Politecnico di Milano
Dipartimento di Elettronica, Informazione e Bioingegneria (DEIB)
PhD in Information Technology

POLITECNICO
MILANO 1863

PROJECT REPORT

Modelling from Measurements

Author: Simone Specchia
Contact: simone.specchia@polimi.it

Contents

List of Figures

List of Tables

Abstract

Modelling a physical system is a fundamental first step of many engineering applications.
For complex, non-linear systems, derivating an accurate mathematical model from prior knowledge of the system is not always possible. In the scientific literature, many approaches have been proposed to derive the model of a system directly from measurements of its inputs/outputs.
This work presents a series of case studies involving different techniques for model identification from data, illustrating the key characteristics of each method.
It also discusses how to handle the main problematics that affect model identification approaches, such as the presence of noise and the management of computational complexity.

1 Introduction

This works presents several techniques that can be found in the scientific literature for modelling identification from data. For each method, an example of application to a challenging case study is provided. In particular the report is organized as follows:

- **Section 2** presents Dynamic mode Decomposition technique
- **Section 3** presents different approaches to derive explicit non-linear differential equations from data
- **Section 4** illustrates the use of Neural Networks for model identification
- In **Section 5** final conclusions are drawn for the results of this work

The code implementation of the algorithms discussed in this report is available at [1]

2 Dynamic Mode Decomposition

2.1 Introduction

Dynamic Mode Decomposition (DMD) [6] is a data-driven, black-box approach to model identification. The main principle of DMD consists in determining the best-fit linear dynamical system representing a physical process.
Consider the following data matrix:

$$\mathbf{X} = \begin{bmatrix} x_1^1 & x_1^2 \cdots & x_1^K \\ x_2^1 & x_2^2 \cdots & x_2^K \\ \vdots & \ddots & \vdots \\ x_m^1 & x_m^2 \cdots & x_m^K \end{bmatrix} \tag{1}$$

where $x^k = [x_1^k, x_2^k, \cdots, x_m^k]$ is an observation of m variables at time sample k.
DMD applies Singular Value Decomposition (SVD) to project $\mathbf{X}$ in a lower dimensional space with rank r and computes the DMD modes ϕ_h, eigenvalues λ_j and mode amplitudes b_j, $j = 1, \cdots, r$.
The output in the original coordinates is reconstructed by superimposing the identified modes:

$$x^k = \sum_{j=1}^{r} \phi_j \lambda_j^{k-1} b_j \;\; \forall k = 1, ..., K. \tag{2}$$

In particular in the following sections opt-DMD [5] is considered, that works by solving an exponential curve fitting problem as the one in Eq.3

$$\min_{b_j, \psi_j, \lambda_j} \|\mathbf{X} - \sum_{j=1}^{r} b_j \psi_j e^{\lambda_j t}\| \tag{3}$$

2.2 Modelling predator-prey interaction

The following sections illustrate some examples of applications of the opt-DMD approach to an experimental dataset.
The reference dataset consists in the population counts of two animal species (lynx and snowshoe hare) in Canada from 1845 to 1903. The dataset is reported in Figure 1.

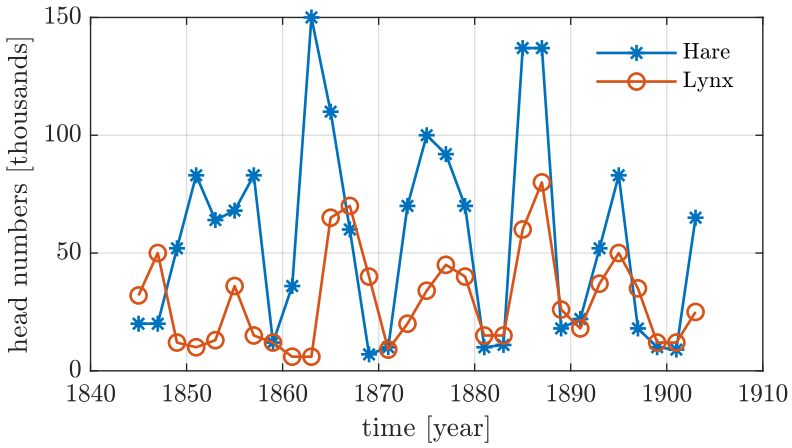

Figure 1: Hares (blue) and lynxes (orange) population dataset

2.2.1 Application of opt-DMD

We apply the algorithm described in Section 2.1 to the animal population dataset. The data matrix is constructed as:

$$\mathbf{X} = \begin{bmatrix} x_1^1 & x_1^2 \cdots & x_1^K \\ x_2^1 & x_2^2 \cdots & x_2^K \end{bmatrix} \tag{4}$$

Where x_1^k is the population of hares and x_2^k the one of lynxes for $k = 1, \cdots, 30$.
Figure 2 shows the original dataset compared to the DMD reconstruction.

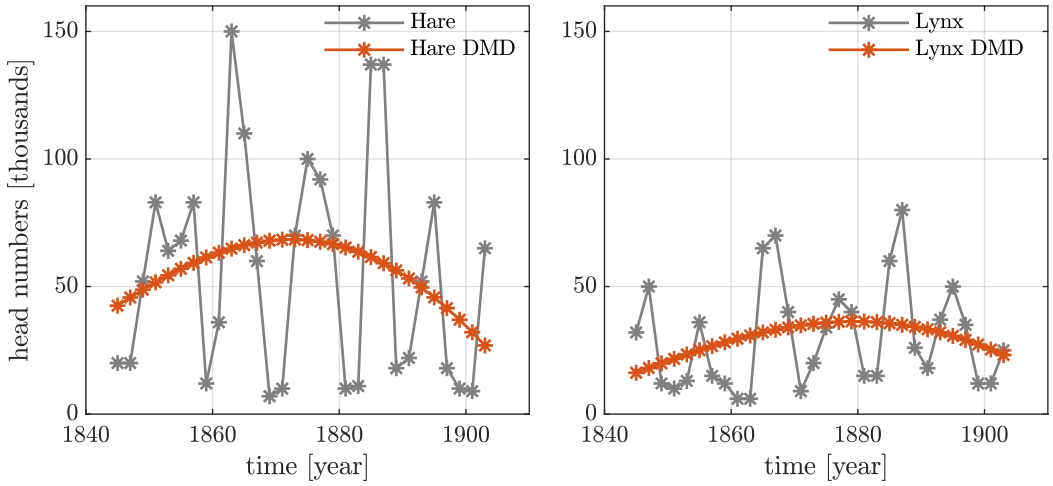

Figure 2: Comparison between population measurements (grey) and DMD reconstruction (orange)

2.2.2 Time delay opt-DMD

The dataset shows a certain seasonality that the linear model identified in Section 2.2.1 fails to represent. Given a non-linear dynamical system, it is possible to define a transformation $g(x)$ to a higher-dimensional space which allows to describe the system with a linear equation:

$$\dot{g}(x) = Fg(x) \tag{5}$$

A common choice for this transformation is time-delay. The new data matrix is constructed as:

$$\mathbf{X} = \begin{bmatrix} x^1 & x^2 \cdots & x^p \\ x^2 & x^3 \cdots & x^{p+1} \\ \vdots & \ddots & \vdots \\ x^{K-p+1} & \cdots & x^K \end{bmatrix} \tag{6}$$

Selecting $p = 25$ the results shown in Figure 3 are obtained. As it can be seen, the DMD output models the seasonal trend of the data.

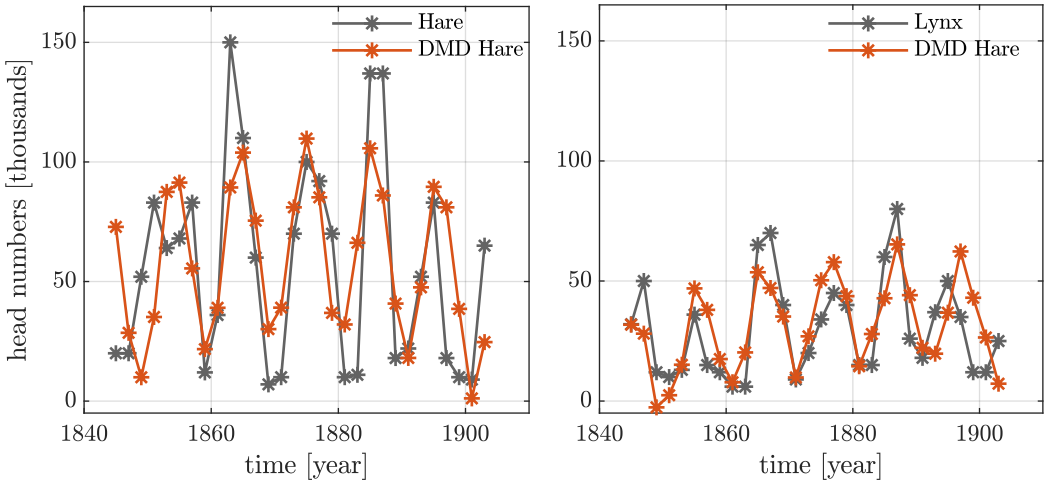

Figure 3: Comparison between dataset (grey) and DMD reconstruction (orange) using time-delay transform

2.2.3 Bagging

The presence of noise in the dataset can greatly affect the results of identification. Bagging is a common technique used to increase robustness.
From the full dataset $\mathcal{D}$, a number N of subsets $\bar{\mathcal{D}}_i \in \mathcal{D}$ are selected. The subsets $\bar{\mathcal{D}}_i$ are composed by selecting randomly K_{sub} observations from $\mathcal{D}$.
The identification with opt-DMD is then performed independently for each subset, generating for each bag a set of $\phi_{ij}, \lambda_{ij}, b_{ij}$.
An average model is then determined by averaging the coefficients obtained for each bag.
Figure 4 reports the reconstruction using the average model and the distribution of the bag models , having run DMD on 100 bags, each obtained selecting a subset of 22 observations from the original time-delayed data matrix.

3 Regression to non linear model

3.1 Regression to known model class

Alternatively to the methods presented in Section 2, we could describe the system that generated the dataset in Section 2.2 as a dynamical one governed by differential equations.
In particular, a common model for the predator-prey interaction is the *Lotka-Volterra* model described by Eq.7, where x, y is the population of predator and prey species, while b, p, r, d are the unknown parameters.

$$\begin{cases} \dot{x} = (b - py)x \\ \dot{y} = (rx - d)y \end{cases} \tag{7}$$

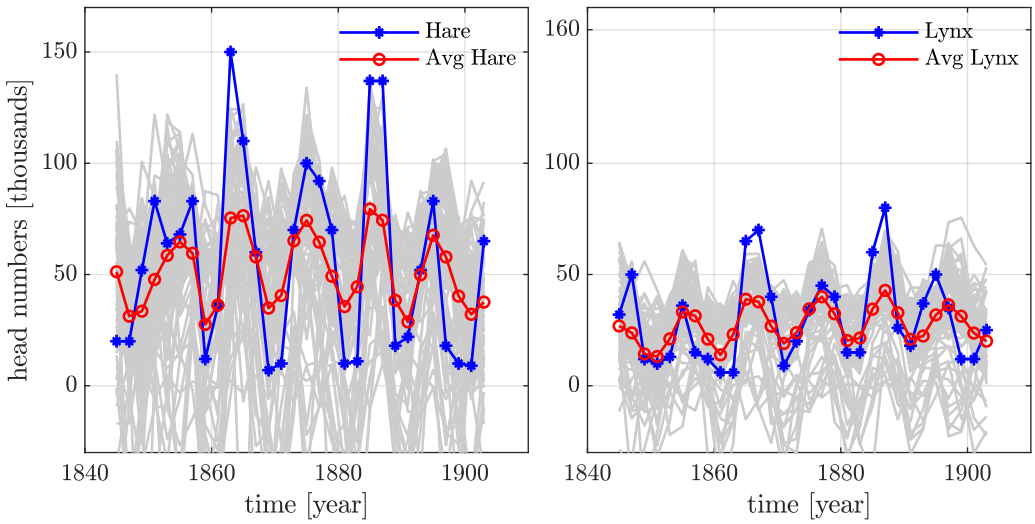

Figure 4: Comparison between dataset (blue) and average DMD model (red). The models from each bag are indicated in grey

The model is linear in the unknown parameters so it can be rewritten as follows:

$$\begin{bmatrix} \dot{x} \\ \dot{y} \end{bmatrix} = \begin{bmatrix} x & -xy & 0 & 0 \\ 0 & 0 & xy & -y \end{bmatrix} \begin{bmatrix} b \\ p \\ r \\ d \end{bmatrix} = A\varphi \tag{8}$$

A Least-squares minimization approach can be used to determine the best fit for the parameter vector φ, by minimizing the cost function described in Eq. 9:

$$\hat{\varphi} = \arg\min_{\varphi} \sum_{i=1}^{N} (\mathbf{y_i} - A\varphi)^2 \tag{9}$$

3.1.1 Results

This procedure applied to the dataset produces the following values for the parameters:

Parameters	b	p	r	d
Values	0.3051	0.0076	0.0047	0.4062

Table 1: Values of the identified parameters for the Lotka-Volterra model

Figure 5 shows the evolution of the identified *Lotka-Volterra* model, compared with the original dataset. As we can see, the identified model quickly diverges, and is not in fact representative of the actual evolution of the population counts.

3.2 Model identification with SINDy

The method described in Section 3.1 requires prior knowledge of the governing equations of the physical phenomena for which we are building a model. It therefore fails whenever this information is not available or incomplete.
In general, a non-linear dynamical system can be written as:

$$\dot{x} = \sum_{f_i \in \mathcal{F}} \alpha_i \cdot f_i(x) \tag{10}$$

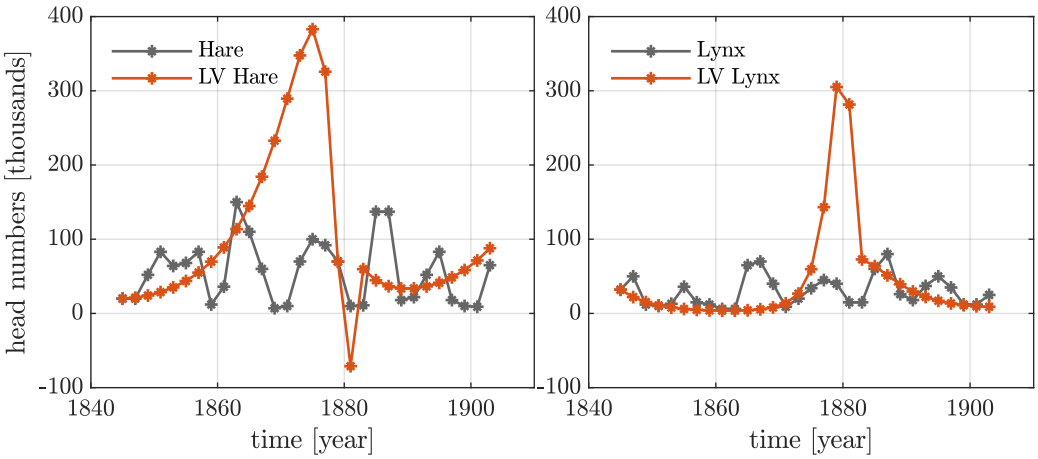

Figure 5: Comparison between the dataset (grey) and the fitted Lotka-Volterra model (orange)

where f_i are non-linear functions of the system state x belonging to a given class $\mathcal{F}$ and α_i are linear coefficients multiplying each function.

Sparse Identification of Nonlinear Dynamics (SINDy) [2], is an approach for the identification of a non-linear dynamical model that performs the following steps:

1. Given a library of basic functions $\mathcal{F}$, determine the coefficients α_i that best fit the dataset
2. In order to avoid overfitting, reduce $\mathcal{F}$ to a new $\bar{\mathcal{F}}$, discarding the terms associated to a coefficient $\alpha_i < \alpha_{th}$
3. Iterate identification, until a convergence condition is reached

3.2.1 Results

SINDy is applied to the dataset described in Section 2.2, making the following considerations for the choice of tuning:

- $\mathcal{F} = \{f(x) : f(x) = x^n \cdot y^m, n + m = 3\}$
- $\alpha_{th} = 0.002$
- Bagging is used to increase robustness to noise, iterating the identification on $B = 700$ bags, each one composed of $N_{sub} = 26$ elements

The output model is described by the following equation:

$$\begin{cases} \dot{x} = 2.361x - 0.047xy \\ \dot{y} = -0.112x + 0.01x^2 - 0.008xy \end{cases} \tag{11}$$

Figure 6 shows the comparison between the simulated evolution of the SINDy model and measurements starting from the same initial condition.

4 Modelling with Neural Networks

4.1 Introduction

The methods presented in Section 3 allow to model non-linear dynamical systems with explicit equations, maintaining interpretability of the physical phenomena.

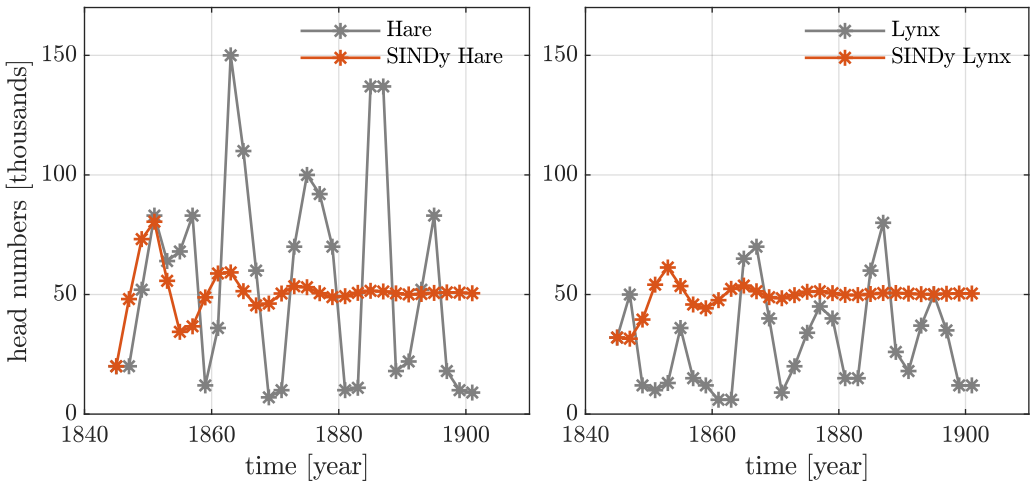

Figure 6: Comparison between the dataset (grey) and the simulated output of the SINDy model (orange)

In alternative to these methods, we can use neural networks to construct a black-box model of a system. Considering the following generic representation of a dynamical system in discrete time:

$$x(t + \Delta t) = g(x(t)) \tag{12}$$

A neural network allows us to represent $g(x(t))$ as a non linear function of the type:

$$g(x) = f^L(W^L f^{L-1}(W^{L-1} ... f^1(W^1 x))) \tag{13}$$

Where:

- f^l is the activation function of the l-th layer
- W^l is the matrix of weights between the nodes of layer $l-1$ and the ones of layer l

Fitting a model consists in determining the set of weights that minimizes a loss function of the predicted outputs versus the measured ones $C(x(t + \Delta t), g(x(t)))$.
This problem is typically solved using backpropagation with stochastic gradient descent [3].
The following sections report some examples of the application of the method to various complex dynamical systems.

4.2 Kuramoto-Sivashinsky equation

As a first example, we consider the 1-D Kuramoto–Sivashinsky (KS) equation, a fourth-order non-linear partial differential equation that takes the following form:

$$\dot{u} = -u \cdot u_x - u_{xx} - \nu \cdot u_{xxxx} \tag{14}$$

The x dimension is sampled in 1024 positions, and we consider known the evolution of the value of u for each cell of the grid in a time horizon of $T = 9.8s$.

A neural network with the following characteristics is selected:

- 3 hidden layers, each one with 15 nodes
- the activation functions for the hidden layers are respectively a log-sigmoid, a radial basis and a tansig function
- the input training dataset is the timeseries of the values of u and its spatial derivatives for a subset $\mathcal{X}$ of the original 1024 cells.

- the output associated to each input is the value of u for the same cell, 1-step ahead in time

The resulting NN is validated by simulating the evolution of the system for the positions not contained in the training dataset.
Figure 7 shows the simulated output compared to the available dataset.

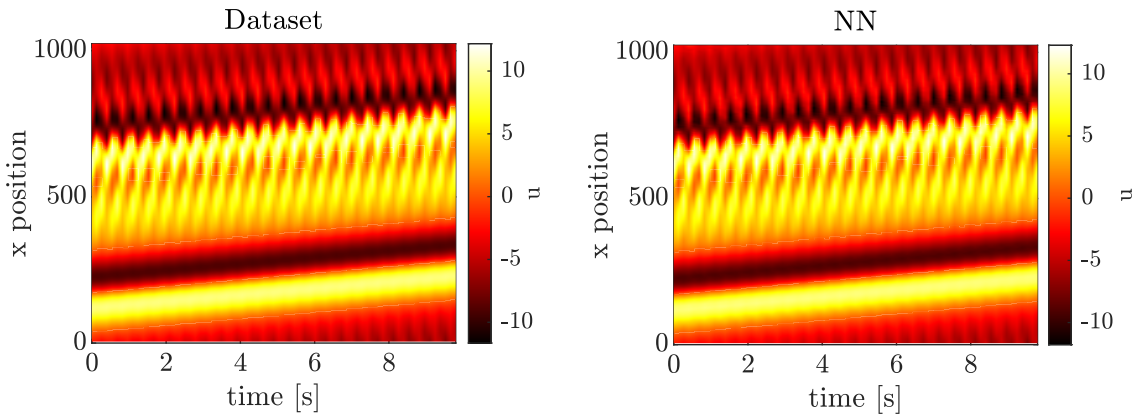

Figure 7: Comparison between the evolution of $u(x,t)$ from dataset and the evolution predicted by the NN

Figure 8 reports The Root Mean Square Error (RMSE) for each cell of the grid.
The NN shows good performance, being the RMSE always below 3%.

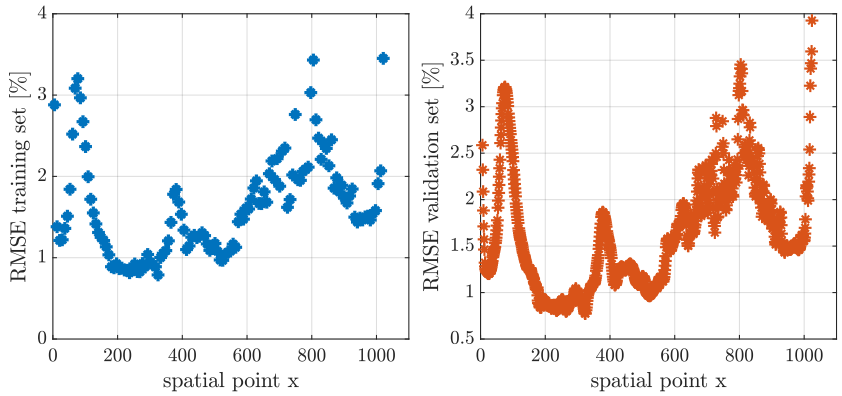

Figure 8: RMSE in training and validation for the trained NN

4.3 Reaction-diffusion system of equations

We consider the following reaction-diffusion equation for a 2-D system:

$$\begin{cases} u_t = \lambda(A)u - \omega(A)v + d_1(u_{xx} + u_{yy}) = 0 \\ v_t = \omega(A)u + \lambda(A)v + d_2(v_{xx} + v_{yy}) = 0 \end{cases} \tag{15}$$

With:

- $A^2 = u^2 + v^2$
- $\lambda(A) = 1 - A^2$
- $\omega(A) = -\beta A^2$

The dataset is composed of the evolutions of $u(x,y,t)$, $v(x,y,t)$ for a grid of 512x512 x-y positions, in a time horizon $T = 10s$.

To predict the evolution of the system, a neural network as in Section 4.2 could be implemented, however this results impractical from a computational standpoint due to the size of the dataset. Consequently, a technique of order reduction can be used to reduce the size of the training dataset and make training the NN feasible.

4.3.1 Principal Components Analysis

Given a set of K observations $x^k = [x_1^k, ..., x_m^k]$, the following data matrix can be defined:

$$\mathbf{X} = \begin{bmatrix} x_1^1 & x_1^2 & \cdots & x_1^K \\ x_2^1 & x_2^2 & \cdots & x_2^K \\ \vdots & \ddots & & \\ x_m^1 & x_m^2 & \cdots & x_m^K \end{bmatrix} \tag{16}$$

Principal Components Analysis (PCA) is a technique commonly applied for model reduction. It works by determining the linear transformation $\mathbf{Z} = \Phi(\mathbf{X})$. In the new frame, the first component of the observation z^k corresponds to the linear combination of the original variables that explains the most variance. The second component explains the most variance in what remains after the first component is removed, etc.
The model reduction is obtained by truncating the new observation matrix to the i-th principal component.

In the case of Eq. 15, we define the observation matrix as:

$$\mathbf{X} = \begin{bmatrix} u_{11}^1 & u_{11}^2 & \cdots & u_{11}^{K_{train}} \\ v_{11}^1 & v_{11}^2 & \cdots & v_{11}^{K_{train}} \\ u_{12}^1 & u_{12}^2 & \cdots & u_{12}^{K_{train}} \\ \vdots & \ddots & & \\ u_{nn}^1 & u_{nn}^2 & \cdots & u_{nn}^{K_{train}} \end{bmatrix} \tag{17}$$

Applying PCA to the so obtained data matrix, we obtain that the first component explains the 51% of the variance of the observations, and the second one explains the 48%, consequently all the remaining components can be discarded.

4.3.2 Results

A neural network with the following characteristics is selected:

- 2 hidden layers, each one with 10 nodes
- the activation functions for the hidden layers are respectively a log-sigmoid and a tansig function
- the input training dataset is composed of the observations $[z^1, z^2, \cdots, z^{K_{train}-1}]$, with $K_{train} = 150$
- the output is composed of the observations $[z^2, z^3, ..., z^{K_{train}}]$

To validate the results of the training, the following steps are performed:

1. Select x^k, $k > K_{train}$
2. Transform the observation in the coordinate z^k
3. Predict $\hat{z}^{k+1}$ from z^k using the trained NN
4. Transform back $\hat{z}^{k+1}$ to $\hat{x}^{k+1}$ and compare with x^{k+1}

Results of prediction are shown for $k_{validation} = 160$ are shown in Figure 9.
Figure 10 shows the average reconstruction error between the 1-step prediction obtained with the NN and the actual output, for different validation time instants.

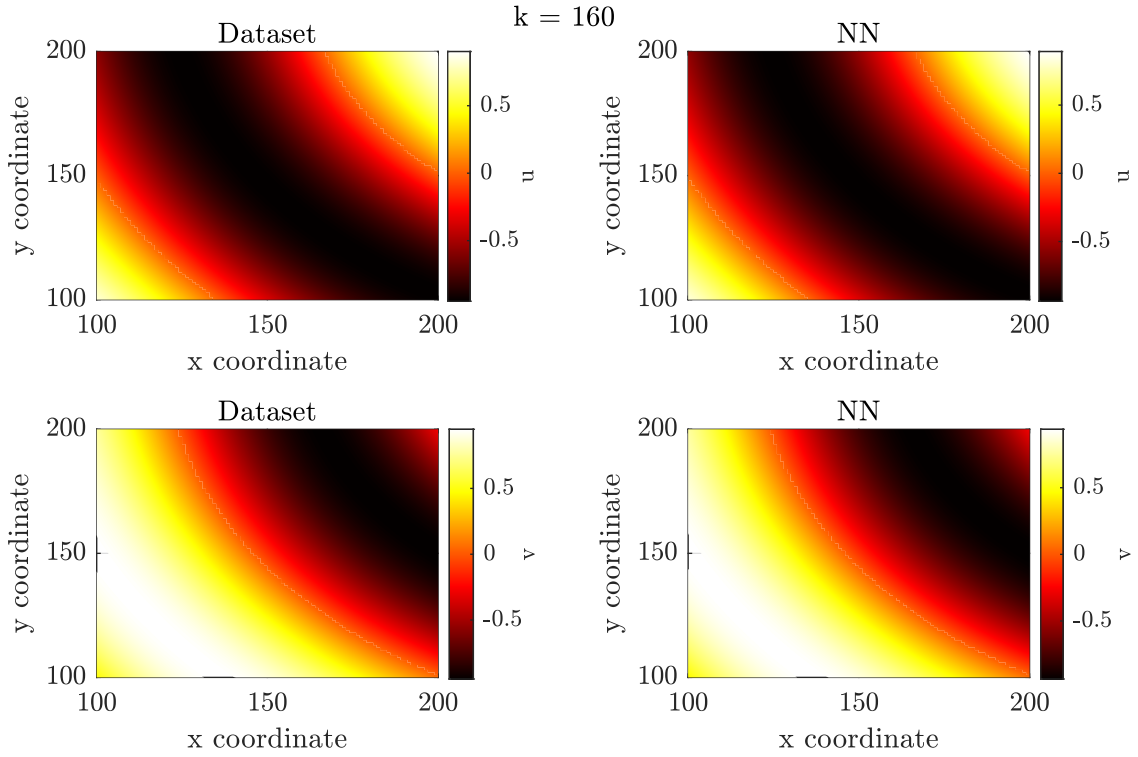

Figure 9: Comparison between actual X-Y grid and output prediction of the NN at time instant $k = 160$

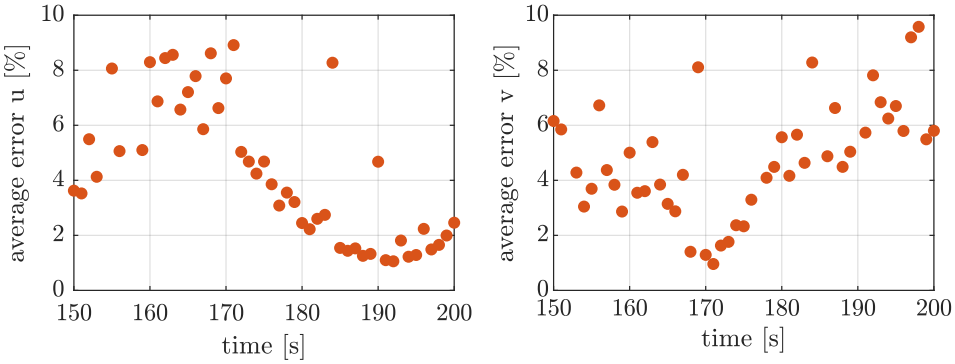

Figure 10: Average prediction error for different validation instants

4.4 Lorenz equations

For the third example we consider Lorenz equations [4], a system of ordinary differential equations that take the following form:

$$\begin{cases} \dot{x} &= \sigma(y - x) \\ \dot{y} &= \rho x - xz - y \\ \dot{z} &= xy - \beta z \end{cases} \tag{18}$$

where σ,ρ and β are parameters.

4.4.1 Solution from random initial condition

The first objective of this example is to train a NN to solve the differential equations, starting from a random initial condition.
After having generated $N = 70$ trajectories of $[x, y, z]$ starting from random initial condition, the following setup is selected for the network:

- 2 hidden layers, each one with 10 nodes

- The activation functions for the hidden layers are respectively a log-sigmoid and a purelin function
- Input and output training dataset is defined as:

$$\begin{cases} X &= [x_j(k)\, y_j(k)\, z_j(k)\ ];\ j = 1, ..., N;\ k = 0, ..., K-1 \\ Y &= [x_j(k+1)\, y_j(k+1)\, z_j(k+1)] \end{cases} \tag{19}$$

Figure 11 compares the simulated output of the NN compared to the actual trajectories starting from two different initial conditions.

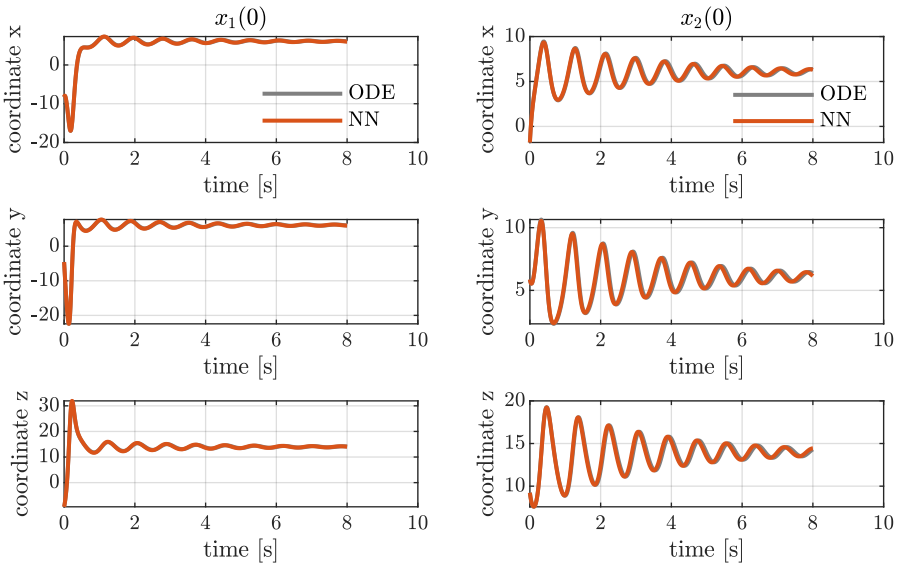

Figure 11: Comparison between data (grey) and output of the NN (orange) for two random initial condition

4.4.2 Effect of varying parameter

The network in Section 4.4.1 was trained on a dataset of trajectories where ρ, σ, β were kept constant. Consider, for example, the effect of a varying ρ. This can be taken into account as an additional input to the net.
We consider three different values of $\rho_{train} = [10, 28, 35]$, for each of them $N = 70$ trajectories from a random initial condition are generated. The training dataset is obtained by modifying Eq. 19 as follows:

$$X = [x_j(k)\, y_j(k)\, z_j(k)\ \rho];\ j = 1, ..., N;\ k = 0, ..., K-1;\ \rho \in \rho_{train} \tag{20}$$

The trained NN is validated by selecting new values for $\rho_{val} = [17, 40]$ and comparing the simulated output of the net to the solution actual evolution.
Figure 12,13 report the comparison of trajectories in 3D space, while Figure 14, 15 represent the evolution in time.

For $\rho = 17$, the NN predicts the evolution of the states accurately over the entire time horizon. For $\rho = 40$, the prediction is only reliable for the first $1.5s$. This can be explained by the fact that $\rho = 40$ is outside the bounds of the values of ρ explored during training.

Table 2 reports the Root Mean Square Errors (RMSE) between the predicted output and the actual one. For $\rho = 40$, the error is computed up to $t = 1.5s$ as the prediction diverges after that time instant.

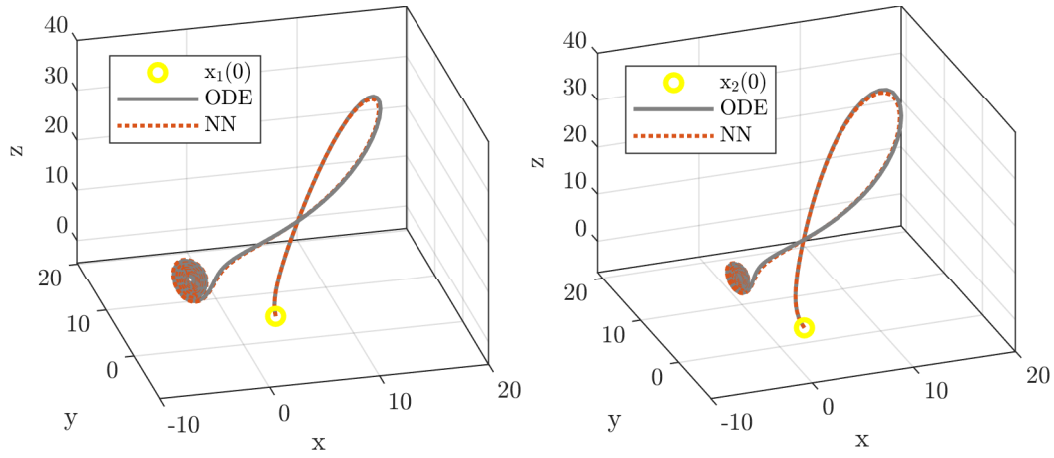

Figure 12: Comparison of 3D state trajectory from 2 random initial conditions for $\rho = 17$.

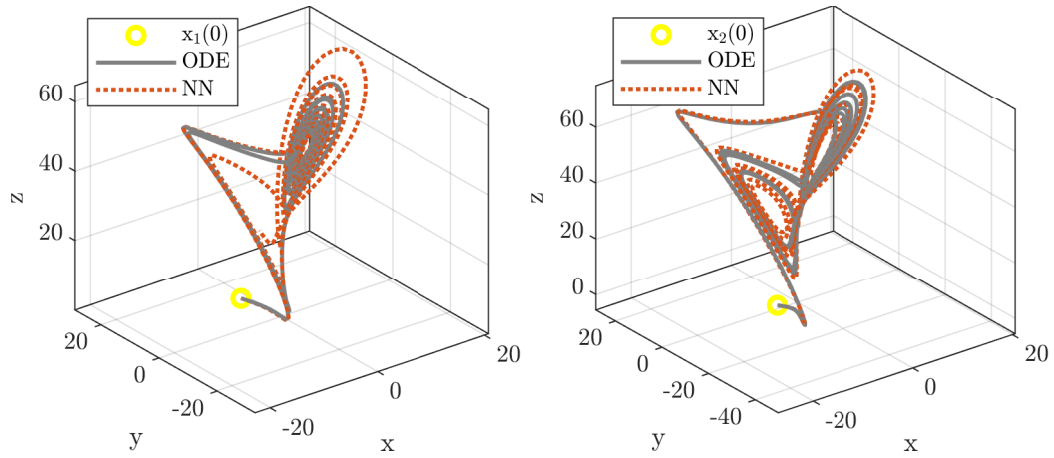

Figure 13: Comparison of 3D state trajectory from 2 random initial conditions for $\rho = 40$.

5 Conclusions

In this work, several techniques for data-driven modelling were described and analyzed.
Given an engineering application that requires data-driven modelling, the choice of the right technique must be made keeping into consideration different factors:

- **Computational requirements:** the various approaches greatly differ for the computational resources necessary to perform identification: DMD is a very fast and low-effort approach, while the resources necessary to train a Neural Network grow exponentially depending on the required characteristics
- **Availability of prior knowledge:** prior knowledge of the physical system is very useful to drive model identification to more accurate solutions. Approaches such as the ones presented in Section 3 easily allow the selection of model classes and libraries compatible with the physical system to be described
- **Interpretability of the model:** depending on the application, we may want to preserve the interpretability of the mathematical equations. This is not possible for example using Neural Networks, however they are a very powerful tool to model complex systems, whenever a black-box relation between input and output is sufficient for the application

The code implementation of the algorithms discussed in this report is available at [1]

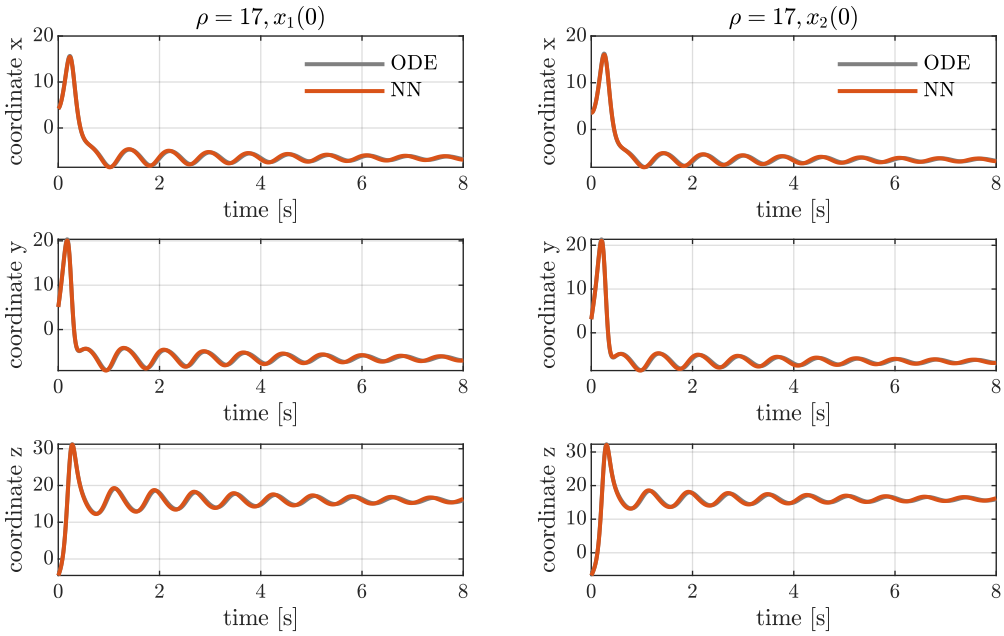

Figure 14: Comparison of the evolution of system states in time for $\rho = 17$.

RMSE[%]	$\rho = 17$	$\rho = 40,\ \Delta t = 1.5s$
x_1	3.05%	2.48%
y_1	3.81%	3.2%
z_1	2.15%	1.18%
x_2	2.80%	13.5%
y_2	3.49%	17.5%
z_2	1.98%	7.98%

Table 2: RMSE for different initial conditions and different values of ρ

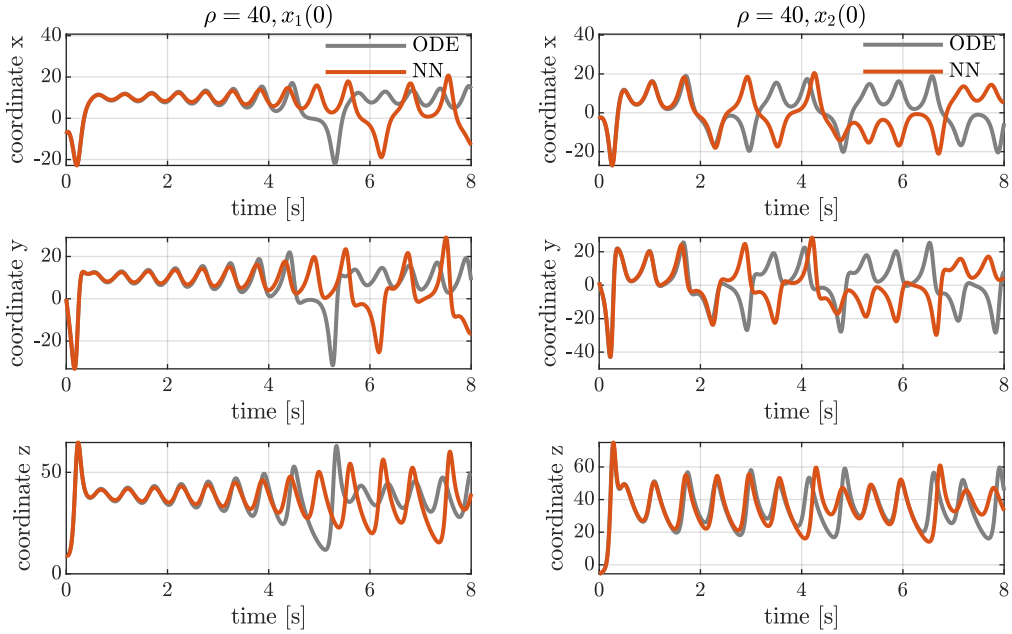

Figure 15: Comparison of the evolution of system states in time for $\rho = 40$.

References

[1] Github code page. `https://github.com/SimoneSpecchia/MFM-homework-Specchia`.

[2] S.L. Brunton, J.L. Proctor, and J.N. Kutz. Discovering governing equations from data by sparse identification of nonlinear dynamical systems. *Proceedings of the National Academy of Sciences*, 2016.

[3] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`.

[4] E. N. Lorenz. Deterministic nonperiodic flow. *Journal of the atmospheric sciences*, 1963.

[5] Diya Sashidhar and J.Nathan Kutz. Bagging, optimized dynamic mode decomposition (bop-dmd) for robust, stable forecasting with spatial and temporal uncertainty-quantification. *arXiv*, 2021.

[6] P.J. Schmid and Sesterhenn. Dynamic mode decomposition of numerical and experimental data. *61st Annual Meeting of the APS Division of Fluid Dynamics*, 2008.