AIMS Mathematics, 9(1): 998–1022.
DOI: 10.3934/math.2024050
Received: 19 June 2023
Revised: 24 August 2023
Accepted: 31 August 2023
Published: 05 December 2023

http://www.aimspress.com/journal/Math

Research article

Autoencoding for the "Good Dictionary" of eigenpairs of the Koopman operator

Neranjaka Jayarathne[1,*] **and Erik M. Bollt**[2]

[1] Center for Complex Systems Science, Clarkson University, Potsdam, NY 13699-5815
[2] Department of Electrical and Computer Engineering and The Clarkson Center for Complex Systems Science, Clarkson University, Potsdam, New York 13699, USA

* **Correspondence:** Email: neranjaka.jayarathne@ieee.org; Tel: +18063178638.

Abstract: Reduced order modelling relies on representing complex dynamical systems using simplified modes, which can be achieved through the Koopman operator(KO) analysis. However, computing Koopman eigenpairs for high-dimensional observable data can be inefficient. This paper proposes using deep autoencoders(AE), a type of deep learning technique, to perform nonlinear geometric transformations on raw data before computing Koopman eigenvectors. The encoded data produced by the deep AE is diffeomorphic to a manifold of the dynamical system and has a significantly lower dimension than the raw data. To handle high-dimensional time series data, Takens' time delay embedding is presented as a preprocessing technique. The paper concludes by presenting examples of these techniques in action.

Keywords: deep learning; autoencoders; data driven science; reduced order modelling; Koopman analysis
Mathematics Subject Classification: 37M05, 68T07

1. Introduction

There has been a rapid advancement of computing machinery, scientists acquiring the capability to analyze large data sets and extract information and data-driven techniques have gathered momentum with this recent growth. The modern discipline of data-driven sciences can be viewed as a combined outcome of available advanced mathematical techniques and high performance computing. According to [1] data-driven discoveries are revolutionizing the modeling, prediction, and control in a diverse range of complex systems found in turbulence, the brain, climate, epidemiology, finance, robotics, and autonomy.

As opposed to analyzing the behaviors of complex dynamical systems in terms of the examining

single trajectories, it has become possible to empirically discuss global questions in terms of evolution of density. Apart from the traditional analysis using geometrical techniques, we can analyze the evolution of dynamical systems using the *Frobenius-Perron* transfer operator [2]. The left-adjoint of the *Frobenius-Perron* operator is the *Koopman* operator(KO). KO produces the values produced by a measurement function applied to the dynamical system. Instead of producing the value of the measurement, it produces a value in future.

The perspective of data-oriented KO analysis for analyzing dynamical systems has recently become extensively popular and relevant in science and engineering [3–6], and it can be used to interpret dynamical systems irrespective of them being linear or nonlinear [3]. However, the compromise is that the finite dimensional nonlinear system may or may not require infinitely many dimensions to be represented as a linear system. For most of the computation requirements, a small error could be tolerated. Therefore the infinite dimensionality can be truncated at the cost of reduced accuracy [3]. Furthermore, it explains why the low dimensional dataset could be used to do the same. This paper brings out a machine learning technique, to reduce the dimensionality of the dataset produced by a dynamical system using deep autoencoders(AEs).

A *good dictionary* in this context refers to a set of eigenvectors for an efficient representation of the dynamical system. When representing arbitrary observable functions by a series of eigenvectors, there is freedom to define an efficient representation. Choosing the best from the rich and infinite eigenvectors develops a good dictionary of eigenvectors. The reader is referred to [7–10] for *Koompan mood decomposition* as a global analysis of dynamical systems. This can be used for forecast, as well as to do descriptive decompositions such as growing, fluctuating and decaying components.

The utility of the initial Dynamic Mode Decomposition(DMD) and exact DMD methods as numerical techniques for computing Koopman eigenpairs persists, as evidenced by their continued use in research [3, 5, 11, 12]. Several variants of DMD exist, including sparse DMD [13], extended DMD (EDMD), and those that employ basis vectors to interpret data flow [14, 15]. In machine learning nomenclature, these basis vectors are commonly referred to as a *dictionary of features* [3]. Following this direction [16] describes an extended dynamic mode decomposition with optimal dictionary learning (EDMD-DL). DMD with control laws are introduced in [5, 17] and are emphasized on developing control laws.

In this paper we are developing a computationally efficient way of computing a good dictionary of Koopman eigenpairs. The technique of computing eigenpairs is based on [3]. Computational efficiency gain was induced by making a geometric transformation to the data manifold by means of deep learning techniques. In section 2 we will be presenting the mathematical and numerical foundations that the Koopman eigenpairs computed. In section 3 the machine learning technique is introduced and in section 4 examples are presented using the technique. In section 5 the conclusions are derived with a direction for future developments.

2. Mathematical foundations

2.1. Koopman analysis

First, we review the underlying mathematics of Koopman analysis. Let,

$$\dot{X} = F(X), \quad F : M \to \mathbb{R} \tag{2.1}$$

be a dynamical system. The flow can be stated as a function for each $t \in \mathbb{R}$, such that $X(t) = \rho_t(X_0)$ where $X_0 = X(0) \in M$. This means that the trajectory starts at X_0 when $t = 0$. A *KO* describes the evolution of observables along the flow [3, 4, 18]. This study is based on analyzing these observables. Let us call these observables observation functions. With reference to [3], in this work we consider the observation function to be

$$g : M \to \mathbb{C}, \tag{2.2}$$

where

$$g \in L^2(M) = \left\{ g : \int_M |g(s)|^2 ds < \infty \right\}. \tag{2.3}$$

The scalar observation functions introduced here can be stacked to form vector observations. *KO* defines how the observation functions evolve over time along the orbits of the dynamical system.

Definition 2.1. Let ρ_t be a semiflow. That is, ρ_t is defined for $t \geq 0$. Then, the *KO* $\mathbb{K}_t$ is defined for $L^2(M)$ such that

$$\mathbb{K}_t[g](X) = g \circ \rho_t(X), \tag{2.4}$$

where $\rho_t(X_0) = X_t$.

In lehman's terms, for each X we observe the value of an observable $g(x)$, not at x itself, but after a time t at $\rho_t(X)$. The *KO* is linear on $L^2(M)$. This may be at the cost of it being infinite dimensional. *Eigenvectors* and *eigenvalues* are studied under the spectral analysis of the *KO* [4, 18, 19].

Definition 2.2. Suppose ϕ_λ is an eigenvector of $\mathbb{K}_t$ and λ is the corresponding eigenvalue. Then the eigenpair should satisfy the equation

$$\mathbb{K}_t[\phi_\lambda](X) = b^t \phi_\lambda(X) = e^{\lambda t} \phi_\lambda(X), \tag{2.5}$$

or

$$\mathbb{K}_t[\phi_\lambda](X) = e^{\lambda t} \phi_\lambda(X). \tag{2.6}$$

For each eigenvalue λ there are uncountable eigenvectors. In general, these are not linearly independent. It is interesting to explore how an observable function can be decomposed into eigenvectors of the *KO*.

Definition 2.3. Let $\overline{g(X)}$ be a vector of observable functions $g_k(X)$.

$$\overline{g(X)} = [g_1(X), \ldots, g_D(X)] : M \to \mathbb{C}^D. \tag{2.7}$$

Suppose $\overline{g(X)}$ could be written as follows,

$$\overline{g(X)} = \sum_{j=1}^{\infty} \phi_{\lambda_j}(X) \mathbf{v}_j. \tag{2.8}$$

Then it follows that,

$$\overline{g(X_0)} = \sum_{j=1}^{\infty} \phi_{\lambda_j}(X_0) \mathbf{v}_j. \tag{2.9}$$

Apply KO to both sides of the equation,

$$\mathbb{K}\overline{g(X_0)} = \mathbb{K}\sum_{j=1}^{\infty}\phi_{\lambda_j}(X_0)\mathbf{v}_j \tag{2.10}$$

$$g \circ \rho_t(X_0) = \sum_{j=1}^{\infty}\mathbb{K}\left[\phi_{\lambda_j}\right](X_0)\mathbf{v}_j, \tag{2.11}$$

where $X_t = X(t)$.

Since eigenvectors are not unique, the authors are interested in finding an efficient representation using eigenvectors.

Definition 2.4. In [3], a **k**-efficient finite set of Koopman eigenpairsare defined as follows. Let $\{\psi_{\lambda_i}(X)\}_{i=1}^{k}$ and $\{\phi_{\lambda_i}(X)\}_{i=1}^{k}$ be sets of **k** unit eigenvectors.

Further, let q be an observable function,

$$q : M \to \mathbb{C}. \tag{2.12}$$

If

$$\min_a \| \sum_{i=1}^{k} a_i\psi_{\lambda_i}(X) - q(X)\| \leq \min_b \| \sum_{i=1}^{k} b_i\phi_{\lambda_i}(X) - q(X)\|. \tag{2.13}$$

then $\{\psi_{\lambda_i}(X)\}_{i=1}^{k}$ is a set of **k**-efficient Koopman eigenvectors.

2.2. Computational technique of determining Koopman eigenpairs

The following definition of the eigenvector was rephrased from [3].

Theorem 2.5. *Let the Koopman eigenfunciton Partial Differential Equation(PDE) is defined for an Ordinary Differential Equation(ODE) $\dot{X} = F(X)$, with a flow $X(r) = \rho_r(X_0) : M \times \mathbb{R} \to M$. Assume a co-dimension* 1 *[20] initial data manifold $\Lambda \subset M$ is non-recurrent for some time epoch $r \in [t_1, t_2]$, that contains* 0*, and is transverse to the flow. Further let $\mathbb{U} = \cup_{t\in[t_1,t_2]}\rho_t(\Lambda)$ be the resulting non-recurrent closed domain. Furthermore, let $h : \Lambda \to \mathbb{C}$ be an initial data function, then the Koopman eigenpair $(\lambda, \phi_\lambda(X))$ $\phi_\lambda : \mathbb{U} \to \mathbb{C}$ has the form*

$$\phi_\lambda(X) = h \circ s^*(X)e^{\lambda r*(X)}. \tag{2.14}$$

Now, we present a computational explanation of **Theorem 1** of [3]. This is the mathematical infrastructure of the work presented in this paper.

Suppose $\dot{X} = F(X)$ is an ODE satisfying the Koopman eigenvector PDE. Let the corresponding flow be given by $X(r) = \rho_r(X_0) : M \times \mathbb{R} \to M$. Further, let $\Lambda \subset M$ be a co-dimension-1 data manifold. We should pick a time interval where the data values are non-recurrent. Let us call Λ a non-recurrent data manifold, and Let us assume that the non-recurrent time span is $[t_1, t_2]$ and $0 \in [t_1, t_2]$.

If Λ is transverse to the flow, we can define $\mathbb{U}$ as follows.

$$\mathbb{U} = \cup_{t\in[t_1,t_2]}\rho_t(\Lambda). \tag{2.15}$$

That is, $\mathbb{U}$ is the collection of all the trajectories starting from Λ. This includes reverse trajectories as well. Furthermore, let $h : \Lambda \to \mathbb{C}$ be an initial data function. Then, an eigenpair $(\lambda, \phi_\lambda(X))$ satisfies the equation 2.14 where

$$r^*(X) = \{t : \rho_{-t}(X) \cap \Lambda \neq \phi\} \tag{2.16}$$

and

$$s^* = s \circ \rho_{-r^*(X)}(X), \tag{2.17}$$

where s is the parametrization of Λ. That is, s is the data function available on Λ.

The exercise of [3] was to introduce this formulation and introduce the algorithm to numerically compute h and optimal λ for a given stream of data.

Following is a summary of the numerical algorithm described in [3].

Suppose $q \in L^2(\mathbb{U})$ be an arbitrary data function. $q : \mathbb{U} \to \mathbb{C}$. Let,

$$\psi = argmin_{\lambda,h} \|\phi_{\lambda,h} - q\|^2_{L^2(\mathbb{U})} \tag{2.18}$$

be the eigenvector that closely estimates q. $\psi : \mathbb{U} \to \mathbb{C}$. It is worth noting that the optimization is over both λ and h. Furthermore,

$$argmax_{\lambda,h}\,(q, \phi_{\lambda,h}) = argmin_{\lambda,h} \|\phi_{\lambda,h} - q\|^2_{L^2(\mathbb{U})}, \tag{2.19}$$

where

$$\|f\|_{L^2(\mathbb{U})} = \int_{\mathbb{U}} f(X)dX \tag{2.20}$$

and

$$(f, g) = \int_{\mathbb{U}} f(X)g(X)dX \tag{2.21}$$

for any $f, g \in L^2(\mathbb{U})$.

Let us explain an example scenario for a 2-dimensional dynamical system. The algorithm can be generalized to larger dimensions limited only by the computational resources, then Λ becomes 1-dimensional since it should be co-dimension 1. Let $s_0 < s_1 < \cdots < s_n$ be a uniform partition of Λ. The data function $h : \Lambda \to \mathbb{C}$ could be indexed accordingly as

$$h_i := h(s_i), \tag{2.22}$$

We will consider a uniform grid of time as well, $r_0 = 0 < r_1 < \cdots < r_m$. That is, $m + 1$ is the number of equal time steps. Then, the grid points can be identified as follows: $s_{i,j} = \rho_{r_j} \circ x(s_i), 0 \leq i \leq m, 0 \leq j \leq m$.

In terms of the grid, the optimization problem in equation 2.18 for the function $\psi(x)$ represented on the grid $\psi \circ x(s_{i,j})$ is approximated by solving the finite rank least squares problem,

$$\tilde{\psi} = argmin_h \|\phi_{\lambda,h} \circ x(s_{i,j}) - q \circ x(s_{i,j})\|^2_F = argmin_{h_i} \sum_{j=1}^{m} \sum_{i=1}^{n} |e^{\lambda r_j} h_i - q_{i,j}|^2. \tag{2.23}$$

On this grid, both functions q and $\tilde{psi}_{\lambda,h}$ can be sampled at $\rho_{r_j} \circ x(s_i)$.

Let $q_{i,j} = q(s_{i,j})$ for $0 \leq i \leq n, 0 \leq m$. In [3], for distance the Frobenius norm [21] is used.

We can rewrite the optimal initial data as an $n \times 1$ vector $h^0(\lambda) \in \mathbb{C}^n$. The problem restatement is as follows.

Let us define the $m \times 1$ vector $E(\lambda)$ as $E(\lambda) = \begin{bmatrix} e^{\lambda r_0} & e^{\lambda r_2} \dots e^{\lambda r_m} \end{bmatrix} \in \mathbb{C}^m$.

Then, $A(\lambda) = E(\lambda) \otimes I_n$ where $\otimes$ is the Kronecker product. Furthermore, let b be the vector received by reshaping the matrix $q \in \mathbb{C}^{n \times m}$ into a vector of shape $mn \times 1$. That is, $b = reshape(q, mn, 1)$.

Then $h^0(\lambda)$ is computed by solving the least squares problem,

$$h^0(\lambda) = argmin_h \|A(\lambda)h - b\|_2. \tag{2.24}$$

It is noticed that $q = q(s_{i,j})$ is a grid of values. It follows that

$$p^0(\lambda) = A(\lambda)h^0(\lambda). \tag{2.25}$$

Thus, $\tilde{\psi}_1^0 = reshape(p^0(\lambda), n, m)$.

That is, in order to produce the optimal eigenvector, $p^0(\lambda)$ is being reshaped.

$\tilde{\psi}_1^0$ is the eigenvector computed by carrying out the computation once. This can be successively done to reduce the residuals.

Therefore, we can call the residue at the kth iteration R_k,

then

$$R_k = b - \sum_{i=1}^{k} p_i^0, \tag{2.26}$$

and the successive eigenpairs can be computed by using

$$(\lambda_{k+1}^0, \lambda_{k+1}^0) = argmin_{\lambda,h} \|A(\lambda)h - R_k\|_2, \tag{2.27}$$

$$\tilde{\psi}_{k+1}^0 = reshape(p_k^0, n, m), \tag{2.28}$$

3. Manifold learning

In [20] and [22], a manifold is described as a collection of charts called an atlas. Each chart maps a subset of the manifold to the *Eucledean Space* with an invertible map and these subsests could have intersections that are not empty. The invertible maps should be consistent in the nonempty intersections. Furthermore, a manifold is a topological space that is *Locally Eucledean* [20].

3.1. Manifold hypothesis

The manifold hypothesis is the assumption that higher dimensional data lies on lower dimensional manifolds embedded within the high-dimensional space [23, 24]. This is due to the belief that high dimensional data could be generated from a low-dimensional dynamical system. Specifically in the context of a dynamical systems, orbit data may be attracted to a stable invariant manifold, which occurs in certain systems where dissipation leads to the potential of a reduced order model. A collection of techniques developed to identify such a low-dimensional manifold using the available high-dimensional data is called manifold learning. For a comprehensive study of the manifold hypothesis and its mathematical consequences, the reader is referred to [23].

3.2. Manifold learning

Among the handful of manifold learning techniques, there are techniques such as isometric feature mapping (*ISOMAP*), locally linear embeddings (*LLE*), Laplacian eigenmaps, diffusion maps, nonlinear principal component analysis which stand out from the rest. While introducing manifold learning, [25] introduces it as an algorithmic technique for dimensionality reduction. In paper [26], a geometric framework for nonlinear dimensionality reduction is introduced. The authors claim that their technique efficiently computes the global optimal and is guaranteed to converge asymtotically to the true structure. For a comprehensive study of *LLE*, the reader is referred to [27]. For treatment of *ISOMAP* algorithms and *Laplacian Eigen Maps* the interested reader is referred to [28] and [29], respectively. However, this investigation is more concerned about manifold learning using deep learning techniques. Following is a concise empirical study of deep learning based manifold learning in various engineering applications. For a more descriptive treatment on the topological point of view of manifold learning, refer to [22].

3.3. Mathematical foundations of manifold learning

Dimensional reduction of the data is the main application of manifold learning. The data, which is to be dimensionally reduced, is assumed to be generated from a lower dimensional submanifold [30]. A d-dimensional manifold M has the property that for all $x \in M$, there is a neighborhood $X_x \subset M$ of x such that X_x is homeomorphic to an open set $\Theta_x \subset \mathbb{R}^d$, where $\mathbb{R}^d$ is the d-dimensional Euclidean space [31]. Then, Θ_x can be called a local coordinate of X_x [31].

Since $M = \bigcup_{x \in M} X_x$ where $\{X_x | x \in M\}$ is an *open covering* of M, we can find a homeomorphism $\phi_x : X_x \to \Theta_x$ for all $x \in M$. Then, for all $x \in M$, we have $\phi_x(X_x) = \Theta_x$. Therefore, $\{\Theta_x | x \in M\}$ can be represented as $\{(X_x, \Theta_x) | x \in M\}$. The set of homeomorphisms $\{\phi_x\}_{x \in M}$ need not be unique [31]. This leads to the observation that learnt manifolds need not be unique. Suppose $X = \{X_1, X_2, \ldots, X_N\} \subset M \subset \mathbb{R}^{d'}$ be the set of observable data vectors. Let $Y = \{Y_1, Y_2, \ldots Y_N\} \subset \mathbb{R}^d$ be the reduced dimensional vectors where $d' > d$. It can then be concluded that the observable data was generated by a d-dimensional dynamical system. We find the homeomorphism $\phi : X \to Y$, such that $\phi(X_i) = Y_i$ for $1 \leq i \leq N$ is the process of manifold learning.

Even though the previously surveyed techniques can be used for manifold learning, this investigation is interested in deep learning techniques. The deep learning techniques are favorable in real time computing and the learnt deep neural network can be used to uncover the intrinsic local coordinates. This ability comes as a consequence of the results of this investigation.

Following is a concise survey of applications of deep manifold learning. While introducing the phrase *deep manifold learning (DML)* for the deep learning of hidden lower dimensional manifolds, [32] represents a framework of DML for action recognition using convolutional neural networks. In their paper on image set classification, [33] introduces a technique for *Riemannian* manifold deep learning. This paper also provides a summary of previous efforts on Riemmanian manifold learning.

3.4. Autoencoders

AEs have become a standout machine learning method to find a reduced order model in an invariant manifold within an artificial neural network framework, including dynamical systems with stable reduced order models [34, 35]. Figure 1 depicts a basic neural network topology of a deep AE.

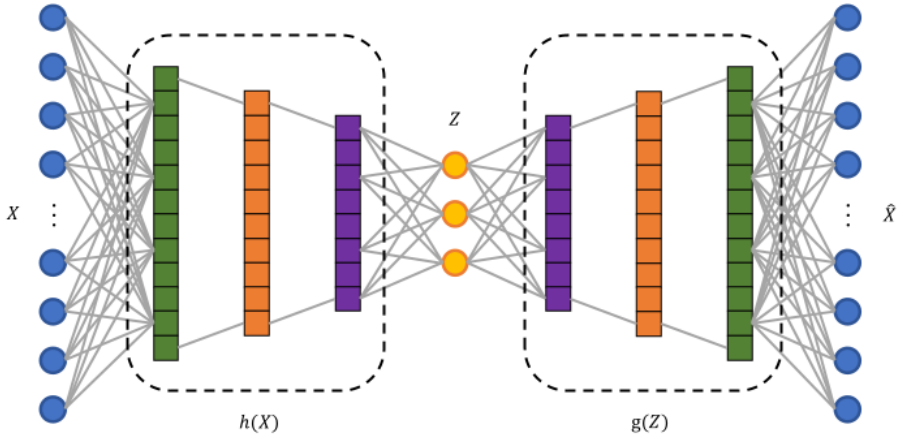


Figure 1. Graphical depiction of a simplified deep AE.

AEs were first proposed and formulated by the PhD thesis of Y. LeCun [36]. These are traditionally used for dimensionality reduction [37]. Based on the formulation of [38], the mathematical foundation of an AE can be viewed as the recursive formula 3.1. The mathematical foundation of the neural networks can be found in [39].

$$x^{(l)} = g\left(b^{(l-1)} + W^{(l-1)}x^{(l-1)}\right) \tag{3.1}$$

Let $K^{(l)}$ denote the number of neurons in each layer where $l = 1, 2, 3, \ldots, L$. Let the output of neuron k in layer l be $x_k^{(l)}$, and the vector of all outputs for this layer be $x^{(l)} = \left(x_1^{(l)}, x_2^{(l)}, x_3^{(l)}, \ldots, x_{K^{(l)}}^{(l)}\right)'$. In each node, a nonlinear activation function $g(\cdot)$ is applied further to the linear operation. In the recursive equation 3.1, $W^{(l-1)}$ is a $K^{(l)} \times K^{(l-1)}$ matrix of weight parameters and $b^{(l-1)}$ is a $K^{(l)} \times 1$ vector of bias parameters [38]. For this work, "*sigmoid*" and "*ReLu*" [40] were proven to be acceptable activation functions. We decided to do our experiments with *sigmoid* consistently.

AEs, in general, takes the form $Z = h(X)$ and $\widehat{X} = g(Z)$, where $\widehat{X}$ is the output of the neural network. Vector value Z is known as the latent vector, which is the encoded/transformed form of X. Transformation g is meant to invert the transformation h. Therefore, ideally, $g = h^{-1}$. The loss function of the neural network is a function of $||\widehat{X} - X||$. We used the mean squared error as the loss function,

$$\mathbb{L} = \frac{1}{N}\sum_{i=1}^{N}||\widehat{X} - X||^2. \tag{3.2}$$

Subsequently, $\mathbb{L}$ is minimized using the gradient descent over a multitude of iterations.

3.5. AEs for reduced order modeling

Full order models require increased utilization of resources. The main goal of a reduced representation is to capture the essential physical features of a system and project them onto a lower-dimensional space or manifold in a way that preserves as much information as possible, while still

allowing for meaningful comparisons to be made with a full-order model (FOM). This approach is therefore called reduced order modeling (ROM) [41]. There have been many attempts in developing ROMs for dynamical systems [42–44]. Compiling a ROM for a complex dynamical system can be an exceedingly arduous task due to the inherent complexity of the system. The projection-based ROM is one of the earliest and most widely used techniques for ROM. This method involves transforming the original space into a lower-dimensional space using the governing PDEs of the physical system [45]. For an elaborate introduction of the dimensionality of the ROM, the reader is referred to [46].

Proper orthogonal decomposition (POD) is a widely used projection-based ROM technique, particularly in the field of computational fluid dynamics. It involves decomposing the original data into a set of orthogonal modes that capture the dominant dynamics of the system. For those interested in delving deeper into the topic of POD, we recommend referring to the following seminal works: [47–49]. Principal component analysis (PCA) [50] is a comparable method to POD, which gathered momentum due to the availability of the computing resources. Empirical orthogonal functions [51], Karhunen-Loeve expansion [52] are also similar techniques for ROM. According to [53] these ROM techniques utilize the eigen-decomposition of the snapshot matrix using singular value decomposition (SVD).

DMD is a ROM technique, that is popular in modeling physics-based systems using their spatio-temporal coherent structures [11]. DMD is being used for a wide range of applications [1], image-processing [54], neuroscience [55], and Robotics [56] to name a few. Koopman operator theory (KOT) itself has direct connections to DMD. Initially KOT was used for the characterization of the dynamics of the Hamiltonian functions [57]. EDMD was used in [16] to decompose the KOT for the analysis of a nonlinear dynamical system. A matching dynamical system was formed using the Koopman operator by decomposing the operator using EDMD in [58].

4. Examples

Now, we provide a couple of worked examples of the developed theory.

4.1. Example 1: Van der Pol Oscillator

We use the *Van der Pol*(*VdP*) oscillator as the first example for simulation. The *VdP* is an oscillator with nonlinear damping [59]. The first step of the process is to simulate trajectories of the *VdP*. In order to simulate geometric transformations using deep AEs, we considered a complete cycle of each trajectory. These trajectories are similar to what was depicted in Figure 2. The original trajectories lie on a 2 dimensional plane. We projected these trajectories to a 3 dimensional space using the mapping

$$\begin{bmatrix} \bar{x}_1 \\ \bar{x}_2 \\ \bar{x}_3 \end{bmatrix} = \begin{bmatrix} x_1 \\ x_2 \\ x_1^2 + x_2^2 \end{bmatrix}. \tag{4.1}$$

This transforms $2d$ trajectories to trajectories in $3d$. These projected trajectories are depicted in Figure 2. At this, state the dynamical system is treated as a 3 dimensional system. The next step would be to reduce the order of the model to 2. We did it using a deep-autoencoder (*DAE*). Following is a brief description of the *DAE* that we tuned for this particular system. For implementing the deep neural network, we used *Keras*, which is based on *Tensorflow*. Both of these were programmed using the

python programing language. The *DAE* had the latent vector size of 2, consistent with our goal. We received optimal results when the first and second hidden layers had the sizes of 100 nodes each. These two hidden layers are the encoder. The decoder, which is the neural network after the latent vector, has the same architecture as the encoder. Optimal results were received when *Sigmoid* was used as the activation function. Comparably similar results were achieved when rectified linear unit (*Relu*) was used as the activation function. *tanh* and *softmax* functions did not provide comparably similar results. These two functions were rejected after visually inspecting the resulting reduced order trajectories, as all the layers in the neural network were dense. As a normalized practice, the networks were trained for $2,000$ epochs throughout the study. Before feeding into the neural network for training, each stream of data has to be scaled from 0 to 1. The scaled 2 dimensional trajectories are depicted in Figure 3. There are many trajectories, each with a different initial point. The nature of the *VdP* is that they converge to a stable limit cycle. The range of initial points are depicted by the anomaly at around points $(0.9, 0.5)$ and $(1, 0.5)$ in Figure 3. All the trajectories, initiated from the range of different initial points, eventually converge to the stable limit cycle and stay on it. Therefore, it is sufficient to simulate one initial period. The transformed trajectory is depicted in Figure 4. There is no guarantee that the transformed trajectory will always be this. Even with the original trajectories being the same, the AE might converge to a different local minimum of the error producing a different transformation. Finding a geometric transformation using a *DAE* is a trial and error process. With the number of hidden layers, nodes on each layer, and epochs to train, the activation function may vary with the application at hand. Therefore, it is to be found and optimized over several iterations during the process of simulation. As it was intended, the ROM of the *VdP* has 3 dimensions.

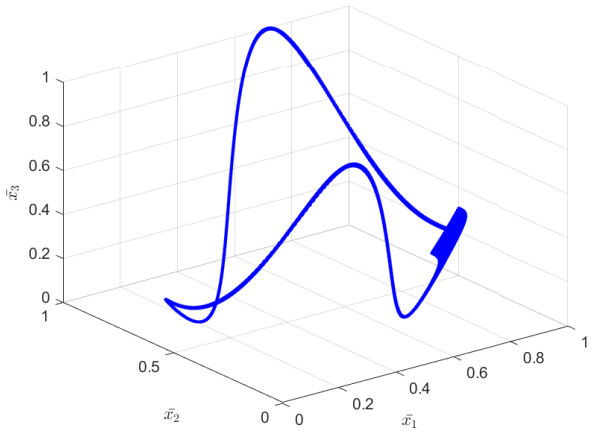


Figure 2. Trajectories from the VdP Oscillator projected onto the 3 dimensions and scaled.

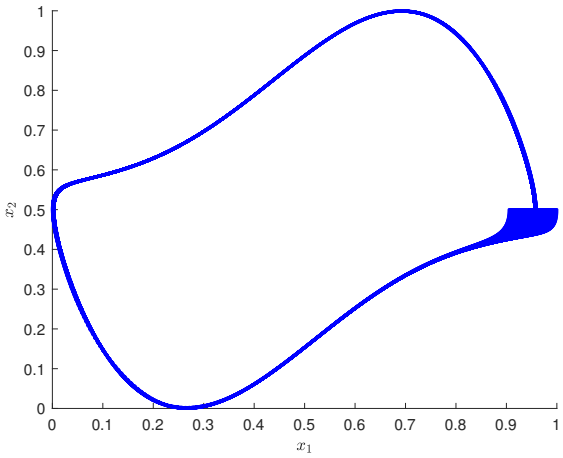


Figure 3. Scaled trajectories from the VdP Oscillator.

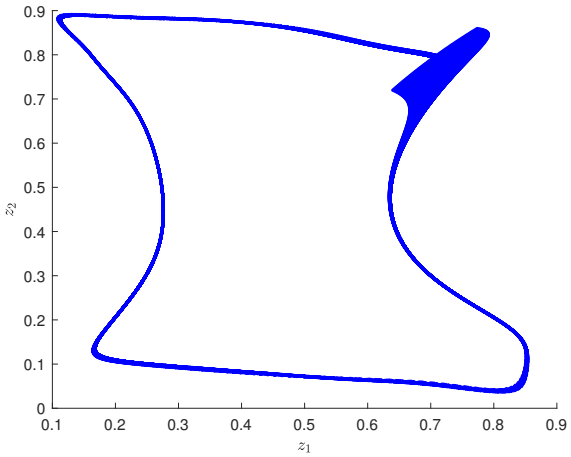


Figure 4. Transformed trajectories from the VdP Oscillator.

4.2. Reduced order modeling for Koopman eigenpair computation

Now, we move into the analysis of the geometric transformation of the trajectory we used for the *Koopman* eigenpair computation. For the *Koopman* analysis, we do not need a full cycle of a trajectory. Therefore, the trajectories are restricted to a shorter time period. The scaled trajectory is depicted in Figure 5. Again, this is projected to the 3 dimensional space using the same transformation as in the complete trajectory case which is depicted by Figure 6. The same neural network architecture as for the previous case was used to autoencode the set of shorter trajectories. The ROM trajectories are depicted in Figure 7. A closer comparison between Figures 4 and 7 will hint the interested reader that the two transformations are not equal to each other. For the *DAE*, the same architecture and hyperparameters were used as the full trajectory *DAE*. This guarantees that accuracy is going to converge to a stable value. The next step is to use the ROM to compute the *Koopman* eigenvectors and eigenvalues.

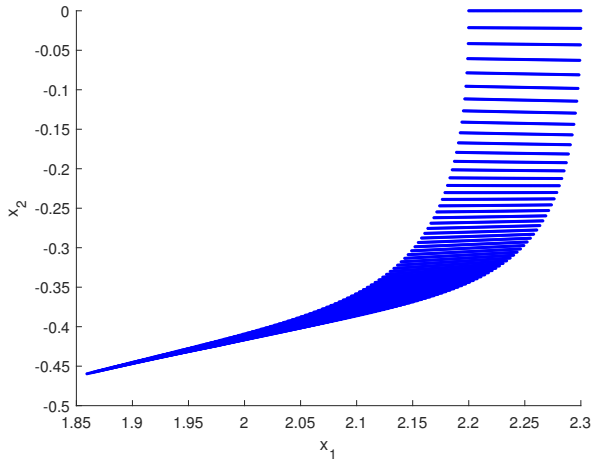


Figure 5. Trajectories from the VdP Oscillator.

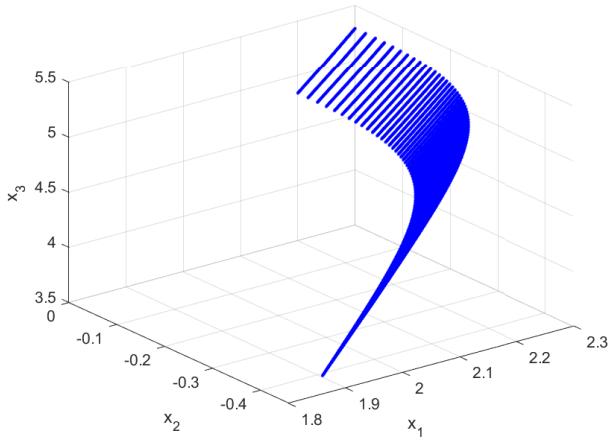


Figure 6. Trajectories from the VdP Oscillator projected to 3 dimensional space.

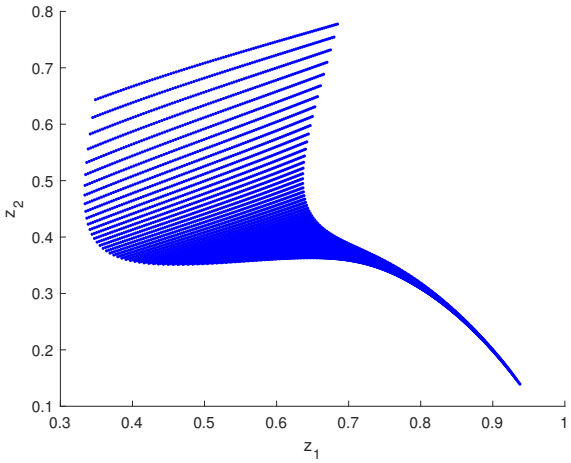


Figure 7. Trajectories in the transformed geometry.

4.3. Computing Koopman Eigenvectors with the ROM

The next step is to find *Koopman* eigenvectors as given by the equation 2.14. Given by the equation 2.17, s^* is derived as a consequence of the flow of the dynamical system. Therefore, we only need to determine the optimal eigenvalue λ and the corresponding h of the equation 2.14. To be consistent with the investigation of [3] we chose the observation function to be $q = 3e^{-\frac{x_1^2+x_2^2}{10}}$. When each x_i is replaced by z_i for $i = 1, 2$, we get $q = 3e^{-\frac{z_1^2+z_2^2}{10}}$. In this case, instead of using x_1 and x_2, we use z_1 and z_2, the transformed coordinates to compute the eigen pairs. We computed the first 10 modes or the first 10 eigenvectors and their corresponding eigenvalues. For each mode the corresponding eigenvalue is the value of λ, which minimizes the error that is depicted in Figure 8. The $h(s)$ vectors for the first 10 modes are depicted in Figure 9. The absolute error between the sum of eigenvectors and the observation function decreases with the mode. This is depicted in Figure 10.

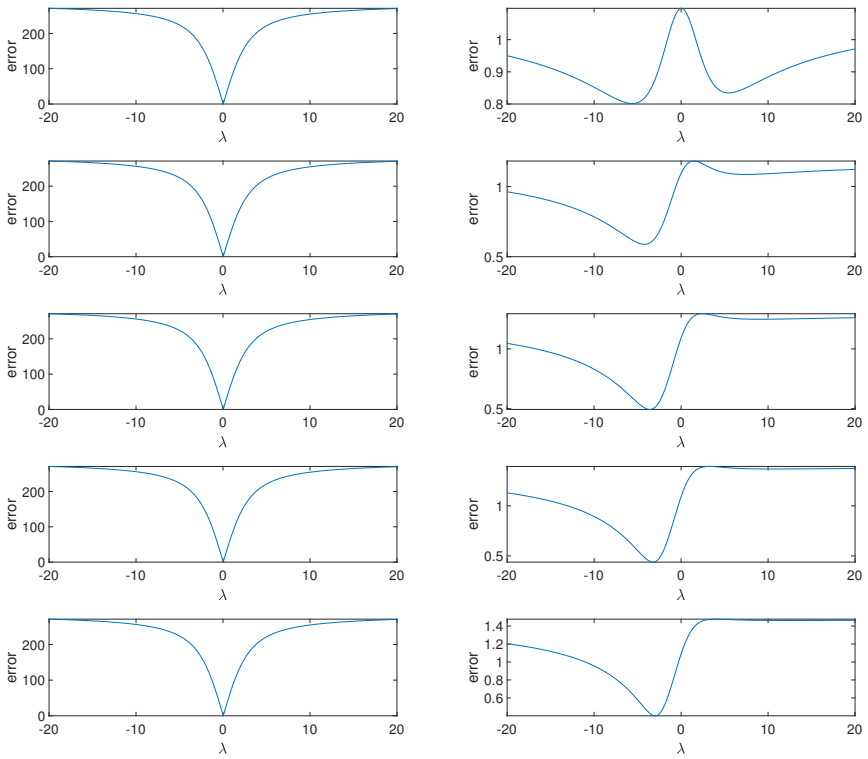


Figure 8. Error vs λ variation through the first 10 modes.

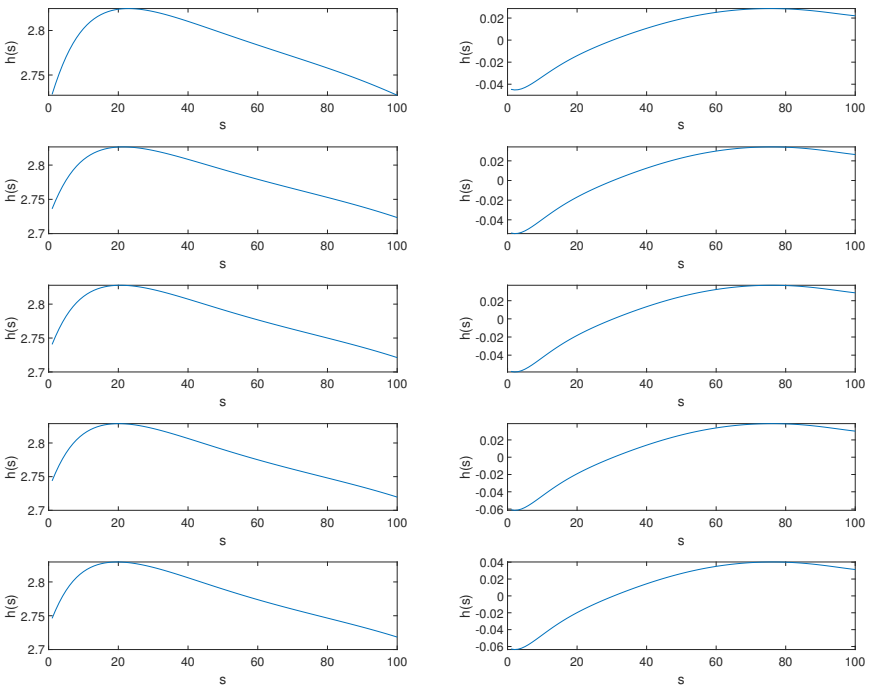


Figure 9. $h(s)$ of the first 10 modes.

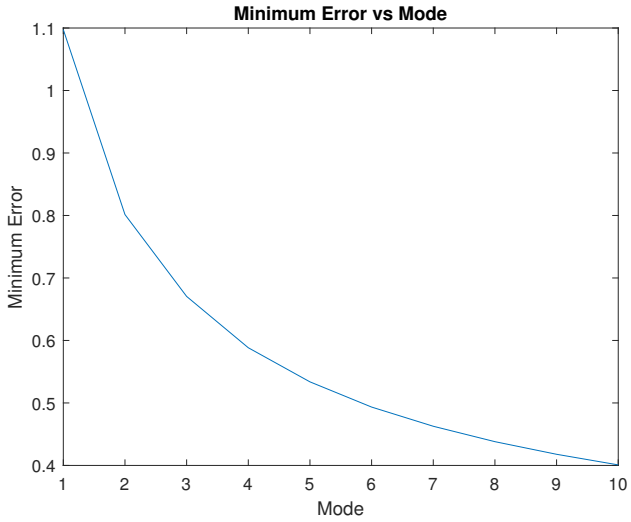
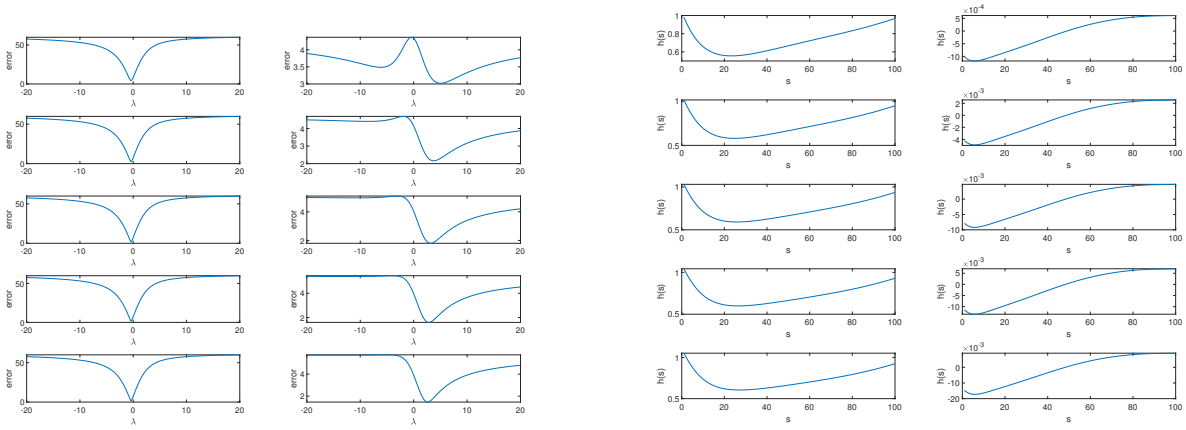


Figure 10. Minimum error vs. mode.

4.4. Eigenpairs with a different observation function

Now, we change the observation function to $q = z_1^2 + z_2^2$. The variation of the error with λ and the function $h(s)$ for the first 10 modes are plotted in subfigures, Figure 11a and 11b, respectively. The corresponding minimum error versus mode graph is presented in Figure 12.



(a) Error vs λ variation **(b)** $h(s)$

Figure 11. VdP trajectories with $q = z_1^2 + z_2^2$.

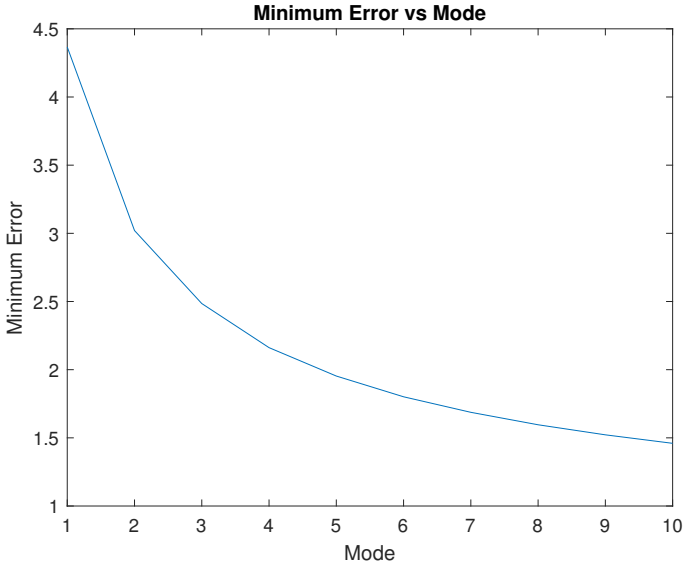


Figure 12. Minimum error vs. mode for $q = z_1^2 + z_2^2$.

4.5. Example 2: Reaction diffusion

Now, we apply our algorithm to a 1 dimensional reaction diffusion equation that appears in [60].

$$\begin{aligned} \frac{\partial u_1}{\partial t} &= D\frac{\partial^2 u_1}{\partial x^2} + \frac{1}{\epsilon}(u_2 - f(u_1)) \\ \frac{\partial u_2}{\partial t} &= D\frac{\partial^2 u_2}{\partial x^2} - u_1 + \alpha. \end{aligned} \tag{4.2}$$

In the system 4.2, $x \in [0, 1]$ and it is subject to *Dirichilet* conditions,

$$\begin{aligned} u_1(x = 0, t = 0) &= u_1(x = 1, t = 0) \\ u_2(x = 0, t = 0) &= u_2(x = 1, t = 0). \end{aligned} \tag{4.3}$$

We chose to fix $\epsilon = 0.01$ and $\alpha = 0.01$ for our experiments. For the simulation we have presented, we used $D = 0.0322$. The surface plots for u_1 and u_2 for these parameters are presented in Figure 13, where $x \in [0, 1]$ and $0 \leq t \leq 60$. It is noted that this reaction-diffusion system produces chaotic [61] behavior depending on the values of ϵ and D. The trajectories of u_1 and u_2 at $x = 1/2$ are graphed in Figure 14. Even though mathematically there are unaccountably many trajectories between $x = 0$ and $x = 1$, computationally this number is countably finite. We used x to be *linspace(0,1,100)*. This leads to the resulting simulation having 100. When there are 100 trajectories, each for u_1 and u_2, there are 200 trajectories in total. Each trajectory from these becomes an input to the *DAE*. In order for the machine to learn a reduced order model, the deep neural network has to have at least 4000 nodes in the first hidden layer. However, the network architecture with $(4,000, 4,000)$ did not converge beyond 25% accuracy. Increasing the number of nodes to $(10,000, 10,000)$ did not improve the accuracy significantly. The available computing cloud resources were not sufficient to increase the number of nodes further. Therefore, we had to look for other means of producing the reduced order model.

The solution we found was to preprocess the data using other means. In particular, we used another embedding technique using time delayed snapshots of the trajectories.

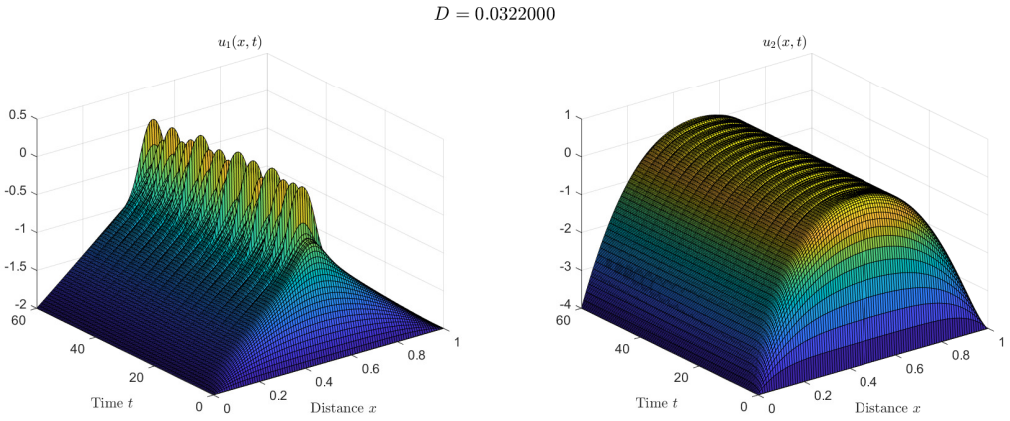


Figure 13. Surface plots of u_1 and u_2 of reaction-diffusion PDE.

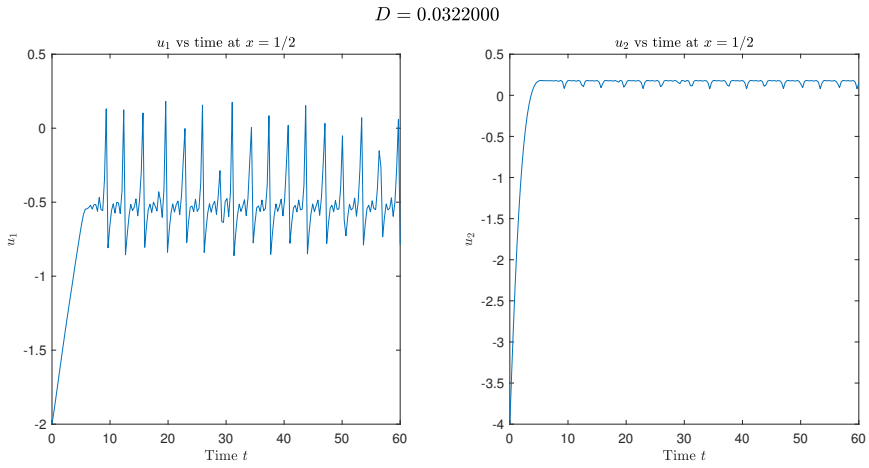


Figure 14. u_1 and u_2 at $x = 1/2$ of reaction-diffusion PDE.

4.6. Takens' time delay embedding

Takens' theorem provides conditions where we can reproduce the dynamical system using sequential data from a single trajectory rather than processing all the trajectories in parallel. Since all the trajectories from the reaction-diffusion cannot be processed together in parallel with available computing resources, Takens' theorem can be used to reconstruct the dynamical system using time delay embedding with one trajectory at a time.

Takens' time delay embedding was introduced in [62] and provided conditions under which a smooth attractor can be reconstructed from the observations made with an observable function. Suppose we have a d-dimensional dynamical system given by a state vector x_t, which is continuous. Further, assume that we have one observable function $y(t)$, which is coupled to all components of x_t. Then, a k dimensional vector of observations can be created by considering k time lagged observations with period τ of $y(t)$, that is, $[\cdots, y_{t-2\tau}, y_{t-\tau}, y_t, y_{t+\tau}, y_{t+2\tau}, \cdots]$, so on and so forth. As $k \to \infty$, the system becomes deterministic and predictable. *Takens'* theorem states that the dynamics of the lagged vector become deterministic at a finite dimension. The finite dimension is given by $k < 2d + 1$.

Let us formally present the *Takens'* embedding theorem.

Theorem 4.1 (**Takens's Embedding Theorem**). *Let $\dot{X} = f(X)$ be a dynamical system defined on the manifold M. $f : M \mapsto M$, and f is smooth. Suppose that the dynamics f has the strange attractor [63] $\mathbb{A}$ with Minkowski-Bouligand dimension [64] $d_{\mathbb{A}}$. Using the Whitney's embedding theorem [65, 66], $\mathbb{A}$ can be embedded in k-dimensional Euclidean space with $k > 2d_{\mathbb{A}}$. That is, there is a diffeomorphism ϕ that maps $\mathbb{A}$ into $\mathbb{R}^k$ such that the derivative of ϕ has full rank.*

Building on Theorem 4.1, it is possible to construct a vector using only a single trajectory from the flow of trajectories of 4.2. In the work presented in this paper, each of the trajectories u_1 and u_2 are time embedded into a vector of size 5. It is further explained by using the trajectory at $x = 1/2$. The values of the trajectory at $x = 1/2$ are isolated for both u_1 and u_2. Since the values are discrete, τ is taken to be the time steps available. Let the time stamp we are considering be $t = t_0$, then the scalar

$u_1\,(x = 1/2, t = t_0)$ is composed to be a vector given by

$$\begin{bmatrix} u_1\,(x = 1/2, t_0 - 2) \\ u_1\,(x = 1/2, t_0 - 1) \\ u_1\,(x = 1/2, t_0) \\ u_1\,(x = 1/2, t_0 + 1) \\ u_1\,(x = 1/2, t_0 + 2) \end{bmatrix}.$$

$$[u_1\,(x = 1/2, t_0 - 2), u_1\,(x = 1/2, t_0 - 1), u_1\,(x = 1/2, t_0), u_1\,(x = 1/2, t_0 + 1), u_1\,(x = 1/2, t_0 + 2)]^T.$$

When both u_1 and u_2 are considered, it becomes the following vector.

$$\begin{bmatrix} u_1\,(x = 1/2, t_0 - 2) \\ u_1\,(x = 1/2, t_0 - 1) \\ u_1\,(x = 1/2, t_0) \\ u_1\,(x = 1/2, t_0 + 1) \\ u_1\,(x = 1/2, t_0 + 2) \\ u_2\,(x = 1/2, t_0 - 2) \\ u_2\,(x = 1/2, t_0 - 1) \\ u_2\,(x = 1/2, t_0) \\ u_2\,(x = 1/2, t_0 + 1) \\ u_2\,(x = 1/2, t_0 + 2) \end{bmatrix}.$$

Then, the trajectories for different x values are fed to the neural network serially.

The first hidden layer had $3,200$ nodes while the second hidden layer had 100 nodes, with a latent vector size of 6 the network produced an accuracy $> 70\%$. Hence this ROM is of dimension 6. Using this ROM, the *Koopman* eigenvectors were computed.

For the simulation of the eigenpair computations, $q = 3e^{-\frac{z_1^2+z_2^2+z_3^2+z_4^2+z_5^2+z_6^2}{10}}$ was taken as the observation function. Similar to the previous experiment, we have produced the graphs corresponding to the first 10 modes.

Figure 15 displays graphs that illustrate how the error changes with the eigenvalue of the initial 10 modes. The binary nature of the graphs are observed again. While the graphs of the odd modes exhibit a similar shape, the graphs of the even modes display a shape that is different yet similar to each other. Figure 16 depicts the graphs of the computed $h(s)$ functions of the corresponding modes. It is visible that there is a "saw-tooth" nature to all the graphs even though there are two distinctive types of graphs. Figure 17 depicts the error variation with mode. The absolute error is initially around 8.9 but decreases to slightly below 8.4, where it then remains constant.

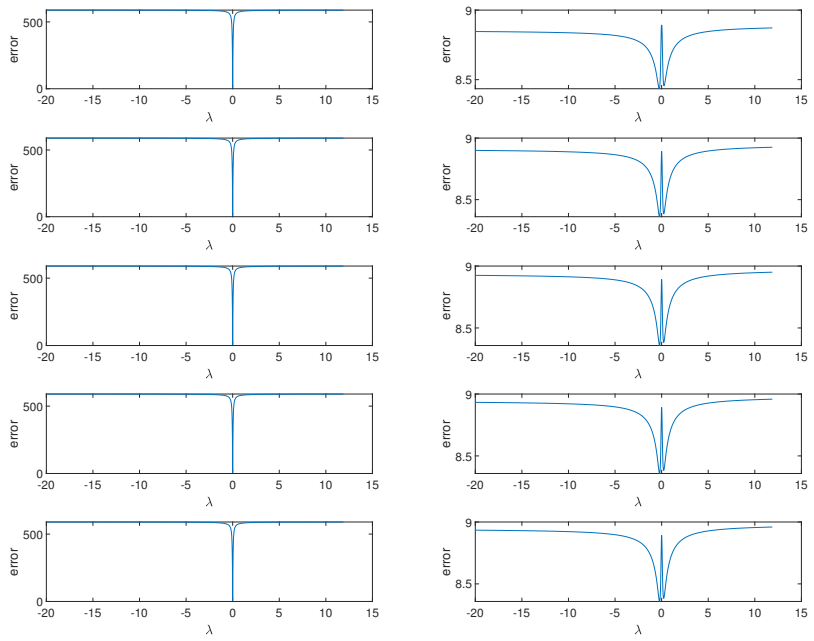


Figure 15. Error vs λ variation through the first 10 modes Reaction-Diffusion PDE.

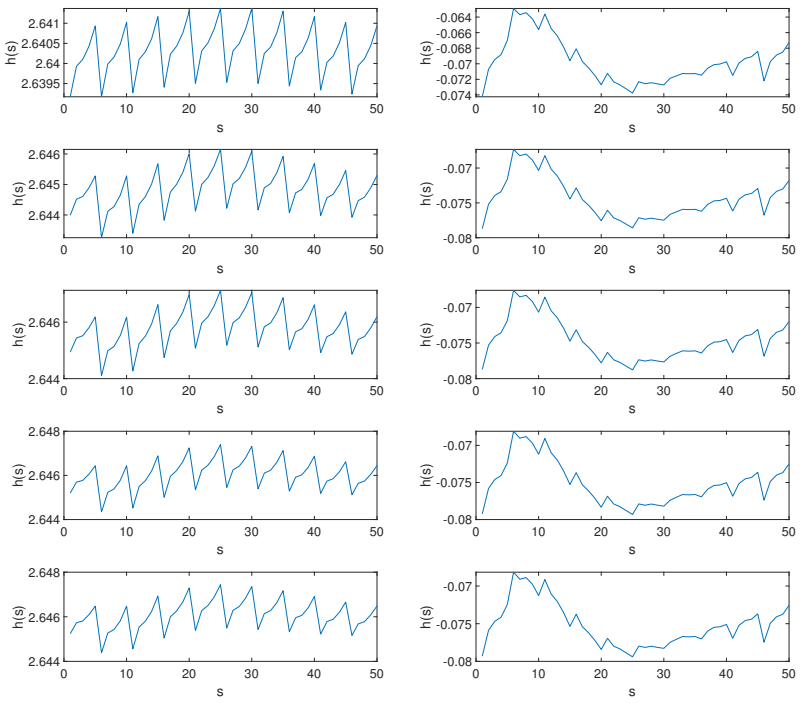


Figure 16. $h(s)$ of the first 10 modes reaction-diffusion PDE.

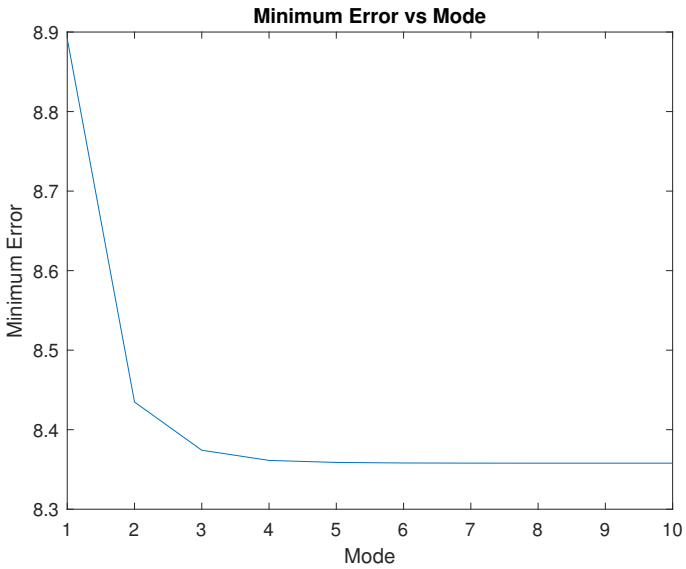


Figure 17. Minimum error vs. mode reaction-diffusion PDE.

5. Conclusions and further work

In this paper we brought out deep autoencoding as a technique for uncovering simple geometries of complex dynamical systems. Autoencoding works as a geometric transformation. The transformed stream of data is then used to compute a dictionary of Koopman eigenpairs. The technique used to compute Koopman eigenpairs is different from the conventional technique introduced in the paper [3]. In that conventional technique the eigenpairs are determined, using the data streams as it is presented. In this paper, we improved on that and computed the eigenpairs using the transformed geometry.

The transformation minimized the vector size of the input data by compressing the data driven dynamical system into a low-dimensional manifold. This reduced the required computational resources to compute the eigenpairs. The transformed data stream was then a reduced order model of the full-order dynamical system.

When the ROM is formed using an AE, there is no mechanism to govern the formulation of the model according to our need. This is because the autoencoding is an unsupervised learning technique. To overcome this, a nontrivial loss function can be introduced. This loss function could partly depend on the latent variables as well as the output of the neural network. Even though this is still unsupervised learning, the ROM can be governed using a nontrivial loss function, and this will, in turn, lead to less error in Koopman eigenpair calculations.

Use of AI tools declaration

The authors declare they have not used Artificial Intelligence (AI) tools in the creation of this article.

Acknowledgments

Neranjaka Jayarathne acknowledges support from the Office of Naval Research. Erik M. Bollt acknowledges support from the Office of Naval Research, Army Research Office, Air Force Office of Scientific Research and the National Institute of Health-Collaborative Research in Computational Neuroscience. The authors are also grateful for the thorough and insightful comments provided by anonymous referees, which greatly contributed to the improvement of this manuscript.

Conflict of interest

The authors confirm that this article content has no conflict of interest.

References

1. S. Brunton, J. Kutz, Data-driven science and engineering: Machine learning, dynamical systems, and control, Cambridge University Press, 2022. https://doi.org/10.1017/9781009089517
2. E. Bollt, N. Santitissadeekorn, Applied and computational measurable dynamics, SIAM, 2013. https://doi.org/10.1137/1.9781611972641
3. E. Bollt, Geometric considerations of a good dictionary for Koopman analysis of dynamical systems: Cardinality, "primary eigenfunction," and efficient representation, *Commun. Nonlinear Sci.*, **100** (2021), 105833. https://doi.org/10.1016/j.cnsns.2021.105833
4. M. Budišić, R. Mohr, I. Mezić, Applied koopmanism, *Chaos: An Interdisciplinary J. Nonlinear Sci.*, **22** (2012), 047510. https://doi.org/10.1063/1.4772195
5. J. Kutz, S. Brunton, B. Brunton, J. Proctor, Dynamic mode decomposition: data-driven modeling of complex systems, SIAM, 2016.
6. Y. Lan, I. Mezić, Linearization in the large of nonlinear systems and Koopman operator spectrum, *Physica D: Nonlinear Phenomena*, **242** (2013), 42–53. https://doi.org/10.1016/j.physd.2012.08.017
7. A. Avila, I. Mezić, Data-driven analysis and forecasting of highway traffic dynamics, *Nat. Commun.*, **11** (2020), 1–16. https://doi.org/10.1038/s41467-020-15582-5
8. I. Mezić, Spectral properties of dynamical systems, model reduction and decompositions, *Nonlin. Dynam.*, **41** (2005), 309–325. https://doi.org/10.1007/s11071-005-2824-x
9. I. Mezić, Spectrum of the Koopman operator, spectral expansions in functional spaces, and state-space geometry, *J. Nonlinear Sci.,* **30** (2020), 2091–2145. https://doi.org/10.1007/s00332-019-09598-5
10. I. Mezić, A. Banaszuk, Comparison of systems with complex behavior, *Physica D*, **197** (2004), 101–133. https://doi.org/10.1016/j.physd.2004.06.015
11. C. Rowley, I. Mezić, S. Bagheri, P. Schlatter, D. Henningson, Spectral analysis of nonlinear flows, *J. Fluid Mech.*, **641** (2009), 115–127. https://doi.org/10.1017/S0022112009992059
12. P. Schmid, Dynamic mode decomposition of numerical and experimental data, *J. Fluid Mech.*, **656** (2010), 5–28. https://doi.org/10.1017/S0022112010001217

13. M. Jovanovic, P. Schmid, J. Nichols, Low-rank and sparse dynamic mode decomposition, *Center Turbulence Res. Annual Res. Briefs*, **2012** (2012), 139–152.

14. I. Kevrekidis, C. Rowley, M. Williams, A kernel-based method for data-driven Koopman spectral analysis, *J. Comput. Dynam.*, **2** (2016), 247–265.

15. M. Williams, I. Kevrekidis, C. Rowley, A data–driven approximation of the koopman operator: Extending dynamic mode decomposition, *J. Nonlinear Sci.*, **25** (2015), 1307–1346. https://doi.org/10.1007/s00332-015-9258-5

16. Q. Li, F. Dietrich, E. Bollt, I. Kevrekidis, Extended dynamic mode decomposition with dictionary learning: A data-driven adaptive spectral decomposition of the Koopman operator, *Chaos: An Interdisciplinary J. Nonlinear Sci.*, **27** (2017), 103111. https://doi.org/10.1063/1.4993854

17. E. Kaiser, J. Kutz, S. Brunton, Data-driven approximations of dynamical systems operators for control, *The Koopman Operator In Systems And Control: Concepts, Methodologies, And Applications*, (2020), 197–234. https://doi.org/10.1007/978-3-030-35713-9_8

18. I. Mezić, Analysis of fluid flows via spectral properties of the Koopman operator, *Annual Rev. Fluid Mech.*, **45** (2013), 357–378. https://doi.org/10.1146/annurev-fluid-011212-140652

19. P. Gaspard, Chaos, scattering and statistical mechanics, *Chaos*, 2005.

20. R. Abraham, J. Marsden, Foundations of mechanics, American Mathematical Soc., 2008. https://doi.org/10.1090/chel/364

21. A. Ackleh, E. Allen, R. Kearfott, P. Seshaiyer, Classical and modern numerical analysis: Theory, methods and practice, Crc Press, 2009. https://doi.org/10.1201/b12332

22. D. Floryan, M. Graham, Charts and atlases for nonlinear data-driven models of dynamics on manifolds, *ArXiv Preprint ArXiv:2108.05928*, (2021).

23. C. Fefferman, S. Mitter, H. Narayanan, Testing the manifold hypothesis, *J. Am. Math. Soc.*, **29** (2016), 983–1049. https://doi.org/10.1090/jams/852

24. H. Narayanan, S. Mitter, Sample complexity of testing the manifold hypothesis, *Adv. Neural Inf. Process. Syst.*, **23** (2010).

25. A. Izenman, Introduction to manifold learning, *Wires. Comput. Stat.*, **4** (2012), 439–446. https://doi.org/10.1002/wics.1222

26. J. Tenenbaum, V. Silva, J. Langford, A global geometric framework for nonlinear dimensionality reduction, *Science*, **290** (2000), 2319–2323. https://doi.org/10.1126/science.290.5500.2319

27. S. Roweis, L. Saul, Nonlinear dimensionality reduction by locally linear embedding, *Science*, **290** (2000), 2323–2326. https://doi.org/10.1126/science.290.5500.2323

28. M. Balasubramanian, E. Schwartz, The isomap algorithm and topological stability, *Science*, **295** (2002), 7. https://doi.org/10.1126/science.295.5552.7a

29. M. Belkin, P. Niyogi, Laplacian eigenmaps and spectral techniques for embedding and clustering, *Adv. Neural Inf. Process. Syst.*, **14**, 2001. https://doi.org/10.7551/mitpress/1120.003.0080

30. Z. Ma, Z. Zhan, Z. Feng, J. Guo, Manifold learning based on straight-like geodesics and local coordinates, *IEEE T. Neural Net. Lear.*, **32** (2020), 4956–4970. https://doi.org/10.1109/TNNLS.2020.3026426

31. W. Boothby, W. Boothby, An introduction to differentiable manifolds and Riemannian geometry, Revised, Gulf Professional Publishing, 2003.

32. X. Chen, J. Weng, W. Lu, J. Xu, J. Weng, Deep manifold learning combined with convolutional neural networks for action recognition, *IEEE T. Neural Net. Lear.*, **29** (2017), 3938–3952. https://doi.org/10.1109/TNNLS.2017.2740318

33. R. Wang, X. Wu, J. Kittler, Symnet: A simple symmetric positive definite manifold deep learning method for image set classification, *IEEE T. Neural Net. Lear.*, **33** (2021), 2208–2222. https://doi.org/10.1109/TNNLS.2020.3044176

34. K. Lee, K. Carlberg, Model reduction of dynamical systems on nonlinear manifolds using deep convolutional autoencoders, *J. Comput. Phys.*, **404** (2020), 108973. https://doi.org/10.1016/j.jcp.2019.108973

35. J. Bakarji, K. Champion, J. Nathan Kutz, S. L. Brunton, Discovering governing equations from partial measurements with deep delay autoencoders, *P Royal Soc. A*, **479** (2023), 20230422. https://doi.org/10.1098/rspa.2023.0422

36. Y. LeCun, PhD thesis: Modeles connexionnistes de l'apprentissage (connectionist learning models), (Universite P. et M. Curie (Paris 6), 1987.

37. J. Zhai, S. Zhang, J. Chen, Q. He, Autoencoder and its various variants, *2018 IEEE International Conference On Systems, Man, And Cybernetics (SMC)*, (2018), 415–419. https://doi.org/10.1109/SMC.2018.00080

38. S. Gu, B. Kelly, D. Xiu, Autoencoder asset pricing models, *J. Econometrics*, **222** (2021), 429–450. https://doi.org/10.1016/j.jeconom.2020.07.009

39. C. Bishop, N. Nasrabadi, Pattern recognition and machine learning, Springer,2006.

40. B. Karlik, A. Olgac, Performance analysis of various activation functions in generalized MLP architectures of neural networks, *Int. J. Artif. Intell. Expert Syst.*, **1** (2011), 111–122.

41. P. Pant, R. Doshi, P. Bahl, A. Barati Farimani, Deep learning for reduced order modelling and efficient temporal evolution of fluid simulations, *Phys. Fluids*, **33** (2021), 107101. https://doi.org/10.1063/5.0062546

42. Z. Bai, Krylov subspace techniques for reduced-order modeling of large-scale dynamical systems, *Appl. Numer. Math.*, **43** (2002), 9–44. https://doi.org/10.1016/S0168-9274(02)00116-2

43. D. Lucia, P. Beran, W. Silva, Reduced-order modeling: new approaches for computational physics, *Prog. Aerosp. Sci.*, **40** (2004), 51–117. https://doi.org/10.1016/j.paerosci.2003.12.001

44. N. Kazantzis, C. Kravaris, L. Syrou, A new model reduction method for nonlinear dynamical systems, *Nonlinear Dynam.*, **59** (2010), 183–194. https://doi.org/10.1007/s11071-009-9531-y

45. O. San, R. Maulik, Neural network closures for nonlinear model order reduction, *Adv. Comput. Math.*, **44** (2018), 1717–1750. https://doi.org/10.1007/s10444-018-9590-z

46. R. Fu, D. Xiao, I. Navon, F. Fang, L. Yang, C. Wang, et al., A non-linear non-intrusive reduced order model of fluid flow by auto-encoder and self-attention deep learning methods, *Int. J. Numer. Meth. Eng.*, (2023). https://doi.org/10.1002/nme.7240

47. N. Aubry, P. Holmes, J. Lumley, E. Stone, The dynamics of coherent structures in the wall region of a turbulent boundary layer, *J. Fluid Mech.*, **192** (1988), 115–173. https://doi.org/10.1017/S0022112088001818

48. G. Berkooz, P. Holmes, J. Lumley, The proper orthogonal decomposition in the analysis of turbulent flows, *Annu. Rev. Fluid Mech.*, **25** (1993), 539–575. https://doi.org/10.1146/annurev.fl.25.010193.002543

49. P. Holmes, J. Lumley, G. Berkooz, C. Rowley, Turbulence, coherent structures, dynamical systems and symmetry, Cambridge university press, 2012. https://doi.org/10.1017/CBO9780511919701

50. H. Hotelling, Analysis of a complex of statistical variables into principal components, *J. Educ. Psychol.*, **24** (1933), 417–441. https://psycnet.apa.org/doi/10.1037/h0071325

51. E. Lorenz, Empirical orthogonal functions and statistical weather prediction, Massachusetts Institute of Technology, Department of Meteorology Cambridge, 1956.

52. M. Loeve, Probability theory: foundations, random sequences, New York, NY: Van Nostrand, 1955.

53. K. Taira, S. Brunton, S. Dawson, C. Rowley, T. Colonius, B. McKeon, et al., Modal analysis of fluid flows: An overview, *Aiaa J.*, **55** (2017), 4013–4041. https://doi.org/10.2514/1.J056060

54. P. Schmid, L. Li, M. Juniper, O. Pust, Applications of the dynamic mode decomposition, *Theor. Comp. Fluid Dyn.*, **25** (2011), 249–259. https://doi.org/10.1007/s00162-010-0203-9

55. B. Brunton, L. Johnson, J. Ojemann, J. Kutz, Extracting spatial–temporal coherent patterns in large-scale neural recordings using dynamic mode decomposition, *J. Neurosci. Meth.*, **258** (2016), 1–15. https://doi.org/10.1016/j.jneumeth.2015.10.010

56. E. Berger, M. Sastuba, D. Vogt, B. Jung, H. Amor, Dynamic mode decomposition for perturbation estimation in human robot interaction, *The 23rd IEEE International Symposium On Robot And Human Interactive Communication*, (2014), 593–600. https://doi.org/10.1109/ROMAN.2014.6926317

57. B. Koopman, Hamiltonian systems and transformation in Hilbert space, *P. Natl. Acad. Sci.*, **17** (1931), 315–318. https://doi.org/10.1073/pnas.17.5.315

58. E. Bollt, Q. Li, F. Dietrich, I. Kevrekidis, On matching, and even rectifying, dynamical systems through Koopman operator eigenfunctions, *SIAM J. Appl. Dyn. Syst.*, **17** (2018), 1925–1960. https://doi.org/10.1137/17M116207X

59. T. Kanamaru, Van der Pol oscillator, *Scholarpedia*, 2007. Available from: `http://www.scholarpedia.org/article/Van_der_Pol_oscillator`

60. I. Triandaf, I. Schwartz, Karhunen-Loeve mode control of chaos in a reaction-diffusion process, *Phys. Rev. E*, **56** (1997), 204–212. https://doi.org/10.1103/PhysRevE.56.204

61. H. Goldstein, C. Poole, J. Safko, Classical mechanics, American Association of Physics Teachers, 2002.

62. F. Takens, Detecting strange attractors in turbulence, *Dynamical Systems And Turbulence, Warwick 1980: Proceedings Of A Symposium Held At The University Of Warwick 1979/80*, (2006), 366–381. https://doi.org/10.1007/BFb00919

63. D. Ruelle, F. Takens, On the nature of turbulence, *Les Rencontres Physiciens-mathématiciens De Strasbourg-RCP25*, **12** (1971), 1–44.

64. K. Falconer, Fractal geometry: Mathematical foundations and applications, John Wiley & Sons, 2004. 10.1002/0470013850

65. M. Adachi, Embeddings and immersions, American Mathematical Soc., 2012. https://doi.org/10.1090/mmono/124

66. A. Skopenkov, Embedding and knotting of manifolds in Euclidean spaces, *London Math. Soc. Lecture Note Series*, **347** (2008), 248. https://doi.org/10.1017/CBO9780511666315.008