Modelling from Measurements Homework Assignment

Stefano Riva[1,*]

[1] *Politecnico di Milano, Department of Energy - Nuclear Energy Division, Via Lambruschini 4, 20156 Milan, Italy*

[*] *stefano.riva@polimi.it*

Abstract: In the last few years, data-driven modelling has become more and more studied and a lot of novel approaches have been proposed in literature. Previously, most the efforts were concentrated on simulations of theoretical models, based on the numerical solution of parametric PDEs. The main advantages of building data-drive models is that they are based on real evaluations on the physical fields (even though they may be affected by uncertainties) and a better insight on the system itself may be obtained. This work aims at assessing the performance of some of these techniques by applying them to either literature datasets or numerical solution of simple dynamical systems.

1. Introduction and Overview

Theoretical modelling has been widely used in engineering and science both for the design of physical systems and for making predictions; as time passes, more and more complexity has been introduced so that accurate models have been derived. The typical mathematical description is carried out with parametric Partial Differential Equations (PDEs), directly derived from conservation principles (e.g., Boltzmann equations, heat diffusion, Navier-Stokes, ...). They are usually referred to as *High Fidelity* models and they have to be solved with suitable numerical methods, such as Finite Elements or Finite Volumes, which are typically computationally demanding, even on high performance computers; therefore, they are not suited for control purposes and real-time applications [Schilders et al., 2008]. In this context, Reduced Order Modelling (ROM) methods [Quarteroni et al., 2015] represent a very promising tool, widely studied in literature.
In the last few years there have also been many improvements in the framework of Data Assimilation (DA) [Carrassi et al., 2018, Brunton and Kutz, 2019], which is a mathematical discipline that deals with the combination of theoretical modelling and experimental observations. They are based on machine learning techniques or on a coupling between Reduced Order Modelling and experimental data, in the framework of Hybrid Data Assimilation (HDA) methods. Nowadays, a lot of efforts are concentrated into the *modelling from data* approach, in which a surrogate model is typically trained using some *a priori* knowledge, either coming from experiments or numerical models.
In this work, some techniques, suited for making predictions on physical systems and for discovering the best fit model from the data, will be tested on literature data and famous dynamical systems. The structure of the paper is the following: in Section 2, the theoretical background will be briefly presented; in Section 3, some comments on the algorithm implementation (made either in Python or MATLAB) will be provided; in Section 4, the main computational results will be discussed; finally, Section 5 is devoted to a short summary and the conclusions.

2. Theoretical Background

Let $\mathbf{u} \in \mathbb{R}^n$ be the state space vector variables[1], so that a generic dynamical system can be described as

$$\frac{\partial \mathbf{u}}{\partial t} = \mathcal{F}(\mathbf{u}, \mathbf{x}, t; \mu) \qquad \mathbf{x} \in \Omega \subset \mathbb{R}^d,\ t \in \mathbb{R}^+,\ \mu \in \mathcal{D} \subset \mathbb{R}^p, \tag{1}$$

with $d = \{1, 2, 3\}$ and $\mathcal{D}$ as the parameter space of dimension p. Let us consider the case in which the parameters μ are fixed and let us introduce the data matrix $X \in \mathbb{R}^{n \times m}$, defined as

$$X = \begin{bmatrix} u(x_1, t_1) & u(x_1, t_2) & \dots & u(x_1, t_m) \\ | & | & | & | \\ u(x_n, t_1) & u(x_n, t_2) & \dots & u(x_n, t_m) \end{bmatrix}, \tag{2}$$

[1] If the dynamical system comes from the spatial discretisation of a PDE, the j-th component of $\mathbf{u}$ is nothing but $u_j = u(x_j, \cdot)$.

in the following, the $i-$th column of X may be indicated with $\mathbf{x}_i$ or $X(:,i)$.

2.1. Singular Value Decomposition. (SVD)

This technique is well known in linear algebra for matrix decomposition, in our context it is a very useful tool to look for low-rank approximations: in fact, when a large amount of data is available, they can be projected onto a smaller space and a model on the low rank variables may be derived. This technique is a foundation of the Dynamic Mode Decomposition (DMD) and the Proper Orthogonal Decomposition (POD), the state-of-the-art of intrusive Reduced Order Modelling techniques [Sirovich, 1987, Berkooz et al., 1993].
Let us consider a generic matrix $A \in \mathbb{R}^{n\times m}$, the SVD reads[2]

$$A = U\Sigma V^T, \qquad U \in \mathbb{R}^{n\times n},\ \Sigma \in \mathbb{R}^{n\times m}, V \in \mathbb{R}^{m\times m}, \tag{3}$$

in which both U and V are unitary matrices. If we sought for an approximation of A, there is the truncated version till rank $r < \mathrm{rk}(A)$ of the SVD, i.e.

$$A \approx \tilde{A}_r = \tilde{U}\tilde{\Sigma}\tilde{V}^T, \qquad \tilde{U} \in \mathbb{R}^{n\times r},\ \tilde{\Sigma} \in \mathbb{R}^{r\times r}, \tilde{V} \in \mathbb{R}^{m\times r}. \tag{4}$$

The columns of the matrix U are orthonormal vectors, also called *spatial modes*, or simply *modes*. In this framework, these vectors are related to the amount of energy that any mode carries [Quarteroni et al., 2015, Brunton and Kutz, 2019]. A very important information on this is provided from the eigenvalues of Σ (i.e., its diagonal), in particular the error, that is performed when approximating the matrix, can be proved to be

$$\left|\left|A - \tilde{A}_r\right|\right|_F = \sqrt{\sum_{i=r+1}^{\mathrm{rk}(A)} \sigma_i^2}. \tag{5}$$

The eigenvalues σ_i^2 can be proved to be the eigenvalues of the correlation matrix $C = AA^T$. In the end, this method is useful when an optimal coordinate system, onto which the data are projected, is sought or when a model on the low rank variables has to be derived.

2.2. Dynamic Mode Decomposition (DMD)

This method has been introduced in 2008 by [Schmid and Sesterhenn, 2008, Schmid, 2010] and its aim consists in finding the best **linear** model that approximates a dynamical (usually non-linear) system. This technique has been linked to the Koopman theory [Rowley et al., 2009, Brunton et al., 2021], which states that any dynamical system can be replaced by a linear model when moving to an infinite-dimension observable space.
Let us define two matrices $X_{old} \in \mathbb{R}^{n\times(m-1)}$ and $X_{new} \in \mathbb{R}^{n\times(m-1)}$, starting from our data matrix X, as in Eq. (2)

$$X_{old}(:,j) = X(:,j) \qquad j = 1,\dots,m-1; \qquad\qquad X_{new}(:,k) = X(:,k) \qquad k = 2,\dots,m; \tag{6}$$

in this way, we are connecting what happens at time t_n with the next step at t_{n+1}, with the following approximation $\mathbf{x}_{new}(t_{n+1}) \approx A\mathbf{x}_{old}(t_n)$; the dynamic matrix A can be found as $X_{new}X_{old}^+$, given X_{old}^+ the pseudo-inverse[3] of X_{old} [Brunton and Kutz, 2019]. Instead of reasoning in terms of the full-rank data, we can project onto the low-rank variables (i.e. $X_{old} \approx \tilde{U}\tilde{\Sigma}\tilde{V}^T$), so that the matrix A may be approximated as

$$A \approx \tilde{A} = X_{new}\tilde{V}\tilde{\Sigma}^{-1}\tilde{U}^T. \tag{7}$$

Now, we are ready to find the **DMD eigenvalues**, necessary to predict the state of the system, as $\tilde{A}W = W\Lambda$[4]. Then, we can compute the high dimensional DMD modes $\mathbf{\Phi}$ as $X_{new}\tilde{V}\tilde{\Sigma}^{-1}W$, the columns of $\mathbf{\Phi}$ are the eigenvectors of the full matrix A, whereas the eigenvalues are still given by Λ. This last property is fundamental and it let us work in the low rank variables, instead of the high dimensional one: A and $\tilde{A}$ have the same set of eigenvalues.

[2] In this work, only real data are going to be considered, however the generic SVD takes the conjugate transposed of V, instead of the simple transpose.
[3] This can be linked to the SVD of X_{old}.
[4] The columns of W are the eigenvector, whereas the diagonal of Λ represents the eigenvalues.

In the end, the state can be predicted using a continuous time representation: let $\mathbf{b}$ be the solution of the following linear system $(W\Lambda)\mathbf{b} = \mathbf{x}_1 \approx V(1,:)$ and finally the state may be expressed as

$$\mathbf{x}(t) = \sum_{j=1}^{r} \phi_j e^{\omega_j t} b_j, \qquad \omega_j = \frac{\ln(\lambda_j)}{\Delta t}, \tag{8}$$

given ϕ_j as the $j-$th column of matrix $\mathbf{\Phi}$. From Eq. (8), the time dependence is expressed through exponential, this implies that oscillations are given by complex conjugate eigenvalues; furthermore, in the limit for $t \to +\infty$, the state may go either to 0 or $+\infty$, depending on the sign of the real part of λ_j.

2.2.1. Bagging Optimized DMD (BOP-DMD)

In order to make the DMD approach more robust, **bagging** trick can be very useful: we can apply the DMD by randomly pick $p < m$ columns of the data matrix X several times. In this way, we would get several values for b_j, ϕ_j and λ_j that will let us find average values and standard deviation of these quantities. This kind of information can be very useful, in order to have an idea on the completeness of the dataset and on the robustness of the DMD model.

2.2.2. Time Delay DMD

Other way to improve the predictions consists in building the Hankel matrix H_s from X as

$$H_s = \begin{bmatrix} \mathbf{x}(t_1) & \mathbf{x}(t_2) & \dots & \mathbf{x}(t_{m-s}) \\ \mathbf{x}(t_2) & \mathbf{x}(t_3) & \dots & \mathbf{x}(t_{m-s+1}) \\ \vdots & \vdots & & \vdots \\ \mathbf{x}(t_s) & \mathbf{x}(t_{s+1}) & \dots & \mathbf{x}(t_m) \end{bmatrix} \in \mathbb{R}^{(n\cdot s)\times(m-s)}, \tag{9}$$

given $s \geq 1$ and then, the DMD is applied to this matrix. This strategy is particularly useful when the dimension n of the vector state space is much smaller than m, the number of time measures, resulting in a horizontal shape of matrix X.

2.3. SINDy

The DMD has let us find the best linear model that fits the data, however we are aware that this is nothing but an approximation of the true model, sometimes this is a good approach, however it may be too inaccurate in some applications. Let us suppose to have access to the data matrix X and its time derivative $\frac{\partial X}{\partial t}$, our goal is to find a way to define the model f_Θ, s.t.

$$\frac{\partial X}{\partial t} = f_\Theta(X). \tag{10}$$

The Sparse Identification of Nonlinear Dynamics (SINDy) algorithm [Brunton et al., 2016] is a novel approach that is able to search through all possible model structures and to determine the main dominant terms (e.g., linear or parabolic terms or first order derivatives). In a nutshell, the aim of SINDy consists in finding an expression of the general model $f_\Theta(\mathbf{x})$, which is usually approximated with the following expansion

$$f_\Theta(\mathbf{x}) \approx \Theta(\mathbf{x})\boldsymbol{\xi} = \sum_k \theta_k(\mathbf{x})\xi_k, \tag{11}$$

where Θ is a library of functions[5], which can be defined as

$$\Theta(X) = \begin{bmatrix} 1 & X & X^2 & \dots & X^d & \dots & \sin(X) & \dots & e^X & \dots \end{bmatrix}. \tag{12}$$

Therefore, the dynamical system (10) can be represented as $\frac{\partial X}{\partial t} \approx \Theta(X)\Xi$, in which the $k-$th column of Ξ is nothing but $\boldsymbol{\xi}_k$, that is a vector of coefficients determining the dominant terms. In order to make this approach sparse, i.e. considering **only** the dominant dynamics, a convex L^1-regularised sparse regression can be introduced:

$$\boldsymbol{\xi}_k = \arg\min_{\boldsymbol{\xi}'_k} \left|\left| \frac{\partial \mathbf{x}_k}{\partial t} - \Theta(\mathbf{x}_k)\boldsymbol{\xi}'_k \right|\right|_2 + \lambda \left|\left|\boldsymbol{\xi}'_k\right|\right|_1. \tag{13}$$

Two main algorithms are implemented in SINDy, i.e. LASSO and the Sequential Thresholded Least-Squares (STLS).

[5] This tool may be used also to find PDEs, the library will be filled with spatial differential operators as $\frac{\partial}{\partial x}, \frac{\partial^2}{\partial x^2}$.

In practice, the time derivative is rarely available, there two main ways to overcome this problem: *i)* the time derivative may be approximated with a numerical method, which must be robust in presence of random noise [Brunton et al., 2016, Brunton and Kutz, 2019]; *ii)* the time derivative may be discretised with a numerical technique and the *difference equation* can be considered.
In the end, the algorithm can be even more robust when **bagging** is applied, so that a statistical assessment is obtained for any coefficient $\boldsymbol{\xi}_k$.

2.4. Neural Networks

Neural Networks (NNs) have been firstly introduced in the '40s as an attempt to model a biological neuron. Even though they haven't been heavily studied in the second half of the 20th century, nowadays big-tech companies are investing a lot on them, due to their great potential is learning from data. The increase in the computational power through the years, the availability of software and modern hardware have been the main reasons why their importance has increased so much [Goodfellow et al., 2016].
In a nutshell, a NN can be conceived as a non-linear transformation between an input and an output, they can be considered as an extension of the classical linear transforms (e.g., Laplace or Fourier). The basic unit of a NN is called **neuron** and each is characterised by an activation function: a neuron accepts some inputs $[x_1, x_2, \dots, x_n]$ and provides an output z through the function σ, i.e.

$$z = \sigma\left(\sum_{k=1}^{n} \theta_k x_k + b\right), \tag{14}$$

where σ can be either linear or not, important examples are *pureLinear*, *ReLU*, *tanh* or *sigmoid*. A set of neurons compose a layer and adding layers makes the net deeper and deeper, increasing the number of parameters (θ_k) to find.
In addition to the main components defined above, it is required to have an input/output data structure for the training of the net. The main goal of a NN consists in finding the best set of parameters $\hat{\Theta}$ so that the distance between the output and the prediction is lower enough, i.e.

$$\hat{\Theta} = \arg\min_{\Theta} \frac{1}{2} ||Y - f_\Theta(X)||_2^2 + \lambda ||R||_* \tag{15}$$

where X is the input matrix, Y the output and $\lambda ||R||$ indicates a generic regularisation term, properly normed. An optimisation problem is found, which must be numerically solved. The main algorithm is the Gradient Descent (GD) method and its generalisation (Stochastic GD or the *Adam* algorithm) coupled with the Backpropagation technique [Brunton and Kutz, 2019].
From a very simplistic point of view, NNs are nothing but curve fitting and the prediction of a certain net is usually limited by the data that are used to train the model. In fact, at the moment, NNs are useful tools but they are not always good at extrapolating information or at generalisation. This statement is in general true for NN that are not so deep: it has been empirically observed that by increasing the depth of the NN, thus enriching it capacity there is the so-called mystery of **Double Descent**, in which the generalisation error (i.e., extrapolation capability) decreases as the depth increases [Belkin et al., 2019, Berner et al., 2021].

3. Algorithm Implementation and development

The techniques described in the previous sections have been implemented either in MATLAB or Python environment, according to the specific task[6]. In additional to the standard libraries, the BOP-DMD code has been taken from [Askham, 2017] and the implementation of PySINDy by [de Silva et al., 2020, Kaptanoglu et al., 2022] have been used. All the codes that have been used for this work are available on Github.

4. Computational Results

In this section, two main tasks will be considered: the development of linear and non-linear model for a classical Data Science dataset and the development of NNs for ODEs/PDEs.

4.1. Predator-Prey dataset

The dataset is composed by 30 measurements (every 2 years), from 1845 to 1903, of the Canadian population of lynxes and snowshoes. At first a linear model is built with DMD, then non-linear model discovery will be developed.

[6] This choice has been made in order to have at disposal codes for both the environments for future developments.

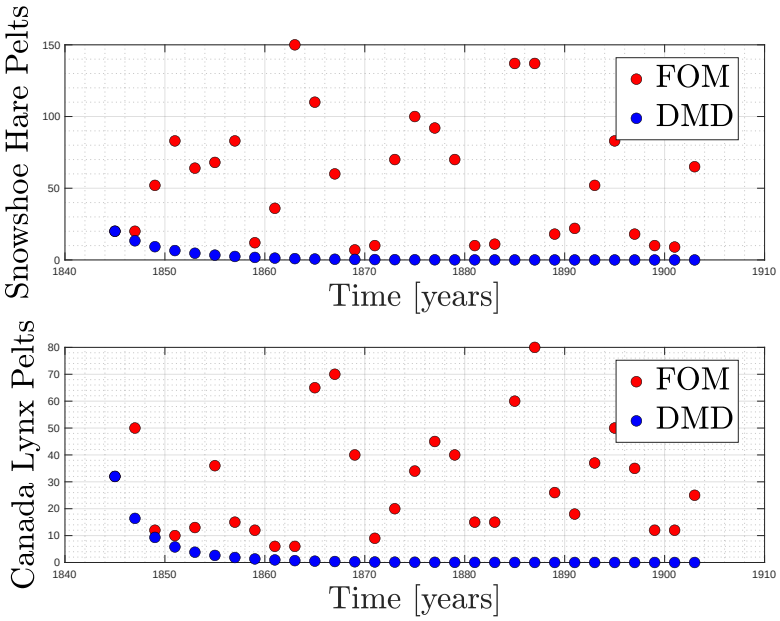


Fig. 1: Lynx-Snowshoe DMD prediction (FOM stands for Full Order Model, in this case it corresponds to the raw data).

4.1.1. Standard DMD

The shape of the data matrix X is 2×30, therefore its rank is, at most, equal to 2, which corresponds to have 2 SVD eigenvalues. This fact has an important impact on the DMD reconstruction: in fact, the linear model prediction will be given by the sum of two exponentials; if the eigenvalues are complex conjugate, oscillations can be predicted, on the other hand with real eigenvalues the prediction will go either to 0 or to $+\infty$. The results are shown in Figure 1. The DMD prediction shows an exponential decay[7], resulting in a very bad reconstruction of the dataset: this is not surprising, due to the scarcity of the data matrix and its low rank. Even bagging the DMD doesn't show any improvements, thus a different approach should be followed, in order to increase the amount of available data.

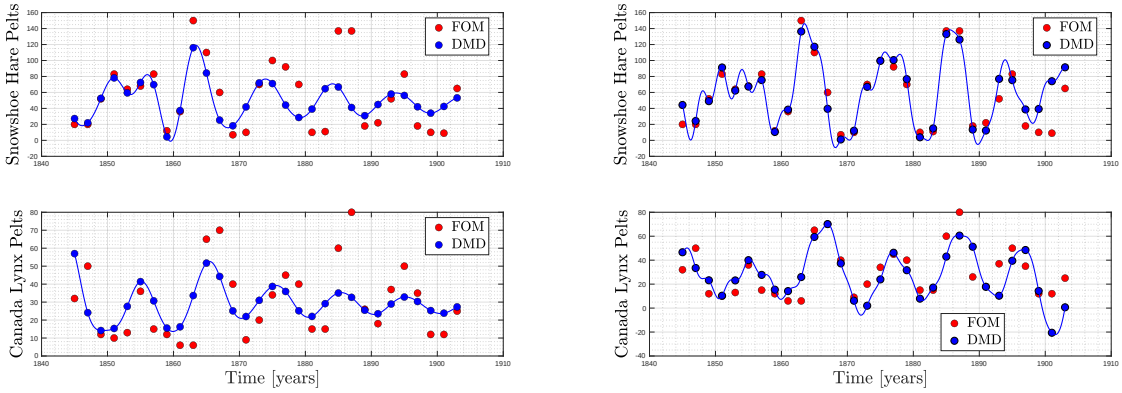
4.1.2. Time-delay DMD

Instead of considering the data matrix X, the Hankel matrix H_s has been built for different values of $s = [5, 10, 15, 20, 25]$. By looking at the SVD eigenvalues decay (in terms of relative energy content), we can say that the best decay is associated to the matrix with the highest rank, corresponding in this case to $s_{opt} = 10$. The truncation has been performed for $r = 11$, since higher order modes are not necessary due to their lower energy content[8]. Both the standard DMD and the optimised one have been tested against the dataset and the prediction is reported in Figure 2. The optimised version is better performing in reconstructing the dataset: in fact, a lower relative error, measured as

$$\epsilon = \frac{||X_{DMD} - X||_F}{||X||_F}, \tag{16}$$

is associated. Nevertheless, even the standard DMD is able to catch the oscillatory behaviour of the populations.
In the end, this first part shows that linear models are able to provide quite good predictions, even with "bad data"; in particular, time-delay coupled with bagging is the best strategy to use. The simplicity of linear models should not be discarded *a priori*, in fact important information on dynamics of the system can be obtained.

[7] This is expected because the DMD eigenvalues are real and negative.
[8] Moreover, choosing $r = 10$ would cause the DMD prediction to be complex, since one of the complex eigenvalues has not its conjugate pair, thus the imaginary part will not be erased.



(a) Standard time-delay DMD.

(b) Optimised time-delay DMD.

Fig. 2: Comparison between the standard time-delay DMD and the optimised version, provided by [Askham, 2017].

4.1.3. Lotka-Volterra equations

This kind of phenomenon is theoretically modelled with the Lotka-Volterra equations

$$\frac{dx}{dt} = (b - p \cdot y) \cdot x, \qquad \frac{dy}{dt} = (r \cdot x - d) \cdot y, \tag{17}$$

in which x represents the preys (i.e., snowshoes), y the predators (i.e., lynxes) and the parameters b, p, r, d are positive quantities, describing the interaction between the species. In order to find an estimation of these parameters, an inverse problem can be defined:

$$\boldsymbol{\mu}^\star = \arg\min_{\boldsymbol{\mu} \in \mathcal{D}} \sum_k \left[x(t_k) - \hat{x}(t_k; \boldsymbol{\mu})\right]^2 + \left[y(t_k) - \hat{y}(t_k; \boldsymbol{\mu})\right]^2 \tag{18}$$

in which $\boldsymbol{\mu} = [b, r, p, d]^T \in \mathcal{D}$ is the vectors of the parameters, $\mathcal{D} \subset \mathbb{R}^4$ their sample space, $\hat{x}(t_k)$ and $\hat{y}(t_k)$ the prediction, given by Eq. (17), at time t_k. This optimisation problem has been solved in MATLAB using the Global Optimisation Toolboox, in particular the *fminsearch* solver has been selected. Figure 3 shows the results of the prediction with respect to the raw data: the agreement is not perfect, however the input data are strongly polluted by noise, hence accurate predictions are not expected. The main issue of such an approach lies on the inverse problem (18), which is usually ill-posed and highly dependent on the initial guess of the optimisation algorithm.

4.1.4. SINDy

The results have been obtained exploiting the code developed by [de Silva et al., 2020, Kaptanoglu et al., 2022]; a complete Python library has been implemented and some of its feature have been exploited for this work. Both built-in and user-defined libraries can be used and several differentiation methods are available (robust even in presence of random noise[9]). Figure 4 shows the coefficients ξ for a polynomial library, of order 2: the main contribution is given by the linear terms, whereas the higher order terms has much lower importance. None of them provides a null contribution, even though the xy coefficients are quite low. The main problem of this dataset is the strong pollution by random noise and the scarsity itself: indeed, the population is sampled every 2 years. In order to improve the results, more data are required: for instance, they could be generated using data augmentation techniques.

4.2. Neural Networks for PDEs

In this section, some NNs will be built and trained to predict how the solution of a certain PDEs advances in time from t to $t + \Delta t$.

[9] As a preliminary check, SINDy has been applied to the Point Kinetics Equations [Duderstadt and Hamilton, 1976], both in presence and absence of noise, to assess the model discovery for this set of ODEs used to describe the kinetics of a nuclear reactor, under some strong assumptions. The code is available on Github. This preliminary assessment shows the potential of this method, to discover models from data, even though it can be seen that the method suffers a bit for high values of random noise.

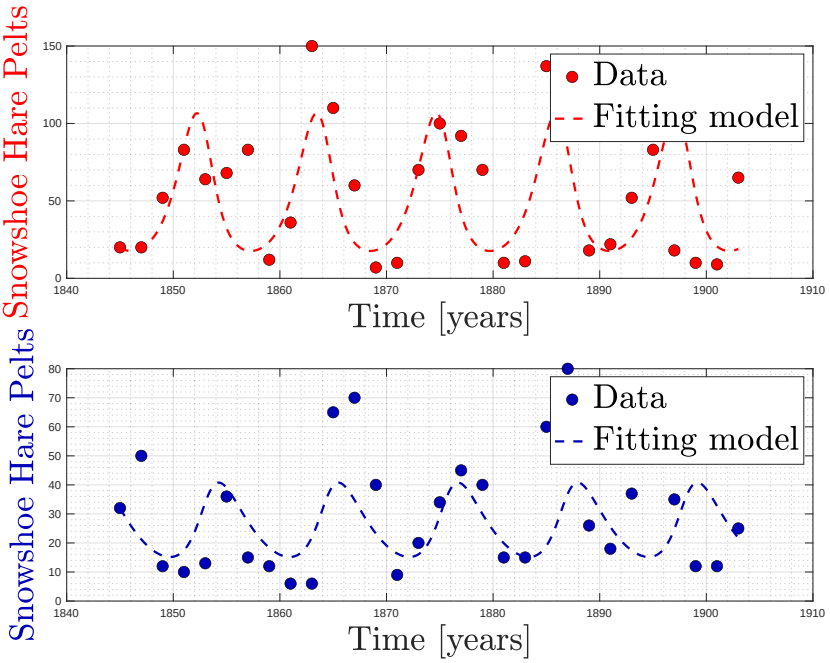


Fig. 3: Comparison between the Lotka-Volterra equations (using the values $\boldsymbol{\mu}^\star$ from Eq. (18)) and the data.

4.2.1. Kuramoto-Sivashinsky equation

Let u be the solution of the Kuramoto-Sivashinsky (KS) equation

$$\frac{\partial u}{\partial t} + \frac{\partial^4 u}{\partial x^4} + u\frac{\partial u}{\partial x} + \frac{\partial^2 u}{\partial x^2} = 0 \quad \text{in } [0,T] \times [0, 2\pi], \tag{19}$$

given an initial condition $u_0(x)$ and periodic boundary conditions. Some applications of the KS equation include flame propagation, flows in pipes and at interfaces, plasmas and chemical reaction dynamics.
At first training data are built by solving numerically the KS equation, using FFT: the time is discretised with a time step $\Delta t = 5 \cdot 10^{-3}$, whereas the spatial variable is divided into 2^7 interval equally spaced. Once the KS equation has been solved, the data are arranged into a matrix X s.t. $X_{ij} = u(x_j, t_i)$, then this matrix is divided into X_{old} and X_{new}, as in Eq. (6)[10]. The state vector, at time t, has been defined as $[t, u(x_1, t), u(x_2, t), \dots, u(x_n, t)]^T$. The network, built to predict the time behaviour of the solution, consists of an input layer with 32 neurons activated with a *sigmoid*, two hidden layers with 64 neurons each, activated with *tanh* and *relu*, and a linear output layer. The cost function is the Mean Square Error and the chosen optimiser is *Adam*.
Figure 5 shows a comparison between the Training data, as a function of time t and space x, and the prediction of the NN. The oscillation at smaller time are quite accurately predicted, whereas the errors are concentrated on higher time $t \geq 1.5\,s$. It should be pointed out that the training data are plotted, thus we can only state that, given the same initial condition and same parameters the NN has learnt how to advance from t to $t + \Delta t$, even in presence of chaotic function as the solution of the KS equation, a NN is able to provide a good approximation; however, if some parameters are changed, it is not always guaranteed that the net will provide good results: for this particular case, if a different initial condition is considered, the net suffers and the prediction is not accurate, this highlights that this kind of NN is very good at interpolation, but less performing at extrapolating.

4.2.2. Reaction diffusion equation

Let u be the solution the following $\lambda - \omega$ Reaction- Diffusion (RD) system of equations

$$\frac{\partial u}{\partial t} = \lambda(A) \cdot u - \omega(A) \cdot v + \delta_1 \Delta u, \qquad \frac{\partial v}{\partial t} = \omega(A) \cdot u - \lambda(A) \cdot v + \delta_2 \Delta v; \tag{20}$$

[10] For this case, the time has been sampled every two time steps, thus the net learns how to predict the behaviour of the solution from t to $t + 2\Delta t$, given Δt the time discretisation step size.

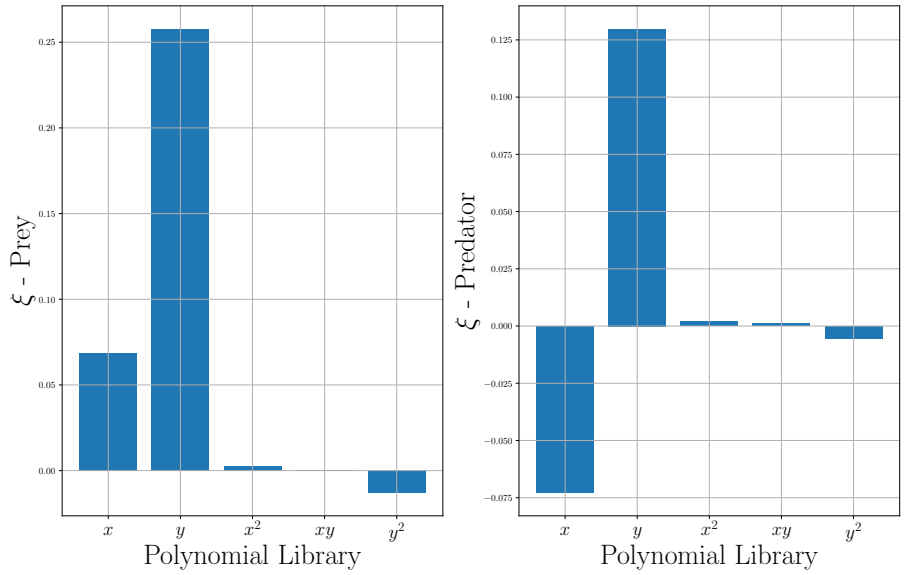


Fig. 4: Coefficients of the SINDy model.

with $A^2 = u^2 + v^2$, $\lambda(A) = 1 - A^2$ and $\omega(A) = -\beta \cdot A^2$. The time domain is $[0, 10]$ and the spatial one is $[-10, 10]$, respectively discretised with 200 sub-intervals and a 512×512 uniform grid. This system of equations is solved with FFT and the solution is later stored in 3D tensors $\mathcal{U}_{ijk} = u(x_i, y_j, t_k)$ and $\mathcal{V}_{ijk} = v(x_i, y_j, t_k)$. The size of this data is too large[11] to be directly used to train the NN, unless a very deep NN is built. This is why, a dimensionality reduction via SVD, on the tensors $\mathcal{U}$ and $\mathcal{V}$, is required (e.g., $\mathcal{U} = U_{\mathcal{U}} \Sigma_{\mathcal{U}} V_{\mathcal{U}}^T$). In this way the spatial behaviour is given by the spatial modes, collected in $U_{\mathcal{U}}$, whereas the time/parametric behaviour is hidden into $V_{\mathcal{U}}$ (see Section 2). By analysing the eigenvalues in $\Sigma_{\mathcal{U}}$ and the relative training error[12], it can be proved that 5 modes are sufficient to have good reconstructions. Since the time behaviour of the modes is relatively simple, it is not required to have a deep net: the NN has 3 hidden layers, two of them with 10 neurons each and the third with 5 (all activated with a linear function). The input and output data are built with the same procedure outlined in Eq. (6). Figure 6 shows the comparison between the true parametric mode and the predicted one by the NN for function u[13]: the agreement is almost perfect, indeed the relative error, measured in L^2 is very low for all the modes. Accordingly, given an accurate prediction of the time behaviour, the overall prediction (in space and time) can be built by getting back to the original variable, using the properties of the SVD decomposition. The global error on the reconstructed function would depend on the truncation error for the SVD (space) and the prediction error of the net (time).

4.2.3. Lorenz equations

Let x, y, z the solution trajectory of the Lorenz equations

$$\frac{\partial x}{\partial t} = \sigma(y - x), \qquad \frac{\partial y}{\partial t} = x \cdot (\rho - z) - y, \qquad \frac{\partial z}{\partial t} = x \cdot y - \beta\, z, \tag{21}$$

given $[0, 2]$ the time domain. The parameters are equal to $\sigma = 10$ and $\beta = \frac{8}{3}$, whereas three different values of ρ are considered for the net training, namely $\rho_t = [10, 28, 35]^T$, so that the interpolation and extrapolation property of the net, at different value of ρ, can be assessed. The training dataset is composed by 100 random trajectories in the box $[-7.5, 7.5]^3$ per each value of ρ_t: by plotting them, the effect of increasing ρ_t is the enhancement of the attractors, meaning that the particles are "more forced" to go into these attractive regions.

[11] The state vector $\mathbf{x}(t_k)$ would have 512^2 elements for each variable.

[12] This quantity is defined as

$$\epsilon(r) = \frac{||u - \mathcal{P}_r[u]||_{L^2}}{||u||_{L^2}}$$

in which $\mathcal{P}_r$ is the projection operator, considering r modes.

[13] The same conclusions can be found also for v.

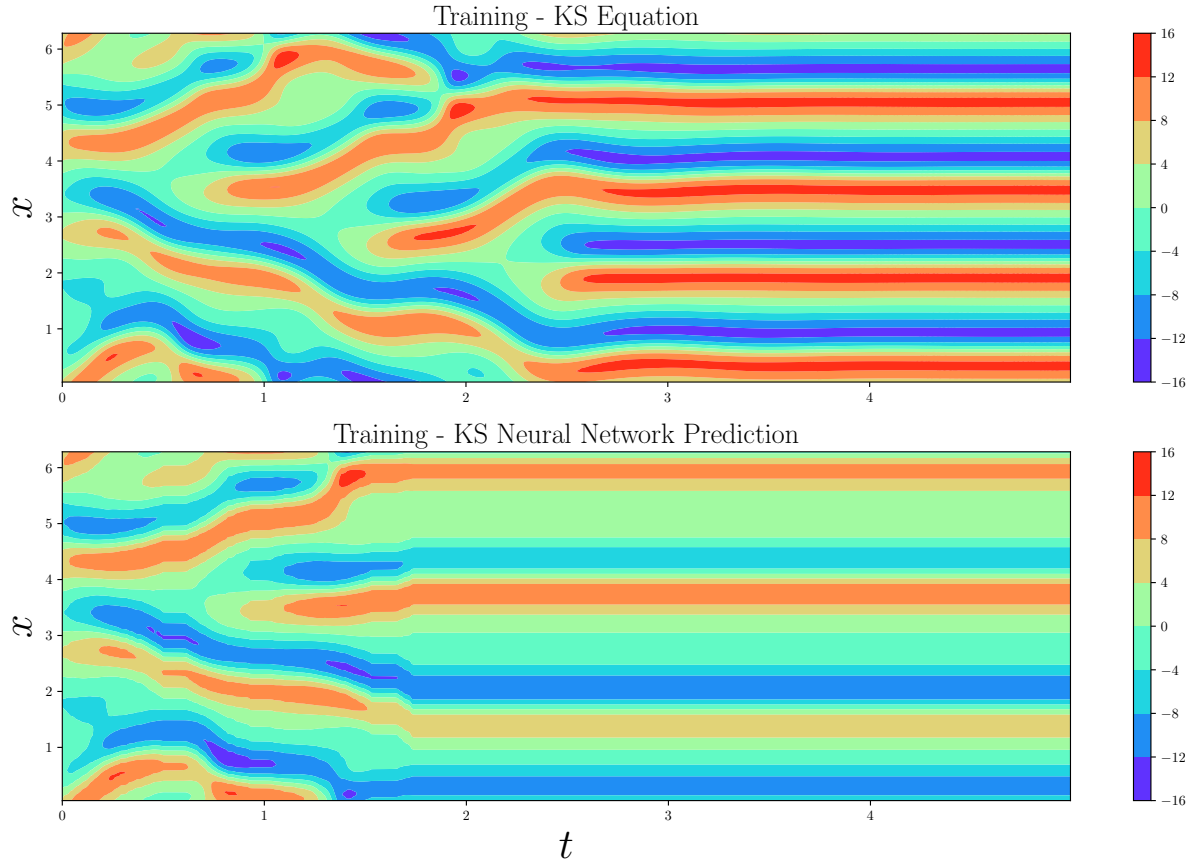


Fig. 5: Comparison between the true training solution and the one predicted by the NN.

A NN is built to learn the behaviour of the trajectories from t to $t + \Delta t$: the net is composed an input layer with 16 neurons activated with a *sigmoid* followed by two hidden layers with 16 neurons each, with the *relu* activation function and the output linear layer. The input and output state vector, at time t for a given initial condition ($\mathbf{x}_0$) and a given parameter ρ_i, is defined as

$$\mathbf{i}_i(t) = \begin{bmatrix} x(t;\mathbf{x}_0) \\ y(t;\mathbf{x}_0) \\ z(t;\mathbf{x}_0) \\ \rho_i \end{bmatrix}, \qquad \mathbf{o}_i(t) = \begin{bmatrix} x(t+\Delta t;\mathbf{x}_0) \\ y(t+\Delta t;\mathbf{x}_0) \\ z(t+\Delta t;\mathbf{x}_0) \\ \rho_i \end{bmatrix}. \tag{22}$$

The parameter ρ_i is included to let the net know which trajectory belong to the particular value of ρ. Figure 7 shows the trajectories, both for interpolation ($\rho = 17$) and extrapolation ($\rho = 40$), for the ODE solution and the NN prediction. Starting from the interpolation case, the NN is able to predict quite well the lobe of all the trajectories, however it is not able to provide a good estimation of what happens at the attractors for the different initial condition. On the other hand, the extrapolation is more inaccurate and the error is in general higher than the interpolation case. The results on interpolation can be improved by increasing the dimension of the training data (e.g., increasing the number of samples for ρ so that a better coverage of the interval is given).

5. Summary and Conclusions

In this work, some Data Assimilation techniques for data-driven modelling have been tested either on raw data or synthetic data, coming from numerical simulations of ODEs/PDEs. All these methods are very promising: the main aim of all of them consists in building the best model, either linear or non-linear, that fit some training data.

The first part was devoted to DMD and its generalisation, applying them to a well known data science dataset (Canadian lynxes and snowshoes): the time-delay DMD is the most promising tool, because it is able to "cure the data", if they are scarse, the Hankel matrix is built and the DMD is directly applied to

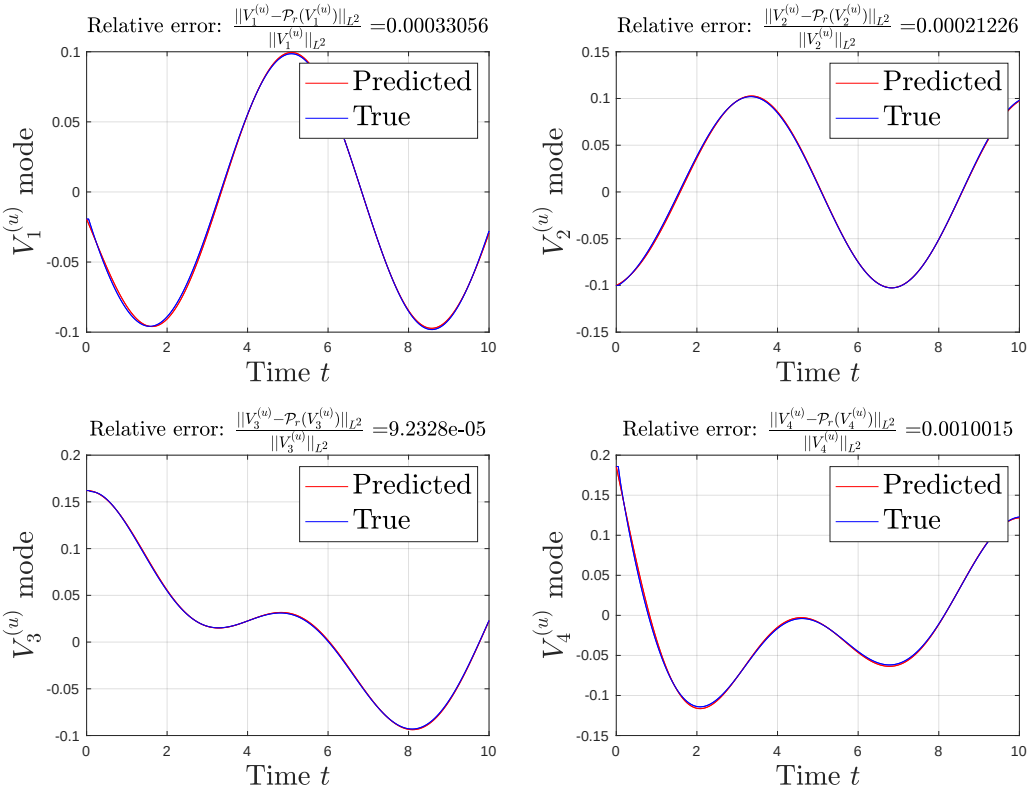
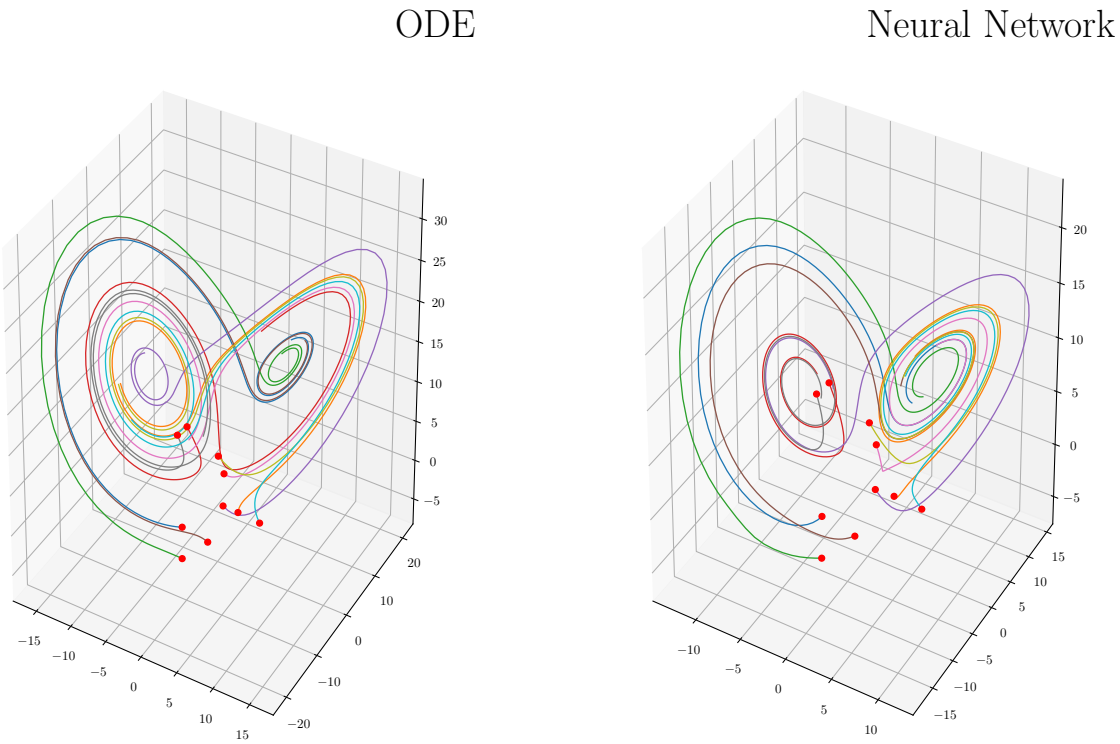
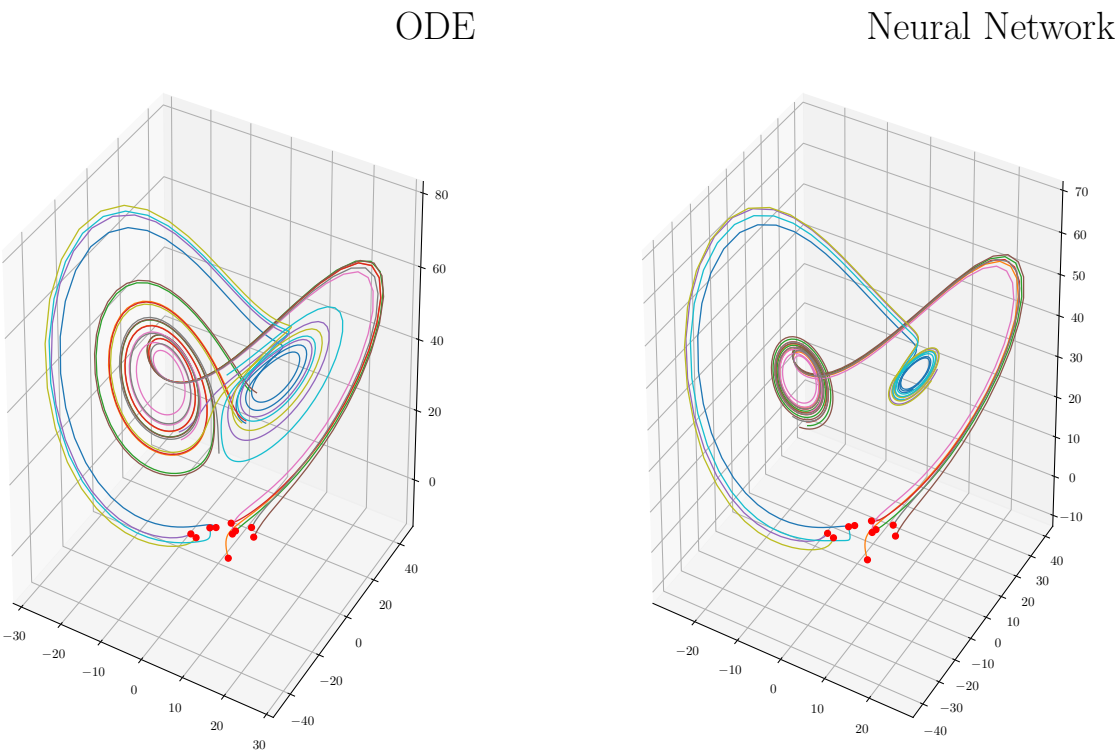


Fig. 6: Comparison between NN prediction and true solution, for the first 4 time modes of function u.

it. Then, the same dataset has been used as input to SYNDy, an efficient method to discover the main dominant terms of a dynamical system, thus it can be seen as a generalisation of the DMD approach. The results are promising and shows the potentiality of this technique.
In the second part, Neural Networks have been used to predict the time evolution of a solution to some ODEs/PDEs: they require a huge amount of data to have a successful training and this may be a big problem in terms of memory storage. In particular, if the mesh has too high dimension, as the example on the Reaction-Diffusion equation, the state variable would have a lot of components. In order to tackle this issue, the data can be projected onto a reduced space by means of the SVD: in this way, the spatial dependence and the parametric (e.g., on time) are de-coupled. The training data are reduced and they are used to make the net learn, this approach is very promising and may be exploited for several high dimensional problems.
Finally, this work may be used a starting point for future developments with the application of some of these methods in the context Hybrid Data Assimilation (i.e., a coupled approach between Reduced Order Modelling and Data Assimilation techniques).



(a) Interpolation: $\rho = 17$.



(b) Extrapolation: $\rho = 40$.

Fig. 7: Comparison between the ODE solution and the NN prediction at different values of ρ, considering both interpolation and extrapolation from the training domain.

References

[Askham, 2017] Askham, T. (2017). duqbo/optdmd: optdmd v1.0.1.

[Belkin et al., 2019] Belkin, M., Hsu, D., Ma, S., and Mandal, S. (2019). Reconciling modern machine-learning practice and the classical bias–variance trade-off. *Proceedings of the National Academy of Sciences*, 116(32):15849–15854.

[Berkooz et al., 1993] Berkooz, G., Holmes, P., and Lumley, J. L. (1993). The Proper Orthogonal Decomposition in the Analysis of Turbulent Flows. *Annual Review of Fluid Mechanics*, 25(1):539–575.

[Berner et al., 2021] Berner, J., Grohs, P., Kutyniok, G., and Petersen, P. (2021). The Modern Mathematics of Deep Learning.

[Brunton et al., 2021] Brunton, S. L., Budišić, M., Kaiser, E., and Kutz, J. N. (2021). Modern koopman theory for dynamical systems.

[Brunton and Kutz, 2019] Brunton, S. L. and Kutz, J. N. (2019). *Data-Driven Science and Engineering: Machine Learning, Dynamical Systems, and Control*. Cambridge University Press, USA, 1st edition.

[Brunton et al., 2016] Brunton, S. L., Proctor, J. L., and Kutz, J. N. (2016). Discovering governing equations from data by sparse identification of nonlinear dynamical systems. *Proceedings of the National Academy of Sciences*, 113(15):3932–3937.

[Carrassi et al., 2018] Carrassi, A., Bocquet, M., Bertino, L., and Evensen, G. (2018). Data assimilation in the geosciences: An overview of methods, issues, and perspectives. *WIREs Climate Change*, 9(5):e535.

[de Silva et al., 2020] de Silva, B., Champion, K., Quade, M., Loiseau, J.-C., Kutz, J., and Brunton, S. (2020). Pysindy: A python package for the sparse identification of nonlinear dynamical systems from data. *Journal of Open Source Software*, 5(49):2104.

[Duderstadt and Hamilton, 1976] Duderstadt, J. J. and Hamilton, L. J. (1976). *Nuclear reactor analysis*. New York: Wiley.

[Goodfellow et al., 2016] Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep Learning*. MIT Press.

[Kaptanoglu et al., 2022] Kaptanoglu, A. A., de Silva, B. M., Fasel, U., Kaheman, K., Goldschmidt, A. J., Callaham, J., Delahunt, C. B., Nicolaou, Z. G., Champion, K., Loiseau, J.-C., Kutz, J. N., and Brunton, S. L. (2022). Pysindy: A comprehensive python package for robust sparse system identification. *Journal of Open Source Software*, 7(69):3994.

[Quarteroni et al., 2015] Quarteroni, A., Manzoni, A., and Negri, F. (2015). *Reduced Basis Methods for Partial Differential Equations: An Introduction*. UNITEXT. Springer International Publishing.

[Rowley et al., 2009] Rowley, C. W., Mezic, I., Bagheri, S., Schlatter, P., and Henningson, D. S. (2009). Spectral analysis of nonlinear flows. *Journal of Fluid Mechanics*, 641:115–127.

[Schilders et al., 2008] Schilders, W., der Vorst, H., and Rommes, J. (2008). *Model Order Reduction: Theory, Research Aspects and Applications*, volume 13. Springer-Verlag Berlin Heidelberg.

[Schmid and Sesterhenn, 2008] Schmid, P. and Sesterhenn, J. (2008). Dynamic Mode Decomposition of numerical and experimental data. In *APS Division of Fluid Dynamics Meeting Abstracts*, volume 61 of *APS Meeting Abstracts*, page MR.007.

[Schmid, 2010] Schmid, P. J. (2010). Dynamic mode decomposition of numerical and experimental data. *Journal of Fluid Mechanics*, 656:5–28.

[Sirovich, 1987] Sirovich, L. (1987). Turbulence and the Dynamics of Coherent Structures Part I: Coherent Structures. *Quarterly of Applied Mathematics*, 45(3):561–571.