and P. A. Eckhoff, "Discovering dynamic patterns from infectious disease data using dynamic mode decomposition," *Int. Health* **7**, 139–145 (2015).
- <sup>32</sup>C. Rowley, I. Mezic, S. Bagheri, P. Schlatter, and D. Henningson, "Spectral analysis of nonlinear flows," *J. Fluid Mech.* **641**, 115–127 (2009).
- <sup>33</sup>P. J. Schmid, "Dynamic mode decomposition of numerical and experimental data," *J. Fluid Mech.* **656**, 5–28 (2010).
- <sup>34</sup>E. Skibinsky-Gitlin, M. Alomar, C. Frasser, V. Canals, E. Isern, M. Roca, and J. L. Rossello, "Cyclic reservoir computing with FPGA devices for efficient channel equalization," *Artif. Intell. Soft Comput.* **1**, 226–234 (2018).
- <sup>35</sup>J. C. Sprott, *Chaos and Time-Series Analysis* (Oxford University Press, Oxford, 2003), Vol. 69.
- <sup>36</sup>A. Surana, "Koopman operator framework for time series modeling and analysis," *J. Nonlinear Sci.* **30**, 1973 (2018).

<sup>37</sup>Y. Susuki and I. Mezić, "Nonlinear Koopman modes and coherency identification of coupled swing dynamics," *IEEE Trans. Power Syst.* **26**, 1894–1904 (2011).

<sup>38</sup>Y. Susuki and I. Mezić, "Nonlinear Koopman modes and power system stability assessment without models," *IEEE Trans. Power Syst.* **29**, 899–907 (2014).

<sup>39</sup>Y. Susuki, and I. Mezic, "A prony approximation of Koopman mode decomposition," in *2015 54th IEEE Conference on Decision and Control (CDC)* (IEEE, 2015), pp. 7022–7027.

<sup>40</sup>N. Takeishi, Y. Kawahara, and T. Yairi, "Learning Koopman invariant subspaces for dynamic mode decomposition," *Adv. Neural Inf. Process. Syst.* **30**, 1130–1140 (2017).

<sup>41</sup>F. Triefenbach, A. Jalalvand, B. Schrauwen, and J.-P. Martens, "Phoneme recognition with large hierarchical reservoirs," in *Advances in Neural Information Processing Systems*, edited by J. Lafferty, C. Williams, J. Shawe-Taylor, R. Zemel, and A. Culotta (Neural Information Processing System Foundation, 2010), Vol. 23, p. 9.

<sup>42</sup>M. O. Williams, I. G. Kevrekidis, and C. W. Rowley, "A data-driven approximation of the Koopman operator: Extending dynamic mode decomposition," *J. Nonlinear Sci.* **25**, 1307–1346 (2015).

<sup>43</sup>E. Yeung, S. Kundu, and N. Hodos, "Learning deep neural network representations for Koopman operators of nonlinear dynamical systems," in *2019 American Control Conference (ACC)* (IEEE, 2019), pp. 4832–4839.