Francesco Gulotta

Modelling from Measurements

May 25, 2020

Abstract

The following report collects the solution of the homework assignment for the course: "Modelling from Measurements". This work is subdivided in three different Chapters, in each of them is analyzed one of the main topics of the curse: dynamic models (PREDATOR-PREY), neural network (Lorentz and reaction diffusion equations) and complex data analysis (Belousov–Zhabotinsky reaction). Each chapter is subdivided in a preliminary introduction and overview Section, which is followed by Section II where the theorical background is summarized. Then, the methodologies applied to develop the algorithm are critically analyzed in the Section III. Finally, the main computational results and the conclusions are described in Section IV.

CHAPTER 1

PREDATOR-PREY DYNAMIC MODEL

Introduction and Overview

In the following Chapter, it is analyzed one classic example of ecological dynamic model, the predator-prey system. In general, the predator-prey scheme is particularly difficult to model because most of predators rely on a variety of preys which introduces difficulties to determine the model that well approximate this complex natural system. However, a few of predators has become specialized in a single prey species, as for example the Canadian lynx. As it can be easily seen form Figure 1, there is a strong correlation between the population of the prey (Snowshore Hale, in orange) and the population of predator (Canada Lynx, in blue). To determine the correlation between the predator and prey, and estimate the future trends of the populations, in this work the Dynamic Mode Decomposition (DMD) is used.

Theoretical Background

The dynamic mode decomposition is a methodology that allows to identify spatial and temporal structures from high-dimensional data. The DMD is based on an orthogonal decomposition, the SVD, but in addition it separates the modes that have the same linear behavior in time, permitting to determine how these modes evolve in time [1]. Indeed, each mode is associated with an eigenvalue and so with an oscillation frequency and growth/decay rate. This feature is particularly interesting since it allows to forecast the trends and evolutions of the analyzed system in the future timesteps.

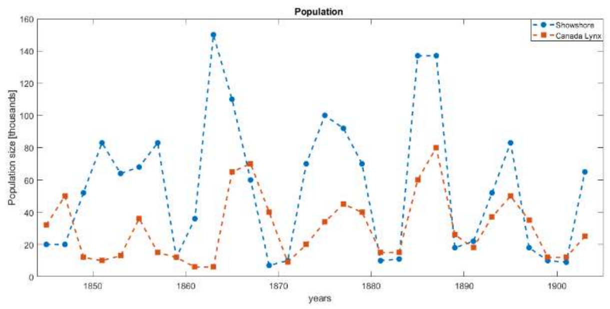

Figure 1 Populations of Showshore and Lynx between 1850 and 1905

To determine the DMD of a dataset, it is firstly required to collect the data in two different matrices as shown in Equations 1.1 and 1.2, the first matrix will consider the data from the first timestep (t_1) to the time step t_{k-1}, which is the second last timestep. The second matrix, called $\boldsymbol{X'}$ collects the data from the second timestep (t_2) to the last one (t_k).

$$\boldsymbol{X} = [\vec{x}(t_1)\ldots\vec{x}(t_{k-1})] \qquad 1.1$$

$$\boldsymbol{X'} = [\vec{x}(t_2)\ldots\vec{x}(t_k)] \qquad 1.2$$

As described before, the DMD allows to determine a linear operator (i.e. a matrix **A**), that advances the measurement forward in time, so the approximate equation presented in (1.3) can be formulated.

$$\boldsymbol{X'} \approx \mathbf{A}\boldsymbol{X} \qquad 1.3$$

The matrix **A** can be computed using the pseudo inverse of the matrix **X**, but this step could be particularly challenging from a computational point of view, especially for large dataset. For this reason, the DMD is typically determined using a procedure that includes the projection of the matrix X into a low rank subspace, thanks to the truncated SVD and a spectral decomposition. In the flowing, a brief description of the algorithm introduced by [2] is provided. Firstly, the SVD decomposition of the matrix **X** is performed and it is approximated with the first r modes. This preliminary step allows to reduce the dimensionality of the system and also subdivided the matrix **X** in two singular matrices ($\tilde{\boldsymbol{V}}, \tilde{\boldsymbol{U}}$) and in a diagonal one ($\tilde{\boldsymbol{\Sigma}}$) as shown in Equation 1.4.

$$\boldsymbol{X} \approx \tilde{\boldsymbol{U}}\tilde{\boldsymbol{\Sigma}}\tilde{\boldsymbol{V}}^* \qquad 1.4$$

Then, the matrix **A** can be computed considering the pseudo inverse of **U** and the DMD decomposition of **X** obtaining Equation 1.5.

$$\mathbf{A} = \mathbf{X}'\tilde{\mathbf{V}}\tilde{\mathbf{\Sigma}}^{-1}\tilde{\mathbf{U}}^* \qquad 1.5$$

Since, we are interested in the first r modes of the matrix **A**, the matrix is projected into the orthogonal subspace as expressed in Equation 1.6, and it possible to prove that the reduced matrix **A** has the same eigenvalues, and so modes, of the reduced matrix $\tilde{\mathbf{A}}$.

$$\tilde{\mathbf{A}} = \tilde{\mathbf{U}}^*\mathbf{A}\tilde{\boldsymbol{U}} = \tilde{\mathbf{U}}^*\mathbf{X}'\tilde{\mathbf{V}}\tilde{\mathbf{\Sigma}}^{-1} \qquad 1.6$$

The third step of the algorithm computes the eigenvalues and eigenvectors of the matrix $\tilde{\mathbf{A}}$**,** collected respectively in the matrix $\boldsymbol{\Lambda}$ and **W**, so that it is possible to reconstruct the DMD modes using Equation 1.7, and finally the spectral expansion can be written:

$$\begin{gathered}\mathbf{\Phi} = \mathbf{X}'\tilde{\mathbf{V}}\tilde{\mathbf{\Sigma}}^{-1}\boldsymbol{W} \\ x(t) = \sum_{i}^{r} \phi_i \, e^{\omega_i t} \, b_i\end{gathered} \qquad 1.7$$

Where ω_i is the natural logarithm of the i-th eigenvalue divided by Δt, and b_i is the amplitude of the i-th mode, calculated using the initials conditions.

Algorithm Implementation and Development

As previously described, the DMD algorithm allows to forecast the behavior of the system analyzed in future timesteps using the spectral expansion of Equation 1.8. The population data shown in Figure 1 are collected in a single matrix with dimension 2x30. As previously explained, two matrices are defined, the matrix **X** in which are collected the data of the two species from the first timestep to the second last (t_{k-1}), and the matrix $\mathbf{X}'$ that contains the data form the second timestep to the last one. Then the algorithm described in the previous section is applied, so firstly the "economy" SVD is applied. Since the matrix **X** has a number of row equal to 2, only two modes can be extracted from the matrix (i.e. $r_{max} = rank(\mathbf{X})$) and this will reduce the capability to forecast the future trend as will be described in the next Section.

To enhance the capability to detect the latent variables in the data, it is possible to introduce the concept of delay coordinates. This procedure consists in introducing a new matrix, in which in each row, the data are time shifted of a measurement. This new arrangement allows to enhance the rank of the matrix and so to detects new modes that better describe the system under analysis. The time-delay matrices are defined as follows (Equation 1.8):

$$\boldsymbol{X_{delay}} = \begin{bmatrix} x_1(t_1) & \cdots & x_1(t_n) \\ x_2(t_1) & \dots & x_2(t_n) \\ x_1(t_2) & \dots & x_1(t_{n+1}) \\ \vdots & \ddots & \vdots \\ x_1(t_d) & \dots & x_1(t_{fin}) \\ x_2(t_d) & \cdots & x_2(t_{fin}) \end{bmatrix} \qquad 1.8$$

Where d is the number of time-delay applied, and $x_1(t_k)$ and $x_2(t_k)$ are respectively the population of the Snowshore Hale and of the Canada Lynx in the timestep (t_k). The same procedure should be applied to the matrix $\boldsymbol{X}'$ obtaining the matrix $\boldsymbol{X}'_{\boldsymbol{delay}}$.

Starting from these two time-delayed matrices, the algorithm of the DMD is applied and thanks to the higher number of rows, it is possible to obtain more modes and so to predict the future behavior of the system with higher accuracy. To assess the forecast quality obtained using the DMD, an empirical dynamic model that simulate the predator-prey behavior will be used: The Lotka-Volterra that is shown in Equation 1.9:

$$\begin{cases}\dot{x} = (b - py)x \\ \dot{y} = (rx - d)y\end{cases} \qquad 1.9$$

Where:

x is the prey populations, y is the predator populations and b, p, r and d are tuning parameters that can assume values between zero and one.

The four parameters (b, p, r and d) should be accurately tuned using data to optimize the performance of the Lotka-Volterra model. In this work, these parameters will be tuned using two different optimization algorithms. Indeed, it is possible to determine the best set of b, p, r and d that minimize the error between the results obtained considering the Lotka-Volterra model and the real data. The first optimization tool adopted in the “fminsearch” used in MATLAB. The second algorithm implemented to determine the best set of parameters is a Natural-Based optimization known in literature as “Artificial Bee Colony (ABC)” [3]. The ABC is a swarm-based meta-heuristic algorithm, which exploits the analogy between food forage of a bee colony and research of the optimal solution in the allowed domain. In the ABC, the research of the optimal point is conducted through the introduction of three phases: employed bee, onlooker bee and scout bee phases. Firstly, all the solutions are initialized by randomly dispersing them in the allowed domain. Then, the employed bee phase is started and local researches to improve the solutions are performed by comparing two different solutions and random searches in the nearby of them. In the subsequent phase, called onlooker bee phase, the promising solutions are further investigated by assigning them a higher number of agents (i.e. bees) that conduct the research around these favorable solutions. Lastly, in all cases in which for a consecutive number of onlooker bee researches, the solution is not improved, the domain around it is considered fully exploited, so the solution is abandoned and a new one is randomly generated. These three phases are conducted iteratively until the maximum limit of iterations is reached. Since the ABC is a meta-heuristic algorithm, one of the main drawbacks of the algorithm, as reported in [4], is the high number of objective function evaluation required to obtain the optimal solution, and in this optimization process this drawback could be particularly severe since to evaluate the objective function it is required to solve a system of ODEs. On the other hand, the structure of the ABC algorithm is very simple and so the implementation is not so challenging. For this reason, in this work the ABC algorithm is implemented, and the results obtained are directly compared with the ones obtained using the “fminsearch” function.

Finally, the methodology called Sparse Identification of Nonlinear Dynamics (SINDy), is applied to determine the best nonlinear system that describe the data. Different algorithms will be applied to solve the associated linear system (i.e. lasso technique, pseudoinverse etc.).

Computational Results and Conclusions

The historical data of the population of Snowshore Hale and Canada Lynx are collected in a matrix 2x30 and then the algorithm described in the previous Section is implemented in MATLAB. Since the matrix has a number of rows equal to 2, the maximum rank of this matrix is two and so, during the SVD, only two modes can be detected. In particular in Figure 2 it is shown the diagonal elements of the matrix $\mathbf{\Sigma}$ opportunely normalized. As it can be noticed, it is not possible to neglect one of these two modes since the mode with less influence has in any case a weight of about the 20% with respect to the total. Since we are dealing with a low rank matrix ($r_{max} = 2$) and both of the modes are significant, the matrix $\boldsymbol{X}$ is not truncated. It is possible to analyze these two modes considering the eigenvalue decomposition, so in Figure 3 the two eigenvalues are represented in the complex plane. So the modes produce two negative real numbers, so the spectral expansion (Equation 1.8) consists in two decreasing exponentials with low capacity of predicting future trends.

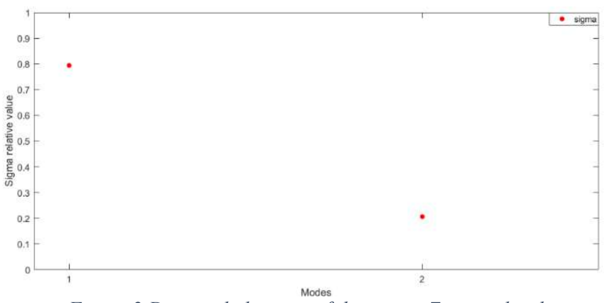

Figure 2 Diagonal elements of the matrix Σ normalized

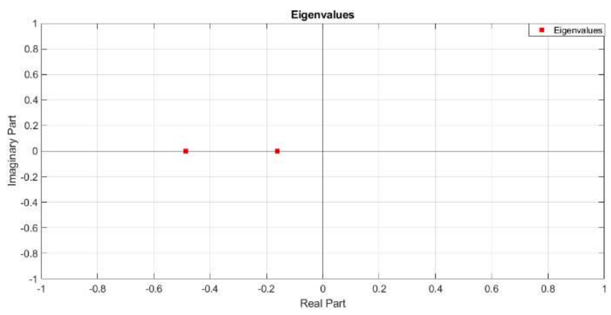

Figure 3 Eigenvalues of the matrix in the Gaussian plane

To increase the rank of the matrix and to detect additionally information (i.e. latent variables), the time-delay procedure is implemented. In this work the two most interesting values of time-delay number ($n_{delay} = [4,12]$) are presented. In Figure 4 and Figure 5 the normalized diagonal elements of the matrix $\mathbf{\Sigma}$ are shown, as can be clearly seen, it is possible to truncate, without introducing excessive errors, for r=6 in the first case and for r=12 in the second one.

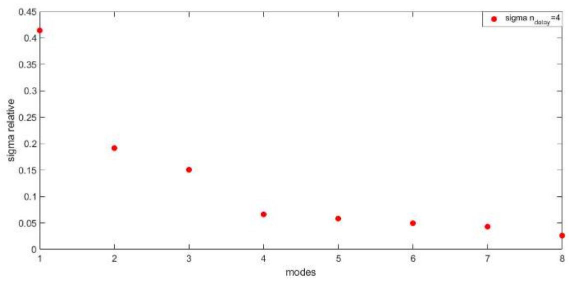

Figure 4 Diagonal elements of the matrix Σ in case of 4 times delay

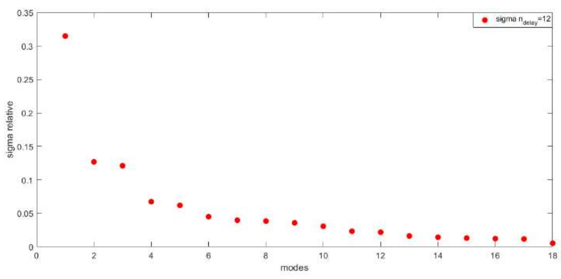

Figure 5 Diagonal elements of the matrix Σ in case of 12 times delay

Analyzing the eigenvalues obtained using 4 time-delay and a r=6 (Figure 6), it can be noticed that they have a negative real part and so the corresponding spectral expansion, obtained with Equation 1.8 will be characterized by a decreasing trend. Therefore, as shown in Figure 7, implementing the DMD with $n_{delay} = 4$ and $r = 6$ is not sufficient to predict accurately the future population trends. Increasing the number of time delay ($n_{delay} = 12$) and truncating the decomposition with r=12, the results obtained, in terms of future population state, are more accurate. Indeed, as shown in Figure 8, the real data curve and the curve obtained using the DMD forecast are similar and often overlapped.

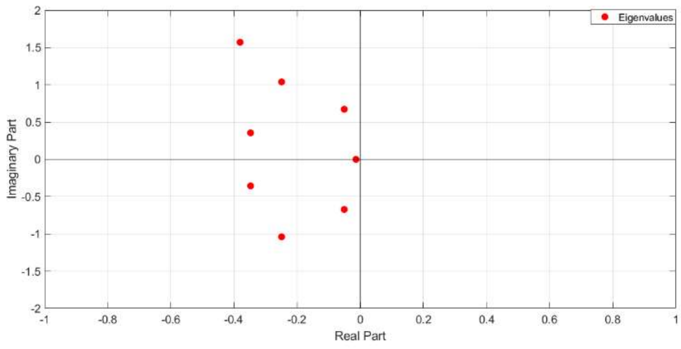

Figure 6 Eigenvalue in the complex plane in case of n_{delay}=4 and r=6

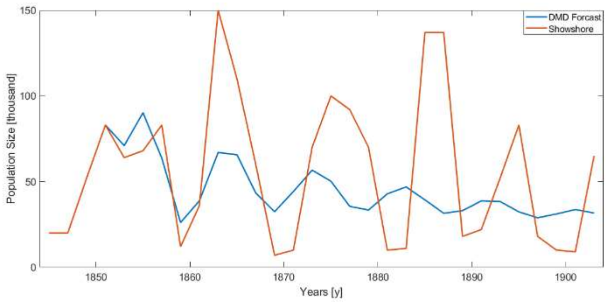

Figure 7 Comparison between the DMD forecast and the real data in case of n_{delay}=4 and r=6 and it is possible to notice the decreasing trend due to the negative real part of the eigenvalues

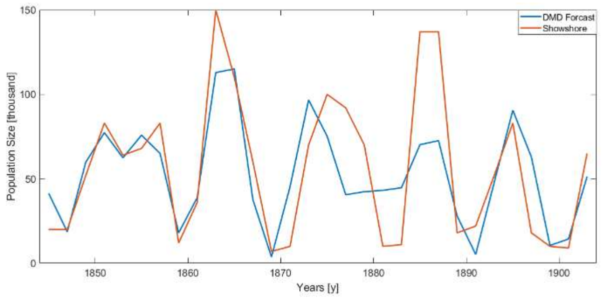

Figure 8 Comparison between the DMD forecast and the real data in case of n_{delay}=12 and r=6

To tune the coefficients of the Lotka-Volterra model, two different optimization processes are executed: the Artificial Bee Colony (ABC) and the "fminsearch" tool implemented in MATLAB. The two algorithms obtain similar results in terms of optimal value of the 4 parameters as shown in Table 1. For what concerns the computational time required to obtain the results, the ABC performs slightly worse since one of the main disadvantages of the Swarm-based optimization algorithm is the high number of objective function evaluation required to obtain the solution. In this particular case, to evaluate the objective function it is required to solve a system of ODE and so it could require few milliseconds during each evaluation. In any case, the difference in terms of computational time to solve the tuning problem between the ABC and "fminsearch" is less than 30 seconds. As shown in Figure 9 and Figure 10, the parameters obtained using the Artificial Bee Colony algorithm are more accurate with respect to the ones obtained with the "fminsearch". Indeed, looking the real data (in orange) and the trends of the Lotka-Volterra tuned with the "fminsearch" and ABC (respectively in yellow and blue), it is clear that the results obtained with the ABC are more precise, especially for the Canada Lynx population.

Table 1 Optimal parameters of the Lotka-Volterra model implementing the fminsearch optimization and the ABC

Parameter	Optimal(fminsearch)	Optimal(ABC)
b	0.6156	0.6023
p	0.0301	0.0309
r	0.5706	0.6033
d	0.0114	0.0116

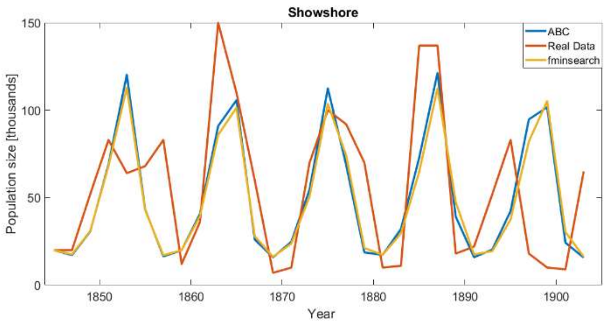

Figure 9 Real data of Showshore population and the results obtained using the Lotka-Volterra model tuned with the ABC (blue) and the fminsearch (yellow)

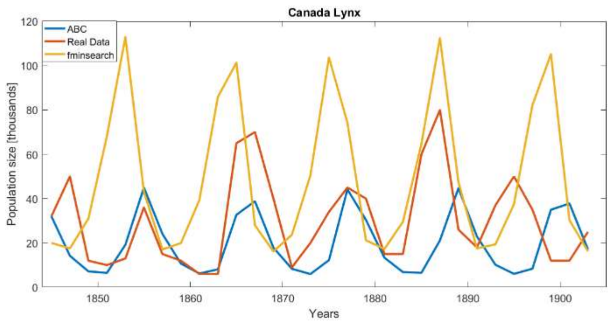

Figure 10 Real data of Canada Lynx population and the results obtained using the Lotka-Volterra model tuned with the ABC (blue) and the fminsearch (yellow)

Then, a Sparse identification of nonlinear dynamics (SINDy) is performed, starting from the population data shown in Figure 1, to identify an optimal non-linear dynamic system that describes the data. In particular three different techniques to solve the associated linear problem are used: i) the backslash operator, ii) the "pinv" function that calculates the pseudo inverse of the matrix and iii) the "lasso" function. For what concerns the matrix $\boldsymbol{A}$, it is created starting from the knowledge of the Lotka-Volterra model and it considers the following library of candidate functions:

$$A = [x_1, x_1 \cdot x_2, x_2, x_1^2, x_1^3, x_2^2, x_2^3, x_1^2 \cdot x_2, x_1 \cdot x_2^2, (\, x_1 \cdot x_2)^2] \qquad 1.10$$

Additionally, three different methodologies have been adopted to calculate the derivates: a) central difference scheme, b) backward scheme and c) forward scheme. Here, only the backward scheme will be analyzed since it produces more accurate results. In Figure 11 and Figure 12 it is shown the results in terms of coefficients obtained solving the linear system associated to the SINDy problem. The Figure 11 collects the coefficients obtained using the backslash (in orange) and the "pinv" (in blue) operators, instead in Figure 12 the ones calculated using the "lazzo" function are presented. As it is possible to see, the backslash and "pinv" methodologies produce the same results, indeed all the bars of the Figure 11 are overlapped. On the other hand, the "lasso" procedure solves differently the linear system, promoting the sparsity. In any case, the model obtained using the SINDy does not depend on the second element of the matrix $\boldsymbol{A}$ (i.e. $x_1 \cdot x_2$), as in the Lotka-Volterra model, but it depends only on x_1, and x_2. In particular, the model obtained using the "lasso" function is:

$$\begin{aligned} \dot{x} &= 0.213 \cdot x - 0.217y \\ \dot{y} &= 0.07 \cdot x - 0.219y \end{aligned} \qquad 1.11$$

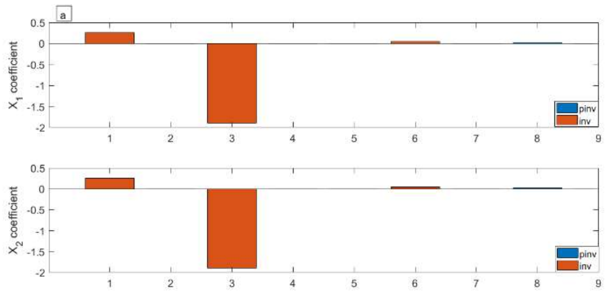

Figure 11 Results obtained adopting the pinv and inv functions to solve the linear problem. All the bars are overlapped

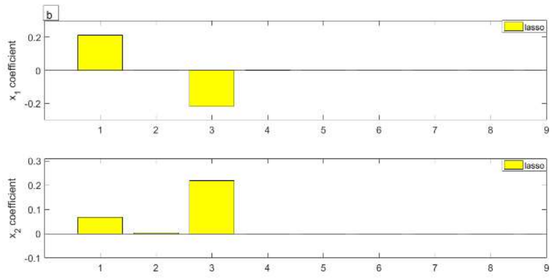

Figure 12 Results obtained adopting the lasso algorithm to solve the linear problem

Chapter 2
Neural Network

Introduction and Overview

In the following Chapter, three different systems will be analyzed: i) a reaction-diffusion, ii) Kuramoto-Sivashinsky (KS) iii) Lorentz systems. For what concerns the reaction-diffusion and the Lorentz problems, the NN will be trained in the full domain dimensional space (t, x, y, z), instead the reaction-diffusion equation will be analyzed considering a lower dimensional subspace obtained through a Singular Value Decomposition (SVD) process.

Theoretical Background

2.1 Reaction diffusion equations

The reaction-diffusion is a set of partial differential equations, which describe different physical phenomenon. In this work the reaction-diffusion equations are solved in a 2D domain, to obtain the horizontal and the vertical velocity, respectively u and v, in all the point of the domain (x-y) and in the considered timesteps (Δt=0.05). Therefore, the solutions are two tensors ($N_x \times N_y \times N_t$). Then, the results obtained are analyzed considering the SVD and the DMD methodology to evaluates the performance in predicting the future development of the system.

2.2 Kuramoto-Sivashinsky equations

The Kuramoto-Sivashinsky equations are a set of fourth-order nonlinea PDF typically adopted to describe the diffusion flame front. They are one of the simplest nonlinear PDEs that exhibit spatiotemporally chaotic behavior

2.3 Lorentz equations

Neural Network is a methodology used to determine the relationship between input and output data. The general structure of a NN is composed of an input and an output layers, and between them there are different hidden layers connected together. Figure 13 presents the structure of a simplified NN. From the mathematical point of view, a NN can be represented by a composite function which includes all the possible connection between the input and output. Therefore, the NN procedures aim to optimize the parameters ($\vec{\omega}_i$) of this complex function in order to determine the optimal map between the input and the output data, as shown in Equation 2.1.

$$\vec{\boldsymbol{O}} = f(\vec{\omega}_i, \vec{\boldsymbol{I}}) \qquad 2.1$$

The activation functions which connects two different layers are typically non-linear in order to increase the performance and the flexibility of the NN. Then to enhance the stability of the algorithm during the optimization process, the non-linear activation functions are usually simple and differentiable. Indeed, during the optimization algorithms implemented to tune the parameters ($\vec{\omega}_i$), they are based on the gradient calculation.

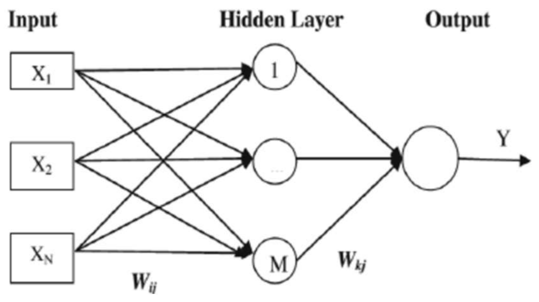

Figure 13 Simplified representation of a Neural Network structure

Algorithm Implementation and Development

2.1 Reaction diffusion equations

To develop a tool that allows to forecast the development of the system, the DMD methodology is applied to the tensors u and v. In particular the two tensors are firstly reshaped into matrices adopting the horizontal slice techniques shown in Figure 14. Using this approach, it is possible to transform a tensor into matrices in which each matrix collects the data of the velocity registered for a fixed x-coordinate in all the y-coordinates and in all the timesteps, as reported in Figure 15. For each of these matrices, the DMD algorithm, described in Chapter 1, is applied to forecast the future development of the flow.

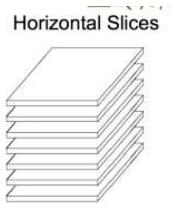

Figure 14 Horizontal slice method to transform a 3D tensor into a set of matrices

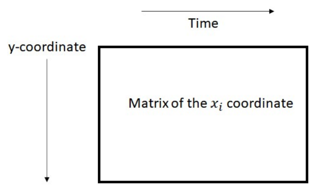

Figure 15 Each slice of the tensor collects the time data evolution along the rows and the y-coordinate data into the columns

2.3 Lorentz equations

For what concerns the Lorentz equations, a NN is trained to advance the solution from the timestep t to the timestep $t + \Delta t$ also in case of variation of the ρ parameter. In particular the goal is to obtain a network that can determine the next timestep $(t + \Delta t)$ also in case of variation of the ρ with respect to the training data. To reach this goal, in this work the matrix shown in Equation 2.2 has been used to train the NN:

$$\boldsymbol{I} = \begin{bmatrix} \rho_1 & x(t_0) & y(t_0) & z(t_0) \\ \rho_1 & x(t_1) & y(t_1) & z(t_1) \\ \vdots & \vdots & \vdots & \vdots \\ \rho_1 & x(t_{fin}) & y(t_{fin}) & z(t_{fin}) \\ \rho_2 & x(t_0) & y(t_0) & z(t_0) \\ \vdots & \vdots & \vdots & \vdots \\ \rho_2 & x(t_{fin}) & y(t_{fin}) & z(t_{fin}) \\ \vdots & \vdots & \vdots & \vdots \\ \rho_n & x(t_{fin}) & y(t_{fin}) & z(t_{fin}) \end{bmatrix} \qquad 2.2$$

Therefore, each row of the input matrix ($\boldsymbol{I}$) collects:

- in the first column the value of the ρ parameter,
- in the other three columns the values of the three coordinates in corresponding timestep.

The input dataset is composed by the solution of the Lorentz equation in the interval $T = [t_0; t_{fin}] = [0; 8]$ with a time resolution equal to Δt=0.01. For what concerns the ρ parameter, a large set of values has been used to train the NN. In particular, all the even number between 10 and 40 are used, so $\overrightarrow{\rho_{train}} = (\rho_1, \rho_2 \dots \rho_n) = (10{,}12 \dots 40)$. Then, as described in the next Section, the performances of the NN will be tested considering the capability to determine the evolution of the Lorentz equation with different values assumed by the ρ parameter. In particular will be tested the performances considering two different values of ρ (i.e. $\overrightarrow{\rho_{test}} = (17{,}35)$). Therefore, since the value assumed by $\overrightarrow{\rho_{test}}$ are not included in $\overrightarrow{\rho_{train}}$, the NN should interpolate correctly to determine the solution of the Lorentz equations.

The structure of the NN used in this work is shown in Figure 16 and it has:

- four inputs $(\rho, x(t), y(t), z(t))$;
- four outputs $(\rho, x(t + \Delta t), y(t + \Delta t), z(t + \Delta t))$;
- three hidden layers, each of them composed by 15 nodes.

The transfer functions adopted for the hidden layers are respectively: i) Log-sigmoid, ii) radial basis and iii) linear.

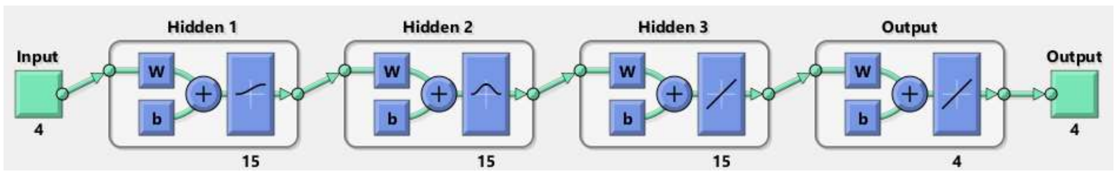

Figure 16 General structure of the NN adopted in this work

For what concerns the final exercise, in which it is asked to predict the transition from one lobe to another, different techniques have been tried to solve to determine the transition. The most promising, but particularly challenging to implement, implies the calculation of the second derivatives of the data, since during the transition from one lobe to another the concavity of the curve changes. The Figure 17 shows the proposed methodology. Unfortunately, the approach proposed does not result to be successful, probably due to an insufficient quality of the second derivatives calculation.

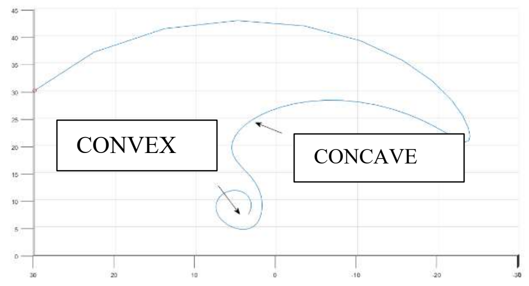

Figure 17 Methodology proposed to determine the transition between two lobes

Computational Results and Conclusions

2.1 Reaction diffusion equations

To test and validate the capability of the DMD techniques to forecast the development of the simulated flow using the diffusion reaction equation, only the first 150 timesteps are used. The other ones (the remaining 50 timesteps) are forecasted using the DMD model, and then, the outcomes obtained forecasting the flow evolution are compared with the ones coming from the real data. In order to strongly reduce the computational time required to obtain the DMD decomposition, a low rank subspace of the domain is adopted. In particular, each matrix, which collects the velocity data in a fixed x-coordinate as a function of the time and the y-coordinates, is transformed, using the DMD, into a set of three matrices, opportunely truncated. As shown in Figure 18, only the first 20 modes of the matrix have an important role in the dynamic of the system. Therefore, in this work, it is chosen to consider only the first 20 modes of the SVD matrices ($\boldsymbol{U_r}, \boldsymbol{V_r}, \boldsymbol{S_r}$). After the truncation, the DMD algorithm is implemented and Equation 2.3 is used to evaluate the system forecast.

$$\vec{x}(t) = \sum_{i}^{r} \boldsymbol{\phi_i}\, e^{\overline{\omega}_i t}\, \vec{b}_i \qquad 2.3$$

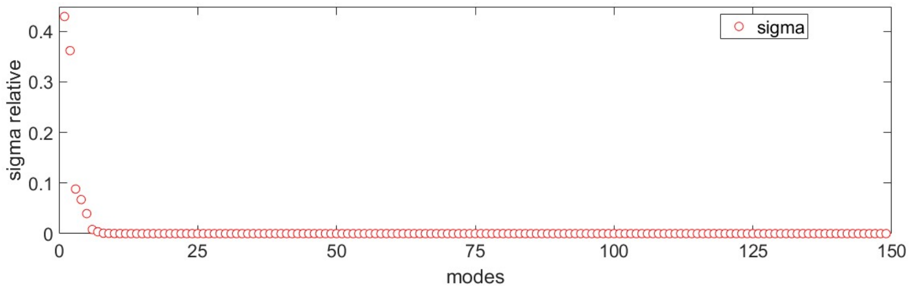

Figure 18 Diagonal elements of the matrix Σ opportunely normalized

In the next figures, the real data (on the top) and the results obtained using the DMD algorithm (the bottom figure) are compared, for different timesteps. In particular in Figure 19 the results for t=100 are presented: observing this figure it is possible to underline the capability of the algorithm to represent the reaction-diffusion system by using only 20 modes. While, in Figure 20 and Figure 21Figure 21 the results obtained for t=175 and t=200 are shown. As described in the previous Section, the data used to define the DMD collect only the first 150 timesteps. Therefore, these last two figures are produced by an extrapolation of the data and they can be used to establish the quality of the forecast produced by the DMD spectral expansion (Equation 2.3). The outcomes obtained are particularly interesting: indeed, the truncate expression (r=20) well describes the system not only during the timesteps used to define the spectral expansion (i.e. for t<150), but also in case of forecast (i.e. t>150). So, Figure 20 and Figure 21 highlight the quality of the DMD spectral expansion to predict the future development of the system since the real data for t>150 are not used to define the DMD spectral expansion. In conclusion these two figures represent an example of extrapolation.

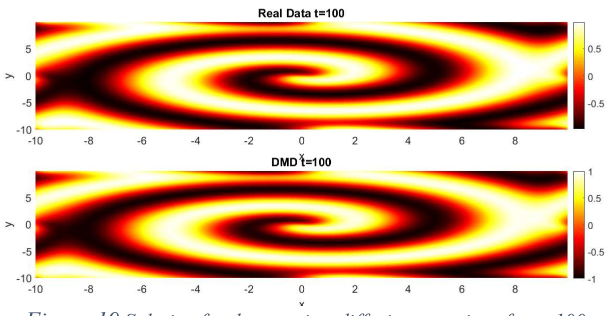

Figure 19 Solution for the reaction-diffusion equations for t=100, real data on the top, approximation with DMD and r=20 on the bottom

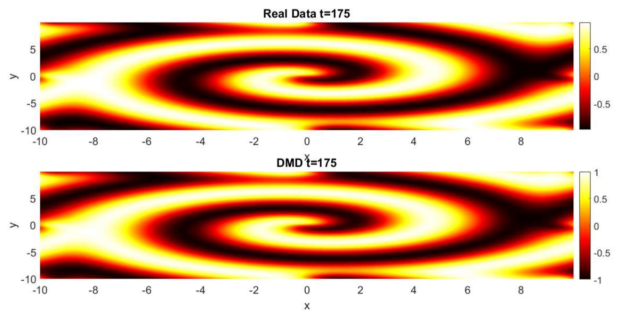

Figure 20 Solution for the reaction-diffusion equations for t=175, real data on the top, approximation with truncated DMD (r=20) on the bottom

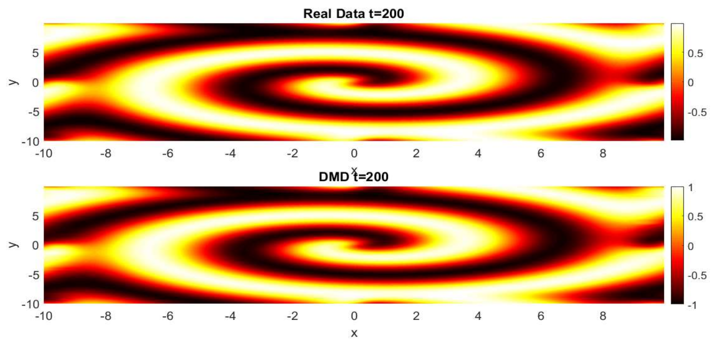

Figure 21 Reaction-diffusion solution for t=200. It should be underlined the quality of the results obtained, considering that the DMD was defined using the firsts 150 timesteps.

2.2 Lorentz equations

For what concerns the Neural Network (NN) to model the Lorentz equations, it was trained using the results (i.e. $x(t), y(t), z(t)$) of the Lorentz oscillator obtained with even numbers for the rho parameters ($\overrightarrow{\rho_{train}} = (\rho_1, \rho_2 \dots \rho_n) = (10{,}12 \dots 40)$). To test the capability to correctly interpolate, it is asked to the proposed NN to solve the Lorentz equations in case of odd values of the rho parameters (i.e. $\overrightarrow{\rho_{test}} = (17{,}35)$). The results obtained are shown in the next pictures, where the figures on the left part are characterized by ρ=17, while Figure 23 and Figure 25 collect the outcomes for ρ=35. As it is clearly shown, the value assumed by the ρ parameter has a huge role in the quality of the results obtained. Indeed, the solutions achieved using the NN in case of ρ=17 overlap with the real solutions achieved solving the system of ODEs. On the other hand, increasing the value assumed by ρ, the outcomes found by the proposed NN tend to be different with respect to the results of the ODE solver implemented in MATLAB. Indeed, as reposted in [5] the ρ parameter has an important role in the definition of the stability/instability of the trajectory.

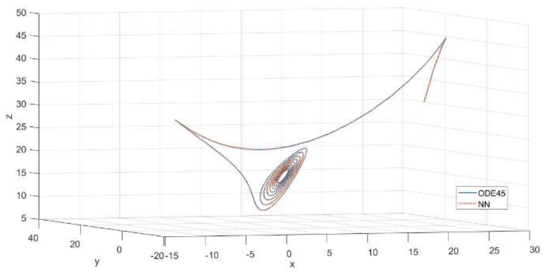

Figure 22 Lorentz equation the 3D space for ρ=17. The blue line is the trajectory found solving the system with the ODE solver and the orange one is obtained using the NN

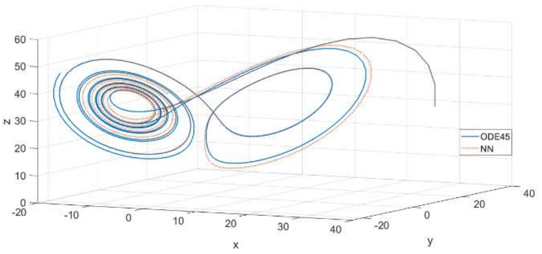

Figure 23 Lorentz equation the 3D space for ρ=35 The blue line is the trajectory found solving the system with the ODE solver and the dotted orange one is obtained using the NN trained.

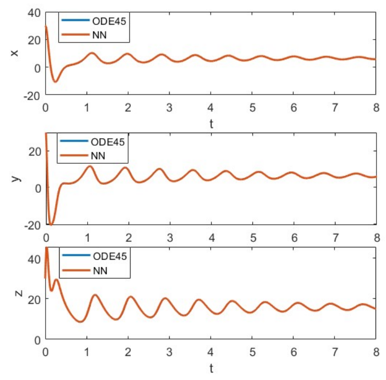

Figure 24 Comparison between the results obtained with the ODE solver and the NN for ρ=17

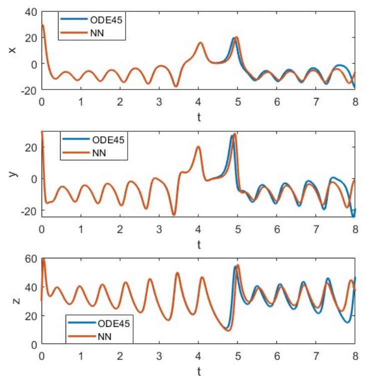

Figure 25 Comparison between the results obtained with the ODE solver and the NN for ρ=35

CHAPTER 3
BELOUSOV-ZHABOTINSKY CHEMICAL OSCILLATOR

Introduction and Overview

The Belousov–Zhabotinsky reaction is a classic example of non-equilibrium thermodynamic reaction and it is particularly interesting since it shows that some chemical reactions are not dominated by thermodynamic equilibrium [6]. In this Chapter, the truncated SVD has been applied to assess the capability of this method to compress the data. In particular, the dimension of the tensor which contains the original data will be compared to the dimension of the data obtained through a truncated SVD process and then, using the KPIs defined in [7], the quality of the compression and the errors introduced by it will be analyzed.

Theoretical Background

The SVD is a technique that factorized a generical matrix $\boldsymbol{A}$ into the form expressed by Equation 3.1:

$$\boldsymbol{A} = \boldsymbol{U\Sigma V}^* \tag{3.1}$$

Where $\boldsymbol{U}$ and $\boldsymbol{V}$ are unitary matrices and $\boldsymbol{\Sigma}$ is a rectangular diagonal matrix with non-negative real elements. As described in Chapter 1, it is possible to approximate the initial matrix $\boldsymbol{A}$ by considering the truncate representation, presented in Equation 3.2, where r is the truncation value and it is included between one and the maximum rank of the matrix $\boldsymbol{A}$.

$$\boldsymbol{A} \approx \boldsymbol{U}_r \boldsymbol{\Sigma}_\mathrm{r} \boldsymbol{V}_r^* \tag{3.2}$$

And the compressed matrix is defined by Equation 3.3:

$$\boldsymbol{C} = \boldsymbol{U}_r \boldsymbol{\Sigma}_\mathrm{r} \boldsymbol{V}_r^* \tag{3.3}$$

To evaluate the capability of the truncated SVD to compress the data, it is firstly defined the Compression Ratio (CR) as the ratio between the total number of bytes required to store uncompressed data (i.e. the matrix $\boldsymbol{A}$) and the total number of bytes required to store compressed data (i.e. the matrix $\boldsymbol{C}$). Since in this work we adopt the SVD algorithm to compress the data, the total number of bits required to store the compressed data is represented as the sum of the bits dimension of the three truncated matrices (i.e. $\boldsymbol{U}_r, \boldsymbol{\Sigma}_\mathrm{r}\ and\ \boldsymbol{V}_r^*$). The truncated SVD is a compression technique that introduces compression loss, so the reconstructed data actually varies from original data. Therefore, to assess the level of distortion, fidelity and quality of the reconstructed data, the Peak Signal to Noise Ratio (PSNR) is defined as:

$$PSNR = 20\log_{10}\frac{\max[A_i]}{\sqrt{MSE}} \tag{3.4}$$

$$MSE = \frac{1}{MN}\sum_{i=0}^{M-1}\sum_{j=0}^{N-1}\left\|A_{i,j} - C_{i,j}\right\|^2 \tag{3.5}$$

Computational Results and Conclusions

In this last section, the results obtained using the truncated SVD to compress the data of the BZ reaction will be analyzed. In particular three main aspects will be considered: the size in bit of the compressed data, the quality of the compression (calculated using the KPI PSNR) and finally the Compression Ratio (CR). In Figure 26, the PSNR values are represented for different values of r as a function of the x-coordinate. Indeed, as explained in the previous chapter, the tensor that contains the data of the BZ reaction, is analyzed considering the subdivision in matrices thanks to the horizontal slice method. Then, all these matrices are considered separately to determine the SVD decomposition. In Figure 27 it is shown the size (in bytes) of the different matrices (S_r, U_r, V_r) as function of the truncation value (r) and the corresponding CR. As can be clearly seen from these two figures, the truncated SVD allows to detect the modes that are most important to describe the system. Therefore, it is not surprising that adding new modes to the system (i.e. increasing r), the PSNR increases but not so strongly. Indeed, the SVD method is capable to extract from the initial matrix the fundamental features with only few modes and increasing r only allows to introduce details that does not modify so strongly the quality of the reconstructed data, as shown in Figure 28 and Figure 29.

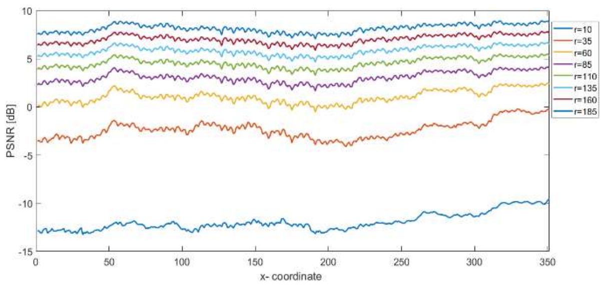

Figure 26 PSNR as function of the x-coordinate for different truncation value (r)

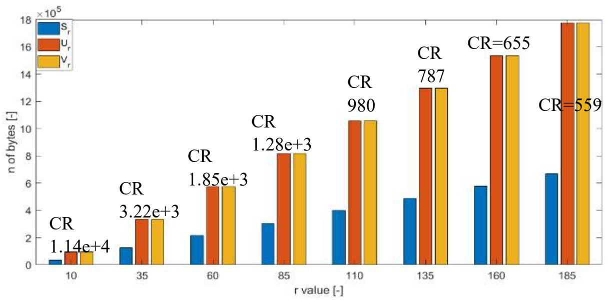

Figure 27 Size in Bytes of the tree matrices, for different truncation value. The corresponding CRs are also indicated

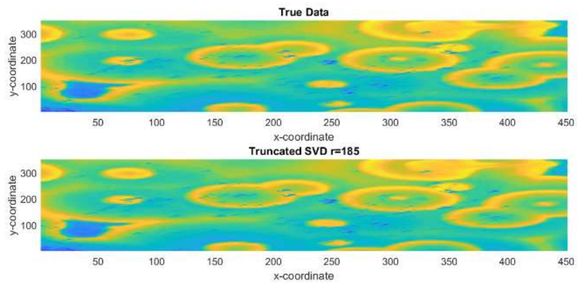

Figure 28 Real data (on the top) and reconstructed data for r=185 (bottom) of the BZ reaction

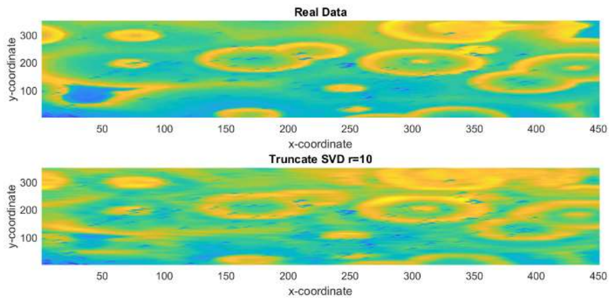

Figure 29 Real data (on the top) and reconstructed data for r=10 (bottom) of the BZ reaction

Bibliography

[1] S. L. Brunton and J. N. Kutz, *Data-Driven Science and Engineering*. 2019.

[2] J. H. Tu, C. W. Rowley, D. M. Luchtenburg, S. L. Brunton, and J. N. Kutz, "On dynamic mode decomposition: Theory and applications," *J. Comput. Dyn.*, 2014, doi: 10.3934/jcd.2014.1.391.

[3] D. Karaboga and B. Basturk, "A powerful and efficient algorithm for numerical function optimization: Artificial bee colony (ABC) algorithm," *J. Glob. Optim.*, 2007, doi: 10.1007/s10898-007-9149-x.

[4] T. K. Sharma, M. Pant, and T. Bhardwaj, "PSO ingrained artificial bee colony algorithm for solving continuous optimization problems," in *ICCAIE 2011 - 2011 IEEE Conference on Computer Applications and Industrial Electronics*, 2011, doi: 10.1109/ICCAIE.2011.6162114.

[5] J. Hateley, *The Lorenz system*. available online: http://web.math.ucsb.edu/~jhateley/paper/lorenz.pdf .

[6] V. Petrov, V. Gáspár, J. Masere, and K. Showalter, "Controlling chaos in the Belousov - Zhabotinsky reaction," *Nature*, 1993, doi: 10.1038/361240a0.

[7] J. Uthayakumar, T. Vengattaraman, and P. Dhavachelvan, "A survey on data compression techniques: From the perspective of data quality, coding schemes, data type and applications," *J. King Saud Univ. - Comput. Inf. Sci.*, 2018, doi: 10.1016/j.jksuci.2018.05.006.