Data-driven science and engineering: mathematical methods to model systems from measurements

Massimo Brunetti

Leonardo Spa, via Giovanni Agusta 520, 21017 Samarate (VA), ITALY

[massimo.brunetti@leonardocompany.com]

Abstract. This project report is in partial fulfilment of the requirements of course “Modelling from Measurements” #055461, managed by prof. A.Berizzi from Politecnico di Milano and held by prof. N.J.Kutz from University of Washington. The project has been developed in the shed of the mathematical methods presented during the course for the purpose to understand their use and assess their effectiveness against data from different domains, either pre-calculated or generated by physical equation solvers. Following a brief introduction to the course and relevant research field, all the main methods adopted in the project are presented from a theoretical perspective. In the central section, method implementation is reviewed while in the final two sections results and conclusions are drawn for comparison with the theoretical background. All codes generated by MATLAB® and available on GitHub (https://github.com/massimo-brunetti/modelling-from-measurements).

Keywords – Singular Value Decomposition, Dynamic Model Decomposition, Neural Networks, Machine Learning

BZ	Belousov-Zhabotinsky	NN	Neural Network
DMD	Dynamic Mode Decomposition	ODE	Ordinary Differential Equation
DOF	Degree of Freedom	PDE	Partial Differential Equation
FF	Feed Forward	PP	Predator-Prey
LA	Lorenz Attractor	ROM	Reduced Order Modelling
LV	Lotka-Volterra	RS	Reaction-Diffusion
IC	Initial Condition	SINDY	Sparse Identification of Nonlinear DYnamics
KS	Kuramoto-Sivashinsky	SVD	Singular Values Decomposition
SML	Supervised Machine Learning	TDE	Time Delay Embedding

I. INTRODUCTION AND OVERVIEW

Dynamic modelling from measurements is a thriving research field that aims at synthetizing models of the process generating the data from the field.

Synthetized models are agnostic (i.e. not based on the physical equations governing the process) and typically sensitive to noise and interferences, but oftentimes the only viable approach to practitioners. Moreover, if a physical insight is available, this can be advantageously combined to capture the discrepancy between the expected and the measured data (residual). For instance, dynamic modeling can improve data assimilation (Kalman theory) by refining the model available for the process so as to yield the best dynamical system representation.

The model can then be exploited to forecast the system status, to optimize it or to control it.

From this perspective, the optimal low rank data approximation by singular value decomposition (SVD); the reconstruction by spatially correlated modes having a linear behavior in time (DMD); the dynamic forecast by trained neural networks (NN) and data regression/classification by supervised machine learning (MSL): are some of the most interesting techniques presented in the course.

In the following sections, examples of their application to different non-linear systems are provided along with numerical evidence of their supposed effectiveness.

II. THEORETICAL BACKGROUND

Singular Value Decomposition (SVD). Let X be a matrix of data from the field and structured in such a way that each column represents a "snapshot" of the system status (nx1), captured at a given time k from 1 to m:

(1) $$\underset{nxm}{X} = \begin{bmatrix} | & & | & & | \\ x_1 & \dots & x_k & \dots & x_m \\ | & & | & & | \end{bmatrix}$$

Then the Singular Value Decomposition that exists for every complex-valued matrix $X \in \mathbb{C}^{nxm}$ is according **(2),** where Σ is a diagonal matrix and U, V are two distinguished matrices having the columns and the rows respectively orthogonal and with unitary L^2 norm:

(2) $$\underset{nxm}{X} = \underset{nxn}{U} \cdot \underset{nxm}{\Sigma} \cdot \underset{mxm}{V^*} = \underset{nxm}{X} = \underset{nxm}{U} \cdot \underset{mxm}{\Sigma} \cdot \underset{mxm}{V^*} \text{ (n ≥ m) = } \underset{nxm}{X} = \underset{nxn}{U} \cdot \underset{nxn}{\Sigma} \cdot \underset{nxm}{V^*} \text{ (n ≤ m)}$$

The geometrical interpretation of SVD is that the columns of U provide an orthonormal basis for the column of X, and V an orthonormal basis for the rows of X. Since U encodes spatial patterns, its columns are also regarded as "spatial modes" and since V encodes temporal patterns, its columns as "temporal modes".

(3) $$U \cdot U^* = V \cdot V^* = \mathrm{I}$$

Typically, singular values curve exhibits a hard threshold **[1]** that corresponds to the real DOF number of the system. U is the privileged coordinate system in which data turn into a simple, linear combination of temporal modes, modulated by the relevant singular values:

(4) $$U^* \cdot X = \Sigma \cdot V^*$$

Notably, this is a similar result to that achievable by Eigen decomposition **(5),** which however works with one single eigenvectors basis only (non-orthogonal and limited to full-rank square matrices or operators only):

(5) $$W^{-1} \cdot X = \Lambda \cdot W^{-1}$$

Actually, one of the most useful and defining property of SVD is that it provides an optimal low-rank approximation to a matrix X, in a least square sense, by keeping the leading r singular values and vectors and discarding the rest. In consideration of this, SVD is widely used in system identification and control theory to obtain reduced order models that are balanced in the sense that states are hierarchically ordered.

(6) $$\underset{nxm}{X} = \underset{nxr}{U} \cdot \underset{rxr}{\Sigma} \cdot \underset{rxm}{V^*} = \sum_{j=1}^{r} \sigma_j \cdot u_j \cdot v_j{}'$$

Dynamic Mode Decomposition (DMD). Originally conceived by P.Schmid in 2010 **[2]**, this methods seeks the leading spectral decomposition (i.e. eigenvalues and eigenvectors) of the best-fit linear operator A that relates the two data matrices X and X' defined below:

(7) $$X' = A \cdot X \Leftrightarrow \begin{bmatrix} | & & | \\ x_2 & \dots & x_{m+1} \\ | & & | \end{bmatrix} = A \cdot \begin{bmatrix} | & & | \\ x_1 & \dots & x_m \\ | & & | \end{bmatrix}$$

In principle, the linear operator A could be computed through the pseudo inverse of matrix X, but given its large dimension this would be computationally demanding. Instead, SVD of matrix X appears a better option in consideration of the fact that Σ is diagonal and the possibility to perform a model order reduction by rank-truncation:

(8) $$\underset{nxn}{A} = \underset{nxm}{X'} \cdot \underset{mxr}{V} \cdot \underset{rxr}{\Sigma^{-1}} \cdot \underset{rxn}{U^*}$$

Actually, the reconstruction (and forecast) of the process can be achieved even without resort to the computation of the operator A but through the Eigen decomposition of a similar operator $\tilde{A}$ projected via the matrix U:

(9) $$\underset{rxr}{\tilde{A}} = \underset{rxn}{U^*} \cdot \underset{nxn}{A} \cdot \underset{nxr}{U} = U^* \cdot X' \cdot V \cdot \Sigma^{-1} = W \cdot \Lambda \cdot W^{-1}$$

The high dimensional DMD modes Φ can then be expressed using the eigenvectors W of the reduced order system and the time-shifted matrix X':

(10) $$\Phi = X' \cdot V \cdot \Sigma^{-1} \cdot W$$

Recalling the fact that A and $\tilde{A}$ share the same eigenvalues, it's easy to verify that the DMD modes Φ are indeed the eigenvectors of the high-dimension matrix A, corresponding to eigenvalues Λ :

(11) $$A \cdot \Phi = \left(X' \cdot V \cdot \Sigma^{-1} \cdot U^* \right) \cdot \left(X' \cdot V \cdot \Sigma^{-1} \cdot W \right) = \Phi \cdot \Lambda$$

The SVD of data matrix X along with the Eigen decomposition of operator $\tilde{A}$ and computation of Φ is the so called "DMD algorithm", which enables the reconstruction (and forecast) of the process, directly at any time step in the future:

(12) $$X_k = \Phi \cdot \Lambda^{k-1} \cdot \vec{b} = \sum_1^r \phi_j \cdot \lambda^{k-1} \cdot b_j \Leftrightarrow \Phi \cdot \Lambda^{\Omega \cdot t} \cdot \vec{b} = \sum_1^r \phi_j \cdot e^{\omega_j \cdot t} \cdot b_j$$

Usually, the mode amplitudes b are obtained using the initial conditions while for sake of interpretability, the spectral expansion is re-formulated using continuous eigenvalues $\omega = \log(\lambda) / \Delta t$ (ω complex frequency).

Time Delay Embedding (TDE). In many real-world systems, the underlying dynamics will be non-linear, and thus the DMD reconstruction and forecast of the process will not yield good results. Unlike linear systems, non-linear systems do not satisfy the superposition principle, can exhibit chaotic behavior and do not generally admit analytical solutions. Koopman operator theory is an emerging framework for analyzing such systems. According to Koopman, it is always possible to find a transformation that increasing the state dimension can make the global system dynamics linear:

(13) $$\begin{cases} \dfrac{dx}{dt} = f(\mathrm{x}) \\ x_{k+1} \approx A \cdot x_k \end{cases} \Rightarrow \begin{cases} \dfrac{dy}{dt} = g(f(\mathrm{x})) \\ y_{k+1} = K \cdot y_k \end{cases}$$

On the other hand, Takens' embedding theorem establishes that under certain technical conditions, Time Delay Embedding (TDE) can better capture the "true" system dynamics and dig-out possible latent variables **[3].** This is done by augmenting the information contained in the system state with measurements of its history:

(14) $$H = \begin{bmatrix} x_1 & x_2 & \cdots & x_m \\ x_2 & x_3 & \cdots & x_{m+1} \\ \cdots & \cdots & \cdots & \cdots \\ x_{1+d} & x_{2+d} & & x_{m+d} \end{bmatrix}$$

More recently, the TDE/Hankel representation and Koopman theory have been connected **[4]**. In practice, Time Delay Embedding can be regarded as an approximated expression of Koopman transformation and the Hankel matrix, the augmented data set on which to perform DMD (as if generated by a more-linear system).

(15) $$H' = K \cdot H$$

This approach provides a simple back-path to the original state dimension and coordinate system (by simply extracting the original data from the Hankel matrix) and it is especially useful whenever the data set from field is "poor" in connection to the non-linearity of the system

Sparse Identification of Nonlinear DYnamics (SINDY). The problem of extrapolating dynamical systems models from data has always played a central role, with the eminent example of Newton and the planetary motion. Historically, this process has relied on a combination of high-quality measurement, expert intuition … and luck, leading to an explicit form of f :

(16) $$\frac{d}{dt} X = f(\mathrm{X})$$

The Sparse Identification of Nonlinear DYnamics (SINDY) is an alternative, automated approach which leverages on the fact that many systems have a dynamics with only few terms. Under this condition, the extrapolation problem turns into a more circumscribed fitting problem of a library of governing equations terms, each one weighted by an unknown coefficient ξ :

(17) $$f(\mathrm{X}) \approx \sum_{k=1}^{p} \theta_k(x) \cdot \xi_k = \Theta(X) \cdot \Xi$$

In practice, having arranged the data in snapshots, numerical derivatives can be computed from them using a finite difference (e.g. centered) scheme:

(18) $$X = [\mathrm{x}(\mathrm{t}_1) \quad \mathrm{x}(\mathrm{t}_2) \quad \ldots \quad \mathrm{x}(\mathrm{t}_m)]^T$$

(19) $$\dot{X} = [\dot{\mathrm{x}}(\mathrm{t}_1) \quad \dot{\mathrm{x}}(\mathrm{t}_2) \quad \ldots \quad \dot{\mathrm{x}}(\mathrm{t}_m)]^T$$

It is then possible to solve for the relevant coefficients using sparse regression algorithms that penalize the number of terms in the dynamics and scales well to large problems.

(20) $$\dot{\mathrm{X}} = \Theta(X) \cdot \Xi$$

Neural Networks (NN). Inspired by Noble prize winning work of Hubel and Wiesel on the primary visual cortex of cats, artificial neural networks were firstly mathematically modelled in 1980 under the name of “Neocognitron”. Neocognitron already had many of today’s NN characteristics, including a multi-layer structure and non-linear dynamics nodes.

The architecture of the simplest (i.e. feed-forward) NN consists of an input layer, a variable number of hidden layers and an output layer. Each layer can have a variable number of nodes (neurons) each feeding those it is connected to through its output (y), as a function of its inputs (x). If this function (Activation Function) is linear, then the input-output mapping of the NN turns into a simple linear system, where each matrix A_j contains the coefficients (weights) that map each layer into the next.

(21) $$y = \left[A_m \cdot A_{m-1} \cdot \ldots \cdot A_2 \cdot A_1\right] \cdot x$$

Although the dimensionality of the input layers $X \in \mathbb{R}^n$ is known, there is great flexibility in choosing the dimension of the hidden layers as well as how to structure the output layer. The number of layers and how to chain the layers is also selectable by the user. For instance, a NN for binary classification may consist of just an input layer (with a number of neurons equals to the input dimension) and an output layer (with one single neuron). In this case, the determination of the matrix coefficients (training) mapping the input into the output space, would require to solve an overdetermined linear equation system:

(22) $$\underset{nxm}{Y} = \underset{nxn}{A} \cdot \underset{nxm}{X} \qquad \mathrm{n} \le \mathrm{m}$$

Similar considerations apply to non-linear NN, where non-linearity comes from a non-linear Activation Function. This can be of any kind but typically with fast derivative computation (for back-propagation), each layer and each neuron potentially featuring a different one:

(23) Binary $$f(x) = \begin{cases} 0 & x \le 0 \\ 1 & x > 0 \end{cases}$$

(24) Logsig $$f(x) = \frac{1}{1 + \mathrm{e}^{-x}}$$

(25) Radbas $$f(x) = \mathrm{e}^{-x^2}$$

With this approach, the linear input-output map can be re-formulated as a compositional function, which error (distance) from a given data set must be minimized (supervised learning):

(26) $$y = f_m(A_m, f_{m-1}(A_{m-1}, \ldots (f_1(A_1, x_1))))$$

NN training is then performed using stochastic gradient descent and back-propagation algorithms that try to minimize the error. Clearly, the training of a non-linear NN becomes less predictable (both in term of computational time and final result) but also more powerful in that non-linearity can help to map low into high dimensional space (e.g. Koopman transformation) or capture “deep” features **[5],** otherwise filtered out by linear NN**:**

(27) $$E = \left\| Y_0 - Y \right\|$$

Anyway, regardless of the selected architecture, activation function, training set and optimization method, it should be remarked that NN are still, essentially, “maps” that once developed represent a straightforward and very fast computational tool.

III. ALGORITHM IMPLEMENTATION AND DEVELOPMENT

FILE	PP_1 (ref. to Homework point 1.1)
STRUCTURE	▪ DATA INITIALIZATION ▪ DMD ALGORITHM ▪ DMD PREDICTION ▪ RESULTS PLOTTING
DESCRIPTION	After data initialization, **DMD modes** are computed as per the canonical algorithm. DMD reconstruction and forecast (on extended time basis) are then plotted next to the original data for comparison. **DMD eigenvalues** (discrete and continuous) are sub-plotted to ease the recognition of the mode damping (real part) and mode frequency (imaginary part).
NOTES	*This script embodies the data set defined in **Appendix A.1***
FILE	PP_2 (ref. to Homework point 1.2)
STRUCTURE	▪ DATA INITIALIZATION ▪ TDE-DMD ALGORITHM ▪ TDE-DMD PREDICTIOM ▪ RESULTS PLOTTING
DESCRIPTION	This script is similar to PP_1 but Time Delay Embedding technique is applied before performing DMD on **Henkel matrix**. Maximum **delay is adjustable** (variable "d") to see the effect on DMD eigenvalues/modes, hence reconstruction & forecast.
NOTES	*This script embodies the data set defined in **Appendix A.1***
FILE	PP_3_A / B (ref. to Homework point 1.3)
STRUCTURE	▪ DATA INITIALIZATION ▪ PARAMETERS FITTING ▪ DISCREPANCY DMD ← version B only ▪ RESULTS PLOTTING
DESCRIPTION	[Version A] After data initialization, parameters b,p,r,d are roughly estimated according to **[6],** that is: using the population oscillation period; using the two species equilibrium points; using the preys Malthusian growth when predators are few. This is the initial guess that is further refined using **fmeansearch.** This function calls a separate cost function **PP_3_sub**, implemented as a separate file. PP_3_sub solves the system for the input parameters passed by PP_3 using **ODE23** and returns the error from the original data set. Optimized parameters and resulting population are then plotted next to original data set for comparison. [Version B] This is similar to the previous with the addition of discrepancy modelling by DMD. "Corrected" curve is then plotted for comparison.
NOTES	*This script embodies the data set defined in **Appendix A.1*** *This script requires **PP_3_sub** function to operate*
FILE	PP_4 (ref. to Homework point 1.4)
STRUCTURE	▪ DATA INITIALIZATION ▪ LINEAR SYSTEM COMPUTATION ▪ WEIGHTS FITTING & PREDICTION ▪ RESULTS PLOTTING
DESCRIPTION	After data initialization, this scripts computes the numerical derivatives (centered scheme), required to build the (overdetermined) linear systems associated with the equation library (**SINDY**). The weights of each term are determined using **lasso** with a hard threshold to promote sparsity. Resulting weights and population curves (computed by ODE23) are then plotted next to original data set for comparison.
NOTES	*This script embodies the data set defined in **Appendix A.1***
FILE	KS_A / B (ref. to Homework point 2.1-2.2)
STRUCTURE	▪ DATA INITIALIZATION ▪ NN TRAINING ▪ ODE SOLUTION Vs NN PREDICTION ▪ RESULTS PLOTTING
DESCRIPTION	[Version A] After data initialization (which calls **KS_solver** to compute a single numerical solution, starting from a reduced IC dimension) this script trains a FF NN with three hidden layers (5x5x5) and nonlinear activation functions (logsig, radbas, pureline). Training input is equal to the solution at different time steps while the output is equal to a **time-shifted copy** of the input. The trained NN is then used to predict a new solution, which is plotted next to the "true" (computed via KS_solver). [Version B] This is similar to the previous, but here the NN is trained on multiple solutions (computed via KS_solver for different IC) in order to teach the NN about the "chaotic behavior" of KS (outlined by a **waterfall** graph).
NOTES	*This script requires **KS_solver** function to operate*

FILE	RD (ref. to Homework point 2.3)
STRUCTURE	▪ DATA INITIALIZATION ▪ SVD & ROM ▪ NN TRAINING ▪ NN PREDICTION ▪ RESULTS PLOTTING
DESCRIPTION	After data initialization (which loads **reaction_diffusion_big** data file) this script performs a SVD on pre-calculated data to discover the real DOF number. SVD matrices are truncated according to the number of significant singular values and a FF NN with three hidden layers (10x10x10) and nonlinear activation functions (logsig, radbas, pureline) is trained to map the temporal modes. Training input is equal to the **temporal modes** up to a given threshold while the output is equal to a time-shifted copy of the input. NN prediction (reconstruction and forecast) is plotted for comparison with original data set.
NOTES	*This script requires **reaction_diffusion_big** data file to operate*
FILE	LA_1_2_A / B (ref. to Homework point 3.1)
STRUCTURE	▪ DATA INITIALIZATION ▪ NN TRAINING ▪ ODE SOLUTION VS NN PREDICTION ▪ RESULTS PLOTTING
DESCRIPTION	[Version A] After data initialization (which calls ODE23 to compute a set of trajectories for different IC and rho values) this script trains a FF NN with three hidden layers (5x5x5) and nonlinear activation functions (logsig, radbas, pureline). Training input is equal to the set of 3D trajectories while the output is equal to a **time-shifted copy** of the input. The trained NN is then used to predict two new solutions, for different IC and a rho value not in the training set. Predicted trajectories are projected on X Y X axes for comparison with the "true" (ODE23) solution. [Version B] This is similar to the previous, but here the NN is trained on a set of "augmented" trajectories including X Y Z + rho. NN prediction also includes rho, which average is reported on top of the graph.
NOTES	*This script is self-contained*
FILE	LA_3_A / B (ref. to Homework point 3.2)
STRUCTURE	▪ DATA INITIALIZATION ▪ LABELLING FOR SML (version A) / TRIGGERING FORCE COMPUTATION (version B) ▪ NN TRAINING ← version A only ▪ ODE SOLUTION VS NN PREDICTION ← version A only ▪ RESULTS PLOTTING
DESCRIPTION	[Version A] After data initialization (which calls ODE23 to compute a set of trajectories for different IC and a single value of rho) this script trains a FF NN with three hidden layers (5x5x5) and nonlinear activation functions (logsig, radbas, pureline). Training input is equal to trajectory projections over the X axis and a time window (C1,W1) at distance (horizon) from a second (C2,W2). Training output is equal to a **set of labels** (1/-1) determined upon the transitioning behavior, from one lobe to another, over this second window. The trained NN is then used to predict the transition of a new set of solutions for different IC and same rho value. The entire simulation is repeated for increasing values of the horizon, to see the effect on NN prediction. [Version B] After data initialization, this script performs a SVD on trajectory projection (X-axis) to extract the temporal mode that does not fit into a linear scheme and which can be interpreted as a **triggering force** causing the transition **[7].** The triggering force and the 1st (dominant) mode are plotted overlapped to appreciate the cause-effect relationship.
NOTES	*This script is self-contained*
FILE	LA_4_A / B (ref. to Homework point 4)
STRUCTURE	▪ DATA INITIALIZATION ▪ DATA RESHAPE AND SVD ▪ NN TRAINING (version A) / DMD ALGORITHM (version B) ▪ NN PREDICTION (version A) / DMD PREDICTION (version B) ▪ RESULTS PLOTTING
DESCRIPTION	[Version A] After data initialization (which loads and rescale **BZ** data file, for computational handling purpose) this script performs a SVD on pre-calculated data to discover the real DOF number. SVD matrices are truncated according to the number of significant singular values and a FF NN with three hidden layers (10x10x10) and nonlinear activation functions (logsig, radbas, pureline) is trained to map the **temporal modes**. Training input is equal to the temporal modes up to a given threshold and the output is equal to a time-shifted copy of the input. NN prediction (i.e. reconstruction and forecast) is then plotted for comparison with original data set. [Version B] This is similar to the previous but here the NN is replaced by DMD on projected/truncated data (proportional to temporal modes through sigma matrix).
NOTES	*This script requires **BZ** data file to operate*

IV. COMPUTATIONAL RESULTS

PREDATOR-PREY

The graph of Predator-Prey data (see Appendix A.1) shows an oscillatory but non-harmonic behavior, indicative of a nonlinear system dynamics (Fig. 1). Regression by DMD to a linear system model, leads on two damped modes, offering a poor reconstruction and forecast (Fig. 2). Expansion by TDE technique up to the maximum Sigma rank (d = 15) reveals a more complex nature indicative of latent variables (Fig. 3). Continuous eigenvalues gathered by conjugate couples give rise to oscillatory modes, with small damping (real part < 0.05) (Fig. 4). Reconstruction and forecast by TDE-DMD (truncated up to 11th mode) shows a much better correlation with original data, although somehow delayed (see the prey double peak shift from 1850/55 to 1855/65) (Fig. 5). Assigned data are then fitted using Lotka-Volterra model, which is based on physical insight and governing equations. Initial guess is made qualitatively, by observing the oscillation period and the equilibrium points of the two species along with the prey Malthusian growth **[6]**. This leads to a first parameter guess [p, p, r, d] = [0.71, 0.05, 0.04, 1.01] (Fig. 6), further refined using Lagarias method [1.1, 0.04, 0.02, 0.76] (Fig.7). However, Lotka-Volterra reconstruction ends up with too many and too small oscillations (Fig. 8), possibly due to a rough parameter guess and/or noise affecting the data. Discrepancy modelling attempted by DMD and following "corrected" reconstruction does not yield the expected improvement either (Fig. 9). Finally, use of SINDY in conjunction with a simple equation library (purely polynomial, up to 2nd order) and fixed threshold to eliminate the least significant weights (w<0.01) discovers a reasonable cross-dependency between prey and predators (Fig. 10, 11, 12).

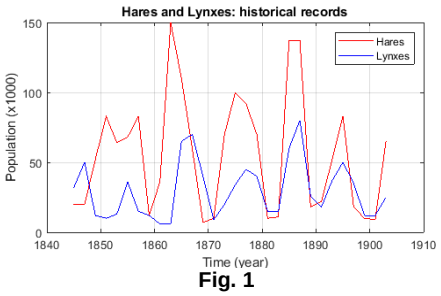

Fig. 1

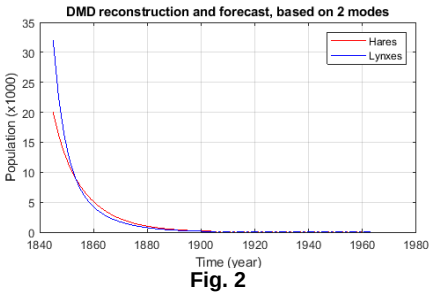

Fig. 2

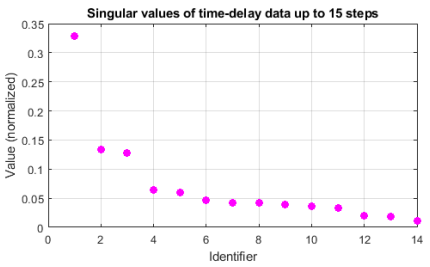

Fig. 3

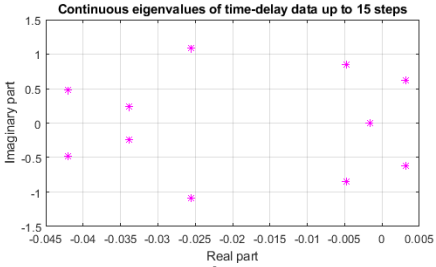

Fig. 4

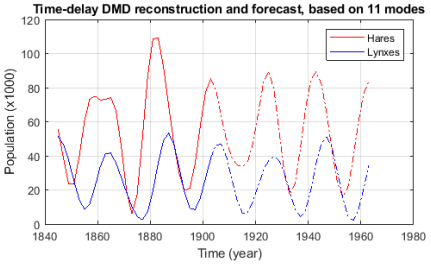

Fig. 5

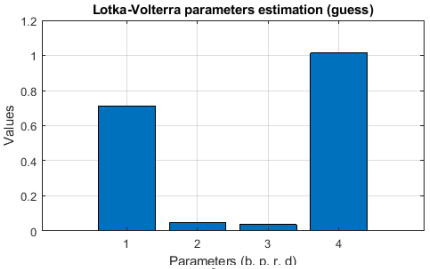

Fig. 6

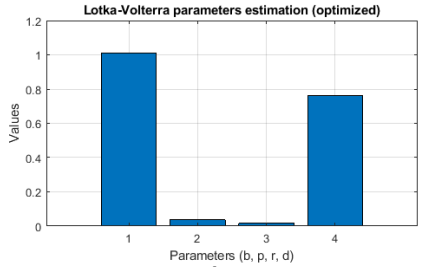

Fig. 7

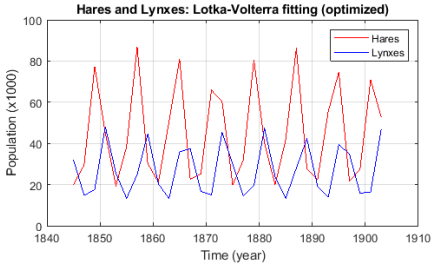

Fig. 8

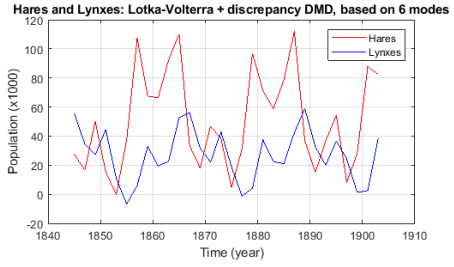

Fig. 9

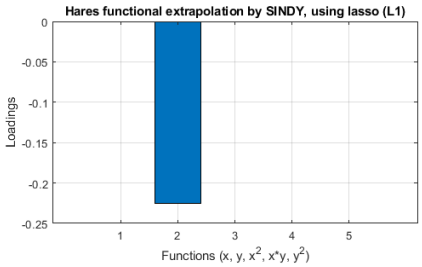

Fig. 10

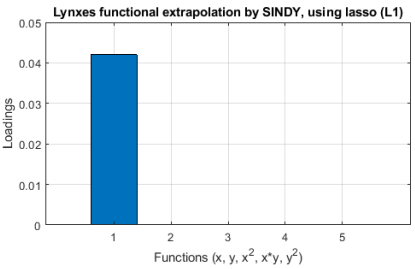

Fig. 11

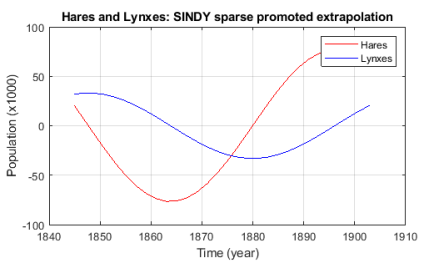

Fig. 12

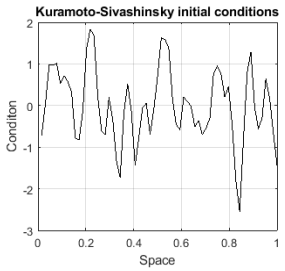

Fig. 13

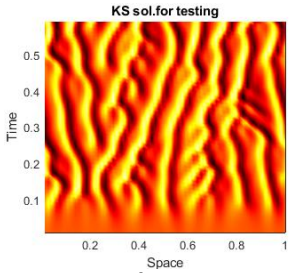

Fig. 14

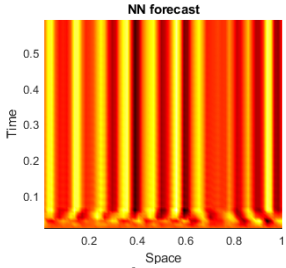

Fig. 15

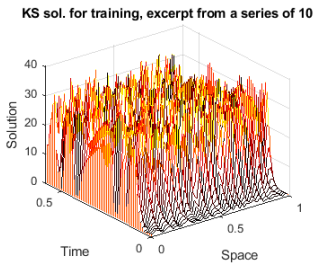

Fig. 16

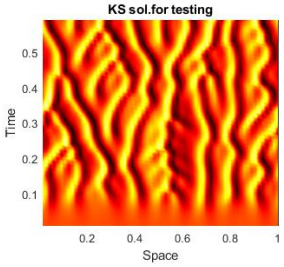

Fig. 17

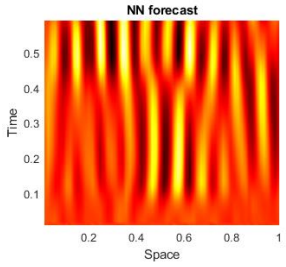

Fig. 18

KURAMOTO-SIVASHINSKY

Kuramoto-Sivanshinsky is a partial differential equations that even in one dimension tends to be numerically unstable **[8]**. It is thus non-surprising that training a simple FF NN to advance a single solution from t to t+Δt (Fig. 14), for a given initial condition (Fig. 13) does not yield satisfactory results (Fig. 15). Actually, looking at the waterfall plot of KS solution, it becomes apparent that event the faintest variation in the initial condition (time = 0) is amplified in time, leading to very different flame temperature profiles (Fig. 16). However, training the same FF NN on a set of ten different solutions obtained solving the equation for as many different initial conditions yields better results (Fig. 18). The improvement can be appreciated observing the NN forecast along space slices that - by comparison to the previous attempt - appear less uniform.

REACTION-DIFFUSION

Reaction-Diffusion solution at time = 0 (Fig. 19) is first reshaped to a 2D matrix then processed via SVD to reveal a singular value hard threshold around 4 (fig. 20). The first temporal modes are actually almost harmonic (Fig. 23) – unlike the higher modes, much similar to white noise. A FF NN is then trained to advance the first temporal modes, from t to t+Δt, on 80% of the data (Fig. 24). Temporal modes predicted by the very same NN enable the reconstruction (up to time = 8) and forecast (up to time = 10) of RD solution. However, as anticipated by "spaghetti-incident" at time = 6 (Fig. 24), whereas RD reconstruction at time = 5 is good (Fig. 21 vs. Fig. 25) RD forecast at time = 10 is highly distorted (Fig. 22 vs. Fig. 26).

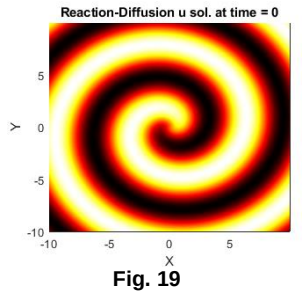

Fig. 19

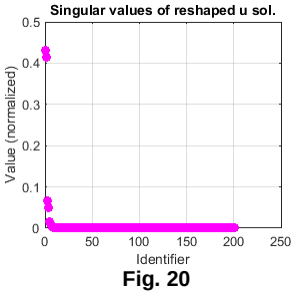

Fig. 20

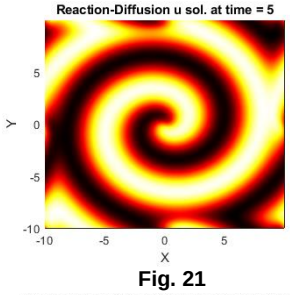

Fig. 21

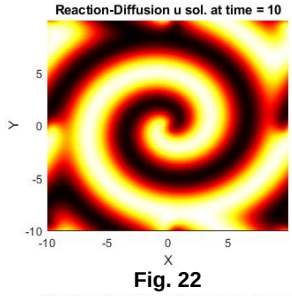

Fig. 22

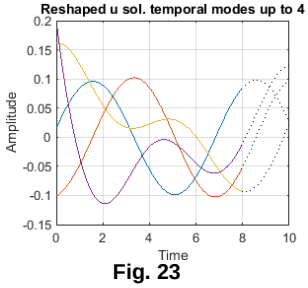

Fig. 23

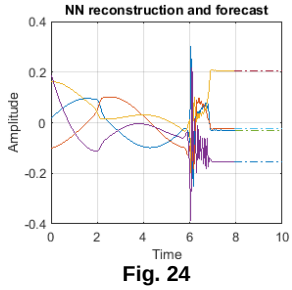

Fig. 24

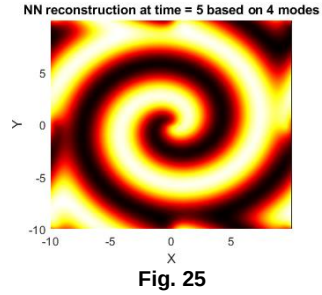

Fig. 25

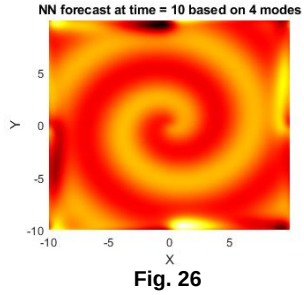

Fig. 26

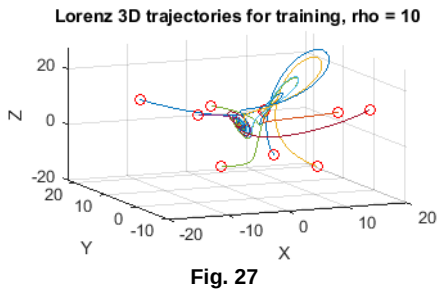

Fig. 27

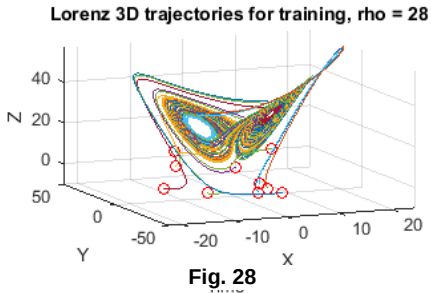

Fig. 28

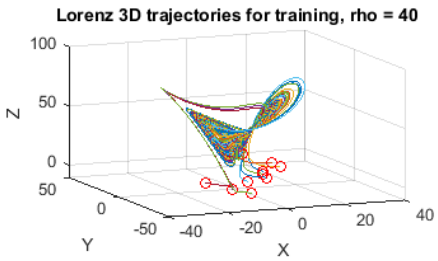

Fig. 29

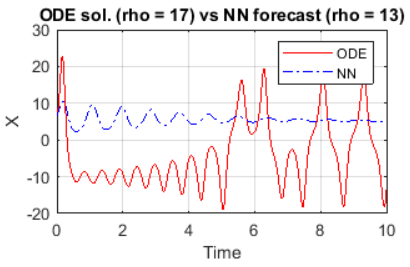

Fig. 30

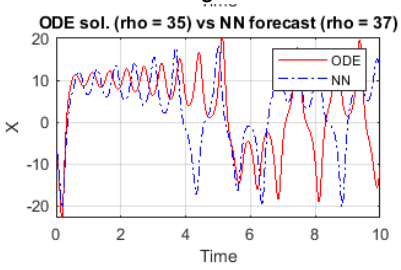

Fig. 31

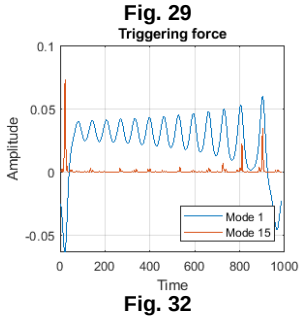

Fig. 32

LORENZ ATTRACTOR

Lorenz attractor system is well known for its chaotic behavior **[9]**: small variations in the initial conditions lead to totally different trajectories, with multiple transitions from one lobe to another, and especially for large values of rho (Fig 27, 28, 29). Here, training a NN to learn the system dynamics on a large set of trajectories for different values of rho is not effective. However, if the set of trajectories is augmented including rho as a 4th coordinate, the NN seems to be able to capture the relationship existing between trajectories and this latter. Moreover, since rho goes into the training set, it is also returned as part of the NN prediction and results indicate that the better the rho prediction is, the better the trajectory forecast too (Fig. 31, 32). Notably, transitions from one lobe to another can be interpreted as a reflection of the system non-linearity, or the r-1 temporal mode of trajectory SVD that does not fit into a linear scheme **[7].** Compared to the dominant mode, this other mode is mostly negligible except for some peaks well related to the lobe transition (Fig. 32). An alternative approach to forecast the transitioning behavior of the system is Machine Supervised Learning. In this case, for a fixed value of rho, a FF NN is trained to learn the transitioning behavior of the system (labelled as 1/-1) over a future time window (indicated in red), based on the trajectory elongations over a previous (Fig. 33, 34). The NN is then asked to forecast the transitioning behavior of a new set of trajectories, leading to very accurate (100% correct) forecast (Fig. 37, 38). The same experiment is then repeated for increasing values of the temporal horizon (from 2 to 5), showing a progressive degradation of the forecast accuracy (Fig. 39, 40).

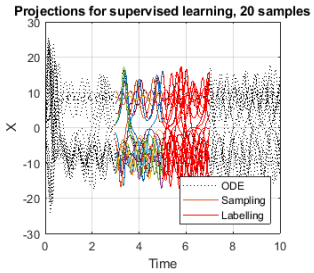

Fig. 33

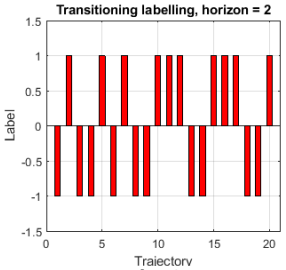

Fig. 34

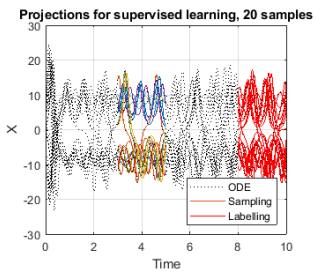

Fig. 35

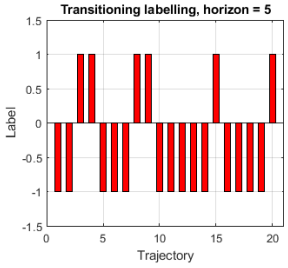

Fig. 36

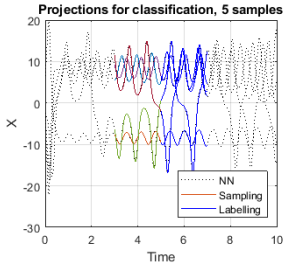

Fig. 37

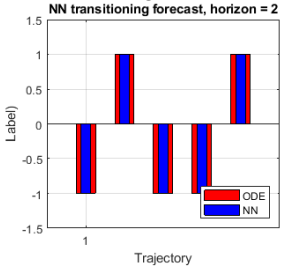

Fig. 38

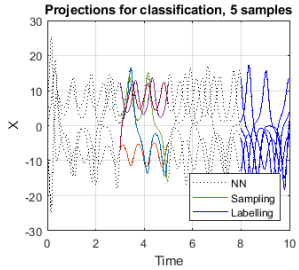

Fig. 39

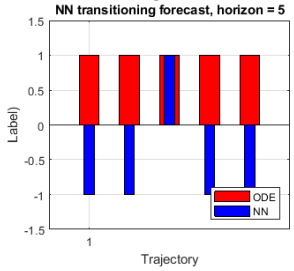

Fig .40

BELOUSOV-ZHABOTINSKY

Belousov-Zhabotinsky solution at time = 0 (Fig. 41) is firstly rescaled /reshaped, then processed by SVD to extract the dominant singular values (Fig. 42). Unfortunately, the decay is slow, so that a high number of temporal modes is needed to approximate the original data set properly (Fig. 45). Consistently, data are projected through a reduced (same) number of spatial modes (Fig. 49). Then, two different approaches are pursued for comparison. The first leverages a FF NN to capture the temporal modes (training performed on 80% of total duration). In this case, the NN prediction makes for "not so a terrible" reconstruction (Fig. 47 vs. Fig. 43) and forecast (Fig. 48 vs. Fig. 44). Actually, looking at the NN prediction it can be noticed that the modes are slightly distorted and propagated periodically (Fig. 46), so that reconstruction and forecast are almost identical. The second approach leverages instead a DMD regression of projected data to a linear model approximation (regression performed on 80% of total duration). In this second case, the DMD prediction makes for a flawless reconstruction (Fig. 51 vs. Fig. 43), but very poor forecast (Fig. 52 vs. Fig. 44). Actually, looking at DMD prediction it can be noticed that on the long term, DMD modes tend to diverge due to the positive real part of the dominant eigenvalues (Fig. 50). So that the two approaches are somehow complementary: if DMD is better at reconstruction (*image compression*), the NN is better on long term forecast.

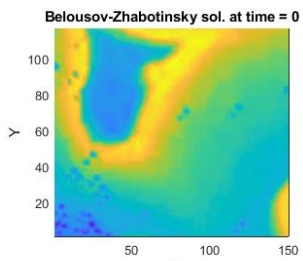

Fig. 41

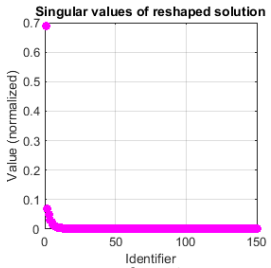

Fig. 42

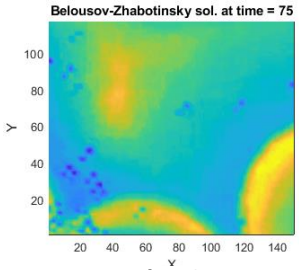

Fig. 43

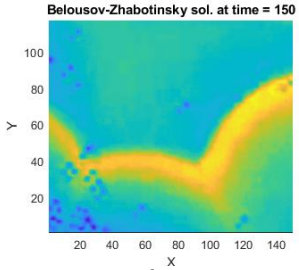

Fig. 44

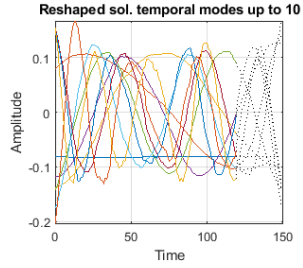

Fig. 45

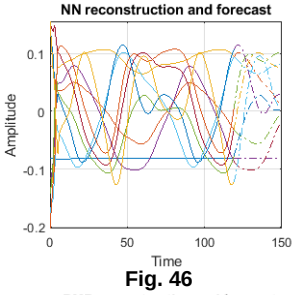

Fig. 46

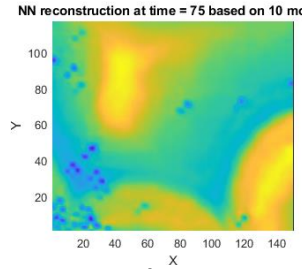

Fig. 47

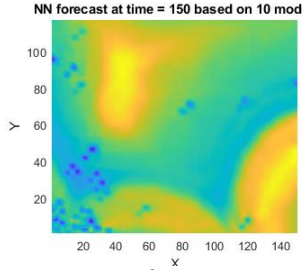

Fig. 48

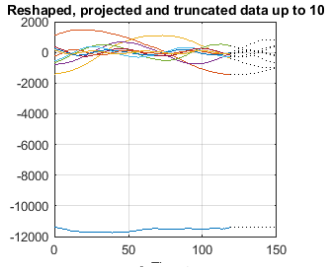

Fig. 49

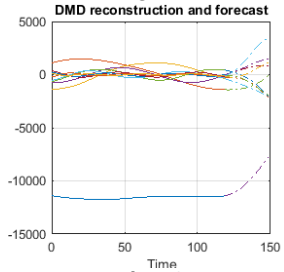

Fig. 50

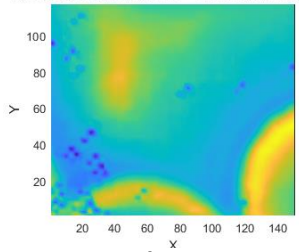

Fig. 51

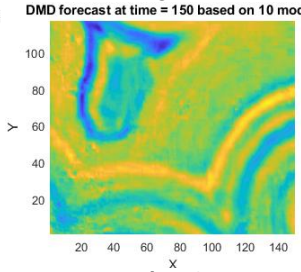

Fig. 52

V. SUMMARY AND CONCLUSIONS

Dynamic modelling from measurements is a central research field, virtually connected to any other and that aims at finding useful results for the real-world problems inspired by parsimony, interpretability and generalizability principles.

Some of the mathematical techniques presented in this course are still quite young and unsatisfactory results are just the confirmation of the existing opportunities both in terms of algorithm and strategy improvement. Actually, the solution to a given problem can blossom out of the combination or adaptation of different techniques, such as a dynamic mode decomposition cascaded to a time-delayed-embedding or singular-value-decomposition. Other tools, such as neural networks, can yield positive results only when trained on enough, clean data under very circumvented, very specific computational framework.

Today's data-driven big challenge revolves around autonomy - in either ground transports, robotics, energy or medicine - but it is worth to recall that everything began with observing the sky, and wondering whether it existed an alternative perspective and model of the planetary motion.

APPENDIX A – GUIDELINES

1) Predator-Prey (PP). Consider the following historical (and classic) data set concerning Canadian lynx and snowshoe hare populations from 1845 to 1903.

Year ('800)	45	47	49	51	53	55	57	59	61	63	65	67	69	71	73	75	77	79	81	83	85	87	89	91	93	95	97	99	01	03
Hare pelts ($x10^3$)	20	20	52	83	64	68	83	12	36	150	110	60	7	10	70	100	92	70	10	11	137	137	18	22	52	83	18	10	9	65
Lynx pelts ($x10^3$)	32	50	12	10	13	36	15	12	6	6	65	70	40	9	20	34	45	40	15	15	60	80	26	18	37	50	35	12	12	25

1.1 Develop a DMD model to forecast the future population states
1.2 Do a time-delay DMD model to produce a forecast and compare with regular DMD; Determine if it is likely that there are latent variables.
1.3 Empirical Predator-Prey models such as Lotka-Volterra are commonly used to models such phenomenon. Consider following model and use data to fit the values of b, p, r, d.

$$\begin{cases} \dot{x} = (b - py)x \\ \dot{y} = (\mathrm{r}x - d)\,\mathrm{y} \end{cases}$$

1.4 Find the best nonlinear, dynamical system model fitting to the data using sparse regression

2) Kuramoto-Sivashinsky (KS) and Reaction-Diffusion (RD). Further to the accompanying MATLAB solver for a Reaction-Diffusion (RD) system of equations, and Kuramoto-Sivashinsky (KS) equation.
2.1 Train a NN that can advance the solution from t to t + Δt for the KS equation
2.2 Compare your evolution trajectories for your NN against using the ODE time-stepper provided with different initial conditions
2.3 For the Reaction Diffusion system, first project to a low-dimensional subspace via the SVD and see how forecasting works in the low-rank variables.

3) Lorenz-Attractor (LA). For the Lorenz equations, consider the following:
3.1 Train a NN to advance the solution from t to t + Δt for ρ = 10; 28; 40. Now see how well your NN works for future state prediction for ρ = 17 and ρ = 35.
3.2 See if you can train your NN to identify (for ρ = 28) when a transition from one lobe to another is imminent. Determine how far in advance you can make this prediction. (NOTE: you will have to label the transitions in a test set in order to do this task)

4) Belousov-Zhabotinsky (ZB). Further to the data set BZ.mat (which is a snipet from a Belousov-Zhabotinsky chemical oscillator) see what you can do with the data

APPENDIX B – REFERENCES

[1] M. Gavish and D.L Donoho. The optimal hard threshold for singular values is $4/\sqrt{3}$. IEEE Transactions on Information Theory – 2014

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

[3] F. Takens. Detecting strange attractors in turbulence. Dynamical Systems and Turbulence, Lecture Notes in Mathematics – 1981.

[4] M. Kamb, E. Kaiser, S.L. Brunton, J.N. Kutz. Time-Delay Observables for Koopman. Journal on Applied Dynamical Systems, – 2020

[5] M.Z. Alom et al. The History Began from AlexNet: A Comprehensive Survey on Deep Learning Approaches. arXiv:1803.01164 – 2018

[6] J.M. Mahaffy. Lotka-Volterra Continuous Model. State University of San Diego, Lecture Notes in Mathematical Modeling – 2018

[7] S.L. Brunton, B.W. Brunton, J.L. Proctor, E. Kaiser and J.N. Kutz. Chaos as an intermittently forced linear system. Nature Communications, 8 – 2017.

[8] https://encyclopediaofmath.org/wiki/Kuramoto-Sivashinsky_equation

[9] https://encyclopediaofmath.org/wiki/Lorenz_attractor