

## THESIS / THÈSE

### DOCTOR OF SCIENCES

#### A Journey Through Reciprocal Space from Deep Spectral Learning to Topological Signals

Giambagli, Lorenzo

*Award date:*  
2024

*Awarding institution:*  
University of Namur

[Link to publication](#)

#### General rights

Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

- Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
- You may not further distribute the material or use it for any profit-making activity or commercial gain
- You may freely distribute the URL identifying the publication in the public portal ?

#### Take down policy

If you believe that this document breaches copyright please contact us providing details, and we will remove access to the work immediately and investigate your claim.



UNIVERSITÀ  
DEGLI STUDI  
FIRENZE



Physics and Astronomy - University of Florence (IT)  
Mathematics - University of Namur (BE)

CYCLE XXXVI

## A Journey Through Reciprocal Space: from Deep Spectral Learning to Topological Signals

Italian Academic Discipline (SSD) FIS/02

### Doctoral Candidate


Dr. Lorenzo GIAMBAGLI

  
\_\_\_\_\_

### Supervisors

Prof. Duccio FANELLI (Florence)

Prof. Timoteo CARLETTI (Namur)

  
\_\_\_\_\_  
\_\_\_\_\_

### Coordinator (Florence)

Prof. Giovanni MODUGNO

  
\_\_\_\_\_

### Composition of the Jury:

Timoteo CARLETTI (Advisor)

Cecilia CLEMENTI

Duccio FANELLI (Advisor)

Mattia FRASCA

Anne-Sophie LIBERT

Michael SCHAUB

# Contents

<b>1</b>	<b>Introduction</b>	<b>5</b>
<b>2</b>	<b>Introduction to Spectral Learning</b>	<b>9</b>
2.1	Spectral analysis of NN . . . . .	9
2.2	Spectral Parametrization of a layer . . . . .	10
2.2.1	Adjacency matrix of a Feed Forward Network . . . . .	10
2.2.2	Spectral Parametrization of $A^{(k)}$ . . . . .	11
2.2.3	Spectral Convolutional Layer . . . . .	13
<b>3</b>	<b>Spectral Learning</b>	<b>17</b>
3.1	Training of a Spectral-MLP . . . . .	17
3.1.1	Eigenvalues training . . . . .	18
3.1.2	Eigenvectors and eigenvalues training . . . . .	18
3.2	Numerical Experiments . . . . .	18
3.2.1	Deep Linear Network . . . . .	19
3.2.2	Linear layer in non-linear network . . . . .	22
<b>4</b>	<b>Filling the gap with direct space</b>	<b>25</b>
4.1	Inter-Layer Transfer Decomposition . . . . .	25
4.1.1	Spectral Layer . . . . .	26
4.1.2	Spectral SVD . . . . .	27
4.2	Spectral QR . . . . .	27
4.2.1	$S$ -QR, Sparse R . . . . .	28
4.3	Spectral training of sparse networks . . . . .	28
4.4	Conclusions . . . . .	29
<b>5</b>	<b>Spectral Pruning</b>	<b>35</b>
5.1	Introduction . . . . .	35
5.2	Conventional Pruning Techniques . . . . .	35
5.3	Methods . . . . .	37
5.4	Results . . . . .	39
5.4.1	Single hidden layer ( $\ell = 3$ ) . . . . .	40
5.5	Conclusions . . . . .	45
<b>6</b>	<b>Spectral Regularization</b>	<b>49</b>
6.1	Teacher-Student framework . . . . .	49
6.2	Experimental Framework . . . . .	51
6.3	Invariant Core . . . . .	52

6.4	Other datasets . . . . .	54
6.5	Linear Core . . . . .	56
6.6	Conclusions . . . . .	57
<b>7</b>	<b>Recurrent Spectral Networks</b>	<b>59</b>
7.1	Classification as a dynamical system . . . . .	59
7.2	The mathematical foundation . . . . .	60
7.3	Testing RSN: a simple dataset in $\mathbb{R}^2$ . . . . .	62
7.4	Applying RSN to the MNIST dataset . . . . .	66
7.4.1	Comparison with Recurrent Network . . . . .	68
7.5	Sequential Learning . . . . .	70
7.5.1	Quasi-Orthogonal $\Phi$ . . . . .	71
7.6	Conclusions . . . . .	75
7.7	From low to high order . . . . .	76
<b>8</b>	<b>Topological Signals</b>	<b>79</b>
8.1	Why simplicial complexes . . . . .	80
8.2	Geometric viewpoint . . . . .	81
8.3	Algebraic Topology, an overview . . . . .	82
8.3.1	Boundary Operator . . . . .	83
8.3.2	Topological Signals . . . . .	85
8.3.3	Coboundary operator . . . . .	86
8.3.4	Hodge-Laplacians and the Dirac operator . . . . .	88
8.3.5	Major Spectral properties . . . . .	89
<b>9</b>	<b>Pattern formation of topological signals</b>	<b>91</b>
9.1	Onset of diffusion driven instability . . . . .	95
9.1.1	Dirac reaction term . . . . .	95
9.1.2	Numerical results on a benchmark network . . . . .	99
9.1.3	Dirac cross-diffusion term . . . . .	101
9.2	Conclusions . . . . .	102
<b>10</b>	<b>Global Synchronization in Simplicial Complexes</b>	<b>105</b>
10.1	Dynamical Framework . . . . .	106
10.2	Construction of eulerian discrete manifolds . . . . .	108
10.2.1	2-simplicial complex . . . . .	108
10.2.2	3-simplicial complex . . . . .	109
10.2.3	3-cell complex: 3D-torus . . . . .	111
10.3	Master Stability Equation for Topological Signals . . . . .	113
10.3.1	Simplicial Stuart-Landau model . . . . .	114
10.4	Conclusions . . . . .	116
<b>11</b>	<b>Conclusions and outcomes</b>	<b>119</b>
<b>A</b>	<b>Construction of a simplex satisfying <math>L_k \mathbf{u} = 0</math></b>	<b>121</b>
A.1	Introduction . . . . .	121
A.2	Graph representation of Boundary operator $\mathbf{B}_k$ . . . . .	121
A.3	Top-down construction of $P$ . . . . .	124

A.3.1 $\mathbf{B}_{k+1}^\top$ . . . . .	125
A.4 Construction of <i>eulerian</i> simplicial complex . . . . .	126
A.4.1 Non-homogeneous vector . . . . .	128
A.5 Higher dimension . . . . .	128
A.6 Graph formulation of the problem . . . . .	129
A.7 Results in deep networks: Sparsity and SVD . . . . .	130
<b>B Square Lattice with periodic boundary conditions</b>	<b>133</b>
<b>C Analysis of the Stuart-Landau model</b>	<b>137</b>



# Chapter 1

## Introduction

In the realm of theoretical physics and machine learning, the intertwined world of network dynamics and computational intelligence opens up new horizons for research. This dissertation aims to contribute to this flourishing field by tackling two diverse yet connected aspects: Neural Network Interpretability and Signal Dynamics in Simplicial Complexes. Both avenues explore the fundamental role played by spectral properties in shaping the behavior and capabilities of network systems. Remarkably, as we are about to show in this manuscript, concepts and ideas inherited from network dynamics and complex systems find fertile ground in the field of mechanistic AI, paving the way for a stronger understanding of the learning process.

Furthermore the notion of emergence and the relevance of the spectral proprieties of coupling operators are, however, still at their infancy in the fast growing field of Topological Signal, justifying out characterization and analysis of synchronization and pattern formation.

### Spectral Parametrization of Deep Neural Networks

We will start with our contribution in the field of Deep Learning whose unprecedented success in a myriad of applications from natural language processing to image recognition cannot be overstated. Despite this, one of the persistent challenges has been the difficulty in interpretability and analysis of the weights within a neural network. Modern neural networks are a huge (even thousands of billions or more) collection of interacting nodes, whose connections are dictated by the characteristic of the task to be learned. Using a very large amount of data as training set, the neural network adjust to approximate the non trivial correlations in the dataset progressively gaining inference capability. Remarkably, at the end of a well defined training procedure, a phenomenon emerges: the network is capable of performing well also on data that were not seen in the training session, determining thus the *generalization power* of the method.

This paradigm, where different entities organizes producing a non linear coherent effect is well established in the realm of complex systems where the emergence of a coherent behaviour is very often addressed via the analysis of the proprieties of the coupling between the elements. Starting from this premise we have developed a novel approach to weight parametrization termed “Spectral Parametrization” that will start in Chapter 2 and continue in Chapter 3. By employing spectral graph theory, we frame

the weights within a neural network layer as the elements of the (weighted) adjacency matrix of a bipartite directed graph. Thus, we leverage the eigenvalues and eigenvectors of this adjacency matrix as novel descriptors for the links within the layer. This parametrization enables multiple innovations:

*A novel formalism 2:* Expressing the connections within a neural network through the spectrum of its adjacency matrix shows that the process commonly referred to as 'feature extraction'—which isolates relevant data characteristics—occurs in each layer as a modulation of distinct eigenvalue components. Furthermore, the relevance of a particular feature can be quantified by the magnitude of its corresponding eigenvalue.

*Reduction of trainable parameters (Chapter 4):* The eigenvalues of the adjacency matrix exploited as adjustable 'knobs' capable of changing different weights simultaneously. Such effect, in conjunction with the Singular Value or QR Decomposition of the eigenvectors components, is enough to train on relevant tasks a deep neural network employing a risible number of free parameters.

*Network Slimming Algorithm (Chapter 5 and 6):* The magnitude of the eigenvalues provides a basis for evaluating the nodes relevance in the network. We propose an algorithm that capitalizes on this relationship to streamline the network, preserving its essential functionality while reducing computational overhead.

*Spectral Regularization Technique (Chapter 6):* Building upon the Spectral Parametrization, we introduce a regularization method that acts upon the eigenvalues. The result is an effective compression of the neural network that notably does not compromise its performance and reveals the emergence of a network size invariant core whose structure is deeply related to the learned task.

*Dynamical Learning (Chapter 7):* Leveraging on our understanding of network dynamics, where the adjacency matrix spectrum drives the asymptotic state of the whole system, we frame a novel learning process where the inputs are dynamically steered towards given attractors. Such formulation of a recurrent network is possible only due to our developed spectral theory.

## Topological Signals Dynamics

The second part of the dissertation turns its attention towards complex systems modelled as Simplicial Complexes. The latter generalize the concept of network by allowing to assign signals, not only to nodes, but also to higher-order structures such as link, faces and so on. We will delve into the dynamics of *Pattern Formation (Chapter 9)* and *Synchronization (Chapter 10)* of topological signals where a plethora of phenomena is produced intertwining algebraic topology and non linear dynamics.

In this case, the focal point is the Dirac operator - a mathematical entity fundamental in several areas in physics allowing signals defined on structures with different dimension to interact. We uncover that the spectrum of this operator provides indispensable insights into the dynamics of pattern formation and synchronization within simplicial complexes. In essence, the spectral properties are fundamental tools to understanding and predicting the complex interaction of topological signals.



Interestingly, the two parts, while developed independently, converge on a common theoretical substrate: the role of spectral properties in understanding and manipulating networks. Whether in the reparametrization of neural network weights or the dynamics of simplicial complexes, spectral theory provides a unified framework for analysis and optimization. Moreover the two parts are continuously merged in the development of a learning dynamical system (the Recurrent Spectral Network), showing how the interconnection between neural networks and dynamical system can be profound and prolific.

In sum, this dissertation aims to shed light on how spectral methods can elucidate, simplify, and optimize the complex realms of neural networks and simplicial complexes. Through this dual exploration, we embark on a journey aimed at both deepening the field's foundational understanding and providing practical tools for future research and applications.

## Published Papers

- *L Giambagli, L Buffoni, L Chicchi, D Fanelli*, How a student becomes a teacher: learning and forgetting through Spectral methods, *Advances in Neural Information Processing Systems (36)*, 2023 (Chapter 6)
- *L Giambagli, D Fanelli, G Risaliti, M Signorini*, Non-parametric analysis of the Hubble Diagram with Neural Networks, *Astronomy & Astrophysics*, 678, A13
- *T Carletti, L Giambagli, G Bianconi*, Global topological synchronization on simplicial and cell complexes, *Physical Review Letters*, 130, 187401 (Chapter 10)
- *L Buffoni, E Civitelli, L Giambagli, L Chicchi, D Fanelli*, Spectral pruning of fully connected layers: ranking the nodes based on the eigenvalues, *Scientific Reports* 12 (1), 1-9 (Chapter 5)
- *L Giambagli, L Calmon, R Muolo, T Carletti, G Bianconi*, Diffusion-driven instability of topological signals coupled by the Dirac operator, *Physical Review E* 106, 064314 (Chapter 9)
- *L Chicchi, D Fanelli, L Giambagli, L Buffoni, T Carletti*, Recurrent Spectral Network (RSN): shaping the basin of attraction of a discrete map to reach automated classification, *Chaos, Solitons and Fractals* 168, 113128 (Chapter 7)
- *L Chicchi, L Giambagli, L Buffoni, T Carletti, M Ciavarella, D Fanelli*, Training of sparse and dense deep neural networks: Fewer parameters, same performance, *Physical Review E* 104 (5) (Chapter 4)
- *L Giambagli, L Buffoni, T Carletti, W Nocentini, D Fanelli*, Machine learning in spectral domain, *Nature Communications* 12 (1), 1-9 (Chapter 3)
- *L Chicchi, L Giambagli, L Buffoni, D Fanelli*, Mobility-based prediction of SARS-CoV-2 spreading, *arXiv:2102.08253*



# Chapter 2

## Introduction to Spectral Learning

### 2.1 Spectral analysis of NN

Machine learning (ML) [1]–[3] refers to a broad field of study, with multifaceted applications of cross-disciplinary breadth. ML is a subset of Artificial Intelligence (AI) which ultimately aims at developing computer algorithms that improve automatically through experience. The core idea is that systems can learn from data, so as to identify distinctive patterns and make consequently decisions, with minimal human intervention. The range of applications of ML methodologies is extremely vast [4]–[7], and still growing at a steady pace due to the pressing need to cope with the efficiently handling of big data [8]. Biomimetic approaches to sub-symbolic AI [9] inspired the design of powerful algorithms. These latter sought to reproduce the unconscious process underlying fast perception, the neurological paths for rapid decision making, as e.g. employed for faces [10] or spoken words [11] recognition.

An early example of a sub-symbolic brain inspired AI was the perceptron [12], the influential ancestor of deep neural networks (NN) [13], [14]. The perceptron is indeed an algorithm for supervised learning of binary classifiers. It is a linear classifier, meaning that its forecasts are based on a linear prediction function which combines a set of weights with the feature vector. Analogous to neurons, the perceptron adds up its input: if the resulting sum is above a given threshold the perceptron fires (returns the output the value 1) otherwise it does not (and the output equals zero). Modern multilayer perceptrons (MLP), account for multiple hidden layers with non-linear activation functions. Nowadays we refer to those kind of architectures as Deep Neural Networks. The learning is achieved via minimizing the classification error. Single or multilayered perceptrons should be trained by labeled examples [13], [15], [16]. Supervised learning requires indeed a large set of positive and negative examples, the training set, labelled with their reference category.

The perceptrons' acquired ability to perform classification is eventually stored in a finite collection of numbers, the weights and thresholds that were learned during the successive epochs of the supervised training. To date, it is not clear how such a huge collection of numbers (hundred-millions of weights in state of the art ML applications) are synergistically interlaced for the deep networks to execute the assigned tasks, with an exceptional degree of robustness and accuracy [17]–[21].

## 2.2 Spectral Parametrization of a layer

To introduce the formalism we developed, let us start by describing the linear action of a fully connected layer. Let  $x_j^{(k-1)}$ , with  $j \in 1 \dots N_{k-1}$  and  $k \in 1 \dots \ell$ , to represent the activity of the  $j$ -th neuron of layer  $k - 1$ . The linear action on the neurons of the layer is expressed as a vector-matrix multiplication:  $z_i^{(k)} = \sum_j w_{ij}^{(k)} x_j^{(k-1)}$  where the matrix components  $w_{ij}^{(k)}$  set the weight of the connections from node  $j$  to  $i$ . In vector notation the latter yields:  $\mathbf{z}^{(k)} = \mathbf{w}^{(k)} \cdot \mathbf{x}^{(k-1)}$ . We then apply a suitably chosen element-wise non-linearity to the vector  $\mathbf{z}^{(k)}$  to obtain  $\mathbf{x}^{(k)}$  and repeat this operation for all the layers  $k \in 1 \dots \ell$  to build the neural network architecture of our choice.

It is a well known interpretation [22] that this vector-matrix multiplication results in the projection of  $\mathbf{x}^{(k-1)}$  along each feature  $\mathbf{w}_i^{(k)}$  (the  $i$ -th row of matrix  $\mathbf{w}^{(k)}$ ), which corresponds to the linear activation of the corresponding neuron (ignoring the bias addition). In this framework, one can speculate that the relevance of each feature can be assessed through a further set of scalar parameters, associated to the destination node, that are modified by the underlying optimization process. These latter parameters, that are denoted  $\lambda_i^{(k)}$  for reasons that will become clear in the following, get initialized to one and leverage the projection along each feature  $\mathbf{w}_i^{(k)}$ . The rescaled linear transfer of information then becomes:

$$z_i^{(k)} = \lambda_i^{(k)} \sum_j w_{ij}^{(k)} x_j^{(k-1)} \quad (2.1)$$

or equivalently, in vector notation:

$$\mathbf{z}^{(k)} = \lambda^{(k)} \odot (\mathbf{w}^{T(k)} \cdot \mathbf{x}^{(k-1)}) \quad (2.2)$$

where  $\odot$  stands for the Hadamard (or element-wise) product. As we shall prove in the following, this seemingly simple parametrization holds unexpected capabilities and profound connections with graph theory.

To this end, we begin by noting layers  $k - 1$  and  $k$  can be viewed as a bipartite graph where connections exist only among nodes of different layers. This graph can be fully described in terms of an *adjacency matrix*  $A^{(k)}$ , which encodes all the relevant information on existing connections. We shall now be more specific regarding this point.

### 2.2.1 Adjacency matrix of a Feed Forward Network

We will give here some more details about the spectral parametrization, with specific regards to the construction of the underlying adjacency matrices. As already mentioned layer  $k - 1$  and  $k$  of a feed forward fully connected neural network, can be viewed as a bipartite directed graph connecting the nodes in layer  $k - 1$  to those in layer  $k$ . This graph can be fully described in terms of an *adjacency matrix*  $A^{(k)}$ , which encodes all the relevant information on the weights of existing connections.  $A^{(k)}$  is a  $(N_{k-1} + N_k) \times (N_{k-1} + N_k)$  matrix. If we label the neurons belonging to layer  $k - 1$  from 1 to  $N_{k-1}$  and those in layer  $k$  from  $N_{k-1} + 1$  to  $N_{k-1} + N_k$ , the elements  $A_{lm}^{(k)}$  exemplify the weighted connection from node  $m$  to node  $l$ , with  $l, m \in 1 \dots N_{k-1} + N_k$ .

Due to the directed nature of the graph, the structure of  $A^{(k)}$  is lower-block diagonal and can be represented as:

$$A^{(k)} = \begin{pmatrix} 0 & & \dots & 0 & 0 \\ \vdots & & & & \\ 0 & & \dots & 0 & 0 \\ w_{11}^{(k)} & \dots & w_{1N_{k-1}}^{(k)} & \vdots & \vdots \\ \vdots & & & \vdots & \\ w_{N_k 1}^{(k)} & \dots & w_{N_k N_{k-1}}^{(k)} & 0 & \dots & 0 \end{pmatrix} \quad (2.3)$$

In this framework, the activity of nodes across the graph can be described through a vector, termed  $v$  for clarity, made of  $N_{k-1} + N_k$  components, where  $v_m$  points to the activity of node  $m$ . This will refer to a neuron in layer  $k - 1$  if  $m \leq N_{k-1}$ , or, conversely, to layer  $k$  if  $N_{k-1} + 1 \leq m \leq N_k + N_{k-1}$ . The structure of such vector for activities localized on layer  $k - 1$ , and before the linear transfer gets applied, is shown in Eq.(2.4).

The feedforward transfer from layer  $k - 1$  to layer  $k$  can be hence described as the action of the square matrix  $A^{(k)}$  on the vector  $v$ . The resulting vector components, which are connected to the transferred activity  $\mathbf{z}^{(k)}$ , will be identically equal to zero for  $m \leq N_{k-1}$ . This can be expressed mathematically as:

$$\begin{pmatrix} 0 \\ \vdots \\ 0 \\ z_1^{(k-1)} \\ \vdots \\ z_{N_k}^{(k-1)} \end{pmatrix} = A^{(k)} v = A^{(k)} \begin{pmatrix} x_1^{(k-1)} \\ \vdots \\ x_{N_{k-1}}^{(k-1)} \\ 0 \\ \vdots \\ 0 \end{pmatrix} \quad (2.4)$$

The above graph-oriented way of describing the activity paves the way to a different parametrization of the linear transfer between layers.

### 2.2.2 Spectral Parametrization of $A^{(k)}$

The spectral parametrization builds on the observation that another class of adjacency matrices can be written, whose action on vector  $v$  is equivalent to that illustrated above. This latter matrix can be assembled starting from two other matrices: a diagonal *eigenvalue* matrix and one lower-block triangular *eigenvector* matrix.

This parametrization is made possible by the fact that the feedforward action is solely encoded in the lower-block diagonal elements, and will always operate on vectors whose structure is depicted in Eq.(2.4). Take then two matrices, denoted respectively  $\Phi^{(k)}$  and  $\Lambda^{(k)}$ , with the following structure:

$$\Phi^{(k)} = \begin{pmatrix} 1 & & \dots & & 0 \\ \vdots & \ddots & & & \\ 0 & & 1 & 0 & 0 \\ \phi_{11}^{(k)} & \dots & \phi_{1N_{k-1}}^{(k)} & 1 & \vdots \\ \vdots & & \vdots & 0 & \ddots & 0 \\ \phi_{N_k 1}^{(k)} & \dots & \phi_{N_k N_{k-1}}^{(k)} & 0 & \dots & 1 \end{pmatrix} \quad (2.5)$$

$$\Lambda^{(k)} = \begin{pmatrix} \lambda_1^{in(k)} & 0 & & \dots & 0 \\ 0 & \ddots & & & \\ & & \lambda_{N_{k-1}}^{in(k)} & 0 & 0 \\ \vdots & & & \lambda_1^{out(k)} & \vdots \\ & & & 0 & \ddots & 0 \\ 0 & \dots & 0 & \dots & \lambda_{N_k}^{out(k)} \end{pmatrix}$$

The block structure of matrix  $\Phi^{(k)}$  is the one responsible for the feedforward arrangement of the resulting adjacency matrix. Specifically, this structure enables activity localized in the  $k-1$  layer (i.e., vectors whose first  $N_{k-1}$  components are non-zero) to be transferred to the  $k$ -th layer, resulting in a vector whose last  $N_k$  components are non-zero. Additionally, it is worth mentioning that the inverse of  $\Phi^{(k)}$  can be computed analytically and it is equal to  $(\Phi^{(k)})^{-1} = 2\mathbb{I}_{N_{k-1}+N_k} - \Phi^{(k)}$ .

Using this property, we can explicitly express the result of the spectral composition  $\tilde{A}^{(k)} = \Phi^{(k)} \Lambda^{(k)} (\Phi^{(k)})^{-1}$  as a lower-triangular matrix, shown in Eq.(2.6).

$$\tilde{A}^{(k)} = \begin{pmatrix} \lambda_1^{in(k)} & & \dots & & 0 \\ \vdots & \ddots & & & \\ 0 & & \lambda_{N_{k-1}}^{in(k)} & 0 & 0 \\ \tilde{w}_{11}^{(k)} & \dots & \tilde{w}_{1N_{k-1}}^{(k)} & \lambda_1^{out(k)} & \vdots \\ \vdots & & \vdots & 0 & \ddots & 0 \\ \tilde{w}_{N_k 1}^{(k)} & \dots & \tilde{w}_{N_k N_{k-1}}^{(k)} & 0 & \dots & \lambda_{N_k}^{out(k)} \end{pmatrix} \quad (2.6)$$

We can conclude that the action of this matrix on vector  $v$  is analogous to that exemplified in Eq.(2.4) with a matrix that has a structure like (2.3), provided the weights  $\tilde{w}_{ij}^{(k)}$  are parametrized according to a specific recipe that can be easily derived (see [23] and [24]) and that we shall recall hereafter. Indeed we can write the elements  $\tilde{w}_{ij}$  in terms of the off-diagonal elements of the eigenvector matrix  $\Phi^{(k)}$  and the diagonal eigenvalue matrix  $\Lambda^{(k)}$ :

$$\tilde{w}_{ij}^{(k)} = (\lambda_j^{(k)in} - \lambda_i^{(k)out}) \phi_{ij}^{(k)} \quad (2.7)$$

The matrix  $\tilde{A}$  acts as an equivalent feedforward network, transferring  $(k-1)$ -layer localized activity to the following layer through a simple vector-matrix multiplication. By substituting  $\tilde{A}$  for  $A$  in Eq.(2.4), we can express the linear activity of neurons in

the  $k$ -th layer,  $z_i^{(k)}$ , in terms of the spectral parameters introduced earlier and the  $k-1$  layer activity  $x_j^{(k-1)}$ , namely:

$$z_i^{(k)} = \sum_{j=1}^{N_{k-1}} (\lambda_j^{(k)in} - \lambda_i^{(k)out}) \phi_{ij}^{(k)} x_j^{(k-1)} \quad \forall i \in 1 \dots N_k \quad (2.8)$$

or, in vector notation

$$\mathbf{z}^{(k)} = \boldsymbol{\phi}^{(k)} \cdot (\boldsymbol{\lambda}^{(k)in} \odot \mathbf{x}^{(k-1)}) - \boldsymbol{\lambda}^{(k)out} \odot (\boldsymbol{\phi}^{(k)} \cdot \mathbf{x}^{(k-1)}) \quad \forall i \in 1 \dots N_k \quad (2.9)$$

Interestingly the eigenvalues  $\lambda^{(k)in}$  modulate the density at the tail of the directed link, while  $\lambda^{(k)out}$  appears to regulate the local node's excitability relative to the network activity in the head of the same link. This is the artificial analogue of the *homeostatic plasticity*, the strategy implemented by living neurons to maintain the synaptic basis for learning, respiration, and locomotion [25].

This simple decomposition allows for an intuitive understanding of the role of eigenvalues and eigenvectors, keeping the computational cost almost invariant, since the inverse of the eigenvector matrix  $\Phi^{(k)}$  is given analytically<sup>1</sup>. It is now immediate to see that, by setting  $\lambda_j^{(k)in} = 0$  for all  $j \in 1 \dots N_{k-1}$ , we recover Eq.(2.1). The features can now be seen as components of the eigenvectors of the adjacency matrix describing the sub-network made by layers  $k$  and  $k-1$ . Their relative weight, instead, can be interpreted as the magnitude of the eigenvalue  $\lambda_j^{(k)out}$ . Since the feedforward fully-connected nature of the transfer from layer  $k-1$  to  $k$  is encoded in the structure of  $\Phi^{(k)}$ , the entries of this latter matrix enable one to resolve the relationship between different eigenvector structures and dive into the obtained network topology.

### 2.2.3 Spectral Convolutional Layer

Having encoded the topological structure of the computing network in the underlying eigenvectors' matrix for a specific case study, it is now straightforward to generalize the reasoning to other relevant settings. In particular we shall consider the case of the so called Convolutional Neural Network (CNN). This latter setting can be conceptualized as a genuine eigenvalue training for a peculiar structure of  $\Phi^{(k)}$ . Here the eigenvalues gauge the relative importance of the employed convolutional filters.

In order to show that we shall first describe the CNN action as a classical vector matrix multiplication. Working with CNNs one needs to specify (i) the filter dimensions (ii) the  $x, y$  striding ( $s_{x,y}$ ), i.e. how much each filter is shifted in each direction during the convolution, and (iii) the  $x, y$  padding ( $p_{x,y}$ ) i.e. how many zeros are added at each side of the input so that the convolution starts and ends with a given "offset". See Figure 2.1 for a graphical representation of those definitions, where a  $3 \times 3$  filter is considered. We are aware of the fact that we are showing a simplified case of the general convolutions, indeed in the latter, different channels are present as well. However the underlying maths is basically the same and, for a matter of clarity, only this simplified case is presented. The same operation can be framed in terms of a

<sup>1</sup>Moreover, when implemented in Tensorflow, the additional computational cost with respect to a conventional vector matrix multiplication results in just a Hadamard product with parameters  $\lambda^{in}$  and  $\lambda^{out}$

$$\begin{array}{c}
 \begin{array}{|c|} \hline M \\ \hline \end{array} \\
 \begin{array}{|c|} \hline N \\ \hline \end{array} \\
 \hline
 \end{array}
 x^0
 \quad * \quad
 \begin{array}{c}
 m \\
 \begin{pmatrix} w_1 & w_2 & w_3 \\ w_4 & w_5 & w_6 \\ w_7 & w_8 & w_9 \end{pmatrix} \\
 n
 \end{array}
 =
 \begin{array}{c}
 \begin{array}{|c|} \hline M - m + p_x + s_x \\ \hline \end{array} \\
 \begin{array}{|c|} \hline N - n + p_y + u \\ \hline \end{array}
 \end{array}$$

Figure 2.1: Setting the notation for dealing with CNN. In yellow the input rectangular matrix and in purple the filter structure.

matrix-vector multiplication as soon as every parameter is fixed (similar constructions can be seen in [26]). To this end fix the padding to 0 and the striding to 1, without loss of generality. In fact, the non trivial padding and striding can be reframed in a change on the input vector  $x^{(k)} \mapsto x^{(k)}(p_{x,y}, s_{x,y})$  and the action of the convolution considered with zero padding and unitary striding. The convolution of the input  $x^0$  with a filter  $w$  can be written as the action of a matrix  $\mathcal{M}(w)$  on the input written with an equivalent vectorial notation.  $\mathcal{M}(w)$  is a matrix with a peculiar structure called *Toeplitz matrix*. A paradigmatic example is shown for the aforementioned choice of parameters in Figure 2.2 We point out that this formulation is nothing but the fully

$$\begin{array}{c}
 \overleftarrow{M \times N} \\
 \overleftarrow{M} \\
 \begin{pmatrix} w_1 & w_2 & w_3 & 0 & 0 & w_4 & w_5 & w_6 & 0 \dots 0 & \dots w_9 \\ 0 & w_1 & w_2 & w_3 & 0 & 0 & & & & \\ 0 & 0 & w_1 & w_2 & w_3 & 0 & 0 & & & \end{pmatrix} \\
 \mathcal{M}(w) \\
 \begin{pmatrix} x_{00} \\ x_{01} \\ \dots \\ \end{pmatrix} \\
 x^0 \\
 \overrightarrow{M \times N}
 \end{array}$$

Figure 2.2: The figure shows how a convolution operation of Figure 2.1 can be translated into a matrix-vector product via an opportune remapping of the filter into a Toeplitz matrix and a rearrangement of the rectangular image into a column vector. This is equivalent to represent the single Convolutional layer (with a single filter) as a Feed-forward fully connected one, the weight of the latter given by  $\mathcal{M}(w)$ .

connected representation of a Convolutional layer. In the following, we will show how, the above representation opens up the perspective to a spectral interpretation of the Convolutional layer.

### Convolution parameters are eigenvalues

The first thing to point out is that convolutions are already formulated in what we could call *reciprocal space*. Let us consider the transformation in Figure 2.2. First of



all, by a proper duplication of the input components the structure of matrix  $\mathcal{M}(w)$  can be simplified into one with only a single non-zero value per column.

By taking the structure of the matrix  $\phi$  as the binarization version of such simplified Toeplitz matrix, the  $\lambda^{in}$  of the notation introduced in (2.7) play the exact same role of the filters degree of freedom in the CNN layer.

By inspecting equation (2.7) the effect of  $\lambda^{(in)}$  can be interpreted as a column-wise product whereas the  $\lambda^{(out)}$  as a row-wise one. By setting  $\lambda^{(out)} = 0$  each  $\lambda^{(in)}$  can be adjusted to the same value of the convolutional weight that is solely present in a given column. This operation is graphically presented on the bottom part of Figure 2.3.

$$\underbrace{\begin{pmatrix} w_1 & w_2 & w_3 & 0 & & 0 & & & & & w_4 & w_5 & w_6 & 0 \dots 0 & \dots & w_9 \\ 0 & 0 & 0 & w_1 & w_2 & w_3 & 0 & & & & & & & & & \\ 0 & 0 & 0 & 0 & 0 & 0 & w_1 & & & & & & & & & \end{pmatrix}}_{\mathcal{M}(w)} \begin{pmatrix} x_{00} \\ x_{01} \\ x_{02} \\ x_{01} \\ x_{02} \\ x_{03} \\ \dots \end{pmatrix}$$

$$\begin{pmatrix} \lambda_1 & \lambda_2 & \lambda_3 & \lambda_1 & \lambda_2 & \lambda_3 & \dots & \lambda_4 & \lambda_5 & \lambda_6 & \dots & \dots & \lambda_9 \end{pmatrix} \odot$$

$$\begin{pmatrix} 1 & 1 & 1 & 0 & & 0 & & & & & 1 & 1 & 1 & 0 \dots 0 & \dots & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 & & & & & & & & & \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & & & & & & & & & \end{pmatrix}$$

$$\mathcal{M}(\phi, \lambda)$$

Figure 2.3: In the upper part a feedforward layer equivalent to a CNN one is shown. Input components have been duplicated in order to have a single non zero component per column. In the bottom part the same weight matrix is rephrased in terms of the spectral decomposition of (2.7). The equivalence can be achieved setting  $\lambda_{1,2,3\dots} = w_{1,2,3\dots}$ ,



# Chapter 3

## Spectral Learning

### 3.1 Training of a Spectral-MLP

We will now discuss how this novel parametrization of the linear transfer of information can impact the learning dynamics. More specifically, in this chapter, we will show how two different learning strategies arise spontaneously. The first is the one involving only the eigenvalues  $\lambda$  (all or a subset of them); the second is the one involving also the eigenvectors components.

In a conventional supervised learning scheme the Hypothesis space where functions will be searched is very often selected via suitable parametrization of certain family of functions. The parameters describing the chosen space are adjusted via an optimization process that aims at minimizing an appositely defined Loss function. In the case of a feedforward fully connected neural network (FFNN) spectral learning accounts for the minimization of the following function:

$$\mathcal{L} = \mathbb{E}_{x,y \sim p_{emp}(x,y)} L(f(\lambda_1, \phi_1, \dots, \lambda_\ell, \phi_\ell; x), y) \quad (3.1)$$

where  $p_{emp} = \sum_{i=1}^{|D|} \delta(x - x^{(i)})\delta(y - y^{(i)})$  is the empirical distribution and  $L$  a suitable loss function like the Categorical Cross Entropy Loss or the Square Loss (the two major losses we will be dealing with in this thesis). The scope of this function is to quantitatively express how much is the model predicting well the labelled data. If the problem is formalized correctly then  $\mathcal{L}$  is a good proxy of  $\mathbb{E}_{x,y \sim p(x,y)} L(f(\lambda_1, \phi_1, \dots, \lambda_\ell, \phi_\ell; x), y)$ , where this time  $p(x, y)$  is the true (almost always unknown) probability distribution of the data.

In the spectral learning scheme by extending the definition of the conventional MLP, one can write

$$f(\lambda_1, \phi_1, \dots, \lambda_\ell, \phi_\ell; x) = \sigma_\ell(\tilde{w}_\ell(\lambda_\ell, \phi_\ell)\sigma_{\ell-1}(\dots\sigma_1(\tilde{w}_1(\lambda_1, \phi_1)x))) \quad (3.2)$$

where with  $\sigma_k$  we intend the non linear function of layer  $k$ . The two alternative learning strategies to which we alluded above can be formalized respectively as minimizing the Loss function with respect to the eigenvalues  $\lambda_{1\dots\ell}$  or with respect to the full set of parameters  $\lambda_{1\dots\ell}, \phi_{1\dots\ell}$ . In the following we will comment on the two aforementioned strategies, highlighting their respective peculiarities.

### 3.1.1 Eigenvalues training

When the gradient is computed only with respect to the eigenvalues the number of adjustable parameters drops significantly. More specifically, for a  $\ell$ -deep MLP of respective sizes  $N_i, i \in 1 \dots \ell$ , we have a total of  $N_\lambda = N_1 + \sum_{i=2}^{\ell-1} 2N_i + N_\ell$  eigenvalues and thus target parameters for the optimization algorithm. Indeed, by excluding the input and output layer, we have that every neuron on layer  $j \in 2 \dots \ell - 1$  can be made in correspondence with a pair  $\lambda_j^{out}$  and  $\lambda_{j+1}^{in}$ .

In principle both eigenvalues can be freely trained. This general setting has been however poorly explored so far. For this reason we will restrict in the following to a reduced framework by positing  $\lambda^{in} = 0$  (or alternatively by setting these quantities to quenched, randomly assigned entries). With this choice one consequently gains a very clear interpretation of the role of the eigenvalues, as veritable markers of feature relevance (as already said in the precedent chapter).

We would like to emphasize that learning based solely on eigenvalues can be equated to a random features model during each instance of information transfer. In this context, the learning process effectively acts as a progressive non linear filter for various random features. To be more specific, in the  $k$ -th layer of the network, the components of the eigenvectors, denoted by  $\phi_{:,j}^{(k)}$ , serve as the  $j$ -th feature. Its strength is then modulated by the corresponding eigenvalue magnitude  $\lambda_j^{out}$ .

Given this understanding, it is not unexpected to observe that learning is possible in this specific setting where each random feature get either excited or inhibited. Moreover, increasing the number of nodes, which correspond to random features, enhances the model's expressivity and consequently improves the fit to the data.

### 3.1.2 Eigenvectors and eigenvalues training

When also the eigenvectors components are added to the optimization scheme, clearly, the spectral parametrization is as efficient as the conventional one and we should expect a similar generalization propriety. However, as we will analyze in following Chapters, the effect is still far from trivial in terms of the ensuing information localization as obtained post-training.

## 3.2 Numerical Experiments

To build and train the aforementioned models we used TensorFlow [27] and created a custom spectral layer matrix that could be integrated in virtually every TensorFlow or Keras model. That allowed us to leverage on the automatic differentiation capabilities and the built-in optimizers of TensorFlow. Recall that we aim at training just a portion of the diagonal of  $\Lambda_k$  and a block of  $\Phi_k$ . We then trained all our models with the AdaMax optimizer [28] by using a learning rate of 0.03 for the linear case and 0.01 for the non-linear one. The training proceeded for about 20 epochs and during each epoch the network was fed with batches of images of size 300. These hyperparameters have been chosen so as to improve on GPU efficiency, accuracy and stability. However, we did not perform a systematic study to look for the optimal setting. All our models have been trained on a virtual machine hosted by Google Colaboratory. Standard neural networks have been trained on the same machine using identical software and

hyperparameters, for a fair comparison. Further details about the implementation, as well as a notebook to reproduce our results, can be found in the public repository of this project [29].

We shall start by reporting on the performance of the linear scheme using the renowned MNIST dataset [30]: a collection of 70,000 grayscale images of handwritten digits ranging from 0 to 9, each with dimensions of 28x28 pixels. The simplest model setting we tested is that of a perceptron made of two layers: the input layer with  $N_1 = 28 \times 28 = 784$  nodes and the output one made of  $N_2 = 10$  elements. The perceptron can be trained in the spectral domain by e.g. tuning the  $N = N_1 + N_2 = 794$  eigenvalues of  $\mathbf{A}_1$ , the matrix that links the input ( $x^{(0)}$ ) and output ( $x^{(1)}$ ) vectors. The learning restricted to the eigenvalues returns a perceptron which performs the sought classification task with an accuracy (the fraction of correctly recognized images in the test-set) of  $(82 \pm 2)\%$  (averaging over 5 independent runs). This figure is to be confronted with the accuracy of a perceptron trained with standard techniques in direct space. For a fair comparison, the number of adjustable weights should be limited to  $N$ . To this aim, we randomly select at the beginning of every optimization process a subset of weights to be trained and carry out the optimization on these latter; the others are set to the Glorot Uniform initialization [31]. The process is repeated a few (5 in this case) times and, for each realization, the associated accuracy computed. Combining the results yields an average performance of  $(79 \pm 3)\%$ , i.e. a slightly smaller score (although compatible within error precision) than that achieved when the learning takes place in the spectral domain. When the training extends to all the  $N_1 \times N_2$  weights (plus  $N_1 + N_2$  bias), conventional learning yields a final accuracy of  $(92.7 \pm 0.1)\%$ . This is practically identical to the score obtained in the spectral domain, specifically  $(92.5 \pm 0.2)\%$ , when the sub-diagonal entries of the eigenvectors matrix (the components  $\phi_{ij}^{(1)}$ ) are also optimized (for a total of  $N_1 + N_2 + N_1 \times N_2$  free parameters). The remarkable observation is however that the distribution of the weights as obtained when the learning is restricted on the eigenvalues (i.e using about the 10 % of the parameters employed for a full training in direct space) matches quite closely that retrieved by means of conventional learning schemes, see Fig. 3.1 . This is not the case when the learning in direct space acts on a subset of  $N$ , randomly selected, weights (data not shown). Based on the above, it can be therefore surmised that optimizing the eigenvalues constitutes a rather effective pre-training strategy, which engages a modest computational load.

### 3.2.1 Deep Linear Network

To further elaborate on the potentiality of the proposed technique, we modify the simple two-layers perceptron, with the inclusion of supplementary computing layers, creating a linear MLP or deep linear network. Although it seems like a trivial exercise (as basically it is a very intricate way of regressing a plane) those networks behaves in a very interesting manner and have been widely studied and deeply understood in the last decade [32]. The newly added layers plays an active role during the learning stage, but can be retracted in inference so as to return a two-layers perceptron. The weights of this latter bear however an imprint of the training carried out for the linear network in the expanded configuration. Two alternative strategies will be in particular contemplated. On the one side, we will consider a sole additional layer,

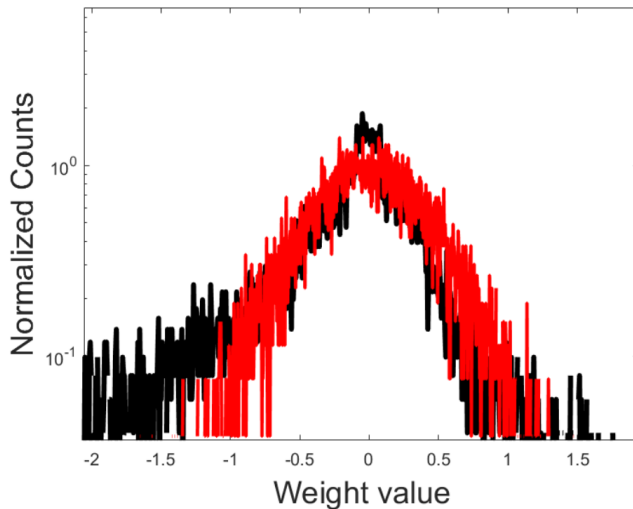


Figure 3.1: Distribution of the weights of a perceptron. The red line follows the spectral training limited the  $N_1 + N_2$  eigenvalues. The black line follows the training in direct space where  $N_1 \times N_2$  parameters are adjusted in the space of the nodes. The distributions are very similar, but the spectral learning employs about 10% of the parameters used in direct space. The distributions obtained when forcing the training in direct space to operate on a subset of  $N_1 + N_2$  weights are very different from the one displayed (for every choice of the randomly selected family of weights to be trained).

endowed with  $N_2$  nodes, interposed between the input and output layers made of, respectively,  $N_1 = 784$  and  $N_\ell \equiv N_3 = 10$  nodes. We will refer to this as to the *wide linear* configuration. The performance of the method can be tested by letting  $N_2$  to progressively grow. On the other side, the *deep linear* configuration is obtained when interposing a sequence of successive (linear) stacks between the input ( $N_1 = 784$ ) and the output ( $N_\ell = 10$ ) layers.

In Fig. 3.2, we report on the performance of the wide learning scheme as a function of  $N_2 + N_3$ . As we shall clarify, this latter stands for the number of trained parameters for (i) the spectral learning acted on a subset of the tunable eigenvalues and for (ii) the conventional learning in direct space restricted to operate on a limited portion of the weights. The red line in the main panel of Fig. 3.2 refers to the simplified scheme where a subset of the eigenvalues are solely tuned (while leaving the eigenvectors fixed at the random realization set by the initial condition). We have in particular chosen to train the second bunch of  $N_2$  eigenvalues of the transfer matrix  $\mathbf{A}_1$  and the  $N_3 = 10$  non trivial eigenvalues of matrix  $\mathbf{A}_2$ , in line with the prescriptions reported in the preceding Section. The blue line reports on the accuracy of the neural network trained in direct space: the target of the optimization is a subset of cardinality  $N_2 + N_3$  of the  $N_1 N_2 + N_2 N_3$  weights which could be in principle adjusted in the space of the nodes. The performance of the spectral method proves clearly superior, as it can be readily appreciated by visual inspection of Fig. 3.2. The black line displays the accuracy of the linear neural network when the optimization acts on the full set of  $N_1 N_2 + N_2 N_3$  trainable parameters. No improvement is detectable when increasing the size of the intermediate layer: the displayed accuracy is substantially identical to that obtained for the basic perceptron trained with  $N_1 N_2 = 7840$  parameters. The spectral learning

allows to reach comparable performance already at  $N_2 = 1000$  (13% of the parameters used for the standard two layers perceptron with  $N_1 \times N_2$  parameters, as discussed above).

Being, de facto, a composition of linear operators, the deep linear network can be collapsed into a zero hidden layer perceptron by simply computing all the matrix product of the linear operators that, via iterative application, transfer the activity from the input to the output layer. Of course the same concept can be rephrased in terms of multiplication of each layers' adjacency matrix, if needed, thanks to the correspondence between connections and adjacency matrix components set in the precedent chapter. In the inset of Fig. 3.2, the distribution of the off diagonal entries in matrix  $\mathcal{A}_c$ , the equivalent perceptron, is depicted in red for the setting highlighted in the zoom. The black line refers to the two-layers equivalent of the neural network trained in direct space, employing the full set of trainable parameters (black dot enclosed in the top-left dashed rectangle drawn in the main panel of Fig. 3.2). The two distributions look remarkably close, despite the considerable reduction in terms of training parameters, as implemented in the spectral domain (for the case highlighted, 0.13% of the parameters employed under the standard training). Similarly to the above, the distribution obtained when forcing the training in direct space to act on a subset of  $N_1 + N_2$  weights are just a modest modulation of the initially assigned profile, owing to the *local* nature of the learning in the space of the nodes.

In Fig. 3.3, we report the results of the tests performed when operating under the deep linear configuration. Symbols are analogous to those employed in Fig. 3.2. In all inspected cases, the entry layer is made of  $N_1 = 784$  elements and the output one has  $N_\ell = 10$  nodes. The first five points, from left to right, refer to a three layers (linear) neural network. Hence,  $\ell = 3$  and the size of the intermediate layer is progressively increased,  $N_2 = 20, 80, 100, 500, 800$ . The total number of trained eigenvalues is  $N_2 + N_3$ , and gets therefore larger as the size of the intermediate layer grows. The successive four points of the collections are obtained by setting  $\ell = 4$ . Here,  $N_2 = 800$  while  $N_3$  is varied ( $= 100, 200, 400, 600$ ). The training uses  $N_2 + N_3 + N_4$  parameters. Finally the last point in each displayed curve is obtained by working with a five layers deep neural network,  $\ell = 5$ . In particular  $N_2 = 800$ ,  $N_3 = 600$  and  $N_4 = 500$ , for a total of  $N_2 + N_3 + N_4 + N_5$  tunable parameters. Also in this case, the spectral algorithm performs better than conventional learning schemes constrained to operate with an identical number of free parameters. Similarly, the distribution of the weights of an equivalent perceptron trained in reciprocal space matches that obtained when operating in the space of the nodes and resting on a considerably larger number of training parameters.

To sum up, eigenvalues are parameters of key importance for neural networks training, way more strategic than any other set of equivalent cardinality in the space of the nodes. As such, they allow for a global approach to the learning, with significant reflexes of fundamental and applied interest. In all cases here considered, the learning can extend to the eigenvectors: an optimized indentation of the eigendirections contribute to enhance the overall performance of the trained device.

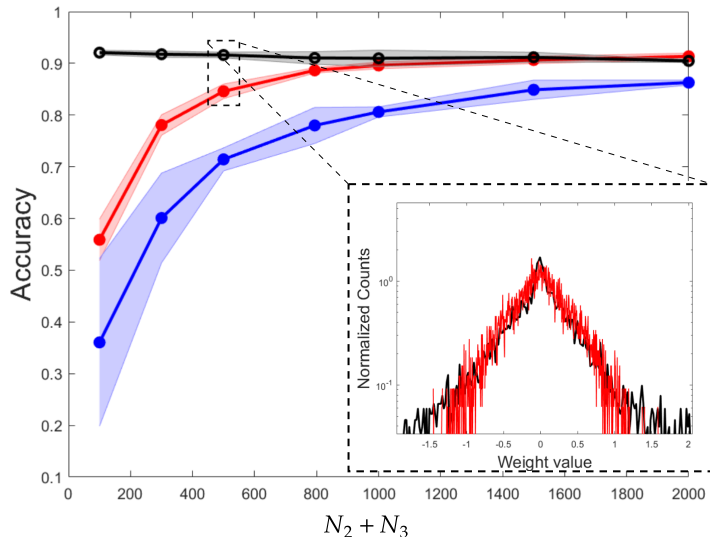


Figure 3.2: A three layers neural network is considered. The accuracy of the neural network is plotted as a function of the number of parameters that we chose to train with the spectral algorithm,  $N_2 + N_3$ . The red line reports on the performance of the spectral training. The blue line refers to the neural network trained in direct space: the optimization runs on  $N_2 + N_3$  parameters, a subset of the total number of adjustable weights  $N_1N_2 + N_2N_3$ . The black line stands for the accuracy of the linear neural network when training the full set of  $N_1N_2 + N_2N_3$  parameters. Notice that the reported accuracy is comparable to that obtained for a standard two layers perceptron. Inset: the distribution of the entries of the equivalent perceptrons are plotted. The red curve refer to the spectral learning restricted to operate on the eigenvalues; the black profile to the neural network trained in direct space, employing the full set of adjustable parameters. In both cases, the weights refer to the two layers configuration obtained by retracting the intermediate linear layer employed during the learning stage.

### 3.2.2 Linear layer in non-linear network

We now turn to considering a non-linear architecture. More specifically, we will assume a four layers network with, respectively,  $N_1 = 784, N_2, N_3 = 120, N_4 = 10$ . The non-linear ReLU filter acts on the third layer of the collection, while the second is a linear processing unit. As in the spirit of the wide network configuration evoked above, we set at testing the performance of the neural network for increasing  $N_2$ . For every choice of  $N_2$ , the linear layer can be retracted yielding a three-layered effective non-linear configurations. We recall however that training the network in the enlarged space where the linear unit is present leaves a non trivial imprint in the weights that set the strength of the links in direct space. In Fig 3.4, we plot the computed accuracy as a function of  $N_2$ , the size of the linear layer. In analogy with the above analysis, the red curve refers to the training restricted to  $N_2 + N_3 + N_4$  eigenvalues; the blue profile is obtained when the deep neural network is trained in direct space by adjusting an identical number of inter-nodes weights. As for the case of a fully linear architecture, by adjusting the eigenvalues yields better classification performances. The black line shows the accuracy of the neural network when the full set of  $N_1N_2 + N_2N_3 + N_3N_4$  is



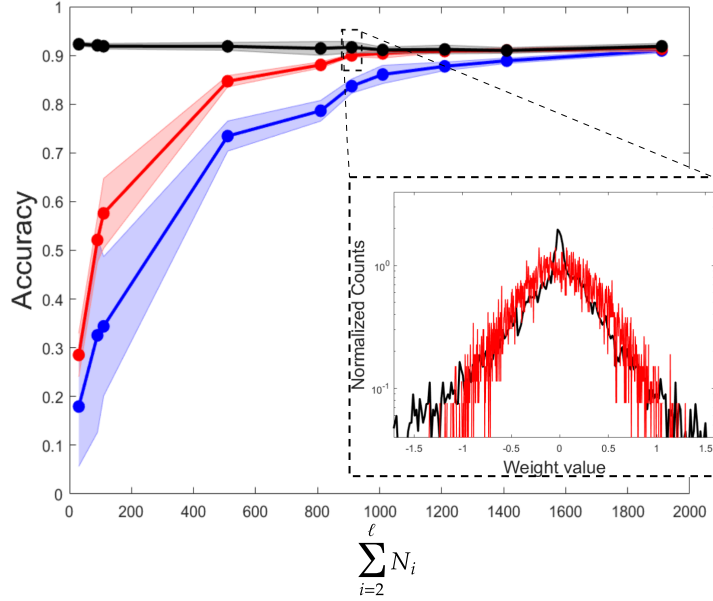


Figure 3.3: The performance of the spectral algorithm are tested for a multi-layered linear configuration. Symbols are chosen in analogy to Fig. 3.2. In all cases, the input layer is made of  $N_1 = 784$  elements and the output layer has  $N_\ell = 10$  nodes. The first five points, from left to right in each of the curves depicted in the main panel, refer to a three layers (linear) neural network. The size of the intermediate layer is progressively increased, as  $N_2 = 20, 80, 100, 500, 800$ . The total number of trained eigenvalues is  $N_2 + N_3$ . The subsequent four points are obtained by considering a four layers architecture. In particular,  $N_2 = 800$  while  $N_3$  takes values in the interval  $(100, 200, 400, 600)$ . The training acts on  $N_2 + N_3 + N_4$  eigenvalues. The final point in each curve is obtained with a four layers deep neural network. Here,  $N_2 = 800$ ,  $N_3 = 600$  and  $N_4 = 500$ , for a total of  $N_2 + N_3 + N_4 + N_5$  tunable parameters in the spectral setting. Inset: the distribution of the entries of the equivalent perceptrons are displayed, with the same color code adopted in Fig. 3.2. Also in this case, the weights refer to the two layers configuration obtained by retracting the intermediate linear layers employed in the learning stage.

optimized in direct space. The green line refer instead to the spectral learning when the eigenvalues and eigenvectors are trained simultaneously. The accuracies estimated for these two latter settings agree within statistical error, even if the spectral scheme seems more robust to overfitting (the black circles declines slightly when increasing  $N_2$ , while the collection of green points appears rather stable).

Summing up, in this Chapter we have presented the effect of a new training procedure, which is bound to the spectral, hence reciprocal, domain. The eigenvalues and eigenvectors of the adjacency matrices that connects consecutive layers via directed feed-forward links are trained, instead of adjusting the weights that bridge each pair of nodes of the collection, as it is customarily done in the framework of conventional Deep Learning approaches.

The first conclusion of our analysis is that optimizing the eigenvalues, when freezing the eigenvectors, yields performances which are superior to those attained with conventional methods *restricted* to a operate with an identical number of free parameters

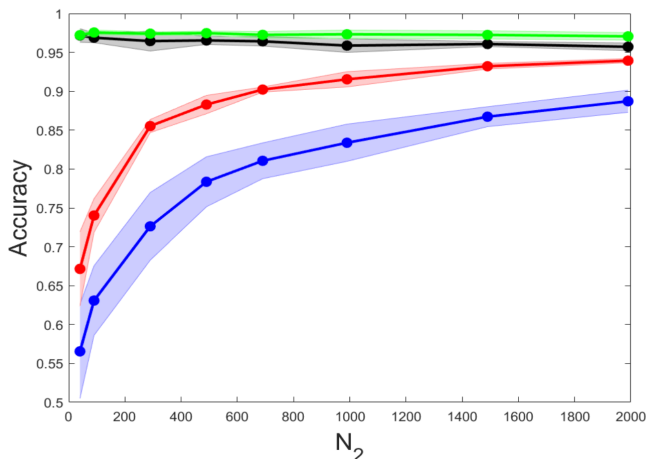


Figure 3.4: The accuracy of the non-linear deep neural network is tested. We assume a four layers network with, respectively,  $N_1 = 784, N_2, N_3 = 120, N_4 = 10$ ;  $N_2$  is changed so as to enlarge the set of parameters to be trained. The red line refers to the spectral training, with  $N_2 + N_3 + N_4$  adjusted eigenvalues. The blue line stands for a neural network trained in direct space, the target of the optimization being a subset made of  $N_2 + N_3 + N_4$  weights, randomly selected from the available pool of  $N_1N_2 + N_2N_3 + N_3N_4$  tunable parameters. The black line reports the accuracy of the linear neural network when training the full set of  $N_1N_2 + N_2N_3 + N_3N_4$  weights. The green line refer to the spectral learning when eigenvalues and eigenvectors are simultaneously trained.

in the direct space. It is therefore surmised that eigenvalues are key target parameters for neural networks training, in that they allow for a *global* handling of the learning. This is at variance with conventional approaches which seek at modulating the weights of the links among mutually connected nodes. Secondly, the spectral learning restricted to the eigenvalues yields a distribution of the weights which resembles quite closely that obtained with conventional algorithms bound to operate in direct space. Indeed, as we will show in Chapters 5 the proposed method could be used in combination with existing algorithms for an effective (and computationally advantageous) pre-training of deep neural networks. We have also shown that linear processing units inserted in between consecutive, non-linearly activated layers produce an enlargement of the learning parameters space, with beneficial effects in terms of performance of the trained device.

In the next Chapter we will expand further the concept of spectral decomposition and merge it with the Singular Value Decomposition, showing how the accuracy gap between direct and spectral learning can be filled by employing a minimal number of free parameters. To substantiate this claim, the techniques will be applied to conventional multi-layer perceptrons featuring non-linear activations. Furthermore, the global attribute of this novel learning paradigm, which allows for the simultaneous modification of multiple network connections through the adjustment of a single eigenvalue, offers considerable advantages for the establishment of a learning procedure that culminates in a network that is both sparse and computationally efficient.

# Chapter 4

## Filling the gap with direct space

Reformulating the learning in reciprocal space enables one to shape key collective modes, the eigenvectors, which are implicated in the process of progressive embedding, from the input layer to the detection point. Even more interestingly, one can assume the eigenmodes of the inter-layer transfer operator to align along suitable random directions and identify the associated eigenvalues as target for the learning scheme. This results in a dramatic compression of the training parameters space, yielding accuracies which are superior to those attained with conventional methods restricted to operate with an identical number of tunable parameters. Nonetheless, neural networks trained in the space of nodes with no restrictions on the set of adjusted weights, achieve better classification scores, as compared to their spectral homologues with quenched eigendirections. In the former case, the number of free parameters grows as the product of the sizes of adjacent layer pairs, thus quadratically in terms of hosted neurons. In the latter, the number of free parameters increases linearly with the size of the layers (hence with the number of neurons), when the eigenvalues are solely allowed to change. Also training the eigenvectors amounts to dealing with a set of free parameters equivalent to that employed when the learning is carried out in direct space: in this case, the two methods yield performances which are therefore comparable.

### 4.1 Inter-Layer Transfer Decomposition

Starting from this setting, we begin by discussing a straightforward generalisation of the spectral learning scheme presented in Chapter 2 and in [23], which proves however effective in securing a significant improvement on the recorded classification scores, while still optimising a number of parameters which scales linearly with the size of the network. The proposed generalisation paves the way to a bio-mimetic interpretation of the spectral training scheme. The eigenvalues can be tuned so as to magnify/damp the contribution associated to the input nodes. At the same time, they modulate the excitability of the receiving nodes, as a function of the local field. Further, in this Chapter, we will show that the residual gap between conventional and spectral trainings methods can be eventually filled by resorting to apt decompositions of the non trivial block of the eigenvectors matrix, which place the emphasis on a limited set of collective variables. Finally, we will prove that working in reciprocal space turns out to be by far more performant, when aiming at training sparse neural networks. Because of the improvement in terms of computational load, and due to

the advantage of operating with collective target variables as we will make clear in the following, it is surmised that modified spectral learning of the type here discussed should be considered as a viable standard for deep neural networks training in artificial intelligence applications.

To test the effectiveness of the proposed method we will consider classification tasks operated on three distinct database of images. The first is the celebrated MNIST database of handwritten digits [30], the second is Fashion-MNIST (F-MNIST) [33], a dataset of Zalando’s article images, the third is CIFAR-10 [34] a collection of images from different classes (airplanes, cars, birds, cats, deer, dogs, frogs, horses, ships, and trucks). In all considered cases, use can be made of a deep neural network to perform the sought classification, namely to automatically assign the image supplied as an input to the class it belongs to. The neural network is again, customarily trained via standard backpropagation algorithms to tune the weights that connect consecutive stacks of the multi-layered architecture. The assigned weights, target of the optimisation procedure, bear the information needed to allocate the examined images to their reference category. The idea of collectively parametrize different weights has been explored in [35] where the authors show that in modern architectures a subset of optimized weights is enough to infer the rest of the trainable parameters. Our study extends this idea using a network based parametrization, the spectral decomposition, chained with the SVD decomposition, de facto linearly correlating different links in the network instead of using a custom kernel. The idea of employing SVD in order to obtain low rank matrices has been explored in [36] where, singular values are regularized and the gradient is performed with respect to the orthogonal matrices.

### 4.1.1 Spectral Layer

Consider a deep feedforward network made of  $\ell$  distinct layers and label each layer with the progressive index  $i$  ( $= 1, \dots, \ell$ ). Following the convention already introduced, denote by  $N_i$  the number of computing units, the neurons, that belong to layer  $i$ . The total number of parameters that one seeks to optimise in a dense neural network setting (all neurons of any given layer with  $i < \ell - 1$  are linked to every neurons of the adjacent layer) equals  $\sum_{i=1}^{\ell-1} N_i N_{i+1}$ , when omitting additional bias. As we shall prove in the following, impressive performance can be also achieved by pursuing a markedly different procedure, which requires acting on just  $N_1 + N_\ell + 2 \sum_{i=2}^{\ell-1} N_i$  free parameters (not including bias).

To illustrate the effectiveness of the proposed methodology we make reference to Fig. 4.1 (a), which summarises a first set of results obtained for MNIST. To keep the analysis as simple as possible we have here chosen to deal with  $\ell = 3$ . The sizes of the input ( $N_1$ ) and output ( $N_3$ ) layers are set by the specificity of the considered dataset. Conversely, the size of the intermediate layer ( $N_2$ ) can be changed at will. We then monitor the relative accuracy, i.e. the accuracy displayed by the deep neural networks trained according to different strategies, normalised to the accuracy achieved with an identical network trained with conventional methods. In the upper panel of 4.1 (a), the performance of the non linear neural networks trained via the modified spectral strategy (referred to as to *Spectral*) is displayed in blue (triangles). The recorded accuracy is satisfactory (about 90% of that obtained with usual means and few percent more than that obtained with the spectral method of original conception [23]), despite

the modest number of trained parameters. To exemplify this, in the bottom panel of Fig. 4.1 (a) we plot the relative ratio of the number of tuned parameters (*Spectral* vs. conventional one) against  $N_2$  (blue triangles) : the reduction in the number of parameters as follows the modified spectral method is staggering. Working with the other employed dataset, respectively F-MNIST and CIFAR-10, yields analogous conclusions shown in Figures 4.1 (b) and 4.1 (c).

### 4.1.2 Spectral SVD

One further improvement can be achieved by replacing  $\phi^{(k)}$  with its equivalent singular value decomposition (SVD), a factorization that generalizes the eigendecomposition to rectangular (in this framework,  $N_{k+1} \times N_k$ ) matrices (see [37] for an application to neural networks). In formulae, this amounts to postulate  $\phi^{(k)} = \mathbf{U}_k \mathbf{\Sigma}_k \mathbf{V}_k^T$  where  $\mathbf{V}_k$  and  $\mathbf{U}_k$  are, respectively,  $N_k \times N_k$  and  $N_{k+1} \times N_{k+1}$  real orthogonal matrices. On the contrary,  $\mathbf{\Sigma}_k$  is a  $N_{k+1} \times N_k$  rectangular diagonal matrix, with non-negative real numbers on the diagonal. The diagonal entries of  $\mathbf{\Sigma}_k$  are the singular values of  $\phi_k$ . The symbol  $(\cdot)^T$ , stands for the transpose operation. The learning scheme can be hence reformulated as follows. For each  $k$ , generate two orthogonal random matrices  $\mathbf{U}_k$  and  $\mathbf{V}_k$ . These latter are not updated during the successive stages of the learning process. At variance, the  $M_{k+1} = \min(N_k, N_{k+1})$  non trivial elements of  $\mathbf{\Sigma}_k$  take active part to the optimisation process and its diagonal values are tuned. For each  $k$ ,  $M_{k+1} + N_k + N_{k+1}$  parameters can be thus modulated to optimize the information transfer, from layer  $k$  to layer  $k + 1$ . Stated differently,  $M_{k+1}$  free parameters adds up to the  $N_k + N_{k+1}$  eigenvalues that get modulated under the original spectral approach. One can hence count on a larger set of parameters as compared to that made available via the spectral method, restricted to operate with the eigenvalues. Nonetheless, the total number of parameters scales still with the linear size  $N$  of the deep neural network, and not quadratically, as for a standard training carried out in direct space. This addition (referred to as the *S-SVD* scheme) yields an increase of the recorded classification score, as compared to the setting where the *Spectral* method is solely employed, which is however not sufficient to fill the gap with conventional schemes (see Fig. 4.1 (a)). Similar scenarios are found for F-MNIST and CIFAR-10 4.1 (b,c), with varying degree of improvement, which reflects the specificity of the considered dataset. Analogous results have been proven for a simplified multi layer scenario and are shown in Appendix A.7.

## 4.2 Spectral QR

A decisive leap forward is however accomplished by employing a QR factorization of matrix  $\phi^{(k)}$ . For  $N_{k+1} > N_k$ , this corresponds to writing the  $N_{k+1} \times N_k$  matrix  $\phi_k$  as the product of an orthogonal  $N_{k+1} \times N_k$  matrix  $\mathbf{Q}_k$  and an upper triangular  $N_k \times N_k$  matrix  $\mathbf{R}_k$ . Conversely, when  $N_{k+1} < N_k$ , we factorize  $\phi_k^T$ , in such a way that the square matrix  $\mathbf{R}_k$  has linear dimension  $N_{k+1}$ . In both cases, matrix  $\mathbf{Q}_k$  is randomly generated and stays frozen during gradient descent optimisation. The  $M_{k+1}(M_{k+1} + 1)/2$  entries of the  $M_{k+1} \times M_{k+1}$  matrix  $\mathbf{R}_k$  can be adjusted so as to improve the classification ability of the trained network (this strategy of training, integrated to the *Spectral* method, is termed *S-QR* ). Results are depicted in Fig. 4.1 with red diamonds (For MNIST,

F-MNIST and CIFAR-10 in panels (a,b,c) respective). The achieved performance is practically equivalent to that obtained with a conventional approach to learning. Also in this case  $\rho < 1$ , the gain in parameter reduction being noticeable when  $N_1$  is substantially different (smaller or larger) than  $N_2$ , for the case at hand.

### 4.2.1 $S$ - $QR$ , Sparse $\mathbf{R}$

Interestingly enough, for a chief improvement of the performance, over the SVD reference case, it is sufficient to train a portion of the off diagonal elements of  $\mathbf{R}$ . In the following, we report the recorded accuracy against  $p$ , the probability to train the entries that populate the non null triangular part of  $\mathbf{R}_k$ . The value of the accuracy attained with conventional strategies to the training is indeed approached, already at values of  $p$  which are significantly different from unit. Introduce  $p \in [0, 1]$ . When  $p = 0$ , the diagonal elements of  $\mathbf{R}$  in the  $S$ - $QR$  method are solely trained. The off-diagonal elements are instead frozen to random values. In the opposite limit, when  $p = 1$  all elements of matrix  $\mathbf{R}$  are assumed to be trained. Intermediate values of  $p$  interpolate between the aforementioned limiting conditions. More specifically, the entries that undergo optimisation, are randomly chosen from the pool of the available ones, as reflecting the selected fraction. In Fig. 4.2 (a) the relative accuracy for MNIST is plotted against  $p$ . Here, the network is made of  $\ell = 3$  layers with  $N_2 = 500$ . A limited fraction of parameters is sufficient to approach the accuracy displayed by the network trained with conventional means. In Figs. 4.2 (b) and 4.2 (c) the results relative to F-MNIST and CIFAR-10 are respectively reported. The same results have been validated also on a multi-layered architecture for F-MNIST and CIFAR-10 and can be found on Appendix A.7.

## 4.3 Spectral training of sparse networks

The quest for a limited subset of key parameters which define the target of a global approach to the training is also important for its indirect implications, besides the obvious reduction in terms of algorithmic complexity. As a key application to exemplify this point, we shall consider the problem of performing the classification tasks considered above, by training a neural network with a prescribed degree of imposed sparsity. This can be achieved by applying a non linear filter on each individual weight  $w_{ij}$ . The non linear mask is devised so as to return zero (no link present) when  $|w_{ij}| < C$ . Here,  $C$  is an adaptive cut-off which can be freely adjusted to allow for the trained network to match the requested amount of sparsity. This latter is measured by a scalar quantity, spanning the interval  $[0, 1]$ : when the degree of sparsity is set to zero, the network is dense. At the opposite limit, when the sparsity equals one, the nodes of the network are uncoupled and the information cannot be transported across layers. Working with the usual approach to the training, which seeks to modulate individual weights in direct space, one has to face an obvious problem. When the weight of a given link is turned into zero, then it gets excluded by the subsequent stages of the optimisation process. Consequently, a weight that has been silenced cannot regain an active role in the classification handling. This is not the case when operating under the spectral approach to learning, also when complemented by the supplemental features tested above. The target of the optimisation, the spectral attributes of the transfer

operators, are not biased by any filtering masks: as a consequence, acting on them, one can rescue from oblivion weights that are deemed useless at a given iteration (and, as such, silenced), but which might prove of help, at later stages of the training. In Fig. 4.3, the effect of the imposed sparsity on the classification accuracy is represented for conventional vs.  $S$ - $QR$  method. The latter is definitely more performant in terms of displayed accuracy, when the degree of sparsity gets more pronounced. The drop in accuracy as exhibited by the sparse network trained with the  $S$ - $QR$  modality is clearly less pronounced, than that reported for an equivalent network optimised in direct space. Deviations between the two proposed methodologies become indeed appreciable in the very sparse limit, i.e. when the residual active links are too few for a proper functioning of the direct scheme. In fact, edges which could prove central to the classification, but that are set silent at the beginning, cannot come back to active. At variance, the method anchored to reciprocal space can identify an optimal pool of links (still constraint to the total allowed for) reversing to the active state, those that were initially set to null. Interestingly, it can be shown that a few hubs emerge in the intermediate layer, which collect and process the information delivered from the input stack. Analogous results have been proven for F-MNIST in a simplified multi layer scenario and are shown in Appendix A.7.

Taken altogether, it should be concluded that a large body of free parameters is *de facto* unessential. The spectral learning scheme, supplemented with a QR training of the non trivial portion of eigenvectors' matrix, enabled us to identify a limited subset of key parameters which prove central to the learning procedure, and reflect back with a global impact on the computed weights in direct space. This observation could materialise in a drastic simplification of current machine learning technologies, a challenge at reach via algorithmic optimisation carried out in dual space. Quite remarkably, working in reciprocal space yields trained networks with better classification scores, when operating at a given degree of imposed sparsity. This finding suggests that shifting the training to the spectral domain might prove beneficial for a wide gallery of deep neural networks applications.

## 4.4 Conclusions

In our study of neural network training, we've closely examined the role of eigenvalues and eigenvectors. Our findings indicate that these quantities can be crucial for optimization processes. We've also introduced a dual-eigenvalue method, termed S-SVD, which has shown to be effective in balancing both network efficiency and computational cost. Moreover, performances equivalent to conventional learning schemes can be attained when the QR decomposition is employed; showing that with smarter parametrizations and correlations a very effective training can be carried out by employing a very small number of trainable parameters with respect to the naive implementation. Furthermore, this paradigm shift that couples different weights by using a spectral derived parametrization, makes a sparsification oriented training extremely effective as the constrains in the direct space - imposed with non linear filters- can be dynamically moulded during the training process, letting turned off connections to become active once again if needed.

Next, we will extend these findings to consider the importance of individual nodes

within the network. If eigenvalues act as tuning parameters for the network's behavior, they can also shed light on which nodes are more or less critical. This leads us to our next area of investigation: the use of eigenvalues as metrics for network pruning strategies.

As we proceed to the following chapter, our focus will move from a broader optimization landscape to the specifics of individual node importance. Through the lens of eigenvalues, we intend to explore a mapping between nodes and their respective eigenvalues. This holds the promise for understanding the network's intrinsic structure and for developing more efficient, yet still effective, network architectures. We will delve into how these spectral insights can guide us in achieving a balance between network compactness and performance showing that such novel parametrization implicitly regularizes the complexity of the network.



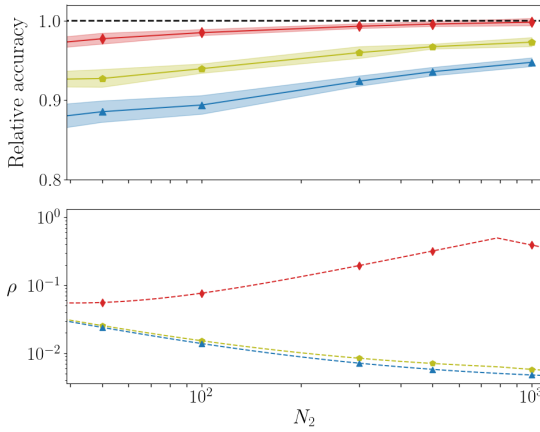
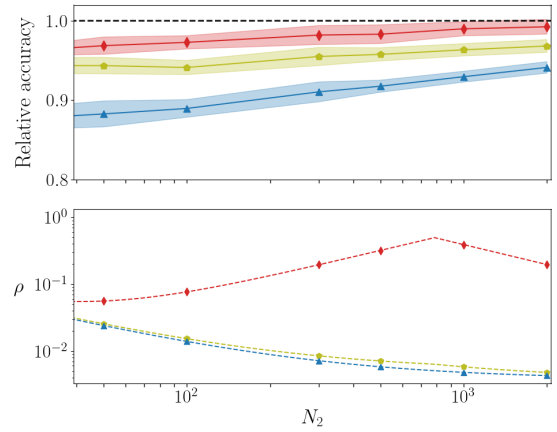
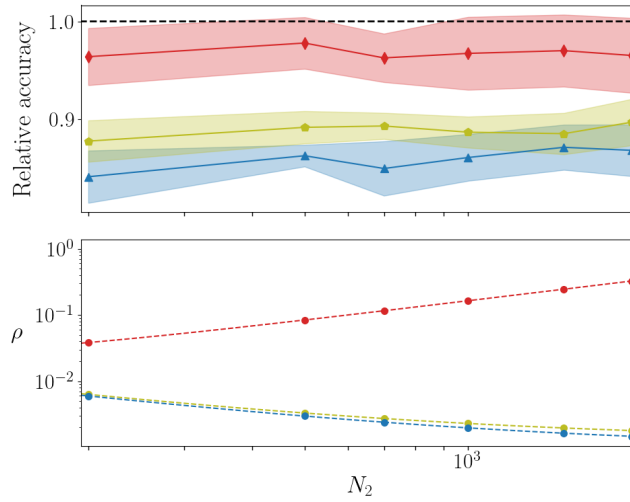
(a) MNIST,  $\ell = 3$ , with  $N_2 = 500$ (b) F-MNIST,  $\ell = 3$ , with  $N_2 = 500$ (c) CIFAR-10,  $\ell = 3$ , with  $N_2 = 700$ 

Figure 4.1: Upper panel: the accuracy of the different learning strategies, normalised to the accuracy obtained for an identical deep neural network trained in direct space, as a function of the size of the intermediate layer,  $N_2$ . Triangles stand for the the relative accuracy obtained when employing the spectral method (*Spectral*). Pentagons refer to the setting which extends the training to the eigenvectors' blocks via a SVD decomposition. Specifically, matrices  $\mathbf{U}_k$  and  $\mathbf{V}_k$  are randomly generated (with a uniform distribution of the entries) and stay unchanged during optimisation. The singular values are instead adjusted together with the eigenvalues which stem from the spectral method (this configuration is labelled *S-SVD*). Diamonds are instead obtained when the eigenvalues and the elements of the triangular matrix  $\mathbf{R}$  (as follows a QR decomposition of the eigenvectors' blocks) are simultaneously adjusted *S-QR*. Here,  $\mathbf{Q}$  is not taking part to the optimisation process (its entries are random number extracted from a uniform distribution). Errors are computed after 10 independent realisations of the respective procedures. Lower panel:  $\rho$  the ratio of the number of tuned parameters (modified spectral, *S-SVD*, *S-QR* methods vs. conventional one) is plotted against  $N_2$ . In calculating  $\rho$  the contribution of the bias is properly acknowledged. In panel (a) the results for MNIST, (b) F-MNIST and (c) CIFAR-10.

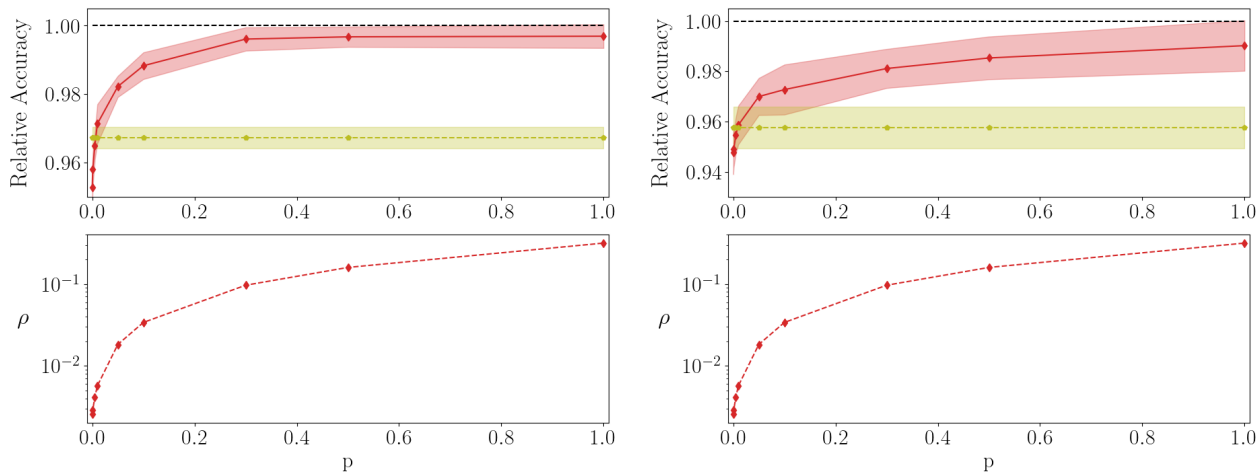
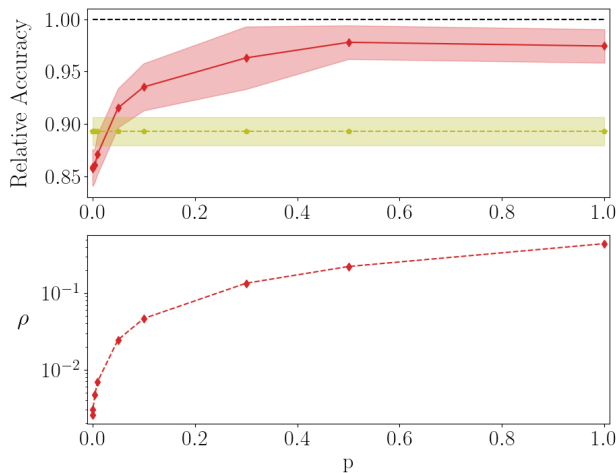
(a) MNIST,  $\ell = 3$ , with  $N_2 = 500$ .(b) F-MNIST  $\ell = 3$ , with  $N_2 = 500$ .(c) CIFAR-10  $\ell = 3$ , with  $N_2 = 700$ 

Figure 4.2: The (relative) classification accuracy is plotted (red, diamond and solid line) against  $p$ , the probability to train the entries that populate the non null triangular part of  $R$ . The corresponding value of the relative accuracy as computed via the  $S$ -SVD is also reported (green, pentagons and solid lines). The averages are carried out over 10 independent realisations.

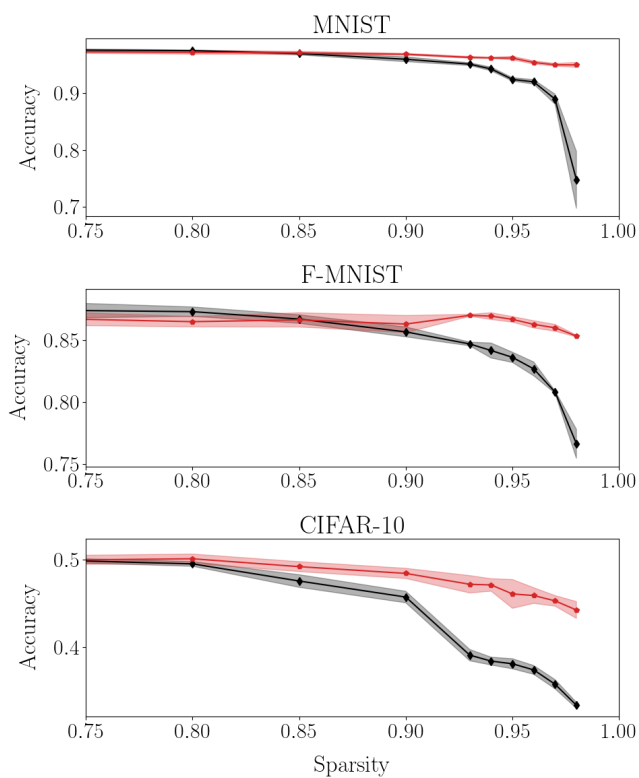


Figure 4.3: **Training sparse networks.** The accuracy of the trained network against the degree of imposed sparsity. Black diamonds refer to the usual training in direct space, while red pentagons refer to the *S-QR* method. From top to bottom: results are reported for MNIST, F-MNIST and CIFAR-10, respectively. In all cases,  $\ell = 3$ .



# Chapter 5

## Spectral Pruning

### 5.1 Introduction

In this Chapter we will discuss a relevant byproduct of the spectral learning scheme. More specifically, we will argue that the eigenvalues do provide a reliable ranking of the nodes, in terms of their associated contribution to the overall performance of the trained network. Working along these lines, we will empirically prove that the absolute value of the eigenvalues is an excellent marker of the node’s significance in carrying out the assigned discrimination task. This observation can be effectively exploited, downstream of training, to filter the nodes in terms of their relative importance and prune the unessential units so as to yield a more compact model, with almost identical classification abilities. The effectiveness of the proposed method has been tested for different feed-forward architectures, with just a single or multiple hidden layers, by invoking several activation functions, and against distinct datasets for image recognition, with various levels of inherent complexity. Building on these findings, we will also propose a two stages training protocol to generate minimal networks (in terms of allowed computing neurons) which outperform those obtained by hacking off dispensable units from a large, fully trained, apparatus. This strategy can be seen as an effective way to discover sub-networks (a.k.a. *winning tickets* [38]) with recorded performance comparable to those displayed by their unaltered homologues, after a proper round of training [38]. More specifically, after a first round of training which solely acts on the eigenvalues, one can identify the most relevant nodes, as follows the magnitude of the associated eigenvalues. Since the first training stage is just targeted to eigenvalues, the eigenvectors obtained after pruning are still bearing reflexes of the random initialization and thus represent a sort of *winning ticket*. In this respect, according to the above reasoning, the proposed two stages strategy can be seen as a novel and efficient way to discover optimal sub-networks. In the next Chapter, we will improve and more carefully analyze its efficacy.

### 5.2 Conventional Pruning Techniques

Generally speaking, it is possible to ideally group various approaches for network compression into five different categories: Weights Sharing, Network Pruning, Knowledge Distillation, Matrix Decomposition and Quantization [39], [40].

*Weights Sharing* defines one of the simplest strategies to reduce the number of parameters, while allowing for a robust feature detection. The key idea is to have a shared set of model parameters between layers, a choice which reflects back in an effective model compression. An immediate example of this methodology are the convolutional neural networks [41]. A refined approach is proposed in Bat et al. [42] where a virtual infinitely deep neural network is considered. Further, in Zhang et al. [43] an  $\ell_1$  group regularizer is exploited to induce sparsity and, simultaneously, identify the subset of weights which can share the same features.

*Network Pruning* is arguably one of the most common technique to compress Neural Network: in a nutshell it aims at removing a set of weights according to a certain criterion (magnitude, importance, etc). Chang et al. [44] proposed an iterative pruning algorithm that exploits a continuously differentiable version of the  $\ell_{\frac{1}{2}}$  norm, as a penalty term. Molchanov et al. [45] focused on pruning convolutional filters, so as to achieve better inference performances (with a modest impact on the recorded accuracy) in a transfer leaning scenario. Starting from a network fine-tuned on the target task, they proposed an iterative algorithm made up of three main parts: (i) assessing the importance of each convolutional filter on the final performance via a Taylor expansion, (ii) removing the less informative filters and (iii) re-training the remaining filters, on the target task. Inspired by the pioneering work in [38], Pau de Jorge et al. [46] proved that pruning at initialization leads to a significant performance degradation, after a certain pruning threshold. In order to overcome this limitation they proposed two different methods that enable an initially trimmed weight to be reconsidered during the subsequent training stages.

*Knowledge Distillation* is yet another technique, firstly proposed by Hinton et al. [47]. In its simplest version Knowledge Distillation is implemented by combining two objective functions. The first accounts for the discrepancy between the predicted and true labels. The second is the cross-entropy between the output produced by the examined network and that obtained by running a (generally more powerful) trained model. In [48] Polino et al. proposed two approaches to mix distillation and quantization (see below): the first method uses the distillation during the training of the so called student network under a fixed quantization scheme while the second exploits a network (termed the teacher network) to directly optimize the quantization. Mirzadeh et al. [49] analyzed the regime in which knowledge distillation can be properly leveraged. They discovered that the representation power gap of the two networks (teacher and student) should be bounded for the method to yield beneficial effects. To resolve this problem, they inserted an intermediate network (the assistant), which sits in between the teacher and the student, when their associated gap is too large.

*Matrix Decomposition* is a technique that remove redundancies in the parameters by the means of a tensor/matrix decomposition. Masana et al. [50] proposed a matrix decomposition method for transfer learning scenario. They showed that decomposing a matrix taking into account the activation outperforms the approaches that solely rely on the weights. In [51], Novikov et al. proposed to replace the dense layer with its Tensor-Train representation [52]. Yu et al. [53] introduced a unified framework, integrating the low-rank and sparse decomposition of weight matrices with the feature map reconstructions. Similar concepts have been explored in [35] where different matrix entries are parametrized via apt kernel expression.

*Quantization*, as also mentioned above, aims at lowering the number of bits used to

represent any given parameter of the network. Stock et al. [54] defined an algorithm that quantize the model by minimizing the reconstruction error for inputs sampled from the training set distribution. The same authors also claimed that their proposed method is particularly suited for compressing residual network architectures and that the compressed model proves very efficient when run on CPU. In Banner et al. [55] a practical 4-bit post-training quantization approach was introduced and tested.

*Node pruning* (or *network slimming*) is a method to reduce network complexity. A very popular example is the one by He et al. in [56]. Once the network has been trained, nodes are classified by means of a node importance function and then removed or retained depending on their score. The authors proposed three different node ranking functions: entropy, output-weights norm and input-weights norm. In particular, the input-weights norm function is defined as the sum of the absolute values of the incoming connections weights. As we will see this latter defines the benchmark model that we shall employ to challenge the performance of the trimming strategy here proposed. In the same spirit we find the Conditional Computation methods [57]–[59]: the aim is to dynamically skip part of the network according to the provided input so as to reduce the computational burden. Related examples can be found in [60]. In this work a neuron importance score function is calculated as function of the average activation and the related weights, making it possible to extract the neuron relevance. At variance with the method that we propose, the relevance is calculated, by Yu and collaborators, after training and using Training set related statistical quantities. In [61], in line with the older and very popular approach of [62], the relevance of nodes is assessed by using the Taylor expansion up to order two, clearly involving further numerical operations. For completeness we would like to point out that, in 2018, Liu and collaborators [63], the value of pruning strategies that follows the canonical pipeline of *Training*  $\rightarrow$  *Extracting Relevance Indicators*  $\rightarrow$  *Pruning*, not always results in better performances when compared with model initialized with random weights.

In contrast with several of those approaches, however, our method involves no further after training elaboration: all the information will be stored in the eigenvalues magnitude.

The method relies on the spectral learning [24], [64] and exploits the fact that eigenvalues are suitable parameters to gauge the importance of a given node among those composing the destination layer. In short, our aim is to make the network more compact by removing nodes classified as unimportant, according to a suitable spectral rating.

## 5.3 Methods

We detail here the spectral procedure to make a trained network smaller, while preserving its ability to perform classification.

To introduce the main idea of the proposed method, we make reference to formula (2.7) and assume the setting where  $\lambda_j^{(k)in} = 0$ . The information travelling from layer  $k$  to layer  $k + 1$  gets hence processed as follows: first, the activity on the departure node  $j$  is modulated by a multiplicative scaling factor  $\phi_{ij}^{(k)}$ , specifically linked to the selected

( $ij$ ) pair. Then, all incoming (and rescaled) activities reaching the destination node  $i$  are summed together and further weighted via the scalar quantity  $\lambda_i^{(k)out}$ . This latter eigenvalue, downstream of the training, can be hence conceived as a distinguishing feature of node  $i$  of layer  $k + 1$ . Assume for the moment that  $\phi_{ij}^{(k)}$  are drawn from a given distribution and stay put during optimization. Then, every individual neuron bound to layer  $k + 1$  is statistically equivalent (in terms of incoming weights) to all other nodes, belonging to the very same layer. The eigenvalues  $\lambda_i^{(k)}$ , dropping the apex *out* from now on, gauge therefore the relative importance of the nodes, within a given stack, and as reflecting the (randomly generated) web of local inter-layer connections (though statistically comparable). Large values of  $|\lambda_i^{(k)}|$  suggest that node  $i$  on layer  $k + 1$  plays a central role in the economy of the neural network functioning. This is opposed to the setting when  $|\lambda_i^{(k)}|$  is found to be small. Stated differently, the subset of trained eigenvalues provide a viable tool to rank the nodes according to their degree of importance. As such, they can be used as reference labels to make decision on the nodes that should be retained in a compressed analogue of the trained neural network, with unaltered classification performance. As empirically shown in the section (5.4) with reference to a variegated set of applications, the sorting of the nodes based on the optimized eigenvalues turns out effective also when the eigenvectors get simultaneously trained, thus breaking, at least in principle, statistical invariance across nodes.

As we will clarify, the latter setting translates in a post-training spectral pruning strategy, whereas the former materializes in a rather efficient pre-training procedure. The non linear activation function as employed in the training scheme leaves a non trivial imprint, which has to be critically assessed.

More specifically, in carrying out the numerical experiments here reported we considered two distinct settings, as listed below:

- (i) As a first step, we will begin by considering a deep neural network made of  $N$  neurons organized in  $\ell$  layers. The network will be initially trained by solely leveraging on the set of tunable eigenvalues. Then, we will proceed by progressively removing the neurons depending on their associated eigenvalues (as in the spirit discussed above). The trimmed network, composed by a total of  $M < N$  units, still distributed in  $\ell$  distinct layers, can be again trained acting now on the eigenvectors, while keeping the eigenvalues frozen to the earlier determined values. This combination of steps, which we categorize as pre-training, yields a rather compact neural network ( $M$  can be very small) which performs equally well than its fully trained analogue made of  $N$  computing nodes.
- (ii) We begin by constructing a deep neural network made of  $N$  neurons organized in  $\ell$  layers. This latter undergoes a full spectral training, which optimizes simultaneously eigenvectors and the eigenvalues. The trained network can be compressed, by pruning the nodes which are associated to eigenvalues (see above) with magnitude smaller than a given threshold. This is indeed a post-training pruning strategy, as it acts *ex post* on a fully spectral trained device.

To evaluate the performance of the proposed spectral pruning strategies (schematically represented in the flowchart of Figure 5.1), we also introduced a reference benchmark model. This latter can be conceptualized as an immediate overturning of the methods in direct space. Simply stated, we train the neural network in the space of



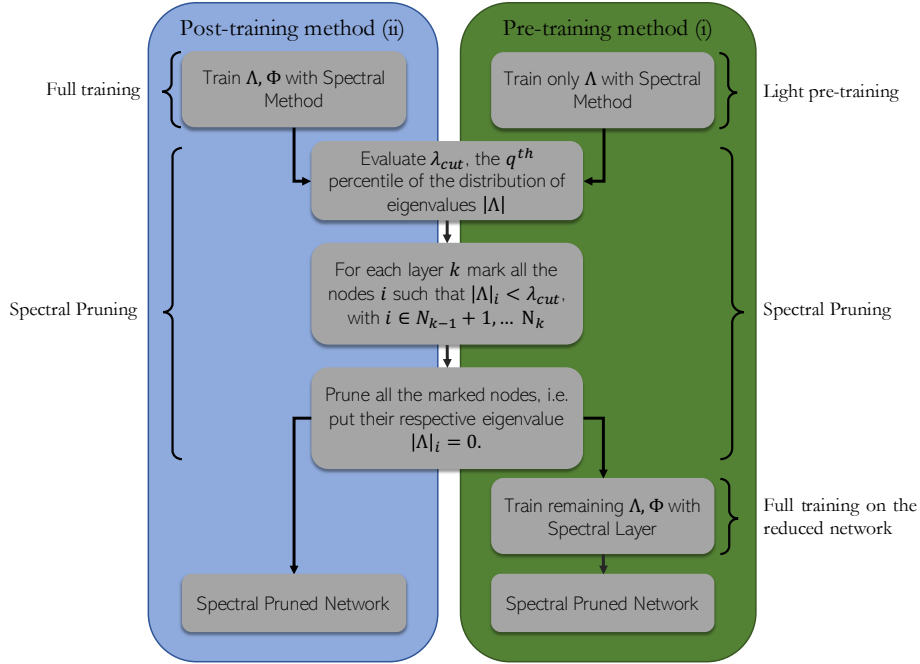


Figure 5.1: Flowchart of the pre- and post-spectral training pruning strategies as presented in section 5.3.

nodes, by using standard approaches to the learning. Then, we classify the nodes in terms of their relevance using a proper metric to which shall make reference below, and consequently trim the nodes identified as less important. When adopting the spectral viewpoint, one can rely on the eigenvalues to rank the nodes importance. As remarked above, in fact, the eigenvalues at the receiver nodes set a local scale for the incoming activity, the larger the eigenvalue (in terms of magnitude) the more important the role played by the processing unit. As a surrogate of the eigenvalues, when anchoring the train in direct space, we can consider the quantity  $\sum_{j=1}^{N_k} |w_{ij}|$ , for each neuron  $i$  belonging to layer  $k + 1$ , see also [56]. The absolute value prevents mutual cancellations of sensible contributions bearing opposite signs, which could incidentally hide the actual importance of the examined node.

In all explored cases, the pruning is realized by imposing a threshold on the reference indicator (be it the magnitude of the eigenvalues or the cumulated flux of incoming –and made positive– weights). Pointedly, the respective indicator is extracted for every node in the arrival layer. Then a percentile  $q$  is chosen and the threshold fixed to the  $q$ -th percentile. Nodes displaying an indicator below the chosen threshold are removed and the accuracy of the obtained (trimmed) neural network assessed on the test-set. The codes employed, as well as a notebook to reproduce our results, can be found in the public repository of this project.

## 5.4 Results

In order to assess the effectiveness of the eigenvalues as a marker of the node’s importance (and hence as a potential target for a cogent pruning procedure) we will consider a fully connected feed-forward architecture. Applications of the explored methods will

be reported for  $\ell = 3$  and  $\ell > 3$  configurations. The nodes that compose the hidden layers are the target of the implemented pruning strategies. As we shall prove, it is possible to get rid of the vast majority of nodes without reflecting in a sensible decrease in the test accuracy, if the filter, either in its pre- or post-training versions, relies on the eigenvalues ranking. Moreover, it is also important to stress that, in general terms, the pruning of unessential nodes improves the computational efficiency of the network. As a matter of fact, reducing the number of output nodes leads a compression in terms of both memory and inference time which is directly proportional to the number of removed elements. As an example, by pruning a fraction  $\alpha$  ( $< 1$ ) of the total nodes, we obtain a new layer with  $\alpha \cdot N$  less neurons and a memory reduction of  $\alpha \cdot N$  times the number of input features.

For our test, we used three different datasets of images. The first is the renowned MNIST database of handwritten digits [30], composed by greyscale images of dimension  $28 \times 28$  pixels. The second is Fashion-MNIST (F-MNIST) [33] (an image dataset of Zalando’s items) which are still depicted with a greyscale with dimension  $28 \times 28$  but display an enhanced degree of inherent complexity for what concerns the type of classification required as compared to the basic MNIST benchmark model (more complex shapes, patterns on items). The last one is CIFAR-10 [65] a richer dataset composed by  $32 \times 32$  RGB images of complex real-world objects divided in 10 classes. Further, different activation functions have been employed to evaluate the performance of the methods. In the following we will show the results obtained for the ELU but, as can be seen from the Figures, similar results are obtained when operating with the ReLU and tanh. In the following we will report into two separate sub-sections the results pertaining to either the single or multiple hidden layers settings.

### 5.4.1 Single hidden layer ( $\ell = 3$ )

In Figure 5.2 the performance of the inspected methods are compared for the minimal case study of a three layers network. The intermediate layer, the sole hidden layer in this configuration, is set to  $N_2 = 500$  neurons. The accuracy of the different methods are compared, upon cutting at different percentile, following the strategies discussed in the Methods and compared with the benchmark model (the orange profile). In the benchmark model, the neural network is trained in direct space, by adjusting the weights of each individual inter-nodes connection. Then, the absolute value of the incoming connectivity is computed and used as an importance rank of the nodes’ influence on the test accuracy (analogous to the way in which we use the eigenvalues). Such a model has been presented and discussed by He et al. in [56]. Following this assessment, nodes are progressively removed from the trained network, depending on the imposed percentile, and the ability of the trimmed network to perform the sought classification (with no further training) tested. We choose this particular type of trimming as a benchmark to our spectral pruning technique for the following reasons. First, it also amount to removing nodes from the collection, and not just sparsify the weight of the associated transfer matrices. Then, both approaches build on the concept of nodes ranking, as obtained from a suitable metric, which is respectively bound to direct vs. spectral domains. We point out that in the next Chapter also another indicator, the *feature norm* will be evaluated, obtaining analogous results. The aforementioned procedure is repeated 5 times and the mean value of the accuracy

plotted in the orange curve of Figure 5.2. The shaded region stands for the semi dispersion of the measurements. A significant drop of the network performance is found when removing a fraction of nodes larger than 60 % from the second layer.

The blue curve Figure 5.2 refers instead to the post-processing spectral pruning based on the eigenvalues and identified, as method (ii), in the Methods Section of this Chapter. More precisely, the three layers network is trained by simultaneously acting on the eigenvectors and the eigenvalues of the associated transfer operators, as illustrated above. The accuracy displayed by the network trained according to this procedure is virtually identical to that reported when the learning is carried out in direct space, as one can clearly appreciate by eye inspection of Figure 5.2. Removing the nodes based on the magnitude their associated eigenvalues, allows one to keep stable (practically unchanged) classification performance for an intermediate layer that is compressed of about 70% of its original size. In this case the spectral pruning is operated as a post-processing filter, meaning that the neural network is only trained once, before the nodes' removal takes eventually place.

At variance, the green curve in Figure 5.2 is obtained following method (i) from the Methods Section, which can be conceptualized as a pre-training manipulation. Based on this strategy, we first train the network on the set of tunable eigenvalues, than reduce its size by performing a compression that reflects the ranking of the optimized eigenvalues and then train again the obtained network by acting uniquely on the ensemble of residual eigenvectors. The results reported in Figure 5.2 indicate that, following this procedure, it is indeed possible to attain astoundingly compact networks with unaltered classification abilities. Moreover, the total number of parameters that need to be tuned following this latter procedure is considerably smaller than that on which the other methods rely. This is due to the fact that only the random directions (the eigenvectors) that prove relevant for discrimination purposes (as signaled by the magnitude of their associated eigenvalues) undergo the second step of the optimization. This method can also be seen as a similar kind of [38]. As a matter of fact, the initial training of the eigenvalues uncovers a sub-network that, once trained, obtains performances comparable to the original model. More specifically, the uncovered network can be seen as a *winning ticket* [38]. That is, a sub-network with an initialization particularly suitable for carrying out a successful training. Further analysis on this direction will be, however, carried out in the next Chapter, where also a Spectral Regularization will be involved.

Same results are shown for Fashion-MNIST and MNIST dataset and with different activation functions in Figure 5.4 and 5.3 respectively.

Next, we shall generalize the analysis to the a multi-layer setting ( $\ell > 3$ ), reaching analogous conclusions.

## Multiple hidden layers ( $\ell > 3$ )

Quite remarkably, the results achieved in the simplified context of a single hidden layer neural network also apply within the framework of a multi-layers setting.

To prove this statement we set to consider a  $\ell = 5$  feedforward neural network with ELU activation function. Here, again,  $N_1 = 784$  and  $N_5 = 10$  as reflecting the specificity of the employed dataset, i.e. MNIST. The performed tests follows closely those reported above, with the notable difference that now the ranking of the eigenvalues

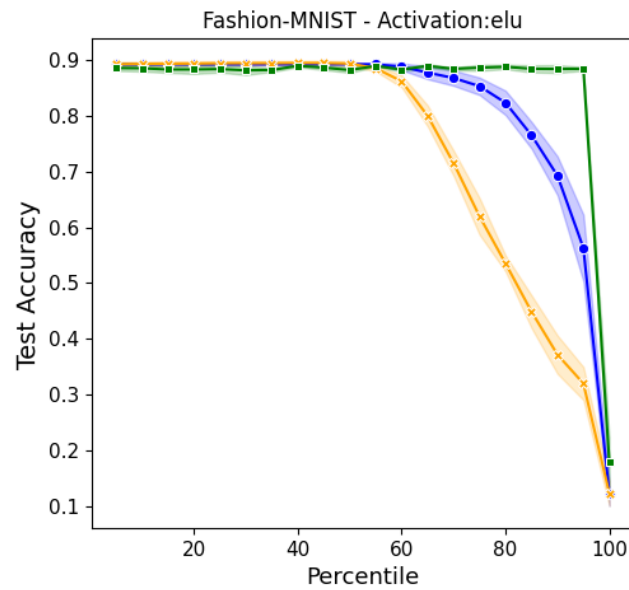


Figure 5.2: Accuracy on the Fashion-MNIST database with respect to the percentage of trimmed nodes (from the hidden layer), in a three layers feedforward architecture. Here,  $N_2 = 500$ , while  $N_1 = 784$  and  $N_3 = 10$ , as reflecting the structural characteristics of the data. In orange the results obtained by pruning the network trained in direct space, based on the absolute value of the incoming connectivity (see main text). In blue, the results obtained when filtering the nodes after a full spectral training (post-training). The curve in green reports the accuracy of the trimmed networks generated upon application of the pre-training filter. Symbols stand for the averaged accuracy computed over 5 independent realizations. The shadowed region is traced after the associated semi-dispersion.

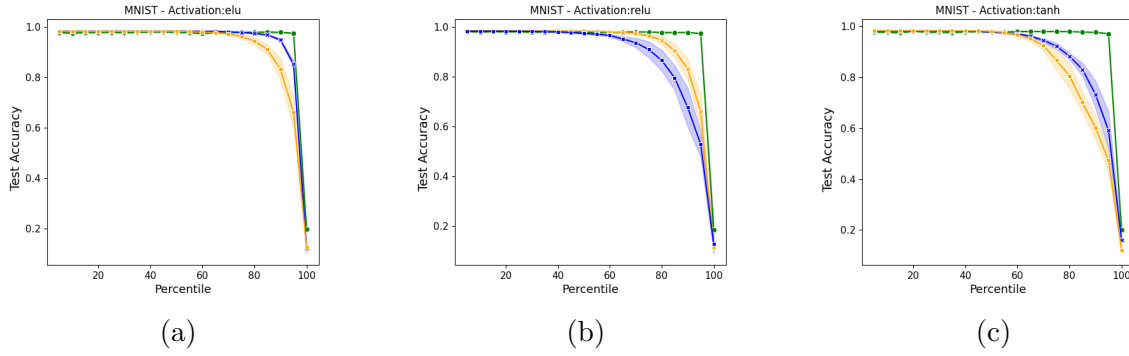


Figure 5.3: Accuracy on the MNIST database with respect to the percentage of trimmed nodes (selected from the 500 neurons that compose the sole hidden layer), in a three layers feedforward architecture. The results reported in each panel refer to a different selection of the nonlinear activation functions, respectively ELU (a), ReLU (b) and tanh (c). In orange, the results obtained by using the trimming procedure based on the absolute value of the incoming connectivity. In blue, the results obtained when filtering the nodes after a full spectral training (post-training). The curve in green displays the accuracy of the trimmed networks generated upon application of the pre-training filter. In this case, the examined network is initially trained on the set of eigenvalues, while keeping the eigenvectors frozen. After having removed unessential nodes, based on their associated eigenvalues, the network undergoes another training phase that is solely targeted to adjusting the entries of the residual eigenvectors. The shadowed region represents the semi-dispersion over 5 independent realizations. When using the Relu function, trimming on the absolute value of the incoming connectivity yields slightly better results than what found when using the post-training spectral filter. The two stages spectral trimming proves always more effective.

is operated on the pool of  $N_2 + N_3 + N_4$  neurons that compose the hidden bulk of the trained network. In other words, the selection of the neuron to be removed is operated after a global assessment, i.e. scanning across the full set of nodes, without any specific reference to an a priori chosen layer.

In Figure 5.5 the results of the analysis are reported, assuming  $N_2 = N_3 = N_4 = 500$ . The conclusions are perfectly in line with those previously reported for the one layer setting, except for the fact that now the improvement of the spectral pruning over the benchmark reference are even superior. To support this claim, let us consider the orange curve that drops at percentage 20, while the blue begins its descent at about 60 %. The green curve, relative to the sequential two steps training, stays stably horizontal up to about 90 %. Analogues results are shown for different activations for Fashion MNIST in Figure 5.7 and for MNIST in Figure 5.6

## Testing the trimming strategies on CIFAR10 dataset

To assess the flexibility (by testing its relation with a convolutional preprocess) of the schemes outlined we here consider the harder CIFAR10 dataset and assume a modified MobileNetV2 [66] adding two dense layer at the end of the network. During training we freeze all the layers, except for the two appended ones that are trained in the

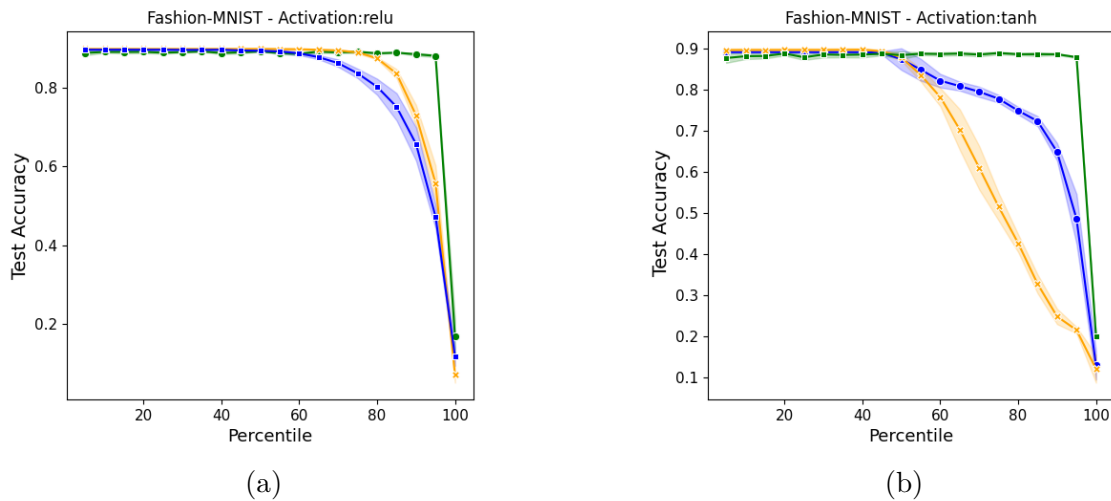


Figure 5.4: Accuracy on the Fashion-MNIST database with respect to the percentage of trimmed nodes (selected from the 500 neurons that compose the sole hidden layer), in a three layers feedforward architecture. The results reported in each panel refer to a different selection of the nonlinear activation functions, respectively ReLU (b) and tanh (c). Symbols and conclusions are in line with those reported for the case of MNIST.

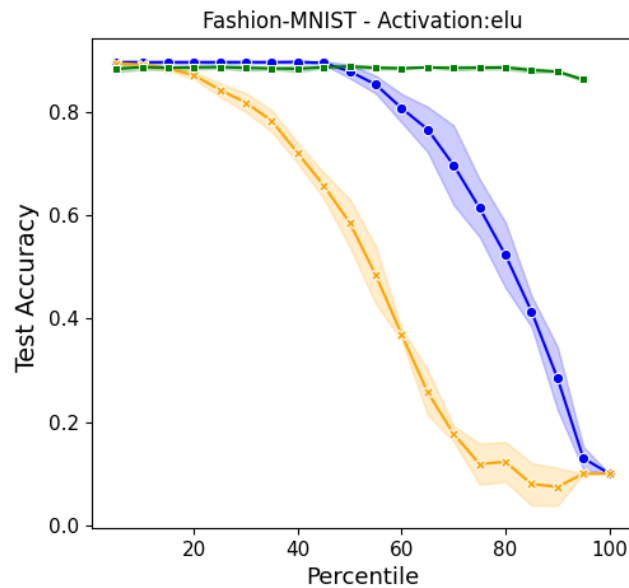


Figure 5.5: Accuracy on the Fashion-MNIST database with respect to the percentage of pruned nodes (from the hidden layers), in a five layers feedforward architecture. Here,  $N_2 = N_3 = N_4 = 500$ , while  $N_1 = 784$  and  $N_5 = 10$ , as reflecting the structural characteristics of the data. Symbols and colors are chosen as in Figure 5.2.

spectral domain. Working in this setting, the pruning is performed on the first dense layer by using both strategies (i) and (ii), as introduced in Section 5.3. Here again the results are compared to those obtained when using the absolute value of the incoming

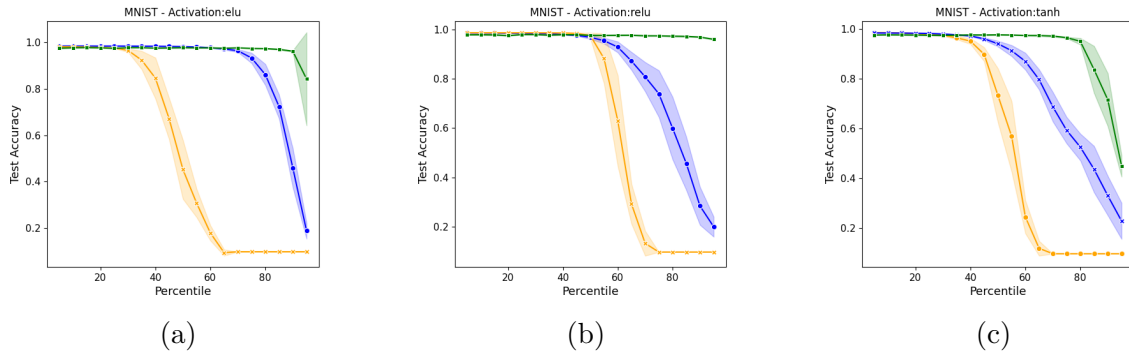


Figure 5.6: Accuracy on the MNIST database with respect to the percentage of trimmed nodes (from the set of  $N_2 + N_3 + N_4$  neurons). The results in each panel refer to different choices of the non linear function, ELU (a), ReLU (b) and tanh (c). Symbols are chosen as for the case of the single hidden layer setting. It should be remarked that the spectral trimming strategies proves definitely more effective than the benchmark model anchored to direct space, also when the Relu function is employed, in the case of multiple hidden layers.

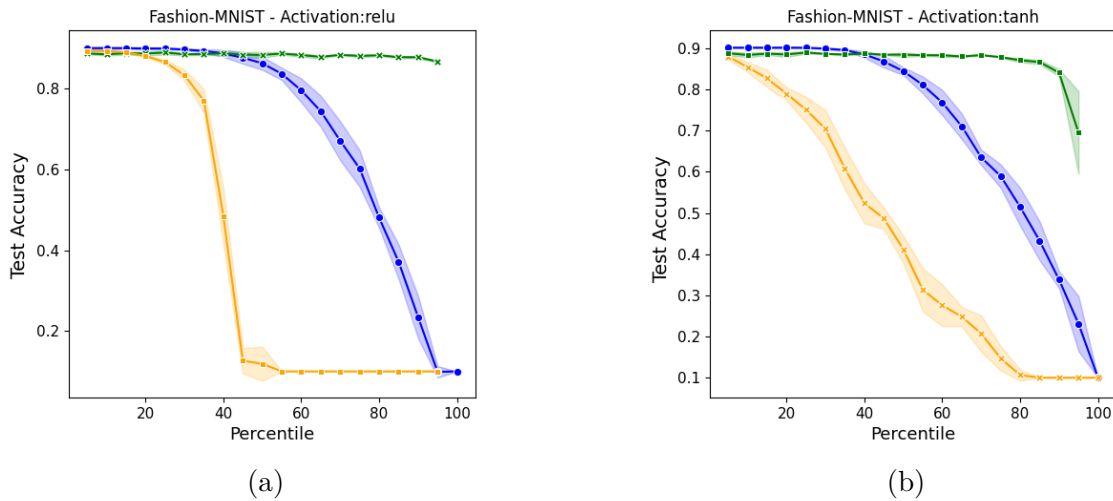


Figure 5.7: Accuracy on the Fashion-MNIST database with respect to the percentage of trimmed nodes (from the set of  $N_2 + N_3 + N_4$  neurons). The results in each panel refer to different choices of the non linear activation function, ReLU (a) and tanh (b). For the symbols, see the caption of the Figures above. Also in this case the spectral filters prove always superior.

connectivity as an alternative trimming criterion (see Figure 5.8). The first dense layer employed is made of 512 nodes with an ELU activation function let us observe that we tested others activation functions and we obtained analogous results.

## 5.5 Conclusions

In this Chapter we have discussed a relevant byproduct of a spectral approach to the learning of deep neural networks. The eigenvalues of the transfer operator that

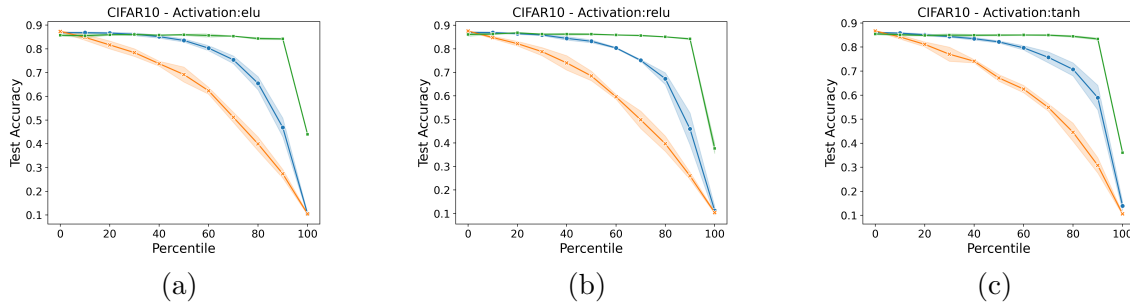


Figure 5.8: Accuracy on the CIFAR10 database with respect to the percentage of trimmed nodes (from the  $\ell - 1$  layer). The results in each panel refer to different non linear functions, respectively ELU (a), ReLU (b) and tanh (c). Symbols are chosen in analogy with the above (the result drawn in green are based on two different runs).

connects adjacent stacks in a multi-layered architecture provide an effective measure of the nodes importance in handling the information processing. By exploiting this fact we have introduced and successfully tested two distinct procedures to yield compact networks –in terms of number of computing neurons– which perform equally well than their untrimmed original homologous. One procedure (referred as (ii) in the description) is acknowledged as a post processing method, in that it acts on a multi-layered network downstream of training. The other (referred as (i)) is based on a sequence of two nested operations. First the eigenvalues are solely trained. After the spectral pruning took place, a second step in the optimization path seeks to adjust the entries of the eigenvectors that populate a trimmed space of reduced dimensionality. The total number of trained parameters is small as compared to that involved when the pruning acts as a post processing filter. Despite that, the two steps pre-processing protocol yields compact devices which outperform those obtained with a single post-processing removal of the unessential nodes.

As a benchmark model, and for a neural network trained in direct space, we decided to rank the nodes importance based on the absolute value of the incoming connectivity. This latter appeared as the obvious choice, when aiming at gauging the local information flow in the space of the nodes, see also [56]. In principle, one could consider to diagonalizing the transfer operators as obtained after a standard approach to the training and make use of the computed eigenvalues to a posteriori sort the nodes relevance. This is however not possible as the transfer operator that links a generic layer  $k$  to its adjacent counterpart  $k + 1$ , as follows the training performed in direct space, is populated only below the diagonal, with all diagonal entries identically equal zero. All associated eigenvalues are hence are zero and they provide no information on the relative importance of the nodes of layer  $k + 1$ , at variance with what happens when the learning is carried out in the reciprocal domain.

Summing up, by reformulating the training of neural networks in spectral space, we identified a set of sensible scalars, the eigenvalues of suitable operators, that unequivocally correlate with the influence of the nodes within the collection. This observation translates in straightforward procedures to generate efficient networks that exploit a reduced number of computing units. Tests performed on different benchmarks corroborate this conclusions.

However, the capabilities of these spectral tools do not end at merely optimizing es-



tablished networks. They extend further, enabling us to draw parallels with theoretical paradigms in Machine Learning, particularly the teacher-student dynamic. Starting from the premise that spectral space allows us to assess node importance and prune accordingly, we will, in the following Chapter, try to get insights into the inherent structures and complexities that exist within student networks, especially when they are modelled after a compact teacher network.

The following Chapter delves deeper into this hypothesis. Building upon our spectral findings, we explore the tantalizing possibility of identifying invariant subnetworks within overparameterized student models, by seeking for structures reminiscent of their teacher counterparts and, more generally, the complexity of the underlying dataset. In order to do so we will leave the realm of implicit bias and shift toward the imposition of a well defined explicit regularization that involves the eigenvalues  $L_2$  norm. As we are about to show, in this simple yet very powerful tool, lies our capability of discovering the structure of a hidden teacher network or, more generally, the function to regress.



# Chapter 6

## Spectral Regularization

### 6.1 Teacher-Student framework

The teacher-student framework is a widely recognized concept in the field of theoretical deep learning, and has gained significant attention due to its effectiveness in knowledge distillation [67]–[69], and theoretical insight [70]–[72]. Introduced by Hinton et al. [73], in the context of neural networks, the teacher-student framework usually involves two neural networks: one called the "teacher" and the other called the "student". The teacher network is generally a pre-trained, well-performing network. Its main role is to guide the training of the student network. By using a softened probability distribution from the teacher model as labels, the student model is encouraged to imitate the teacher's behaviour.

In this Chapter, we will use this particular setting and focus on three different learning regimes. These latter can be hierarchically ranked depending on whether the complexity of the function that the student network learns is greater than, less than, or equal to the complexity of the teacher network function that the student is attempting to regress.

The teacher network is characterized by specific topological features that reflect its inherent structure and path distribution. As a follow-up of the analysis, we will compare these characteristics to those recovered by the student network operated under the spectral paradigm. The degree to which they are preserved in the latter will be one of the metrics that we shall use to determine the effectiveness of the learning process.

To ensure simplicity in our analysis, we will adopt a teacher network with two hidden layers and a scalar output, let us observe that this choice has been grounded on the possibility to have a complete insight into the problem; in the following we will relax this assumption by working with more complex settings. The weights that refer to the final layer are all set to one. The proposed formulation is hence inspired by the theory of soft committee machines [74], [75], but with an additional layer to increase the problem's complexity.

The dataset will be generated by choosing the probability distribution of the Instance space  $p(x)$  to be a Standardized Gaussian in  $\mathbb{R}^{10}$ , namely  $p(x) = \mathcal{N}(x; \mu = 0, \Sigma = \mathbb{I})$ . In this framework, the teacher network  $T(x)$  is used as a Supervisor defining a conditioned probability distribution  $g(y|x) = \delta(y - T(x))$ . The dataset is thus composed by the collection to data-label tuples  $\mathcal{D} = \{(x^{(i)}, y^{(i)})_{i \in 1 \dots |D|} \mid (x^{(i)}, y^{(i)}) \sim p_{data}(x, y) = p(x)g(y|x)\}$  and has a number of elements  $|D| = 1.3 \cdot 10^4$ .

The dimensions of the teacher layers are respectively set to  $10 - 20 - 20 - 1$  and are initialized according to a standard Glorot Uniform distribution [31]. The student network  $S(x)$ , on the other hand, is formed by a two hidden layers deep neural network with dimension  $10 - h - 20 - 1$  where  $h$  is a variable integer that will be tuned across experiments. The second hidden layer is taken to be a standard fully connected layer (the gradient will be computed with respect to the connections). The first hidden layer, whose dimension  $h$  can be changed at will, can be either parametrized as a fully-connected layer or as its spectral analogue (the eigenvalues  $\boldsymbol{\lambda}^{(1)}$  and eigenvectors components  $\boldsymbol{\phi}^{(1)}$  are thus the target of the optimization), depending on the specific aims of the analysis. Accordingly, the loss function will be different depending on the layer employed. Namely, we will use

$$L = \frac{1}{|D|} \sum_{i \in 1}^{|D|} (T(x^{(i)}) - S(x^{(i)}))^2 + \alpha_w \sum_{k \in 1,2} L_2(\mathbf{w}^{(k)}) \quad (6.1)$$

for the student parametrized in the standard way, and

$$L = \frac{1}{|D|} \sum_{i \in 1}^{|D|} (T(x^{(i)}) - S_\lambda(x^{(i)}))^2 + \alpha_\lambda L_2(\boldsymbol{\lambda}^{(1)}) + \alpha_\phi L_2(\boldsymbol{\phi}^{(1)}) + \alpha_w L_2(\mathbf{w}^{(2)}) \quad (6.2)$$

for the student which bears a spectral parametrization of the first hidden layer, here denoted by  $S_\lambda$  for clarity. Note the  $L_2$  penalty term that, as discussed above, acts as an explicit regularization.

To focus on the impact of the spectral parametrization of the layer, we will initialize  $S$  and  $S_\lambda$  such that the set of links is identical but still randomized. We accomplish this by setting  $\lambda_i^{(1)} = 1$  for  $i \in 1 \dots h$  and  $\phi_{ij}^{(1)} = -w_{ij}^{(1)}$  for all  $i \in 1 \dots h$  and  $j \in 1 \dots 10$ . The  $w_{ij}^{(1)}$  are sampled from a Glorot distribution, ensuring that there is no bias in the optimization due to differences in the initialization of the inter-nodes connections. For each run, we will use the minibatch stochastic gradient descent algorithm Adam [28], with a batch size of 300 and 500 for  $S_\lambda$  and  $S$ , respectively, and a total of 2000 epochs. The relevance of the features extracted by  $S$  and  $S_\lambda$  will be computed as the norm of the vectors that modulate the information to be conveyed at the destination nodes on the layer of variable size  $h$ . More specifically, we can extract from the conventionally parametrized student  $S$ , the following indicator bound to node  $i$ :

$$\mathcal{W}_i = \left( \sum_{j=1}^h w_{ij}^2 \right)^{1/2} \quad (6.3)$$

For the network  $S_\lambda$ , on the other hand, we will employ the following expression:

$$\mathcal{L}_i = \lambda_i \left( \sum_{j=1}^h \phi_{ij}^2 \right)^{1/2} \quad (6.4)$$

which scales proportionally to the eigenvalue entry  $\lambda_i$ . It is clear that these quantities hold for every  $i \in 1 \dots h$ , and both support the idea that the larger the  $L_2$  magnitude of the connections, the more relevant the associated processing node. Notice, however, that this latter quantity needs to be appropriately rescaled for the corresponding eigenvalue's magnitude for fair comparison when operating under spectral parametrization.

We speculate that by imposing a node-wise localization of the features when working under the usual parametrization results in a cumbersome exercise which cannot be easily performed. In literature the work by Molchanov et al. [61] use the  $L_2$  norm of the input parameters of every neuron in the network with a regularization that scales with the deepness of the model. In this thesis we will not compare to this work as only preliminary results have been obtained in this direction. Indeed, we would like to stress and explore the multimodal relevance of the spectral parametrization. For what concerns the comparison with state of the art techniques, we are planning to carry out an extensive study in forthcoming papers. On the other hand, this goal can be instead accomplished when working under the spectral parametrization. We also speculate that the same effect could be induced on the input features when using all the spectral components, i.e. when the  $\lambda^{in}$  in Eq.(2.7) are not left untrained. However, we postpone reporting on this alternative scenario in a future publication.

## Formulation of the algorithm

The results presented in this paper are obtained by using an algorithm that can be summarized as follows:

- Replace the conventional feedforward, fully-connected layers with the so called *Spectral* layers, where the links are parameterized as in Equation (2.7).
- Initialize  $\lambda^{(k)in}$  to 0 (and leave it untrained), set  $\lambda^{(k)out}$  to 1, and initialize  $\phi^{(k)}$  by using the Glorot distribution [31].
- Apply  $L_2$  regularization to  $\lambda^{(k)out}$  and  $\phi^{(k)}$ , and proceed with the optimization.
- Examine the entries  $\mathcal{L}_i$  for each node  $i$  from Equation (6.4); a larger value indicates higher node relevance.

The implementation of the *Spectral* layer and the Structural pruning functions can be found in <https://github.com/Jamba15/SpectralTools>.

## 6.2 Experimental Framework

In order to assess the effectiveness of the introduced parametrization and regularization, the dimension of the hidden layer  $h$  has been varied within the set  $\mathcal{I} = [10, 20, 40, 60, 100, 200, 500, 700, 1000]$ . For each  $h \in \mathcal{I}$ , a total of 30 trials have been conducted, train and test losses have been recorded for statistical purposes. This evaluation has been performed for both  $S$  and  $S_\lambda$ , which we will now denote as  $S(h)$  and  $S_\lambda(h)$  respectively, with  $h \in \mathcal{I}$ . The teacher function  $T(x)$  is fixed as described above and the student gets trained over 2000 epochs, to ensure a consistent and proper convergence. The average test loss, evaluated on a different set of 1000 samples distributed according to the same  $p_{data}(x, y)$ , is computed. For all  $h > 20$  (the teacher layer dimension), the average mean squared error (MSE) of  $S(h)$  is  $\langle \text{MSE}(S(h)) \rangle = 9 \pm 1 \times 10^{-3}$  (with a one-sigma error), and the average MSE of the prediction of  $S_\lambda(h)$  reads  $\langle \text{MSE}(S_\lambda(h)) \rangle = 9 \pm 3 \times 10^{-3}$ . Note that averages here are

taken over the 30 repetitions of the experiment. It is worth emphasizing that these values stay stably across all choices of  $h$  above 20. This implies that both models are operated in a properly converging regime, where the two different parametrizations prove equivalent in terms of test error. Building upon this, we can now examine the properties of the network after training in light of the two employed parametrizations. To assess feature localization within the network, we generated aggregated histograms of the scalars (6.3) and (6.4). This aggregation was performed across all trials for a fixed hidden layer size,  $h$ . The range scanned by the aforementioned scalar entries was rescaled in such a way that both  $\mathcal{W}$  and  $\mathcal{L}$  lie in the same range  $[0, 1]$ . This enables for a meaningful comparison between the two distributions. Figure 6.1 demonstrates a stark difference between the empirical distributions of the two computed scalars. The blue histograms stands for (6.3), which refer to the spectral attribute  $\mathcal{L}$ , while the orange histograms represent the feature norm for a conventional regularized training scheme. The latter exhibits a prototypical behavior, with the distribution slowly shifting towards zero as the size of the hidden layer increases. On the other hand, the former displays a more marked dependence on the imposed size  $h$  of the variable hidden layer.

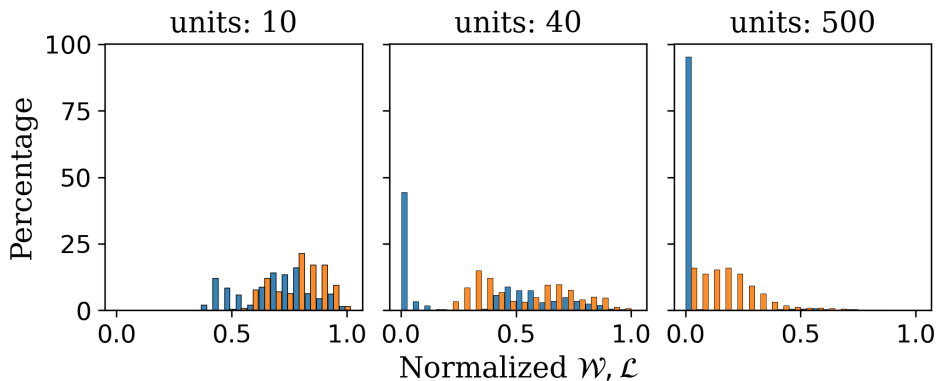


Figure 6.1: Histogram for the quantities (6.4) (in blue) and (6.3) (in orange) for the first hidden layer of the student. The scalars entries are divided by the maximum of the sample so that the two distributions lies in the interval  $[0, 1]$ .

### 6.3 Invariant Core

In the over-parametrized ( $h > 20$ ) student regime, we observe the emergence of a peak in the first bin, which corresponds to values of the feature parameter close to zero. This observation qualifies as a significant change as compared to the under-parametrized regime, where the distribution only contains non-zero values of  $\mathcal{L}_i$ . On the other hand, the behaviour of  $\mathcal{W}$  remains qualitatively similar across different sizes of system being trained. The significance of the zero-peaked  $\mathcal{L}$  values is straightforward: these are the features (i.e. the neurons) that can be safely discarded without affecting the generalization properties of the network  $S_\lambda(h)$ . To support this interpretation, we plot the average dimension size of the first hidden layer, considering only the non-zero features, in the left panel of Figure 6.2. Strikingly enough, we observe the emergence of an invariant processing core within the student, the dimension of which is closely connected to that of the teacher (in this case,  $h = 20$ ). Despite achieving the same

test accuracy, the two networks ( $S$  and  $S_\lambda$ ) exhibit distinct topologies and levels of interpretability. The former demonstrates a sparse structure that distributes information across the entire set of connections, while the latter, obtained as a byproduct of the spectral training complemented with a suitable regularization, drives the information processing towards a minimal and identifiable subnetwork core. The additional feature weighting parameter, which can be interpreted as a particular case of a Spectral parametrization of the transfer operator, could be effectively utilized, along with regularization, to identify the Winning Lottery Ticket features within the network. We speculate that an iterative approach, similar to the one described by Frankle and Carbin [76] but focused on nodes rather than links, may be feasible, but we leave this deepening for future investigation.

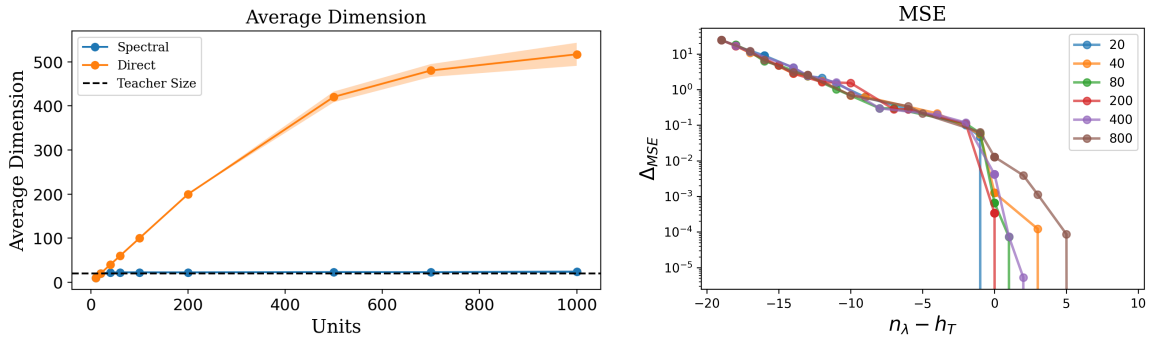


Figure 6.2: (Left) Average dimension of the first hidden layer after the nodes associated to zero values of  $\mathcal{W}$  or  $\mathcal{L}$  have been removed. The average is taken across the 30 trials and the shaded region refer to one standard deviation. (Right) The deviation of the mean squared error (MSE) is plotted against  $n_\lambda - n_T$ . Here  $n_\lambda$  stands for the number of neurons that are left after pruning, while  $n_T$  refer to the actual size of the teacher. A phase transition-like behaviour occurs when the size of the first hidden layer of the student matches that of the teacher network. The vertical scale is logarithmic.

To ensure that the achieved conclusions are not influenced by the size of the second hidden layer, various student configurations have been tested, yielding equivalent results. Specifically, the size of the retrieved computational core remains constant, across different initial choices of the variable hidden layer size. The residual size reflects the inherent complexity of the teacher. As such, it could be thus different from the size of the teacher’s homologous layer.

To provide a complementary view, we also estimate the deviation of the mean squared error (MSE) computed after pruning (i.e. upon removing unessential nodes as ranked via  $\mathcal{L}$ ) from the corresponding value recovered at the end of training, for the full network. This latter quantity is plotted in the right panel of Figure 6.2 against  $n_\lambda - n_T$  where  $n_\lambda$  represents the number of residual neurons and  $n_T$  refers to the number of nodes in the teacher (here  $h_T = 20$ ). The invariant core exhibits remarkable robustness when neurons corresponding to low  $\mathcal{L}$  values are removed. However, as soon as the minimal bulk is perturbed, the MSE deviation starts growing regardless of the initial size of the layer, thereby revealing a universal scaling profile with respect to  $h$ . More specifically, the deviation follows an approximately exponential decrease, and the curve  $\Delta_{MSE}$  vs.  $n_\lambda - h_T$  exhibits a critical point at  $n_\lambda - h_T = 0$ , where a sudden

change in the generalization performance of  $S_\lambda$  is observed.

## 6.4 Other datasets

In order to extend the validity of our results to a less artificial scenario the same analysis has been carried out for various datasets: Shuffled MNIST, Shuffled Fashion MNIST (Standard MNIST datasets but with shuffled pixels), California Housing [77], and a more intricate Teacher structure. In the latter, the size of the first hidden layer differs from that of the second, resulting in a more heterogeneous and realistic structure. The student structure remains consistent with the dimension of the second hidden layer set to 50. In this context, the performance metrics (in terms of accuracy and loss function) for both the spectral Student and the traditional student are comparable across all datasets. The findings presented in Figure 6.3 align with those reported for the simpler scenario presented above: the spectral regularization is capable of finding a computational core (panels  $a - d$ ) whose MSE behaviour, after perturbation, is independent from the size of the initial layer (panels  $e - h$ )



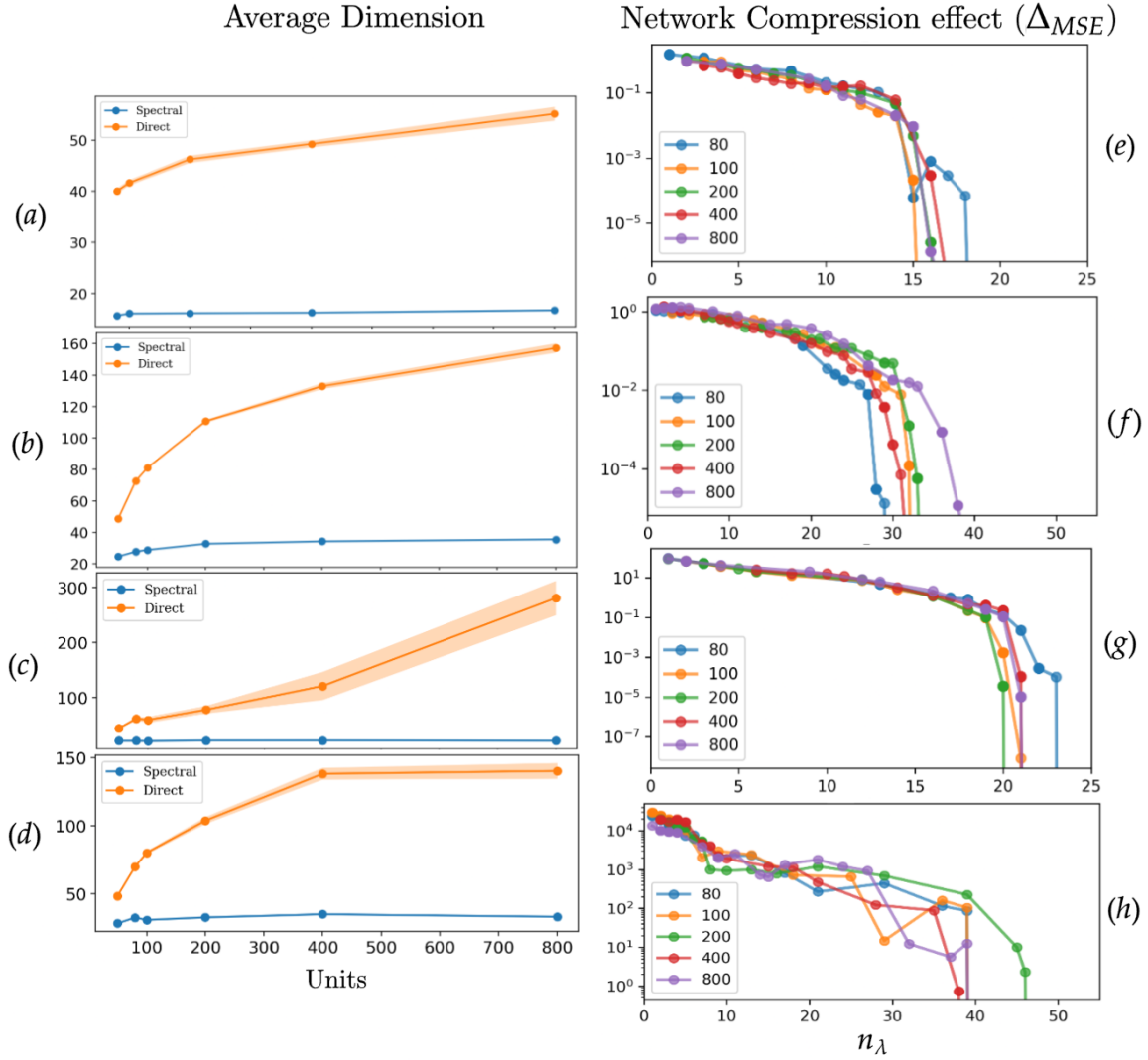
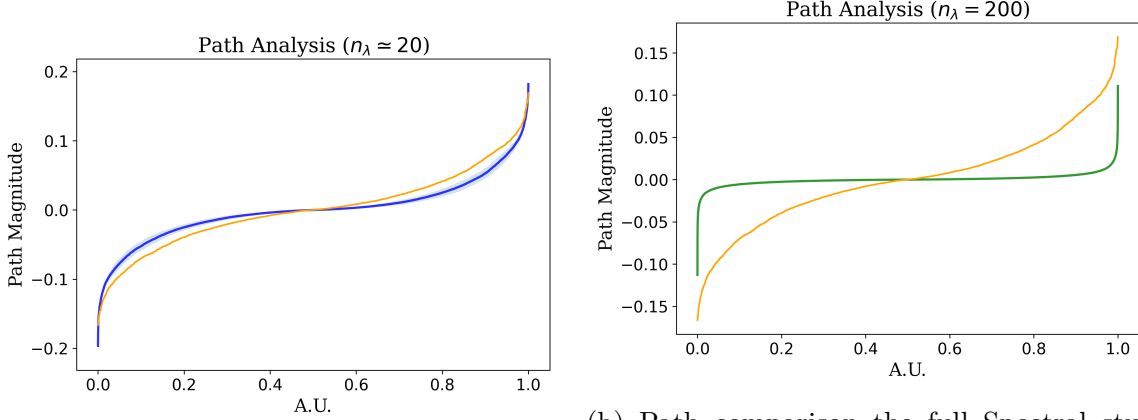


Figure 6.3: Average dimension of the hidden layer (a-d) and effect on the  $\Delta_{MSE}$  (e-f) using the node removal strategy mentioned above. The dataset presented are: Shuffled Fashion MNIST (panels *a* and *e*), Shuffled MNIST (*b* and *f*), two hidden layer teacher but with heterogeneous hidden size (20-40-20-1) (panels *c* and *g*), and California Housing (on panels *d* and *h*). In this case no reference dimension is shown. Regarding the Average dimension of the hidden layer the accuracy and loss of the Spectral and Classical students are the same for each size (within the statistical error, namely the one standard deviation variance across the 30 trials done for each 'Units' value).

It is important to emphasize that the structure of the conventionally parametrized network is fundamentally different, and due to its sparsity, it is impossible to conduct a similar analysis for extracting a compact invariant subnetwork. After applying the node removal strategy to  $S_\lambda$ , we can further analyze the relationship between the pruned student network and the teacher network. Specifically, we investigate whether any properties of the teacher, other than its layer size, can be inferred from the analysis of the invariant core within the student network  $S_\lambda$ . We focus on a natural property given the feedforward structure of both networks: the path magnitude.



(a) Path comparison between compressed (Pruned) Spectral student and the teacher.

(b) Path comparison the full Spectral student, the full Direct space trained and the teacher.

Figure 6.4: Comparison of the path magnitude between the input and second hidden layer of the teacher, displayed in orange, and those obtained for every other case study here considered. The first hidden layer in the student network is set to 200 but equivalent results hold for every other  $h \in \mathcal{I}$ . In panel (a) the first hidden Spectral layer is compressed by removing the nodes that populate with the peak of the normalized distribution of  $\mathcal{L}$  reaching  $n_\lambda = 20$  neurons. In panel (b) no compression is done. As can be seen the correct distribution cannot be retrieved via the usual Direct approach.

## 6.5 Linear Core

To examine the magnitude of each path in the compacted network from a given input node to a specific neuron in the second hidden layer, we construct the expression:

$$\tilde{w}_{i_0, i_1}^{(1)} \cdot \tilde{w}_{i_1, i_2}^{(2)} = \Gamma_{i_0, i_2}^{i_1}, \quad \forall i_0 \in 1 \dots N_0 = 10, i_1 \in 1 \dots h, i_2 \in 1 \dots N_2 = 20 \quad (6.5)$$

Here,  $\tilde{w}_{i_0, i_1}^{(1)}$  represents the weight between input node  $i_0$  and hidden layer neuron  $i_1$ , and  $\tilde{w}_{i_1, i_2}^{(2)}$  represents the weight between hidden layer neuron  $i_1$  and second hidden layer neuron  $i_2$ .  $\Gamma_{i_0, i_2}^{i_1}$  then corresponds to the path from  $i_0$  to  $i_2$  passing through neuron  $i_1$ .

To account for the large amount of permutation invariance in the problem, we transform the tensor  $\Gamma_{i_0, i_2}^{i_1}$  into a vector  $\bar{\Gamma}$  of size  $N_{tot} = N_0 \cdot N_1 + N_1 \cdot N_2$ , where the components  $\Gamma_i$  range from 0 to  $N_{tot}$ .

Next, we sort the components of the vector in ascending order and plot them for the teacher network in orange, and for the student network in blue and green (Figure 6.4). The values are plotted with respect to fraction of the vector components obtained by mapping the index of the sorted vector between zero and one, enabling a meaningful comparison between the networks (due to different number of nodes). Upon visual inspection of panel (a) in Figure 6.4, it is evident that the combined effect of spectral parametrization and regularization yields the emergence of an invariant core within the network. This core can be easily retrieved and exhibits a path structure that closely aligns with that of the teacher network.

In contrast, when analyzing the sparse structure of the conventionally trained and  $L_2$  regularized network, a significantly different path structure is seen, as depicted in panel (b) of Figure 6.4. Specifically, the edges of the path magnitudes differ, as does the overall distribution.

## 6.6 Conclusions

In summary, our study illustrates the advantages of a novel approach to Deep Neural Networks (DNNs) training which is anchored on the spectral domain. The eigenvector and eigenvalues of the underlying linear transfer operators are the target of the optimization. The proposed method weights feature relevance in each layer through a dedicated indicator which scales as the eigenvalues. When regularized, this latter quantity enables us to identify a minimal, efficient subnetwork within the original structure that retains full network performance upon pruning. Tested using the teacher-student paradigm, we found that this reduced network recovers the same topological features as the original teacher network. This approach outperforms conventional weight decay and classical parametrization, opening up new possibilities for the optimization and scalability of DNNs especially when working in the over-parametrized regime.

After discussing how to identify important substructures within feedforward neural networks using spectral tools, we are now moving to a new but related topic. These spectral tools have given us the capability to pinpoint information-rich parts within networks, and as we will see, their uses go beyond merely understanding the structures of these networks.

In the next chapter, we will apply spectral decomposition in a new way: for *dynamical automated classification*. Unlike our previous focus, which was mainly on imitating a known dataset learning a function, we will now use this technique to sort data *dynamically*. In our new model, the Recurrent Spectral Network (RSN), we will employ spectral decomposition to guide data flow based on specific attractors. These attractors are influenced by the decomposition of linear operators, which are essentially the driving force behind data movement in the now-recurrent network.

We will explore this novel and rich learning paradigm describing on how spectral elements like memory kernels, attractors, and decomposition come together to create a new, efficient and more interpretable classification model.



# Chapter 7

## Recurrent Spectral Networks

### 7.1 Classification as a dynamical system

Delving into the principles of the spectral methodology, we will now propose a radically novel approach to computational machine learning which is deeply rooted into the theory of discrete dynamical systems. In a nutshell, the incoming signal is processed by successive iterations across the very same constellation of nodes. The links, and thus the topology of the ensuing network, are fixed and shaped under the spectral paradigm, upon optimization at a given number of iterations. Non-linearities acting on the nodes are imposed a priori or, conversely, learned self-consistently via an apposite deep neural network, which is embedded into the cost function. In either settings, non-linear terms acting in real space at the nodes locations, are forced to vanish asymptotically, iteration after iteration, in such a way that the dynamics eventually turns purely linear. The linear operator, mirroring the processing network, possesses a high dimensional attracting linear manifold spanned by the eigenvectors associated to the eigenvalues equal to one. These latter come in a number that matches the classes to be eventually categorised. A suitable non linear spectral filter is enforced in the loss function to project the ensuing direction along a given eigenvector, assumed as the destination target of a class of homologous entities and selected from those displaying unitary eigenvalues <sup>1</sup>. Stated differently, the classification is accomplished when the processed output - approximately - aligns along a specific direction in dual space, instead of turning active a single node in direct space, as customarily done. This formulation yields a rather natural interpretation of the classifier operational mode: non-linearities, acting at the early stages of the dynamical evolution, drive the discrete dynamical system towards distinct effective stationary equilibria, self-consistently sculpted across the learning scheme and associated to different classes

---

<sup>1</sup>In principle, the system could eventually align along any direction in the manifold spanned by the eigenvectors (of the linear operator) relative to unit eigenvalues. Indeed the learning process, as encoded in the chosen loss function, forces the system to align (as much as possible) along a specific direction - a given eigenvectors selected from those that are associated to eigenvalues identically equal to one. The effectiveness of the procedure is confirmed by a posteriori inspection, as we shall discuss in the following. The proposed method proves indeed remarkably successfully beyond the toy model setting investigated for pedagogical reasons and against classical benchmark datasets. The approximate alignment along the target direction can be made exact by a non linear projection filter that singles out the most prominent among residual directions, in reciprocal space at the time of decision.

of supplied items. Delineating the non-trivial contours that separate the inspected classes in the input space constitutes the tangible outcome of the learning scheme. Remarkably, the trained dynamical system can be iterated forward in time, beyond the limited horizon of the learning procedure: the ability of classifying stays unchanged. The eigenvectors associated to eigenvalues equal to one, are hence veritable memory kernels where the information is kept stored. We name Recurrent Spectral Network (RSN) our novel approach to automated classification via sculpting the attracting invariant subspace of a discrete dynamical map.

Points of connections are found with the framework of reservoir computing. In this latter case, input signals are mapped into higher dimensional computational spaces through the dynamics of a fixed, non-linear system termed reservoir [78]–[80]. Within the RSN, the bulk model is not fixed but self-consistently tailored to the assigned task.

A straightforward variant of the RSN recipe, which accounts for quasi-orthogonal eigen-directions for each processed task, can be also introduced. This latter enables for the sequential handling of different datasets. In simple terms, an artificial computing unit can be assembled which keeps memory of a task, for which it was initially trained, while being exposed to another training session, with an independent dataset to be processed. This is at present arduous with standard approaches to deep learning, as the second learning stage causes the so-called catastrophic forgetting taking over any form of digital consciousness inherited from the first [81]–[83]. Few attempts have been so far reported which aim at overcoming this limitation [84]–[86].

The Chapter is organized as follows. In the next Section we introduce the mathematical notation and the relevant model setting. Then, in the subsequent Section, we will turn to considering a simple example of a dataset defined in  $\mathbb{R}^2$  that will prove useful for clarifying the essence of the proposed methodology. In particular we will show, that the system can effectively trace the boundaries that non-linearly separate different classes within a given datasets. Each class is evolved toward a distinct target, that we identify with a specific direction of the attracting subspace possessed by the underlying linear system. Further, we will proceed by applying the proposed technique to the celebrated MNIST dataset [30]. We will also show that the RSN can also handle multiple datasets with a modest drop in the peak accuracy, and following sequential stages of learning. Finally, we will sum up and draw our conclusions.

## 7.2 The mathematical foundation

Consider  $N$  isolated nodes. Our aim is to assign weighted links among the latter, in such a way that the ensuing network can cope with the assigned task, as e.g., classification of different items in distinct categories. Here,  $N$  can coincide with the number of input variables (e.g., the pixels of a supplied image): in this case, the nodes where reading is performed match the units where calculations are carried out. This is at variance with usual feedforward deep neural networks, where the information to be processed flows from the input to the output, the collection of computing neurons growing with the number of layers that define the underlying architecture. Working within the proposed framework, the topology of the network will unfold as an emerging byproduct of the optimization procedure. As we shall discuss,  $N$  can be larger than the characteristic dimension of the input data, a setting that we will specifically assume

when dealing with the problem of sequential learning, with dedicated memory kernels.

Denote by  $\vec{x}^{(0)}$  the input vector, made of  $N$  entries organized in a column. The idea that we shall hereafter develop is to set up a recursive scheme, the Recurrent Spectral Network (RSN), that takes  $\vec{x}^{(0)}$  as the initial condition and transforms it via successive iterations into a stationary stable output. To stress that the activity vector always belongs to the same space we have decided to keep the ‘top arrow’ notation for vectors in the whole Chapter; the reader shall remind also that, in this context, the apex  $\cdot^{(i)}$  stands for the iteration  $i$  instead of the layer  $i$ . Once the stable output is reached, it should somehow reflect the specific traits of the input items, as identified self-consistently upon dedicated training sessions. Different objects should eventually align along distinct directions of the attracting manifold, depending on the category of specific pertinence. Stated differently, the multidimensional space where the examined objects belong to gets partitioned in mutually exclusive portions, as tailored by suited non-linearities, each associated to a definite asymptotic destination. In the following, we shall label with  $n$  the number of independent target directions, namely the number of independent classes in which the inspected dataset can be eventually partitioned.

Assume  $\vec{x}^{(k)}$  to represent the image of the input vector  $\vec{x}^{(1)}$  after  $k$  application of the iterative scheme. Then:

$$\vec{x}^{(k+1)} = \vec{f}_k(\mathbf{A}\vec{x}^{(k)}) \quad (7.1)$$

where  $\mathbf{A}$  is a  $N \times N$  weighed adjacency matrix that defines the patterns of interactions among nodes;  $\vec{f}_k(\cdot)$  is a non-linear ( $N$ - dimensional) function that depends on the iteration parameter  $k$  and which acts at the level of individual nodes. We require in particular  $\lim_{k \rightarrow \infty} \vec{f}_k \rightarrow \vec{\mathbb{I}} \equiv (\mathbb{I}, \mathbb{I}, \dots, \mathbb{I})^T$ , in such a way that, for large enough  $k$ , the system approximately follows a linear update rule. This is achieved by setting:

$$\vec{f}_k(\cdot) = \vec{\mathbb{I}} + \frac{\vec{g}(\cdot)}{k^\gamma} \quad (7.2)$$

where  $\vec{g}(\cdot)$  is a non-linear function which can be imposed a priori or determined self-consistently via a neural network regression model and  $\gamma$  is a parameter that can be freely adjusted (here we chose to set  $\gamma = 1.5$ ). Focus now on the linear component of the dynamics, as encapsulated in matrix  $\mathbf{A}$ , which takes over for sufficiently large  $k$ . We cast in particular  $\mathbf{A} = \Phi \mathbf{\Lambda} (\Phi)^{-1}$ , by invoking spectral decomposition. Here,  $\mathbf{\Lambda}$  is the diagonal matrix of the eigenvalues  $(\lambda_1, \lambda_2, \dots, \lambda_N)$ . Working in the spectral domain enables us to enforce  $n$ -dimensional attracting subspace. To this end, we impose  $\lambda_1 = \lambda_2 = \dots = \lambda_n = 1$ , and assume  $|\lambda_i| < 1$  for  $i > n$ . These latter  $N - n$  quantities are among the target of the optimization scheme. Moreover, we assume  $\vec{\phi}_1, \vec{\phi}_2, \dots, \vec{\phi}_n$ , namely the eigenvectors relative to the eigenvalues identically equal to one, to identify frozen linearly independent directions of the embedding  $N$ -dimensional space. The remaining eigenvectors ( $\vec{\phi}_i$ , with  $i > n$ , relative to eigenvalues  $\lambda_i$ ) can be freely adjusted, so contributing with a total of  $(N - n) \times N$  tunable parameters to the optimization scheme. When  $k \gg 1$ , non-linear terms fade away and the iterative scheme converges to a linear map,  $\vec{x}^{(k+1)} \simeq \mathbf{A}\vec{x}^{(k)}$ .

By definition,  $\vec{\phi}_i$ , with  $i \leq n$  are stationary solutions of the above system. This latter is hence associated with a high dimensional attracting invariant manifold: any linear combination of  $\vec{\phi}_i$  with  $i \leq n$  is in fact a stationary solution of the linear dynamics that is approached by the examined non linear system, for large enough iterations

$k$ . By acting on the collection of tunable spectral parameters, which ultimately echo on the topology of the network made of  $N$  computing nodes, and by exploiting the non-linearities that act over a finite transient, we aim at steering different input objects toward distinct target solutions, which can be stably maintained beyond the limited horizon of the performed training. To rephrase in words, we postulate that any generically complex classification task is eventually amenable to a multi-dimensional linear problem, with properly tuned interactions strengths and provided non-linearities, imposed or self-consistently learned, are made to initially deform the features landscape.

To implement the learning scheme on these basis, we consider  $\vec{x}^{(\bar{k})}$ , the image on the output layer of the input vector  $\vec{x}^{(0)}$  after  $\bar{k}$  iterations of the iterative algorithm, where  $\bar{k}$  is sufficiently large for the linear approximation to hold true. Then, we calculate  $\vec{c}_{\bar{k}} = (\Phi)^{-1} \vec{x}^{(\bar{k})}$ : the  $i$ -th element  $(\vec{c}_{\bar{k}})_i$  represents the projection of  $\vec{x}^{(\bar{k})}$  along the eigen-direction  $\vec{\phi}_i$ . Each element of the training set is associated to a label  $\ell \leq n$  to identify the category to which  $\vec{x}^{(0)}$  belongs to. Then, an optimization is carried out which seeks at minimizing the squared distance of  $\vec{c}_{\bar{k}}$  (that implicitly depend on the training parameters) with a target  $n$ -dimensional column vector  $\vec{c}_{\ell}$ , made by zeroes except for the element in position  $\ell$  which is set to unit. In such a way, we require that after sufficiently many iterations the dynamical map aligns (as much as possible) along the direction  $\vec{\phi}_{\ell}$ , where  $\ell$  identifies the class to which the supplied entry refers. Different initial conditions, decorated with their reference labels pointing to one of the  $n$  classes, are forced (by a proper use of the non-linearities, as vehiculated by the network arrangement) to yield different asymptotic equilibria, which approximately align along distinct directions in reciprocal space. A perfect alignment along the eigenmodes that flag distinct classes can be eventually forced by performing a projection along the most represented direction, at the end of the iterative update.

Operatively, we begin by initializing the trainable portion of the eigenvectors matrix  $\Phi$  with a random uniform distribution of the assigned entries. Similarly, for the  $N - n$  trainable eigenvalues that enter the definition of matrix  $\Lambda$ . Then, we define a global model (via Tensorflow [87]) that implements a chain of successive applications of the linear mapping  $\mathbf{A}$ . Each linear transfer is followed by the application of the non-linear filter as specified by equation (7.2), which acts at the nodes location. Matrix  $\mathbf{A}$  is written in terms of its spectral decomposition by composing together the three matrices  $(\Phi)^{-1}$ ,  $\Lambda$  and  $\Phi$ , as introduced above. The number of iterations is set to  $\bar{k}$ , a parameter supplied as an input. After iteration  $\bar{k}$ , we apply one more time matrix  $(\Phi)^{-1}$  to obtain the coefficients  $\vec{c}_{\bar{k}}$  that enter the definition of the loss function. The trainable weights of the model are updated according to the gradient descent rule, the loss function gradients being estimated via a standard backpropagation algorithm.

In the following Section, to challenge the effectiveness of the proposed recipe, we set to study a simple dataset defined in  $\mathbb{R}^2$ , which bears pedagogical interest. We will then turn, in a subsequent Section, to examining the ability of the RSN methodologies to cope with a standard datasets of image.

### 7.3 Testing RSN: a simple dataset in $\mathbb{R}^2$

As mentioned above, we aim at testing the RSN as outlined above against a simple dataset, created for this specific purpose. The goals are twofolds. On the one side,



we wish to provide the first consistent implementation of the procedure, by showing that a dynamical system can be trained which preserves its ability to discern beyond the horizon of the training (as instead it is the case for conventional recurrent neural networks). This is an indirect mark of the imposed convergence towards an asymptotic equilibrium, inherent to the dynamical scheme, which flags the class to be identified. Then, we shall convincingly demonstrate that classification by RSN amounts to segmenting the space of the initial conditions in disconnected domains, each pointing to a distinct asymptotic direction, within the invariant attracting manifold. Indeed, the trained map will make a single target mode, representative of the processed class, to stand out as compared to the other. The degree of alignment as observed empirically improves with the complexity of the explored dataset, as we shall remark in the following section. A perfect alignment can be forced by means of a suitable non-linearity that implements a punctual projection along the most represented direction, at final iteration.

The dataset that we shall here consider as a proof of concept is composed by two sets of points, laying on the plane. The points falling inside the unitary circle, centred at the origin, define the first class (displayed in yellow, in Figure 7.1). Those situated outside the circle and inside a square domain of linear width  $L = \sqrt{2\pi}$ , contribute to the second reservoir of datapoints (shown in blue, in Figure 7.1). The size of the square has been chosen in such a way that the surface of the two regions where the dataset insists is equal. The two sets are divided by a non-linear boundary that coincides with the perimeter of the unitary circle. Our objective is to train a RSN, following the prescriptions of the preceding Section, so as to associate any given point - randomly generated to belong to the square domain of width  $L$  - to its reference portion, as introduced above.

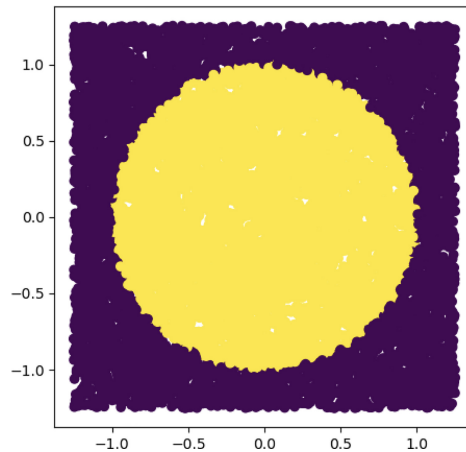


Figure 7.1: The dataset used as a validation test for the RSN scheme. Points populate two different regions, of equal relevance, separated by a sharp non-linear boundary, which we identify as the unitary circle.

For the sake of definiteness we cast  $N = 10$ . Every point of coordinates  $(x, y)$  (constrained so as to fall inside the square of linear size  $L$ ) yields an initial condition for the RSN that we wish at training, i.e.  $(\vec{x}^{(1)}) = (x, y, 0, 0, 0, 0, 0, 0, 0, 0)$ . During the training stage, we generate a sufficiently large reservoir of  $(M)$  points, each complemented with a scalar label that specifies the class, or domain, where the corresponding point

falls. The first two eigenvalues of  $\mathbf{\Lambda}$  are set to unit and the corresponding eigenvectors, respectively  $\vec{\phi}_1$  and  $\vec{\phi}_2$ , are fixed and identify randomly selected (linearly independent) directions in  $\mathbb{R}^{10}$ . The eigenvalues  $\lambda_i$ , as well as the entries of the vectors  $\vec{\phi}_i$ , for  $i > 2$ , contribute to the pool of parameters that one can freely adjust during optimization. Moreover, and to test the method in its general formulation, we do not impose a priori the non-linear function  $g(\cdot)$  (the very same function for each node of the RSN). Rather, we represent  $g(\cdot)$  as a two layered neural network, whose parameters are to be self-consistently adjusted during optimisation. Each of these latter layers is made of 30 neurons and nodes are entitled with a tanh activation function. We label with  $\bar{k}$  the number of iterations of the RSN, assumed during training. Recall that we will also be interested in assessing the behavior of the fully trained systems for  $k > \bar{k}$ . In the following  $\bar{k} = 60$ . The number of epochs is set to 200 and the Early Stopping technique has been employed.

In Figure 7.2, the test-accuracy and the corresponding loss are plotted for  $k < \bar{k}$  and for  $\bar{k} < k < 100$ . As it can be visually appreciated, the accuracy (and the loss) is stable for  $k > \bar{k}$ , i.e., when extending the RSN beyond the iteration number assumed for training.

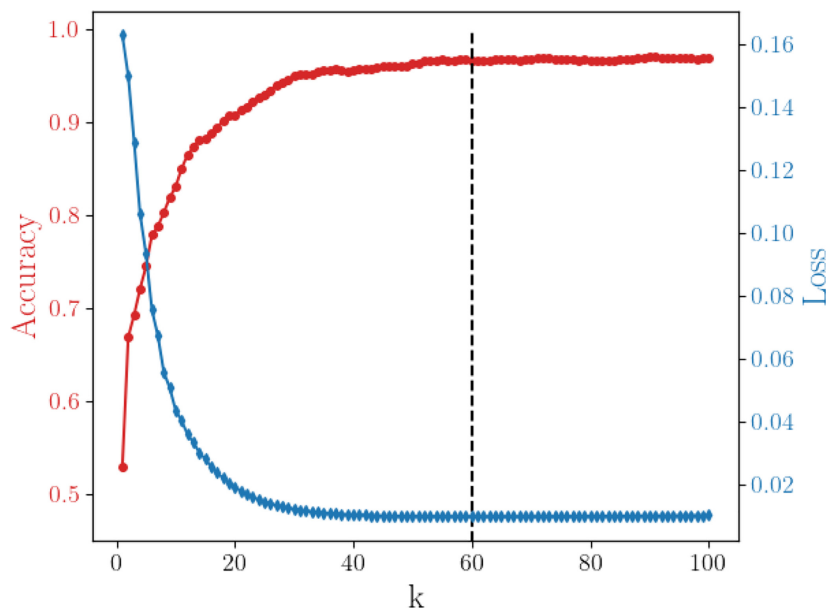


Figure 7.2: Accuracy (in blue) and loss (in red) against the iteration  $k$ , for a trained RSN with  $\bar{k} = 60$  (vertical dashed line). Data refer to just one realization of the training procedure.

The trained RSN classifies points  $(x, y) \in \mathbb{R}^2$ , provided as an input, by generating a late time output in  $\mathbb{R}^{10}$  which tentatively aligns along different target directions: points in the plane contained within the unitary circle with center in the origin, should predominantly activate the spectral mode  $\vec{\phi}_1$ . In this case,  $c_1$  is thus expected to stand out, as compared to all others coefficients, after sufficiently many iterations. At variance, points falling outside the unitary circle are dynamically driven towards a final equilibrium which selectively favours the eigen-direction  $\vec{\phi}_2$ . The coefficient  $c_2$  should therefore prevail over the others. This scenario is confirmed by inspection of Figure 7.3, where  $c_1$  and  $c_2$  are plotted against the iteration number for data points falling

respectively inside (top panel) and outside (lower panel) the unitary circle. Different classes are hence flagging distinct solutions, as stipulated a priori. It is worth recalling that any direction obtained as a linear combination of  $\vec{\phi}_i$  with  $i = 1, 2$ , is also, by construction, a stationary solution of the RSN. This is why a residual activation of the other modes - those relative to eigenvalues one but different from that identified as the target for the class under scrutiny - can in principle manifest when the RSN is challenged against the test-set. A projection along the most represented eigen-mode would enforce a perfect alignment along the sought target direction, with no impact on the performance of the trained device.

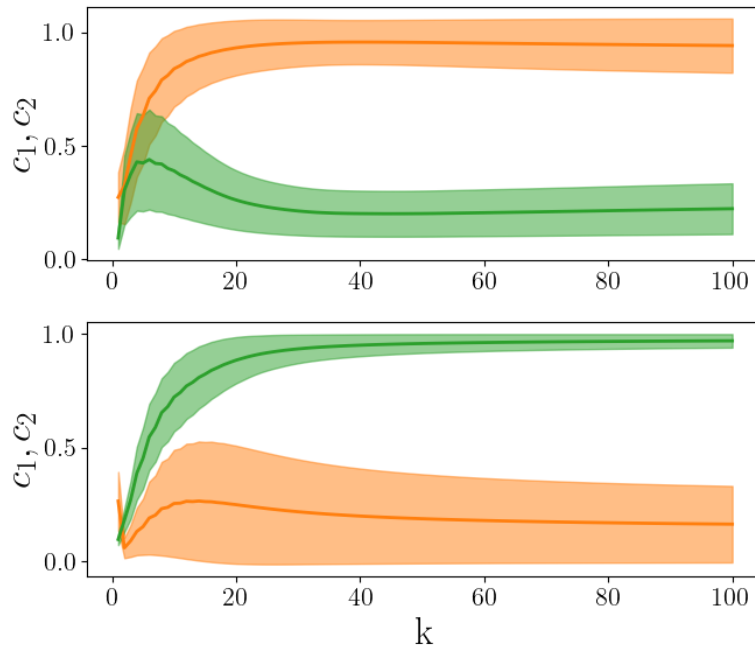


Figure 7.3: The evolution of the coefficients  $c_1$  (orange) and  $c_2$  (green) is plotted for points of the test set positioned respectively inside (top panel) and outside (lower panel) the unitary circle. The shadowed region points to the standard deviation of the collected signal when averaging over the population of supplied input, organized in groups which reflect their domain of pertinence.

The above analysis carried out for a simple benchmark model allowed us to grasp some intuition on the decision making scheme as implemented via the dynamical RSN. Classification is here synonym of convergence towards a specific direction of the attracting manifold. This latter direction is flagged as the destination target of the dynamics, for a homogeneous ensemble of input items. Different classes are hence associated by the RSN to the the eigen-directions of  $\mathbf{A}$  associated to eigenvalues equal to one. For the case at hand, the separatrix between the domains in  $\mathbb{R}^2$  which defines the two classes to be eventually identified matches the unitary circle. To show that the RSN is able to correctly spot out the non-linear separation between the two contiguous domain in  $\mathbb{R}^2$ , and so resolve the distinctive features of the dataset under exam, we consider  $\langle c_i \rangle$ , the average of the  $i$ -th coefficient, across successive phases of the RSN evolution and for different input choices  $(x, y) \in \mathbb{R}^2$ . More specifically,  $\langle c_i \rangle(x, y) = \frac{1}{k_F - k_I} \sum_{k=k_I}^{k_F} (c_k)_i(x, y)$ , for all specific coefficients  $i$  - including those which

will fade away after a transient - and as function of the departure point. In Figure 7.4 the computed coefficients are displayed in the reference plane  $(x, y)$ , with an apposite colorcode and for different choices of  $(k_I, k_F)$ . The panels on the top refers to the initial stages of the evolution ( $k_F = 5, k_I = 1$ ): the separation between the two classes here considered leaves a clear imprint in the distribution of the  $\langle c_i \rangle$  (in particular those with  $i > 2$ ) across  $(x, y)$  (in Figure 7.4 we plot  $\langle c_6 \rangle$ , as an illustrative example as well as  $\overline{\langle c \rangle} = (\sum_{i=3}^{10} \langle c_i \rangle) / 7$ ). An abrupt transition is indeed observed for  $\langle c_i \rangle$ , with  $i > 2$ , when crossing the unitary circle, namely the separatrix between the two adjacent classes that defines our test model. For small  $k_F$  (see top panels), the aforementioned coefficients are in fact remarkably different inside and outside the separatrix. On the other hand, for large  $k_F$ , they are spatially uniformly vanishing (see lower panels, referred to  $k_F = 50, k_I = 40$ ). The patterns associated to  $\langle c_1 \rangle$  and  $\langle c_2 \rangle$  are less clear, at short times, but become evidently distinct when the iterations number is made to increase (see lower panels). Transient modes (those associated to eigenvalues with magnitude smaller than unit) are employed for an early assessment of the examined dataset and get progressively disengaged, at later times. The processed information is in fact passed over the stationary directions, where it is eventually crystallized for classification purposes. Averaged projection coefficients can be employed to trace out, in direct space, key distinctive features that form the basis of decision making. It is here speculated that this is a general attribute of the RSN that can be exported to other, more complex, settings for an a posteriori understanding of the principles that guide artificial reasoning. As a side complement, in Figure 7.5 we depict the non-linear function  $g(\cdot)$  self-consistently obtained via the regression neural model accommodated for in the RSN. In this specific case, it looks like an inverted ReLu (a rectified linear unit) with an additional offset.

Building on these preliminary observations, we will turn in the next Section to considering the application of RSN to MNIST dataset.

## 7.4 Applying RSN to the MNIST dataset

As a further step in the analysis, we apply the RSN to the celebrated and already mentioned MNIST dataset [30]. The images are to be classified in 10 distinct groups (the numbers from 0 to 9). Each element of the training set is associated with an integer label to point to the class to which the selected image belongs to. In the following we will set to train a RSN made by  $N = 784$  nodes: the nodes that receive the information as an input are the very same nodes that carry out the classification, through a dynamical segmentation that originates from the underlying RSN. The network of excitatory (positive weight) or inhibitory (negative weight) interactions is shaped by the optimization scheme which seeks at adjusting the non trivial eigenvalues and eigenvectors of matrix  $\Phi$ . The first 10 eigenvalues are set to unit, as in the spirit of the above, and refer to the eigen-directions employed for discrimination. These latter eigenvectors are a priori fixed and can be engineered so as to return evocative patterns in the space of the inspected images, as we shall demonstrate in the following. Further, we assume  $g(\cdot) = \tanh(\cdot)$ , for the sake of simplicity. Summing up, we can count on a total of  $N \times (N - 10) + (N - 10)$  adjustable parameters to yield a fully trained RSN which can efficiently classify MNIST images.

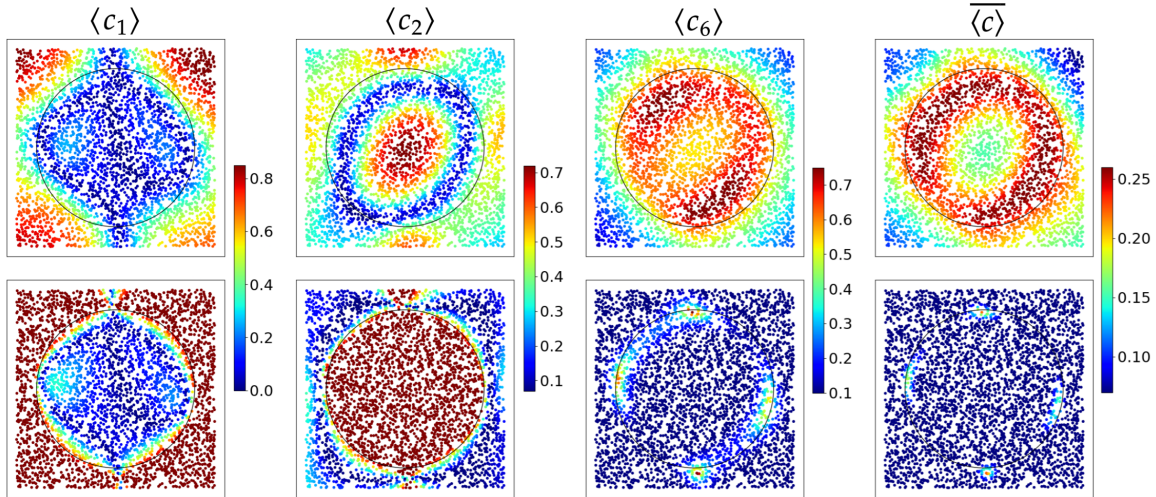


Figure 7.4: The quantities  $\langle c_i \rangle$  for  $i = 1, 2, 6$  and  $\overline{\langle c \rangle}$  are plotted, for different  $(x, y)$ , i.e. moving on the plane of the initial condition. Top panels refer to  $k_F = 5, k_I = 1$ . Lower panel to  $k_F = 50, k_I = 40$ . The separatrix between the two considered classes (which coincides with the unitary circle centered at the origin) is sensed, at short times, by the transient directions. The projections of the generated output along these latter directions fade asymptotically away and the existence of the two classes, as well as the relative domain of definition, leave an imperishable trace in  $\langle c_1 \rangle$  and  $\langle c_2 \rangle$ .

In Figure 7.6, we challenge the ability of the trained RSN to discern images of the test set that respectively corresponds to four (top panel) and five (lower panel). In the former case, as expected,  $c_4$  (depicted in orange) sticks out as the only residual coefficients after sufficiently many iterations of the RSN machinery. All other coefficients (including  $c_5$ , plotted in green) are eventually bound to almost disappear, thus implying that all items belonging to the very same reservoir of images align along a specific direction that can be here traced back to one individual eigen-mode. Remarkably, all coefficients - except for the one that stands for the selected direction - become rapidly negligible. The system is hence directed towards the chosen asymptotic state, without forcing the projection. The shadowed regions that are associated to each average curve refer to the degree of variability inherent to the examined gallery of images. The lower plot in Figure 7.6 shows the response of the RSN when the images displaying a number five are read as an input, and the interpretation is in line with the above. In both cases, the training is performed by arresting the RSN at iteration  $\bar{k} = 10$ : the outcome is however stably maintained well beyond the training horizon, with a modest, although significant in terms of its philosophical implications, improvements in terms of confidence of the assessment. When it comes to the overall performance, the accuracy on the train set is of about 98%, while on the test set the RSN scores 97%, in line with what usually reported when using conventional approaches to machine learning. Figure 7.7 illustrates the progressive convergence of the scheme, for two distinct exemplaries of input images. The RSN converges asymptotically to the deputed solutions, which respectively correspond to eigenvectors  $\vec{\phi}_4$  (left) and  $\vec{\phi}_5$  (right). The entries of these latter eigenvectors are shaped so as to return a stylised version of the digits that define the categories in which the dataset is partitioned. The outcome of

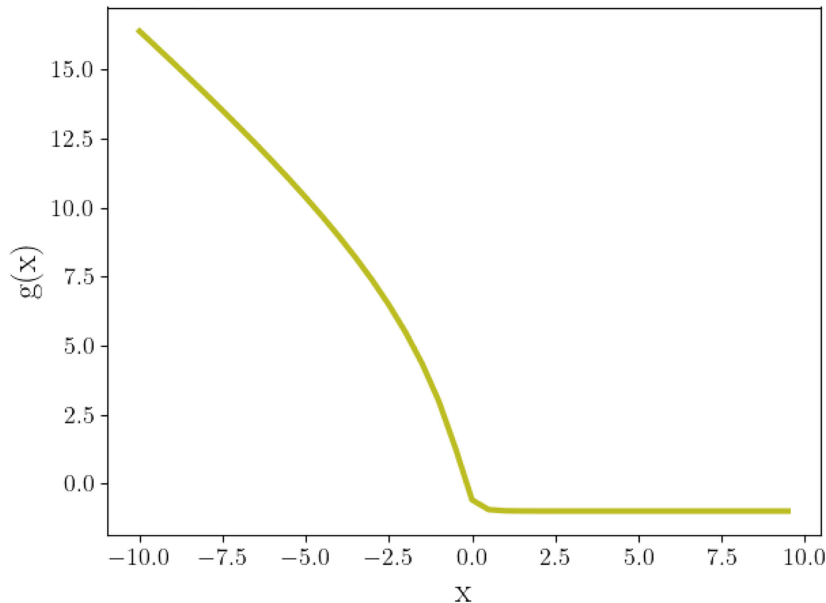


Figure 7.5: The non-linear function  $g(x)$  as obtained from the regression neural model that is associated to each computing neuron of the RSN.

the analysis is hence a stationary stable image, the plastic modulation of the input that is dynamically steered towards a final destination shaped at will by the operator. It is worth stressing that the performances of the method are not affected by the specificity of the target eigenvectors. Stated differently there is no need for them to align with the category that we aim at identifying. Any random eigenvectors would equivalently serve the scope.

#### 7.4.1 Comparison with Recurrent Network

As mentioned earlier, a specific advantage of the RSN model is the ability to keep memory of the final state for  $k > \bar{k}$ . This is a byproduct of the fact that, for sufficiently large times, the non-linear activation terms are virtually silenced and the update rule converges to a simple linear scheme. The dynamics aligns by construction towards stationary directions of the linear mapping, and this makes it possible to operate the RSN for any  $k$  larger than the training horizon  $\bar{k}$ . As a benchmark model, we consider a standard Recurrent Neural Network (RNN) trained in direct space [88]–[90]. The RNN in its simplest version is conceived as a single transfer layer between two adjacent stacks made of  $N = 784$  nodes, iterated  $k$  times (recognition is performed on the first 10 nodes of the final layer). The number of trainable parameters is thus  $N \times N$ , comparable to the number of parameters adjusted by the RSN model. In Figure 7.8, we compare the accuracy measured for the MNIST dataset, for both the RSN and the RNN trained upon completion of iteration  $\bar{k}$ . The accuracy recorded for the RSN (red symbols) converges rapidly and the achieved score is stably maintained for  $k > \bar{k}$  (here  $\bar{k} = 5$ ). Conversely, the RNN (blue symbols) returns its largest accuracy (basically identically to that obtained with the RSN) only for  $k = \bar{k}$ . By taking just one step further (i.e. adding one additional layer to the RNN) is enough to lose predictive power.

As also shown for the case of the simple model discussed in the preceding session,

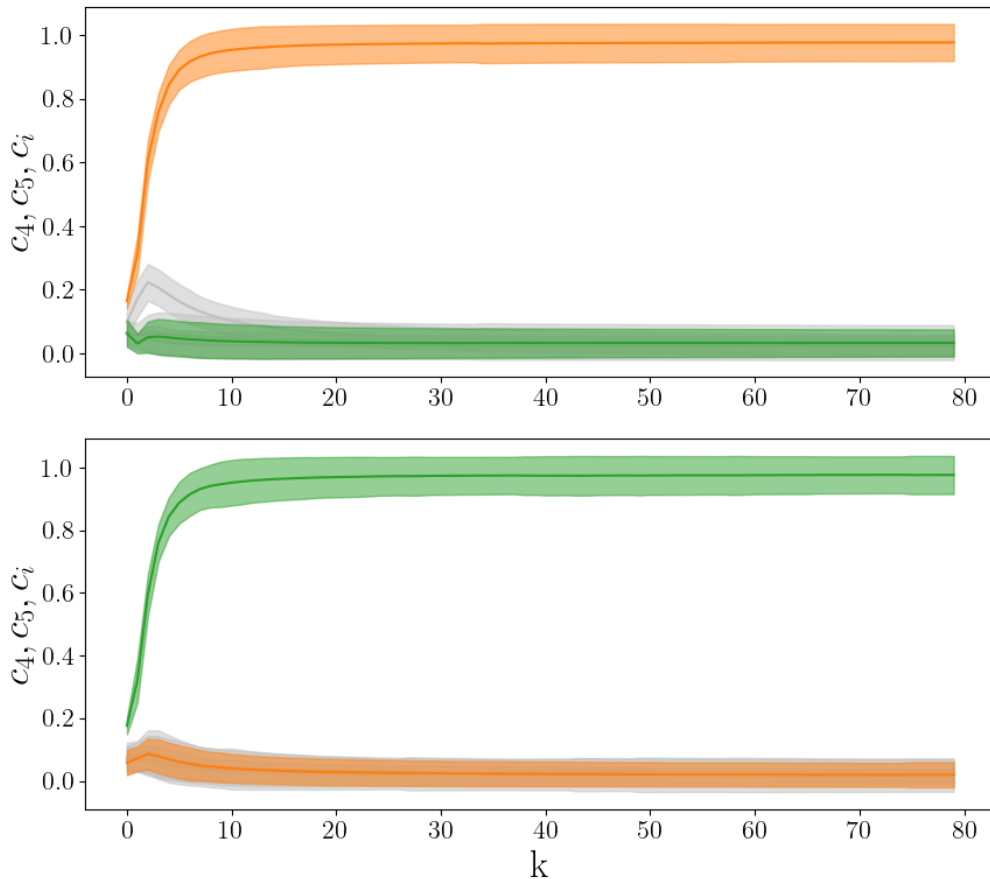


Figure 7.6: Top panel: the full set of handwritten four available in the test set is provided as an input to the trained RSN and the response monitored in terms of the obtained  $c_i$ , with  $i = 1, \dots, 10$ . As expected,  $c_4$  (orange) emerges and converges to unit, for  $k > 10$  ( $\bar{k} = 10$  being the maximum iteration number set during training). All other coefficients, including  $c_5$  (green) disappear. Lower panel: the situation is analogous to that analyzed in the top panel with the notable exception that now handwritten five are analyzed by the RSN. Hence,  $c_5$  (green) converges to unit while,  $c_i$  with  $i \neq 5$  (including  $c_4$ , in orange) fade away. In both cases, the shadowed regions reflect the variability of the images, within any given class of the test set.

there is a progressive tendency to crystallize the final output along the eigen-directions, where recognition is eventually performed. This observation can be made quantitative - see Figure 7.9 - by monitoring the evolution of the coefficients  $c_i$ , as computed from the state vector across successive iterations. In particular, three sets of  $c_i$  are identified: each group clusters together the coefficients associated to eigen-directions relative to eigenvalues that approximately share the same magnitude (a set relative to small eigenvalues, a set relative to larger eigenvalues and the final set of eigenvalues equal to one, i.e., those associated to the eigen-directions where recognition takes place). We evaluate the three sets of coefficients for each image in the test set displaying a four and a five and compute the average distance (square norm) between each set of coefficients, against  $k$ , the iteration of the RSN. The coefficients stemming from the transient modes single out the differences between the analyzed samples, before

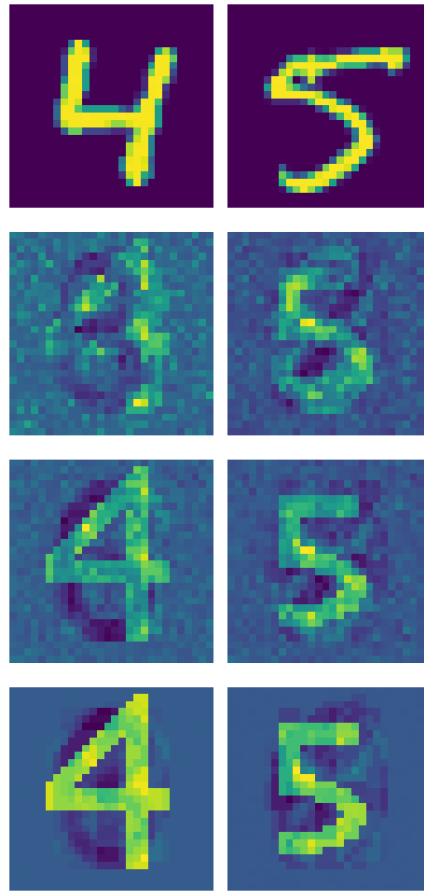


Figure 7.7: In each row we plot the activity on each node of the RSN, at different iterations and for two input numbers that belong to two distinct categories, respectively a four (left) and a five (right), see top panels. After a few iterations the RSN converges asymptotically to the eigenvectors  $\vec{\phi}_4$  (left) and  $\vec{\phi}_5$  (right)) that are triggered by the provided input. Note that the asymptotic solutions can be shaped to manifest as a stylized version of the number to be classified. The more yellow the pixels, the more intense the activity on the associated nodes.

converging to zero when the stationary eigen-modes, inactive at first, get eventually approached

In the next Section we will turn to considering a variant of the RSN which is constructed to yield sequential handling of different datasets, with a long term memory effect. To demonstrate our findings, and as a preliminary proof of concept, we will split MNIST into two distinct, though perfectly balanced, datasets, the first formed by digits from zero to four, and the other populated with the remaining elements, ranging from five to nine.

## 7.5 Sequential Learning

In this Section we will discuss a generalization of the RNS which allows to keep track, to some extent, of a learned task, while dealing with an independent session of training,



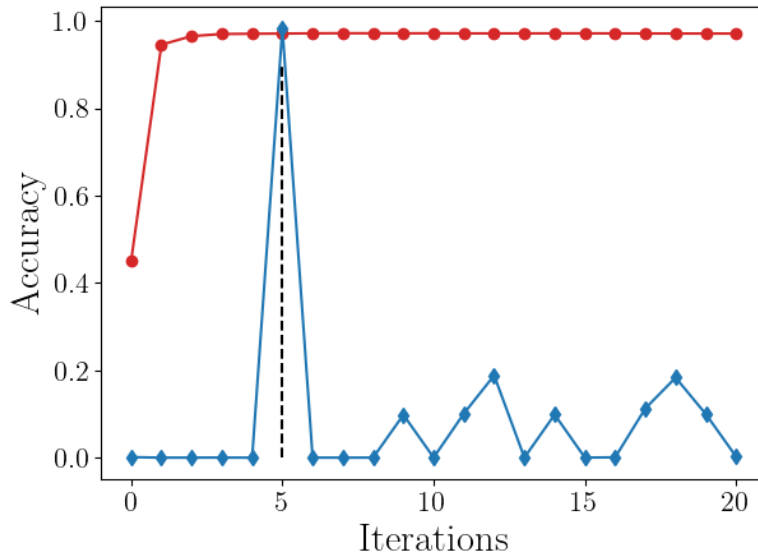


Figure 7.8: Evolution of the accuracy as computed on the test set of the MNIST dataset. Red symbols stand for a RSN model, trained at  $\bar{k} = 5$  (black dashed line); blue symbols refer to a RNN, with  $\bar{k} + 1$  consecutive layers, i.e. with  $\bar{k}$  nested applications of the same  $N \times N$  transfer operator. The RSN quickly converges to the best accuracy, which stays constant for  $k > \bar{k}$ . At variance, the RNN is capable to correctly discriminating the items provided as input entries only punctually, at  $\bar{k} = 5$ . It loses any predictive power for  $k > \bar{k}$ .

on a distinct dataset. To elaborate along these lines, and with the sole aim of providing a preliminary proof of concept of the basic implementation, we shall split the MNIST into two distinct, though balanced datasets. The first will be composed by handwritten digits ranging from zero to four. The remaining images, displaying numbers from five to nine, constitute the second reservoir. We will then train the RSN to classify the images belonging to the first dataset according to the recipe previously presented, namely we fix 5 eigenvalues to 1 and the associated eigenvectors to random and linearly independent directions. Then, the obtained RSN undergoes a second round of training focusing on the images that define the complementary dataset. By assuming sets of quasi-orthogonal<sup>2</sup> eigenvectors with associated memory kernels, yields a fully coupled network, the backbone of the RSN, which is capable to efficiently handle novel tasks while preserving notion of past knowledge. This is at variance of conventional schemes, based on standard deep learning architectures or RNN, which tend to eradicate former imprints by overwriting existing memory slots, as we shall hereafter demonstrate [81]–[83].

### 7.5.1 Quasi-Orthogonal $\Phi$

MNIST images are read as an input by a layer made of  $N_0 = 28 \times 28$ . This information is passed to the  $N$  nodes of the RSN via an all-to-all linear transformation encoded by

<sup>2</sup>As we will explain further this terms means that the off diagonal elements of the matrix  $\Phi^T \Phi$  is a percent of the main diagonal elements

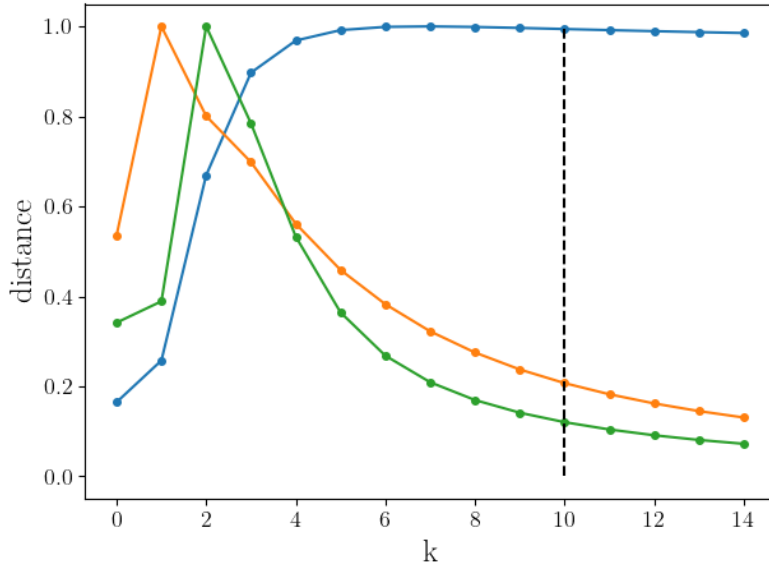


Figure 7.9: Euclidean distance (normalized to its maximum value) between the three sets of coefficients as described in the main body of the Chapter and respectively referred to fours and fives, against the iteration index  $k$ . The orange curve refers to (10) coefficients, associated to modes with small magnitude, as obtained after the training. The green curve is computed by considering the projections along (10) modes with eigenvalues bearing larger absolute values, though smaller than one. The curve depicted in blue refers to the values of the coefficients of the 10 eigen-directions relative to eigenvalues one, where classification is eventually performed. The peak travels horizontally suggesting that the information crawls from the transient towards the stationary modes. The fact that the orange curve seems more persistent than the green at larger  $k$  is just a consequence of the imposed normalization. The vertical dashed line is set at  $\bar{k}$ .

a  $N_0 \times N$  matrix  $\mathbf{w}_0$ , see Figure 7.10. Here,  $\mathbf{w}_0$  is fixed. As such, the entries of  $\mathbf{w}_0$  do not take active part to the optimization process which is instead focused on the RSN component of the dynamics. Further,  $N$  (assumed even, with no loss of generality) can be larger or smaller than  $N_0$  without any limitation whatsoever. We then postulate the following form for matrix  $\Phi$ :

$$\Phi = \begin{pmatrix} \Phi_{11} & \epsilon \Phi_{12} \\ \epsilon \Phi_{21} & \Phi_{22} \end{pmatrix} \quad (7.3)$$

The four blocks  $\Phi_{ij}$ , with  $i, j = 1, 2$  have dimensions  $N/2 \times N/2$ , and comparable norms. The parameter  $\epsilon$  sets the importance of the off-diagonal blocks as compared to those that define the block diagonal terms. In the limiting case  $\epsilon = 0$  the matrix of the eigenvectors is block diagonal. The eigenvectors are hence organized into two distinct ensembles, mutually orthogonal and the corresponding network splits into two disconnected parts. When  $\epsilon \neq 0$  instead the two subparts of the ensuing network get mutually entangled and virtually indistinguishable for a sufficiently large magnitude of the coupling parameter  $\epsilon$ . For  $\epsilon \neq 0$ , though relatively small as we shall assume in the following, the eigenvectors form two quasi-orthogonal blocks. Focus now on

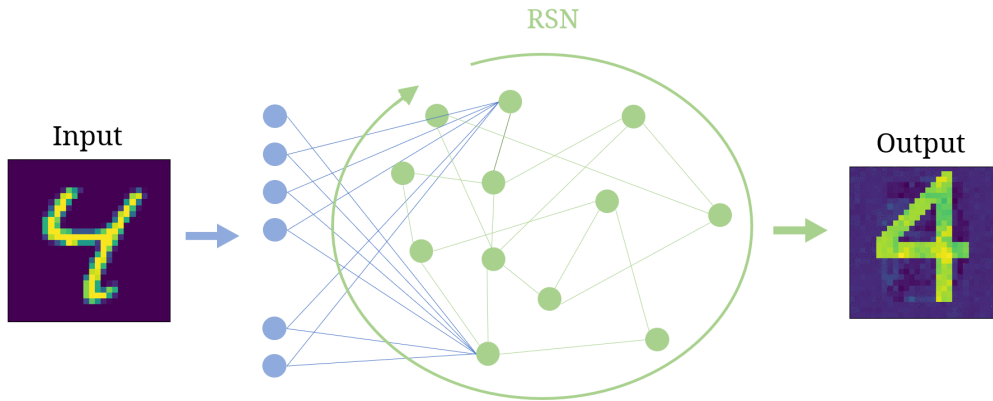


Figure 7.10: A schematic layout of the architecture employed to handle sequential learning. The information stemming from the image presented as an input are passed to the RSN, and therein iteratively elaborated until convergence to the deputed stationary solution (here exemplified as a stylized version of the input number).

the diagonal matrix of the eigenvalues. These are also split into two groups of identical cardinality, which will be eventually structured as follows  $(1, 1, 1, 1, 1, \lambda_6, \dots, \lambda_{N/2})$  and  $(1, 1, 1, 1, 1, \lambda_{N/2+6}, \dots, \lambda_N)$ . Trivial eigenvalues are associated to specific eigen-directions, the target of the RSN, which stay put across optimization. In practice, each eigenvalue equal to unit points to a specific memory slot which can be filled and, at least partially, preserved, across multiple learning stages. Starting from this setting we proceed as follows:

- We set at first to zero the first five eigenvalues belonging to the second group, as identified above. In doing so, we seek at protecting a specific set of memory slots, which should not be contaminated during the first round of training
- We then train the RSN to recognize and correctly classify the first reservoir made of handwritten digits from zero to four, as outlined above. During this operation, the optimization acts on  $\lambda_6 \dots \lambda_{N/2}$  and on (the full set or a limited sub-portion of) the entries of the eigenvectors associated to these latter eigenvalues. Here,  $\epsilon \neq 0$ , which in turn implies that by modulating the entries of the eigenvectors belonging to the first of the two sets, yields an indirect signature on all the inter-nodes weights in direct space. At the end of the optimization, the RSN is capable to correctly classifying analogous images belonging to the test set.
- We then turn to the second round of training by providing to the above RSN (namely, the RSN that has been trained to cope with the first dataset) the elements belonging to the second reservoir of images, those depicting digits ranging from five to nine. The second set of memory slots is turned on, by setting to unit the eigenvalues initialized to be zero: the corresponding eigenvectors define the asymptotic solutions that the trained system should eventually approach. The eigenvalues that identify the target eigen-directions from the preceding training are instead set to zero.

After completion of the optimization, one can check the performance of the RSN, which has been trained across two successive stages, referred to two distinct datasets. To this end we turn on all possible memory slots (trained as follows the above, two steps, procedure): in practice we set to one the eigenvalues relative to the (10) eigen-directions where information is asymptotically conveyed. In Figure 7.11 (top panel), the performance of the RSN, as measured by the reported accuracy, is tested against the epochs of the optimization scheme. The optimization is carried out by assuming  $\bar{k} = 10$  in the RSN, and assuming 100 of epochs for each of the two nested stages of learning. Already after a few epochs the RSN returns a very high accuracy against images of the test set which display digits ranging from zero to four. When the RSN gets also trained on the complementary reservoir of handwritten digits, as follows the sequential scheme highlighted above, it quickly manages to handle the novel task with an adequate success rate, while, at the same time, manifesting a relatively modest drop in performance as referred to the former. Notice that the images, differently from other methods, are supplied as an input with no extra markings, or alert flags, to point to the relevant group of destination patterns. To grasp the interest of the proposed scheme we report in Figure 7.11 the results obtained for a RNN with a number of layers equal to  $\bar{k} + 1$ . As immediately confirmed by visual inspection, any knowledge coming from the first round of training is - almost instantaneously - lost, when the network becomes acquainted with the second task. Similar conclusions (data not shown) are obtained when dealing with a deep neural network, with a standard feedforward architecture [81]–[83]. Summing up, working with a quasi-orthogonal basis, with a set of (almost) mutually exclusive blocks equal to the number of tasks to be eventually handled, yields a RSN which can be sequentially trained, while keeping memory of the previous training sessions. A drop in the recorded accuracy is however found which could be possibly mitigated for increasing RSN size and/or addressing ad hoc solutions that require further investigations, beyond the scope of a mere proof of concept. It is also remarkable that the accuracy displayed against the first dataset, and after the initial sudden jump that follows the second training round, ramps again, epoch after epoch, to align to that referred to the second dataset.

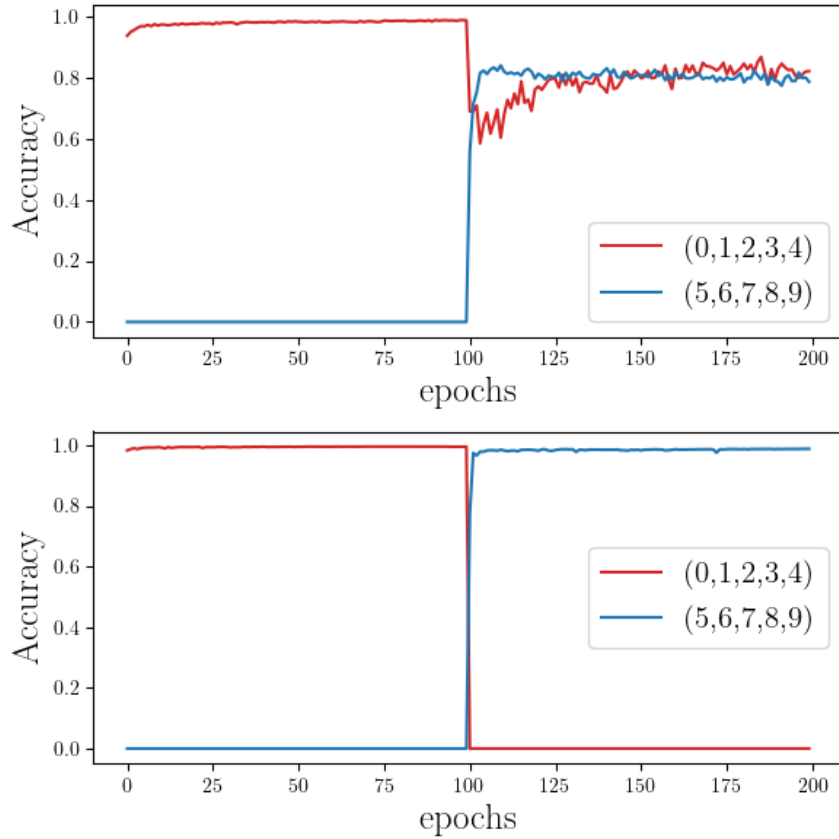


Figure 7.11: Top panel: accuracy against the epochs number for the RSN. The first 100 epochs refer to the RSN confronted with the task of classifying the images of the dataset made of digits from zero to four. Then, the second range of epochs, refers to the RSN while learning to classify numbers from five to nine, after having completed the first stage of training. The accuracy drops but the RSN keeps still memory of the first task, while learning to cope with the second with an almost identical score of reported success. In this specific example, the elements of the off diagonal blocks  $\Phi_{12}$  and  $\Phi_{21}$  are kept fixed, during optimization. Lower panel: sequential learning is ineffective with usual RNN (and standard feedforward deep neural networks, data not shown), since any form of pre-installed knowledge gets washed out during a subsequent, independent, training stage. Here  $\bar{k} = 10$ ,  $\epsilon = 0.25$  and  $N = 1000$ .

## 7.6 Conclusions

In this Chapter we have introduced and tested a novel approach to automated learning, which is rooted in reciprocal space and exploits foundational elements of the theory of discrete dynamical systems. The information under scrutiny is read by a collection of nodes, typically (but not necessarily) the pixels of the image provided as an entry, and further processed by the very same nodes, as follows an iterative update scheme which alternates linear mapping and non-linear filters. Depending on the characteristics of the signal provided as an input, the ensuing dynamics is steered (as a byproduct of the training) towards different asymptotic solutions for the subsequent recognition to take eventually place. The convergence to the asymptotic state is stable by construction,

and the alignment along the selected direction that we demonstrate empirically for a classical benchmark model is guaranteed also when pushing the iterations beyond the limited horizon of the optimization. We have referred to the proposed methodology as to Recurrent Spectral Network RSN, to signify the dynamical nature of the process which is formulated in reciprocal domain.

Neural networks are sometimes called black boxes because it is not immediate to understand how or why they work as well as they do. At variance, the operational mode of a RSN is absolutely transparent and, as such, it could help unveiling the blanked of mystery that surrounds deep learning applications. Indeed, the RSN asymptotically aligns along different directions within the attracting manifold of an underlying - linear - discrete dynamical system. Learning to classify within the RSN amounts to partitioning the high dimensional input space into separated domains, each pointing to a specific stationary eigen-mode of the underlying dynamical system, in its linear approximation. Non-linearities act over a transient and fades eventually away, when the non trivial classification problem has been de facto turned into a linear one.

A variant of the RSN has been also considered which accounts for quasi-orthogonal eigen-directions to carry out a sequential handling of different datasets. In practice, a RSN can be assembled which keeps memory of an initial task, while being subject to another session of training on an independent dataset.

Several directions for further investigations can be outlined. One interesting possibility is to modify the loss function by forcing the contribution at iteration  $k$  to be smaller than that at iteration  $k + 1$ . Preliminary checks shows that the RSN tunes self-consistently its convergence rate, which is hence not a priori imposed as it is here done. It is also tempting to speculate that proceeding along these lines, one could eventually generate a RSN which is capable of improving its accuracy score by iterating further beyond the specific window of training. Another possibility is to introduce apposite frustration mechanisms, which tend to disfavour the accidental convergence towards directions that have been already exploited, when operating with the sequential learning protocol. Also, it would be extremely important to devise other possible strategies, alternative to the one here employed, to structure the eigenvectors matrix for multiple datasets handling.

## 7.7 From low to high order

In this Chapter, our focus revolved around the Recurrent Spectral Network (RSN), a novel strategy in automated classification. There, the spectrum of the adjacency matrix of a network guides the system towards distinct asymptotic stable states whose existence has been engraved in the spectral properties of the adjacency matrix, emphasizing the profound role spectral decomposition plays in defining the activity evolution within a processing network.

As we proceed into the subsequent Chapter, our spectral lens shifts: now focusing on the spectrum of high-order differential operators, specifically, the Dirac operator. As we will discover, indeed, it is possible to define signals not only on the nodes of a network but also in the edges, faces and so on, resulting in a fully alive discrete structure. Those higher order *topological signals*, physically interpreted as fluxes or circulations, can be coupled and the resulting dynamics is deeply dependent on the

spectral structure of the operators used at this purpose. Indeed the Dirac operator allows to let signals defined on structures of different dimension to naturally interact, while the adjacency matrix, or its generalization to higher-order structures, does not immediately possess this property.

While the adjacency matrix spectrum helped us in understanding boundaries and memory kernels in the RSN, the spectrum of high-order differential operators, that will be used as coupling, extends our comprehension of how topological signals interact and evolve. As we are about to see, we will be able to predict the emergent stable state of a dynamical system that couples node defined signals with edge defined fluxes. Again the spectral properties of the linear operators—whether governing discrete or continuous dynamics—critically shape the eventual states making them pivotal quantities for an in-depth comprehension of the system.





# Chapter 8

## Topological Signals

Simplicial complexes are higher-order networks that come with extremely rich and useful structures inherited from discrete topology [91]–[93]. Roughly speaking, a simplicial complex is a topological structure that, besides nodes and links, also contains triangles, i.e., allowing for modelling of three-body interactions, tetrahedra, i.e., four-body interactions, and so on. One can thus consider topological signals defined on nodes and links, but also on higher-order structures [91]. Examples of topological signals are flows associated to links such as synaptic signals between neurons [94], edge signals in large scale brain networks [95], [96], and flows occurring in biological transportation networks [97], [98], in power-grids [99] or in traffic on a road network [100]–[103]. Even higher-order signals, associated to faces, like fluxes, can be taken into account. Moreover edge signals might also represent a number of climate data such as currents in the ocean and velocity of wind that can be projected on a suitable triangulation of the Earth surface [102], [103]. Topological signals can undergo higher-order simplicial synchronization [104]–[110], and higher-order diffusion [106], [111], [112]. Furthermore datasets of topological signals can be treated with topological signal processing [100], [103], [113] and with topological machine learning tools [114]–[117]. Note that this increasing interest in topological signals occurs while the entire field of dynamical processes on simplicial complexes is bursting with significant research activity [118]–[128].

Topological signals of a given dimension can be coupled by the higher-order Laplacians also called the Hodge-Laplacians [91], [129], [130]. However the Dirac operator [131]–[134] is necessary to couple topological signals of different dimension such as interacting signals defined on nodes and links of a network. Interestingly, the Dirac synchronization which stems from the adoption of the Dirac operator to couple topological signals of different dimension, provides a topological and local pathway towards explosive synchronization and rhythmic phases [108].

The Chapter is structured as follows. In the first section we will present the mathematical framework of simplicial complexes and we will emphasize their difference with respect to node signals. We will then introduce the concepts proper of algebraic topology such as Boundary and Coboundary operators, preparing the ground for the definition of the Hodge Laplacian and the Dirac operator: the two most common linear couplings in this field.

## 8.1 Why simplicial complexes

While a comprehensive analysis of the many and different kinds of higher-order interaction is beyond the scope of this thesis, we will detail the assumptions underpinning the use of a simplicial complex and we will differentiate them with other types of popular high-order interaction framework such as hypergraphs.

For the sake of pedagogy, we start by considering the adjacency matrix. In the study of network, the adjacency matrix serves as a critical tool for describing the interactions between arbitrary agents, represented as nodes in a network. These interactions can signify various physical quantities, such as the flow of vehicles or pedestrians on a street, or electrical signals between neurons. Additionally, these matrices can encapsulate more abstract interactions like friendship or animosity between two individuals. At its most basic level, an adjacency matrix can simply denote the presence or absence of an interaction between nodes. In such cases, the matrix is commonly referred to as a "topological adjacency matrix" or a "binarized adjacency matrix," where the elements are restricted to binary values of 1 or 0. These different levels of representation offer a versatile framework for investigating diverse systems, from physical to social, through the lens of network theory. In subsequent sections, we will explore how simplicial complexes enable us to generalize both relational and spatial concepts. Specifically, we can broaden the notion of relationships by introducing multi-body relations among a subset of nodes using simplicial complexes. This approach requires that the presence of any higher-order relation also necessitates the presence of every lower-order relation. To illustrate this concept, consider a three-body relation represented by a two-dimensional simplex. This means not only to deal with three entities (say ABC) related altogether, but also every pairwise relation (AB, BC, and AC) must also exist. This contrasts with hypergraphs where establishing a relation between ABC using a three-dimensional hypernode doesn't necessarily imply lower-order binary relationships.

Because the standard definition of adjacency matrices on network lack the mathematical tools to allow for such definitions the use of simplicial complexes or other high order structures is mandatory to provide a robust framework.

The key idea is to extend the notion of the topological adjacency matrix, defining an equivalent concept that will enable us not only to correctly describe the discrete space that its hosting whatever process we are dealing with, but also to naturally insert the notion of orientation. This concept, deeply rooted to the concept of *boundary*, will enable us to leverage differential geometry and analysis tools in this context. This idea of interlinking signals defined on various spatial dimensions might initially seem unconventional, yet it mirrors concepts familiar in scientific endeavours. Consider the continuity equation,  $\partial_t \rho + \nabla \cdot \vec{J} = 0$ . Here, a scalar quantity (density) interacts with a vector quantity (current) through the differential operator divergence. Think of the density as a node signal, defined on a 0-dimensional object, and the current as a link signal, defined on a 1-dimensional pathway. In this way the scalar or vectorial quantities defined on continuous space can be straightforwardly translated in a discrete domain, enabling de facto differential calculus on graphs.

The initial sections of this Chapter aim to adapt equations like the above for discrete spaces, such as networks. As we are about to discover, shifting the focus toward the spectral properties of the differential operator, which becomes matrices in discrete space, greatly simplifies this analysis.

## 8.2 Geometric viewpoint

An  $n$ -dimensional *simplex* is a set of  $n + 1$  nodes describing a higher-order interaction among  $(n + 1)$  agents or a  $(n + 1)$  dimensional discrete space. In particular a 0-simplex is a node, 1-simplex is a link, a 2-simplex is a triangle and so on. The *faces* of a simplex are the simplices that can be constructed starting from a proper subset of its nodes. For instance the faces of a 2-simplex (i.e., a triangle) are 3 nodes (0-simplices) and 3 links (1-simplices). To simplify the introduction we will start with a geometrical approach, as it is much easier to tackle this subject starting from this viewpoint. In general a simplex  $\sigma_n$  can be geometrically defined as the subset of  $\mathbb{R}^{n+1}$  such that

$$\sigma_n = \left\{ (t_0, \dots, t_n) \in \mathbb{R}^{n+1} \mid \sum_{i=0}^n t_i = 1 \text{ and } t_i \geq 0 \text{ for } i = 0, \dots, n \right\} \quad (8.1)$$

In this way we get the 0 dimensional 0-simplex: the 'point', 1 dimensional 1-simplex, the 'segment', 2 dimensional 2-simplex, the 'triangle', 3 dimensional 3-simplex, the 'tetrahedron' and so on. The simplices constitute the building blocks of simplicial complexes, i.e., a *discrete manifold*. In fact a  $d$ -dimensional simplicial complex  $\mathcal{S}$  is a collection of simplices whose dimension  $n$  is smaller or equal to  $d$ . The intuitive idea is that we can construct a discrete structure whose topology is well defined by glueing together any kind of geometric simplex via its faces. For example we can glue together 2 triangles attaching them via its edges but also via two nodes. This naive way of constructing the simplicial complex is accompanied by a mathematical propriety:  $d$ -dimensional simplicial complex must be closed under the inclusion of the faces of each simplex. In other words, if a triangle is included in the simplicial complex, its nodes and all its links are included in the simplicial complex as well. Moreover if two triangles are glued together via their link also their nodes will be attached. This requirement is central to the understanding of simplicial complexes for two key reasons. First, it solidifies the object as a proper topological 'space' by providing a framework to discuss boundaries. In this context, statements like 'these links form the boundary of this triangle' become meaningful. Second, the interpretation of higher-order interactions that we introduced in the previous section gains validity. Specifically, if a set of three people is considered as higher-order nucleus of interaction, it logically implies that the individual pairs within the set are also interacting.

It's important to note that to properly distinguish a link from a pair of nodes, the link must have an orientation. While this orientation is arbitrary, it serves as an essential aspect of bookkeeping. In a naive sense, a one-dimensional object like a link can support a vector or a flux that is directed from one node to another. This in turn defines what can be considered a 'positive' direction. Referencing to Figure 8.1, the link  $[1, 2]$  assumes a positive orientation when going from node 1 to 2. This ensures that it is a higher-order object with respect to the node (which does not have enough 'space for orientation'). The same idea applies to higher order objects such as the triangle. The latter, should be capable of hosting a face-flux or, equivalently, its boundary should be capable of hosting a *circulating closed flux*.

Therefore, the triangle needs an orientation that is, as in Calculus courses, the orientation of its area in order to be ontologically different from three links. As a pedagogical example we shall refer to Figure 8.1: we could assign the orientation to the triangle  $[1, 2, 3]$  by saying that, indeed, the right hand rule is applied to the path  $1 \rightarrow 2 \rightarrow 3$ ,

as we have written in square brackets. Then, with the right hand rule, this implies that the 'entering the plane' verse is positive. Equivalently we could have said that the triangle is defined as the 2 simplex that has the following closed boundary:  $[[1, 2], [2, 3], [3, 1]]$ .

There are basically three ways of operatively define the simplicial complex:

- Relational: Every set of related nodes is listed as an ordered sequence, usually denoted by using square brackets, remembering that any high order relations implies all the lower order ones.
- Geometric *bottom up*: The discrete manifold is constructed from the lower dimensional entities to the higher dimensional ones. Then the lower entities are labelled and an orientation is induced in all the higher dimensional ones.
- Geometric *top down*: The discrete manifold is constructed from the upper dimensional entities to the higher dimensional ones. Then the upper entities are labelled and an orientation is induced in all the lower dimensional ones thanks to the action of the boundary operator.

In the following of this thesis we will consider the geometric approach. As we will see, the two approaches (*bottom up* and *top down*) comes with their own advantages but are, of course, both equally valid. Crucially, those concepts are intertwined with the definition of the boundary of a simplex that, for the moment, has been only naively introduced; a formal definition will be presented in the next section.

### 8.3 Algebraic Topology, an overview

The structure of simplicial complexes can be investigated using methods coming from algebraic topology. Indeed, the concept of gluing together the pieces of space, as defined in Equation (8.1), can be further abstracted. The properties of the space we construct in this manner can be represented without needing to remember the entire set of points comprising the triangles, or the links that make up the discrete manifold. Instead, it is sufficient to remember the nature of the simplices that we are gluing together. For example, a triangle can be represented as a tuple of numbers, with a triplet corresponding to a specific triangle, and so forth. In this way, as we will see, all the relevant information regarding the structure we have conceptualized by gluing together geometric simplices and their orientations can be deduced.

In algebraic topology, each simplex  $\sigma = [i_0, i_1, \dots, i_n]$  is given an orientation (typically induced by the node labels) obeying

$$[i_0, i_1, \dots, i_n] = (-1)^{\sigma(\pi)} [i_{\pi(0)}, i_{\pi(1)}, \dots, i_{\pi(n)}], \quad (8.2)$$

where  $\pi(\ )$  is any permutation of the indexes and  $\sigma(\ )$  its parity. This is basically an extension of the naive definition of the triangle  $[1, 2, 3]$ .

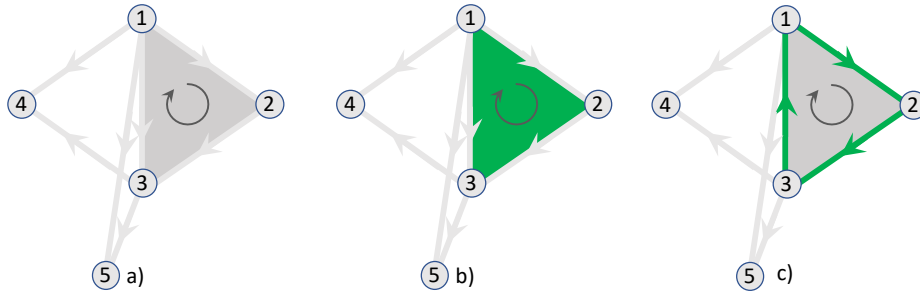


Figure 8.1: Panel a) shows a simplicial complex on dimension 2, with simplicial orientation induced by a labelling of the nodes. The boundary of the 2-simplex  $[1, 2, 3]$  highlighted in panel Panel b) is shown in panel c)

Let us indicate with  $N_n = |S_n|$  the number of  $n$ -dimensional simplices present in the considered simplicial complex. In algebraic topology the simplices  $\sigma_n^{(m)}$  of dimension  $n$  of a simplicial complex define the basis of a vector space  $C_n$  of  $n$ -chains. Therefore a  $n$ -chain  $\mathbf{c} \in C_n$  is a finite linear combination of the  $n$ -simplices  $\sigma_n^{(m)}$  with  $1 \leq m \leq N_n$  with coefficients  $c_m$

$$\mathbf{c} = \sum_{m=1}^{N_n} c_m \sigma_n^{(m)}. \quad (8.3)$$

A more general structure retaining the algebraic richness of simplicial complexes, is given by *cell complexes* [135]–[139]. The latter differ from simplicial complexes because they are obtained by gluing cells, (i.e., regular polytopes) along their faces. In particular 0-cells are nodes, 1-cells links, while 2-cells are generic polygons, and 3-cells are the Platonic polytopes, however nothing changes in their definition with respect to simplicial complexes. Indeed, every aspect of the geometric structure will be encoded in the so called *boundary operator* we are about to define.

### 8.3.1 Boundary Operator

The boundary of a given simplex (or of a chain) can be rigorously defined and computed. Remarkably, the boundary can be obtained from a chain by applying the *boundary operator*  $\partial_n : C_n \rightarrow C_{n-1}$ . The boundary operator is linear, therefore can be completely defined by providing its action on the basis elements of the chain space  $\sigma_n^{(i)} = [i_0, \dots, i_n]$ , i.e., the  $n$ -simplices, hence

$$\partial_n [i_0, \dots, i_n] = \sum_{p=0}^n (-1)^p [i_0, \dots, \hat{i}_p, \dots, i_n], \quad (8.4)$$

where the notation  $\hat{i}_p$  stands for the removal of the index  $i_p$  from the  $n$ -simplex, resulting thus in a  $(n-1)$ -simplex. A fundamental propriety of the boundary operator, which is coherent with the common knowledge of the boundary of a physical object, is that *the boundary of the boundary is closed*, or, stated mathematically:

$$\partial_{n-1} \partial_n = 0 \quad (8.5)$$

This propriety is at the core of algebraic topology and, as can be seen in [130], implies the full set of spectral proprieties and decomposition we are about to show in the

following section.

The action of the boundary operator is illustrated in Fig. 8.1 where the boundary of the  $[1, 2, 3]$  2-simplex (panel b) is shown (panel c). This boundary is nothing but the 1-chain  $[1, 2] + [2, 3] - [1, 3]$ .

The boundary matrix  $\mathbf{B}_n$  is the matrix whose elements are obtained by the action of the boundary operator on the basis elements. The set of all the boundary matrices  $\mathbf{B}_n$  with  $0 \leq n \leq d$  of a simplicial complex fully encodes the topology of the simplicial complex. The boundary matrix  $\mathbf{B}_n$  is a  $N_{n-1} \times N_n$  rectangular matrix of elements  $[B_n]_{\sigma, \sigma'} = +1$  if  $\sigma$  is a  $(n-1)$ -dimensional face of the  $n$  simplex  $\sigma$  with coherent orientation,  $[B_n]_{\sigma, \sigma'} = -1$  if the orientation is not coherent, and  $[B_n]_{\sigma, \sigma'} = 0$  if  $\sigma$  is not a face of  $\sigma'$ . In the particular case of  $\mathbf{B}_1$ , we have for instance

$$[B_1]_{i\ell} = \begin{cases} 1 & \text{if } \ell = [j, i] \text{ and } j < i, \\ -1 & \text{if } \ell = [i, j] \text{ and } i < j, \\ 0 & \text{otherwise} \end{cases} \quad (8.6)$$

once we assume the orientation to be induced by the nodes labels.

The structure of a simplicial complex is completely determined once one defines the set of its oriented simplices and the inclusion relations existing between each simplex and its subsets: e.g., two simplices of order  $k$ ,  $\sigma_k^{(1)}$  and  $\sigma_k^{(2)}$  are lower adjacent if they have a common face of order  $n-1$ ; they are upper adjacent if both are faces of a simplex of dimension  $k+1$ . This information together with the information about their relative orientation can be encoded into the *incidence matrices*  $\mathbf{B}_k$ ,  $k = 1, \dots, d$ .

$$B_n(i, j) = \begin{cases} 1 & \text{if } \sigma_{n-1}^{(i)} \sim \sigma_k^{(j)} \\ -1 & \text{if } \sigma_{n-1}^{(i)} \not\sim \sigma_k^{(j)} \\ 0 & \text{otherwise} \end{cases}, \quad (8.7)$$

Where  $\sim$  stands for equally oriented and  $\not\sim$  for the opposite orientation. Of course we imply that  $\sigma_{n-1}^{(i)} \in \sigma_k^{(j)}$ , namely, the lower dimensional simplex is included in the higher dimensional one that we are each time considering.

The matrix  $\mathbf{B}_1$  and  $\mathbf{B}_2$  of the simplicial complex shown in Fig. 8.1 are given by

$$\mathbf{B}_1 = \begin{matrix} & [1, 2] & [1, 3] & [1, 4] & [1, 5] & [2, 3] & [3, 4] & [3, 5] \\ \begin{matrix} [1] \\ [2] \\ [3] \\ [4] \\ [5] \end{matrix} & \begin{bmatrix} -1 & -1 & -1 & -1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & -1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & -1 & -1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 \end{bmatrix} \end{matrix} \quad (8.8)$$

and

$$\mathbf{B}_2 = \begin{matrix} & & & & & & [1, 2, 3] \\ & & & & & & \\ & & & & & & \\ & & & & & & \\ [1,2] & & & & & & \\ [1,3] & & & & & & \\ [1,4] & & & & & & \\ [1,5] & & & & & & \\ [2,3] & & & & & & \\ [3,4] & & & & & & \\ [3,5] & & & & & & \end{matrix} \begin{bmatrix} 1 \\ -1 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \end{bmatrix} \quad (8.9)$$

### Change of basis

The matrix representation of the boundary operator provides a way to highlight an important fact. When defining the simplicial complex via the *top-down* approach, the action of the boundary operator induces a basis for the lower-order simplices. This in turn implies that the orientation of these lower-order simplices (positive or negative) is inherited from the higher-order ones. Once this basis is established, one is free to change the basis of either the lower or higher-order simplices, such as by flipping the orientation of a link. Such changes require multiplying the corresponding column or row of the boundary operator matrix by  $-1$ . While this approach is valid, one might instead opt for a *bottom-up* strategy.

In the *bottom-up* approach, extra care is needed to ensure that the higher-order simplices are defined with proper closed boundaries, as mentioned earlier. Take, for example, Figure 8.1. Here, the triangle could be defined as  $[[1, 2], [2, 3], [3, 1]]$ , which means that the boundary operator  $\mathbf{B}_2$  will have only positive entries. The legitimacy of this definition can be confirmed by verifying that the equation  $\mathbf{B}_1 \mathbf{B}_2 = 0$  holds true. This equation is only satisfied if the boundary in question is properly closed (its boundary is zero).

### 8.3.2 Topological Signals

We shall now define the generalization of the node signals, namely the *topological signals*, functions defined on the links, faces and so on. A  $k$ -dimensional topological signal  $x$  is a  $k$ -dimensional cochain  $x \in C^k : \mathcal{C}_k \rightarrow \mathbb{R}^d$ , i.e., a linear function that associates to every  $k$ -chain of the simplicial complex a value in  $\mathbb{R}^d$ ; in the following, for pedagogical reasons, we shall set  $d = 1$ . Once we have a basis of the  $k$ -chains (for example the one induced by the boundary operator or the one we have chosen) the cochain can be expressed as a vector  $\mathbf{x}$  of elements

$$x_i = x(\sigma_k^{(i)}), \forall \sigma_k^{(i)} \in S_k \quad (8.10)$$

Note that thanks to the linearity of the cochain  $x$  we always have  $x(\sigma_k^{(i)}) = -x(-\sigma_k^{(i)})$  namely, by inverting the orientation of the simplex, the cochain should also change sign. Let us consider a cochain  $\mathbf{x}$  and a function  $\mathbf{F}$  that associated to  $\mathbf{x}$  a new cochain denoted by  $\mathbf{F}(\mathbf{x})$ . Based on the above, the latter is solely characterized by its values on  $\sigma_k^{(i)}$ , namely its “components”,  $\mathbf{F}(\mathbf{x}) = (f_1(\mathbf{x}), \dots, f_{N_k}(\mathbf{x}))$ . Let us finally assume each component to act component-wise, namely  $f_i(\mathbf{x}) = f_i(x_i)$  for all  $i$ . Being  $\mathbf{x}$  and

$\mathbf{F}(\mathbf{x})$  cochains they should be invariant with respect to the change of orientation of  $\sigma_k^{(i)}$ , namely

$$f_i(x_i) = -f_i(-x_i).$$

We now want to study the evolution of the topological signals by considering dynamical equations for the vector  $\mathbf{x}$ . Exactly in the same way as in conventional node dynamics we can define the *uncoupled* dynamics of a topological signal as

$$\frac{d\mathbf{x}_i}{dt} = f_i(\mathbf{x}_i), \quad (8.11)$$

where  $f_i : \mathbb{R} \rightarrow \mathbb{R}$  is a nonlinear function. Let us observe that we could also consider the case  $f_i : \mathbb{R}^d \rightarrow \mathbb{R}^d$  but in this introductory part we will restrict to  $d = 1$  and  $f_i = f$ . Requiring the invariance of this equation under change of orientation of the simplex  $i$ , imposes that the function  $f(\mathbf{x})$  must be odd. Indeed if the orientation of the simplex  $\sigma_i^{(k)}$  is reversed we have  $x_i \rightarrow -x_i$  and the dynamical Eq.(8.11) becomes

$$-\frac{d\mathbf{x}_i}{dt} = f(-\mathbf{x}_i). \quad (8.12)$$

Therefore to ensure that the dynamical equation (8.11) obeys the necessary invariance conditions for describing the dynamics of topological signals of dimension  $k > 0$ , we need to require  $f(-\mathbf{x}_i) = -f(\mathbf{x}_i)$ , hence that  $f(x_i)$  is an odd function. Note that requiring odd nonlinear functions  $f(x_i)$  is a necessary condition only for higher-order topological signals of dimension  $k > 0$ . Indeed since nodes do not have an orientation this requirement is not necessary for treating topological signals defined on nodes. Therefore this is an important difference with the dynamics defined exclusively on nodes. The reader should not be intimidated by this constrain as it is a consequence of the fact that higher-order signals are vector-like quantities and therefore their meaning is given once a basis is fixed. A 1-topological signal carries the information of directionality *from* one node *to* another one and changing (flipping) the orientation of the links (by flipping the labels of the respective nodes) should consequently impact the sign in front of the corresponding coefficient. Such behaviour should, clearly, be taken into account and preserved once we act on such signals with a non linear function.

### 8.3.3 Coboundary operator

We can now define the *coboundary operator*  $\delta_k : C^k \rightarrow C^{k+1}$  that associates to every  $k$ -cochain of the simplicial complex  $(k + 1)$ -cochain

$$\delta_k x = x \circ \partial_{k+1}. \quad (8.13)$$

Therefore we obtain

$$(\delta_k x) [i_0, i_1, \dots, i_{k+1}] = \sum_{p=0}^{k+1} (-1)^p x \left( [i_0, i_1, \dots, i_{p-1}, \hat{i}_p, i_{p+1} \dots i_{k+1}] \right) \quad (8.14)$$



It follows that if  $g \in C^{k+1} : g = \delta_k x$  then  $\mathbf{g} = \mathbf{B}_{k+1}^\top \mathbf{x}$ <sup>1</sup>.

### Connection between boundary and coboundary

We shall start defining a scalar product between  $k$ -cochains as:

$$\langle x, x \rangle = \mathbf{x}^\top \mathbf{x} \quad (8.17)$$

that can be rewritten as follows by using the coefficients of the cochain

$$\langle x, x \rangle = \sum_{r \in S_k} x_r^2 \quad (8.18)$$

Once we have defined the norm on  $C^k$  we can define the adjoint of the Coboundary with respect to the inner product that we have defined. Clearly the adjoint of the coboundary operator  $\delta_k^* : C^{k+1} \rightarrow C^k$  satisfies

$$\langle g, \delta_k f \rangle = \langle \delta_k^* g, f \rangle \quad (8.19)$$

for any  $f \in C^k$  and  $g \in C^{k+1}$ . It follows that if  $f' = \delta_k^* g$  then  $\mathbf{f}' = \mathbf{B}_{[k+1]} \mathbf{g}$ . We therefore have that the matrix representation of the  $k+1$ -boundary operator and the  $k$ -coboundary operators are linked by the transpose. This fact has a profound interpretation: asking questions regarding the boundary is the same as asking questions regarding functions or, stated differently, defining the action of discrete differential operators.

### Discrete differential operators

As extensively explained in [130] there is a precise correspondence with the intuitive definition of differential operators on network. Indeed assume that  $\rho$  is a 0-cochain (or a 0-topological/node signal),  $\mathbf{J}$  is a 1-cochain (or a 1-topological/edge signal), namely a current or flow and  $\Phi$  is a 2-cochain (or a 2-topological/face signal) a.k.a a flux. Each of them, once we have labelled every simplex, will be a vector whose component is the value on each node, oriented link and oriented face. For example, in the triangle  $[1, 2, 3]$  we could have  $\rho = \rho(1)[1] + \rho(2)[2] + \rho(3)[3] = (\rho(1), \rho(2), \rho(3))$ ,  $\mathbf{J} = J(1, 2)[1, 2] + J(2, 3)[2, 3] + J(1, 3)[1, 3]$  and  $\Phi = \Phi(1, 2, 3)[1, 2, 3]$ .

Then we can define the discrete differential operators as:

$$(\text{grad } \rho)(i, j) = \rho(j) - \rho(i) \quad (8.20)$$

$$(\text{div } X)(i) = \sum_{j: X(i, j) < 0} X(i, j) - \sum_{j: X(i, j) > 0} X(i, j) \quad (8.21)$$

$$(\text{curl } X)(i, j, k) = X(i, j) + X(j, k) + X(k, i) \quad (8.22)$$

---

<sup>1</sup>If  $\sigma_{k+1}^{(\alpha)}, \alpha \in 1 \dots |S_{k+1}|$  are the basis of the  $k+1$ -chains and  $\sigma_k^{(i)}, i \in 1 \dots |S_k|$  of the  $k$ -chains than we can evaluate the following component:

$$(\delta_k x)_\alpha = x \circ \partial_{k+1}(\sigma_{k+1}^{(\alpha)}) = x \left( \sum_{i \in S_k} (\mathbf{B}_{k+1})_{i, \alpha} \sigma_k^{(i)} \right) = \sum_{i \in S_k} (\mathbf{B}_{k+1})_{i, \alpha} x(\sigma_k^{(i)}) \quad (8.15)$$

$$= \sum_{i \in S_k} (\mathbf{B}_{k+1})_{i, \alpha} x_i = \sum_{i \in S_k} (\mathbf{B}_{k+1}^\top)_{\alpha, i} x_i = \mathbf{B}_{k+1}^\top \mathbf{x} \quad (8.16)$$

Remarkably we have  $\text{grad} = \mathbf{B}_1^\top$ ,  $\text{div} = \mathbf{B}_1$  and  $\text{curl} = \mathbf{B}_2^\top$  and, in this setting, we obtain the Laplace operator as  $-\text{div grad} = \mathbf{L}_0 = \nabla^2$ .

In the following of this thesis every time a lower dimensional (0,1 or 2) signal is encountered and modified through a boundary or coboundary operator the reader could think about it as a discrete version of the differential operator of Calculus courses.

As byproduct, there is an insightful and elegant feature in this discrete framework worth highlighting. One could either start by defining the discrete space and labeling its nodes, then proceed to construct the boundary operators (essentially, discrete differential operators), or one could take the opposite approach—starting with the boundary operators and subsequently inferring the discrete space. This reciprocal relationship suggests that *space and differential operators serve as mirrors of each other*. Consequently, we should anticipate a complex interplay between the topology—i.e., the structure of the space—and the dynamics generated by the coupling of differential operators. This intricate relationship underscores the true beauty and generality of this framework.

### 8.3.4 Hodge-Laplacians and the Dirac operator

Starting from the boundary and the coboundary operators, we define the higher-order Laplacians and the Dirac operator.

The Laplace operator [91], [129], [130] of order  $n$ , also called  $n$ -Hodge-Laplacian, describes higher-order diffusion from  $n$ -simplices to  $n$ -simplices and are  $N_n \times N_n$  matrices defined as

$$\mathbf{L}_n = \mathbf{B}_n^\top \mathbf{B}_n + \mathbf{B}_{n+1} \mathbf{B}_{n+1}^\top = \mathbf{L}_n^{\text{down}} + \mathbf{L}_n^{\text{up}} \quad (8.23)$$

for  $1 \leq n < d$ . For  $n = 0$  and  $n = d$  the Hodge-Laplacians  $\mathbf{L}_0$  and  $\mathbf{L}_d$  are respectively given by  $\mathbf{L}_0 = \mathbf{L}_0^{\text{up}} = \mathbf{B}_1 \mathbf{B}_1^\top$  and  $\mathbf{L}_d = \mathbf{L}_d^{\text{down}} = \mathbf{B}_d^\top \mathbf{B}_d$ .

The definition the  $n$ -Laplacian  $\mathbf{L}_n$  can be also related to the notion of neighbour and orientation as follows:

$$L_n(i, j) = \begin{cases} d(\sigma_i^{(k)}) + (p + 1) & i = j. \\ 1 & i \neq j, \sigma_i^{(k)} \frown \sigma_k^{(j)} \vee \sigma_i^{(k)} \smile \sigma_k^{(j)}, \sigma_i^{(k)} \sim \sigma_k^{(j)} \\ -1 & i \neq j, \sigma_i^{(k)} \frown \sigma_k^{(j)} \vee \sigma_i^{(k)} \smile \sigma_k^{(j)}, \sigma_i^{(k)} \approx \sigma_k^{(j)} \\ 0 & \text{otherwise.} \end{cases} \quad (8.24)$$

where  $\smile$  and  $\frown$  indicate lower and upper incident neighbour simplices respectively. Those two equivalent definitions (8.23) and (8.24) show how defining the space operator is just a consequence of the labelling and the definition of a relative orientation. The action of the Hodge-Laplacian can be interpreted as follows. The term  $\mathbf{L}_n^{\text{up}}$  represents the diffusion between  $n$ -simplices through shared  $(n + 1)$ -dimensional simplices. In the case of a network, as previously noticed, this is the combinatorial Laplacian, where concentrations on nodes diffuse through incident edges. The term  $\mathbf{L}_n^{\text{down}}$  represents diffusion between  $n$ -simplices through shared  $(n - 1)$ -simplices, i.e., incident  $(n - 1)$ -faces. For instance in a network (i.e., a 1-simplicial complex)  $\mathbf{L}_1 = \mathbf{L}_1^{\text{down}}$  determines diffusion from links to links through nodes.

From this definition, it is clear that the Hodge-Laplacian of order  $n$  only acts on topological signals of dimension  $n$ . Therefore the  $n$ -Hodge-Laplacian cannot couple signals of different dimension.

In order to couple signal of different dimension, we require the Dirac operator [131]–[134],  $\mathcal{D}$ , which is encoded by an  $M \times M$  matrix where  $M = \sum_{n=0}^d N_n$  and has elements

$$\mathcal{D}_{\sigma,\sigma'} = \begin{cases} [B_n]_{\sigma,\sigma'} & \text{if } |\sigma'| = |\sigma| + 1 = n \\ [B_n^\top]_{\sigma,\sigma'} & \text{if } |\sigma| = |\sigma'| + 1 = n \end{cases}, \quad (8.25)$$

where with  $|\sigma|$  we indicate the dimension of the simplicial complex  $\sigma$ . It follows that in a two dimensional simplicial complex, the Dirac operator has the block structure

$$\mathcal{D} = \begin{pmatrix} 0 & \mathbf{B}_1 & 0 \\ \mathbf{B}_1^\top & 0 & \mathbf{B}_2 \\ 0 & \mathbf{B}_2^\top & 0 \end{pmatrix}, \quad (8.26)$$

while in a network the Dirac operator is given by

$$\mathcal{D} = \begin{pmatrix} 0 & \mathbf{B}_1 \\ \mathbf{B}_1^\top & 0 \end{pmatrix}, \quad (8.27)$$

It follows that the Dirac operator, differently from the Hodge-Laplacian, can couple topological signals of different dimension. In particular the Dirac operator can be used to project a topological signal of any dimension  $n$  onto simplices of dimension  $n + 1$  and  $n - 1$ . One of the most significant properties of the Dirac operator is that it can be considered the “square root” of the Laplacian. In fact we have

$$\mathcal{D}^2 = \mathcal{L} = \mathbf{L}_0 \oplus \mathbf{L}_1 \oplus \dots \oplus \mathbf{L}_d. \quad (8.28)$$

For instance, for a simplicial complex of dimension  $d = 2$  we have

$$\mathcal{D}^2 = \mathcal{L} = \begin{pmatrix} \mathbf{L}_0 & 0 & 0 \\ 0 & \mathbf{L}_1 & 0 \\ 0 & 0 & \mathbf{L}_2 \end{pmatrix}, \quad (8.29)$$

and for a network

$$\mathcal{D}^2 = \mathcal{L} = \begin{pmatrix} \mathbf{L}_0 & 0 \\ 0 & \mathbf{L}_1 \end{pmatrix}. \quad (8.30)$$

Interestingly, both the Hodge-Laplacians and the Dirac operator can be extended to weighted simplicial complexes (see for instance [140]).

### 8.3.5 Major Spectral properties

The  $n$ -order Hodge-Laplacian [91], [129], [130] is a semi-definite positive operator whose kernel has dimension equal to the  $n$ -th Betti number  $\beta_n$ , i.e., the degeneracy of its null eigenvalue is equal to the Betti number  $\beta_n$ . In addition to this, the Hodge-Laplacians obey the Hodge decomposition which implies that the space of  $n$ -chains can be decomposed as

$$C_n = \text{im}(\mathbf{B}_n^\top) \oplus \ker(\mathbf{L}_n) \oplus \text{im}(\mathbf{B}_{n+1}), \quad (8.31)$$

where the kernel of the Hodge-Laplacians are given by

$$\ker(\mathbf{L}_0) = \ker(\mathbf{B}_1^\top) \quad \ker(\mathbf{L}_n) = \ker(\mathbf{B}_n) \cap \ker(\mathbf{B}_{n+1}^\top). \quad (8.32)$$

This same decomposition can be applied to the space of topological signals, namely the one of the  $n$ -cochain in exactly the same way. For us the Boundary operators could either act on the chains or on the cochain and their representation will be the same. Now a connection with discrete differential operators is worth stressing. Let us fix  $n = 1$  than we can decompose the space of 1 dimensional topological signals, namely flows on network, as:

$$C^1 = \text{im}(\mathbf{B}_1^\top) \oplus \ker(\mathbf{L}_1) \oplus \text{im}(\mathbf{B}_2) = \text{im}(\text{grad}) \oplus \ker(\Delta) \oplus \text{im}(\text{rot}^\top), \quad (8.33)$$

Namely each flux can be expressed as a *potential flow*  $\oplus$  *harmonic component*  $\oplus$  *vector potential*, which is a known and widely used decomposition of flows in Calculus.

The Dirac operator [131] has a kernel given by the direct sum of the kernels of the Laplacians,

$$\ker(\mathcal{D}) = \ker(\mathcal{L}) = \ker(\mathbf{L}_0) \oplus \ker(\mathbf{L}_1) \oplus \dots \oplus \ker(\mathbf{L}_d). \quad (8.34)$$

The non-zero spectrum of the Dirac operator is formed by the concatenation of the spectra of the Hodge-Laplacians taken with positive and negative sign.

Let us now focus on the spectrum of the Hodge-Laplacians  $\mathbf{L}_0$  and  $\mathbf{L}_1$  defined on a network, and reveal the relation between their spectrum and the singular values of the boundary operator  $\mathbf{B}_1$ . Since  $\mathbf{L}_0 = \mathbf{B}_1 \mathbf{B}_1^\top$  and  $\mathbf{L}_1 = \mathbf{B}_1^\top \mathbf{B}_1$  it follows that  $\mathbf{L}_0$  and  $\mathbf{L}_1$  are isospectral, i.e., they have the same non-zero eigenvalues and any eigenvalue  $\Lambda_0^k$  of  $\mathbf{L}_0$  can be written as  $\Lambda_0^k = b_k^2$  where  $b_k$  indicates the non-zero singular eigenvalues of the boundary matrix  $\mathbf{B}_1$ . We note however that the degeneracy of the zero eigenvalue  $\Lambda_0 = 0$  is different for  $\mathbf{L}_0$  and  $\mathbf{L}_1$ . Indeed, for  $\mathbf{L}_0$  the degeneracy of the zero eigenvalue is  $\beta_0$ , i.e., the number of connected components of the network, while for  $\mathbf{L}_1$  it is given by  $\beta_1$ , i.e., the number of independent cycles of the network. Therefore for a network that has the topology of a linear chain with periodic boundary conditions then  $N_0 = N_1$ , and thus  $\beta_0 = \beta_1 = 1$ , for a tree when we have  $N_1 = N_0 - 1$  we have  $\beta_0 = 1$  and  $\beta_1 = 0$  and in general for a connected network we have  $\beta_0 = 1$ ,  $\beta_1 = N_1 - N_0 + 1$ .

# Chapter 9

## Pattern formation of topological signals

Nature is a blossoming of patterns, spontaneously emerging from the web of nonlinear interactions existing among the many basic units constituting the system under scrutiny [141], [142]. Scholars have developed theories capable to deal with both the case of stationary patterns [143]–[145] and time varying ones [146]–[150]. Such research has been developed in the framework of network science [151]–[154] relying on the assumption that system interactions can be sufficiently well described by using a pairwise representation: the basic units composing the system exhibit their own dynamics, i.e., a local evolution law associated to each node of the network, and then they interact by diffusing or via non-local (long-range) interactions, by using the available links. In this Chapter we propose a framework to reveal Turing patterns of reacting species described by topological signals defined on the simplices of different dimensions (nodes, links, triangles) coupled through the Dirac operator. Our main goal is to consider reaction-diffusion systems [155] and extend the Turing theory developed so far on networked systems [144] to the framework of simplicial complexes.

Turing’s original framework involved two reacting species whose stable homogeneous equilibrium can turn out unstable once the species are allowed to diffuse and suitable conditions of the species diffusivities are assumed [143]. Gierer and Meinhardt later emphasized that for the Turing instability to set up, one of the two species needs to be an activator while the other should be an inhibitor, and moreover the latter needs to diffuse much faster than the former [156]. The theory was successively extended to regular lattices by Othmer and Scriven [157] and finally to complex networks by Nakao and Mikhailov [144]. The latter framework has been further expanded considering directed networks [158], multiplex [159], temporal networks [160] and non-normal networks [161], just to mention a few. In all the above settings, the two species react in each node while diffusing through the links. For signals defined exclusively on the nodes cross-diffusion terms have been introduced in [162], [163]. Turing patterns on higher-order structures have been recently studied in [164], [165]. Note however that our approach is different because in those works the dynamics is restricted to nodes, while links and high-order structures support the generalized diffusion.

In particular, we consider two different settings: when the reaction term is solely responsible for the coupling of signals of different dimension and when the diffusion also includes cross-diffusion terms coupling the dynamics of signals in different dimension.

For the sake of simplicity, in this Chapter, we will focus our analysis to the case of coupled nodes and links signals which is arguably also the most relevant to applications. We derive the conditions under which stable Turing patterns can be observed and we highlight the differences between the dynamics with and without cross-diffusion terms. The analytical results derived in general are presented with applications to square lattices with periodic boundary conditions and validated by numerical simulations on a benchmark network.

We are interested in studying reaction-diffusion systems defined on simplicial complexes. This entails defining appropriate reaction and diffusion terms. In a network the reaction term is localized on nodes, where the interacting species can be found. When the interacting species are associated to simplices of different dimension, a Dirac reaction term that uses the Dirac operator is required to allow topological signals of different dimension to interact. The theory will be developed in general and then a more realistic application involving lower dimensional topological signal will be presented. In a network, concentrations can flow from one node to one of its neighbours, passing through links, namely the structure one dimension above. A similar idea can be thought in simplicial complexes: quantities defined on links can flow among links by using the faces they share, hence again the structures one dimension above. There is however a second possibility: they can use structures one dimension below, i.e., nodes, to communicate. Such processes can be described by introducing the Hodge Laplace matrix which describes uncoupled diffusion of topological signals of any given dimension. However Hodge-Laplacians describe diffusion terms that act on topological signals of any given dimension separately. Requiring a diffusive coupling of topological signals of different dimension can be only achieved by considering Dirac cross-diffusion terms which involve odd powers of the Dirac operator. Specifically, this includes cross-diffusion terms that are linear or cubic in the Dirac operator.

Here we propose a Turing theory for topological signals and to this end we consider a simplicial complex of dimension  $n$  and species living on nodes, links, triangles, etc. In the present terminology, the concentration of the species living on nodes is a 0-topological signal while the concentration of the species defined on links is a 1-topological signal etc. The dynamical state of the simplicial complex is described by a vector  $\Phi$  which is the direct sum of all topological signals defined on the simplicial complex. For example in a  $n = 2$  dimensional simplicial complex with  $N_0$  nodes,  $N_1$  links and  $N_2$  triangles, assuming a one dimensional dynamical system living in each discrete structure, we have

$$\Phi = \begin{pmatrix} \mathbf{u} \\ \mathbf{v} \\ \mathbf{w} \end{pmatrix}, \quad (9.1)$$

where  $\mathbf{u} \in \mathbb{R}^{N_0}$ ,  $\mathbf{v} \in \mathbb{R}^{N_1}$ ,  $\mathbf{w} \in \mathbb{R}^{N_2}$  are the vectors of concentration of species defined on nodes, links and triangles respectively. In general, if the dimension of every dynamical system is large than one, we have  $\mathbf{u} \in \mathbb{R}^{N_0 \times d_u}$ ,  $\mathbf{v} \in \mathbb{R}^{N_1 \times d_v}$ ,  $\mathbf{w} \in \mathbb{R}^{N_2 \times d_w}$ , where  $d_{u,v,w}$  is the dimension of the node, link and face defined dynamical system respectively.

These signals can only interact with each other when we consider their projection to simplices of one dimension up or one dimension down. This projection is performed

by applying the Dirac operator  $\mathcal{D}$  to  $\Phi$  obtaining new (projected) signals, i.e.,

$$\Psi = \mathcal{D}\Phi = \begin{pmatrix} \hat{\mathbf{u}} \\ \hat{\mathbf{v}} \\ \hat{\mathbf{w}} \end{pmatrix}, \quad (9.2)$$

where  $\hat{\mathbf{u}} \in \mathbb{R}^{N_0}$ ,  $\hat{\mathbf{v}} \in \mathbb{R}^{N_1}$ ,  $\hat{\mathbf{w}} \in \mathbb{R}^{N_2}$  are defined on nodes, links and triangles respectively. In a simplicial complex of dimension  $n = 2$  the Dirac operator  $\mathcal{D}$  is a  $M \times M$  matrix with  $M = N_0 + N_1 + N_2$  which can be expressed in terms of the incidence matrices  $\mathbf{B}_1, \mathbf{B}_2$  (defined in the precedent Chapter) and their transpose as

$$\mathcal{D} = \begin{pmatrix} 0 & \mathbf{B}_1 & 0 \\ \mathbf{B}_1^\top & 0 & \mathbf{B}_2 \\ 0 & \mathbf{B}_2^\top & 0 \end{pmatrix}. \quad (9.3)$$

We therefore obtain that the projected signal  $\Psi$  is given by

$$\Psi = \mathcal{D}\Phi = \begin{pmatrix} \hat{\mathbf{u}} \\ \hat{\mathbf{v}} \\ \hat{\mathbf{w}} \end{pmatrix} = \begin{pmatrix} \mathbf{B}_1 v \\ \mathbf{B}_1^\top u + \mathbf{B}_2 w \\ \mathbf{B}_2^\top v \end{pmatrix}, \quad (9.4)$$

where  $\mathbf{B}_1^\top \mathbf{u}$  and  $\mathbf{B}_2 \mathbf{w}$  describe the irrotational part and the solenoidal part of the link signal  $\hat{\mathbf{v}}$ . Therefore, the dynamical state of the simplicial complex comprises both the topological signals  $\Phi$  and their projections  $\Psi = \mathcal{D}\Phi$ .

Here we propose a Turing theory for topological signals where the topological signals  $\Phi$  can be coupled to the projected topological signals  $\Psi$  either through a Dirac reaction term or through a Dirac diffusion term or both. In presence of a Dirac reaction term and a positive-definite Laplacian<sup>1</sup> diffusion term, the reaction-diffusion process of topological signal is defined as

$$\dot{\Phi} = F(\Phi, \mathcal{D}\Phi) - \gamma \mathcal{L}\Phi, \quad (9.5)$$

where  $F(\Phi, \mathcal{D}\Phi)$  is the Dirac reaction term coupling each topological signal of dimension  $n$  with the nearby topological signals of dimension  $n + 1$  or  $n - 1$  projected to dimension  $n$ . In particular  $F(\Phi, \mathcal{D}\Phi)$  here indicates a generic nonlinear function, assumed to be applied component-wise on the vectors. For instance for  $n = 2$  we have

$$F(\Phi, \mathcal{D}\Phi) = \begin{pmatrix} f_0(\mathbf{u}, \mathbf{B}_1 \mathbf{v}) \\ f_1(\mathbf{v}, \mathbf{B}_1^\top u + \mathbf{B}_2 w) \\ f_2(\mathbf{w}, \mathbf{B}_2^\top v) \end{pmatrix}, \quad (9.6)$$

where  $f_n(\mathbf{x}, \mathbf{y})$  are nonlinear functions, such that :

$f_1(\mathbf{u}, \mathbf{B}_1 \mathbf{v}) = (f_1(u_1, (\mathbf{B}_1 \mathbf{v})_1), \dots, f_1(u_{N_0}, (\mathbf{B}_1 \mathbf{v})_{N_0}))$  etc. The matrix  $\gamma$  in Eq.(9.5) is a diagonal matrix

$$\gamma = \begin{pmatrix} D_0 & 0 & 0 \\ 0 & D_1 & 0 \\ 0 & 0 & D_2 \end{pmatrix}, \quad (9.7)$$

---

<sup>1</sup>We point out that this is at variance with more canonical Turing schemes where, instead, the Laplace operator is negative-definite. This justifies the '-' in Equation (9.5)

where  $D_n$  is the diffusion coefficient acting on topological signals of order  $n$ . Therefore Eq.(9.5) describes topological signals defined on the simplices of the simplicial complex that react with the projection of the topological signals defined in different dimension while undergoing higher-order diffusion.

Note that from the dynamical system given by Eq.(9.5) one can derive the dynamics of the projected signal  $\Psi = \mathcal{D}\Phi$  which is given by

$$\dot{\Psi} = \hat{F}(\Phi, \Psi) - \mathcal{D}\gamma\mathcal{D}\Psi, \quad (9.8)$$

where  $\hat{F}(\Phi, \Psi) = \mathcal{D}F(\Phi, \Psi)$ . In the case of diffusion coefficients independent on the order of the simplices, i.e., for  $D_k = D$ , this equation reduces to

$$\dot{\Psi} = \hat{F}(\Phi, \Psi) - \gamma\mathcal{L}\Psi. \quad (9.9)$$

Therefore in this case the dynamics of the projected signal is the same as the dynamics of the signal  $\Phi$  (Eq. (9.5)) provided that  $F(\Psi, \Phi) = \hat{F}(\Phi, \Psi) = \mathcal{D}F(\Phi, \Psi)$  as for instance in the case of square lattices with periodic boundary conditions (note that  $\Psi$  and  $\Phi$  have been exchanged, due to the duality between links and nodes that holds in this setting).

We now consider Dirac cross-diffusion terms enforcing diffusion of signals across different dimensions.

In particular, we consider including a linear Dirac cross-diffusion term which is proportional the Dirac operator. In [166] also the case of a cubic cross-diffusion term is discussed. In the case of a linear Dirac cross-diffusion term, the reaction-diffusion dynamics takes the form

$$\dot{\Phi} = F(\Phi, \mathcal{D}\Phi) - \tilde{\gamma}\mathcal{D}\Phi - \gamma\mathcal{L}\Phi, \quad (9.10)$$

where  $\tilde{\gamma}$  is the diagonal matrix of cross-diffusion coefficients  $\tilde{D}_n$ ,

$$\tilde{\gamma} = \begin{pmatrix} \tilde{D}_0\mathbb{I}_{N_0} & 0 & 0 \\ 0 & \tilde{D}_1\mathbb{I}_{N_1} & 0 \\ 0 & 0 & \tilde{D}_2\mathbb{I}_{N_2} \end{pmatrix}. \quad (9.11)$$

In this case, the corresponding projected signals  $\Psi = \mathcal{D}\Phi$  obey the dynamical system of equations

$$\dot{\Psi} = \hat{F}(\Phi, \Psi) - \mathcal{D}\tilde{\gamma}\Psi - \mathcal{D}\gamma\mathcal{D}\Psi. \quad (9.12)$$

If the diffusion and cross-diffusion coefficients are the same and  $\gamma$  and  $\tilde{\gamma}$  are proportional to the identity matrix, then we have that both  $\gamma$  and  $\tilde{\gamma}$  commute with the Dirac operator  $\mathcal{D}$  and the dynamics of projected signals becomes

$$\dot{\Psi} = \hat{F}(\Phi, \Psi) - \tilde{\gamma}\mathcal{D}\Psi - \gamma\mathcal{L}\Psi. \quad (9.13)$$

Therefore, in this case too, as long as  $\hat{F}(\Phi, \Psi) = \mathcal{D}F(\Phi, \Psi)$  can be written as the reaction term  $F(\Phi, \Psi)$  (as it happens for square lattices with periodic boundary conditions for example) the equation for the signal is equal to the equation for the projected signals. In all the considered cases, the Turing mechanism requires the presence of a stable homogeneous equilibrium once the diffusion part is silenced. Such state turns out unstable for suitable values of the diffusion coefficients and conditions on the underlying topology; and ultimately evolves into an heterogeneous state. If we think



about a network defined dynamical system with Laplacian coupling, requiring the stability of the homogeneous state implies the homogeneous vector (whose components are all ones) in the kernel of the graph Laplacian. Otherwise the coupling could be able to perturb the homogeneous state towards an heterogeneous one. Despite this condition being trivially satisfied in a connected graph, the situation changes when activity lies in the simplices of a simplicial complex. When dealing with topological signals, a necessary condition is that the homogeneous state vector  $\mathbf{h} = (1, \dots, 1)^\top$  is in the kernel of the Dirac operator  $\mathbf{h} \in \ker(\mathcal{D})$  or, equivalently,

$$\mathcal{D}\mathbf{h} = 0. \quad (9.14)$$

when the state vector includes both nodes and links signals Eq.(9.14) accounts to require

$$\mathbf{B}_1\mathbf{h} = 0 \text{ and } \mathbf{B}_2^\top\mathbf{h} = 0 \quad (9.15)$$

where  $\mathbf{h} = (1, 1 \dots, 1)^\top$  is a homogeneous  $N_1$ -dimensional column vector defined on the links of the network.

By assuming to have a 1-simplicial complex, (i.e., a network) we discard the presence of triangles. In that case  $\mathbf{B}_2 = 0$ , and the second of the conditions in Eq.(9.15) is trivially satisfied. Let us now focus on the remaining condition. Tackling this problem becomes much easier by noticing that the  $i$ -th row of the boundary operator is equal to minus the divergence of node  $i$ . Such equivalence, proved in [130], can be exploited to construct a simplicial complex with the wanted property.

By requiring that every node has an equal amount of in-going and out-going links, we thus ensure that a homogeneous signal, namely an edge-flow directed as indicated by the links orientation, has zero divergence. To sum up, the following analysis grounded on the conditions given in Eq. (9.15), holds for every simplicial complex whose nodes have an even number of connected edges. Notably examples of these networks are square lattices with periodic boundary conditions, however in the following Chapter other examples of such structures for higher dimensions will be given.

## 9.1 Onset of diffusion driven instability

### 9.1.1 Dirac reaction term

In this section we focus on reaction-diffusion systems involving topological signals defined on the nodes and on the links of a network. Our goal is to derive the dispersion relation allowing us to determine the conditions for the Turing instability onset in the presence exclusively of a Dirac reaction term that couples the two topological signals of different dimension, while the diffusion term does not, i.e., driven by Eq.(9.5) which we rewrite here for convenience

$$\dot{\Phi} = F(\Phi, \mathcal{D}\Phi) - \gamma\mathcal{L}\Phi. \quad (9.16)$$

In a network we have  $\Phi = (\mathbf{u}, \mathbf{v})^\top$  and  $F(\Phi, \mathcal{D}\Phi) = \left( f(\mathbf{u}, \mathbf{B}_1\mathbf{v}), g(\mathbf{v}, \mathbf{B}_1^\top\mathbf{u}) \right)^\top$  where  $f$  and  $g$  are two generic nonlinear functions, assumed to be applied component-wise on the vectors, i.e.,  $f(\mathbf{u}, \mathbf{B}_1\mathbf{v}) = (f(u_1, (\mathbf{B}_1\mathbf{v})_1), \dots, f(u_{N_0}, (\mathbf{B}_1\mathbf{v})_{N_0}))$ . Here  $\gamma$  reduces

to the  $(N_0 + N_1) \times (N_0 + N_1)$  block diagonal matrix with structure

$$\gamma = \begin{pmatrix} D_0 \mathbb{I}_{N_0} & 0 \\ 0 & D_1 \mathbb{I}_{N_1} \end{pmatrix}, \quad (9.17)$$

where  $D_0$  and  $D_1$  indicate the diffusion coefficients of the species defined on nodes and links respectively. The Dirac operator  $\mathcal{D}$  and the Laplacian operator  $\mathcal{L}$  are defined as the  $(N_0 + N_1) \times (N_0 + N_1)$  matrices with block structure

$$\mathcal{D} = \begin{pmatrix} 0 & \mathbf{B}_1 \\ \mathbf{B}_1^\top & 0 \end{pmatrix}, \quad \mathcal{L} = \mathcal{D}^2 = \begin{pmatrix} \mathbf{L}_0 & 0 \\ 0 & \mathbf{L}_1 \end{pmatrix}. \quad (9.18)$$

It follows that the dynamics driven by Eq.(9.16) can be rewritten explicitly as

$$\begin{aligned} \frac{d\mathbf{u}}{dt} &= f(\mathbf{u}, \mathbf{B}_1 \mathbf{v}) - D_0 \mathbf{L}_0 \mathbf{u}, \\ \frac{d\mathbf{v}}{dt} &= g(\mathbf{v}, \mathbf{B}_1^\top \mathbf{u}) - D_1 \mathbf{L}_1 \mathbf{v} \end{aligned} \quad (9.19)$$

We would like to stress that, using more familiar mathematical symbols, such system is equivalent to:

$$\begin{aligned} \frac{d\mathbf{u}}{dt} &= f(\mathbf{u}, \nabla \cdot \mathbf{v}) - D_0 \nabla^2 \mathbf{u}, \\ \frac{d\mathbf{v}}{dt} &= g(\mathbf{v}, \nabla \mathbf{u}) - D_1 \Delta_1 \mathbf{v} \end{aligned} \quad (9.20)$$

Where  $\Delta_1$  stands for the Graph Helmholtzian. In the spirit of Turing theory, let us silence the generalized diffusive terms and look for a homogeneous solutions, i.e., the existence of  $\mathbf{u}^* = u_0 \mathbf{h}$  and  $\mathbf{v}^* = v_0 \mathbf{h}$ , for some constants  $u_0$  and  $v_0$ . Because of the assumption on the underlying simplex, we have  $\mathbf{B}_1 \mathbf{v}^* = 0$  and  $\mathbf{B}_1^\top \mathbf{u}^* = 0$ . The existence of a homogeneous fixed point reverberates on the structure of  $f, g$  such that

$$0 = f(\mathbf{u}^*, 0) \text{ and } 0 = g(\mathbf{v}^*, 0), \quad (9.21)$$

which in turn yields that  $u_0$  and  $v_0$  are solutions of  $f(u_0, 0) = g(v_0, 0) = 0$ .

To study the stability feature of the homogeneous equilibrium, we consider a homogeneous perturbation about the latter,  $\delta \mathbf{u} = \mathbf{u} - \mathbf{u}^*$  and  $\delta \mathbf{v} = \mathbf{v} - \mathbf{v}^*$ . Hence by linearising (9.19) we obtain

$$\begin{aligned} \frac{d\delta \mathbf{u}}{dt} &= \partial_{\mathbf{u}} f(\mathbf{u}^*, 0) \delta \mathbf{u}, \\ \frac{d\delta \mathbf{v}}{dt} &= \partial_{\mathbf{v}} g(\mathbf{v}^*, 0) \delta \mathbf{v}, \end{aligned} \quad (9.22)$$

where we used again the conditions  $\mathbf{h} = (1, \dots, 1)^\top \in \ker \mathbf{B}_1$  and  $\mathbf{h} \in \ker \mathbf{B}_1^\top$  to remove some terms in the previous equation. The condition for the stability, considering the component-wise action of the functions, is thus

$$\partial_{\mathbf{u}} f(u_0, 0) < 0 \text{ and } \partial_{\mathbf{v}} g(v_0, 0) < 0. \quad (9.23)$$

Let us observe that Eq. (9.23) implies that both species are self inhibitors, this is the result of the peculiar form of Eq. (9.19), and of the assumption  $\mathbf{B}_1 \mathbf{v}^* = 0$  and  $\mathbf{B}_1^\top \mathbf{u}^* = 0$  which ultimately decouples the dynamics of the two species in the linear

regime. This is at odd with the classical Turing instability where patterns can never emerge in the inhibitor-inhibitor setting, unless some additional assumptions are made [167].

We now focus on the stability of such equilibrium once subjected to heterogeneous perturbations, hence not in the kernels of  $\mathbf{L}_0$  and  $\mathbf{L}_1$ . Let us linearize Eq. (9.19) about the equilibrium solution, obtaining

$$\begin{aligned}\frac{d\delta\mathbf{u}}{dt} &= (\partial_{\mathbf{u}}f) \delta\mathbf{u} + (\partial_{\mathbf{B}_1\mathbf{v}}f) \mathbf{B}_1\delta\mathbf{v} - D_0 \mathbf{L}_0 \delta\mathbf{u}, \\ \frac{d\delta\mathbf{v}}{dt} &= (\partial_{\mathbf{B}_1^\top\mathbf{u}}g) \mathbf{B}_1^\top \delta\mathbf{u} + (\partial_{\mathbf{v}}g) \delta\mathbf{v} - D_1 \mathbf{L}_1 \delta\mathbf{v},\end{aligned}\tag{9.24}$$

where  $\partial_{\mathbf{B}_1\mathbf{v}}f$  and  $\partial_{\mathbf{B}_1^\top\mathbf{u}}g$  denote the scalars indicating the derivative of  $f, g$  with respect to their second argument, (i.e., the projected higher and lower dimensional signal respectively) calculated at the homogeneous stationary solution.

We now recall that the network Laplacians  $\mathbf{L}_0 = \mathbf{B}_1\mathbf{B}_1^\top$  and  $\mathbf{L}_1 = \mathbf{B}_1^\top\mathbf{B}_1$  are isospectral, i.e., they have the same non-zero spectrum. The  $\hat{N}$  non-zero eigenvalues  $\Lambda_0^k$  with  $1 \leq k \leq \hat{N}$  of  $\mathbf{L}_0$  and  $\mathbf{L}_1$  can be expressed as the square of the singular values  $b_k$  of  $\mathbf{B}_1$ , i.e.,  $\Lambda_0^k = b_k^2$ . The eigenvectors  $\psi_0^m$  and  $\psi_1^m$  of  $\mathbf{L}_0$  and  $\mathbf{L}_1$  can be adopted as a basis to perform the singular value decomposition of  $\mathbf{B}_1$ . On a connected network these eigenvectors include the eigenvectors  $\psi_0^k$  and  $\psi_1^k$  corresponding to the non-zero eigenvalue  $\Lambda_0^k = \Lambda_1^k = b_k^2$ , the eigenvector  $\phi_0^h = (1, \dots, 1)^\top$  of  $\mathbf{L}_0$  associated to the zero eigenvalue  $\Lambda_0 = 0$  and the eigenvectors  $\psi_1^l$  associated the zero eigenvalues  $\Lambda_1^l = 0$  of  $\mathbf{L}_1$ . Interestingly the eigenvectors  $\psi_0^k$  and  $\psi_1^k$  associated to the eigenvalue  $\Lambda_0^k = \Lambda_1^k = b_k^2 > 0$  obey

$$\mathbf{B}_1\phi_1^k = b_k\psi_0^k, \quad \mathbf{B}_1^\top\phi_0^k = b_k\psi_1^k.\tag{9.25}$$

Using these results, the signals  $\delta\mathbf{u}$  and  $\delta\mathbf{v}$ , as well as the projected signals  $\delta\hat{\mathbf{u}} = \mathbf{B}_1\delta\mathbf{v}$  and  $\delta\hat{\mathbf{v}} = \mathbf{B}_1^\top\delta\mathbf{u}$ , can be projected onto the basis of the eigenvectors  $\psi_n^m$  of  $\mathbf{L}_n$  (with  $n = 0, 1$  for the analyzed case) corresponding to the non-zero eigenvalues  $\Lambda_0^k = b_k^2$ . We obtain

$$\langle \psi_0^k, \delta\mathbf{u} \rangle = \delta\hat{u}_k, \quad \langle \psi_1^k, \delta\mathbf{v} \rangle = \delta\hat{v}_k,\tag{9.26}$$

$$\langle \psi_0^k, \mathbf{B}_1\delta\mathbf{v} \rangle = b_k\delta\hat{v}_k, \quad \langle \psi_1^k, \mathbf{B}_1^\top\delta\mathbf{u} \rangle = b_k\delta\hat{u}_k,\tag{9.27}$$

where  $\langle \cdot, \cdot \rangle$  denotes the scalar product. By using Eq.(9.26) and Eq.(9.27), we can project in Eq.(9.24) the equations for  $\delta\mathbf{u}$  onto  $\psi_0^{k^\top}$  and the ones for  $\delta\mathbf{v}$  on  $\psi_1^{k^\top}$ , with  $k$  such that  $\Lambda_0^k = \Lambda_1^k = b_k^2 \neq 0$ , to eventually obtain:

$$\begin{aligned}\frac{d\delta\hat{u}_k}{dt} &= (\partial_{\mathbf{u}}f) \delta\hat{u}_k + (\partial_{\mathbf{B}_1\mathbf{v}}f) b_k\delta\hat{v}_k - D_0 b_k^2 \delta\hat{u}_k, \\ \frac{d\delta\hat{v}_k}{dt} &= (\partial_{\mathbf{v}}g) \delta\hat{v}_k + (\partial_{\mathbf{B}_1^\top\mathbf{u}}g) b_k\delta\hat{u}_k - D_1 b_k^2 \delta\hat{v}_k.\end{aligned}\tag{9.28}$$

It is interesting to notice that the leftover modes are those associated to the eigenvectors spanning the kernel space of both  $\mathbf{L}_0$  and  $\mathbf{L}_1$ . Since in the relevant case of a connected network, the eigenvector associated to the zero eigenvalue is the homogeneous one, i.e it is aligned to the stationary state  $\mathbf{u}^*$  of the nodes, it follows that  $\delta\mathbf{u}$  will never have a component along this eigenvector. However, we need to consider

the projection of  $\delta\mathbf{v}$  onto the eigenvectors  $\psi_1^l$  associated to the zero eigenvalues of  $\mathbf{L}_1$ , obtaining

$$\frac{d\delta\hat{v}_l}{dt} = (\partial_{\mathbf{v}}g) \delta\hat{v}_l. \quad (9.29)$$

Hence these modes are always stable due to the second condition in Eq. (9.23).

The instability is realized if the linear system (9.28) admits at least one unstable mode; more precisely we have to compute the eigenvalues of the matrix

$$\mathbf{J}_k = \begin{pmatrix} \partial_{\mathbf{u}}f - D_0b_k^2 & b_k\partial_{\mathbf{B}_1\mathbf{v}}f \\ b_k\partial_{\mathbf{B}_1^\top\mathbf{u}}g & \partial_{\mathbf{v}}g - D_1b_k^2 \end{pmatrix}, \quad (9.30)$$

and determine if there is  $k$  for which the associated eigenvalue,  $\lambda(b_k)$ , has a positive real part. Let us notice that the latter is usually named dispersion relation in the literature. The eigenvalues of  $\mathbf{J}_k$  can be obtained by solving

$$\lambda^2 + \lambda\Gamma_1(b_k^2) + \Gamma_2(b_k^2) = 0. \quad (9.31)$$

where  $\Gamma_1(b_k^2)$  and  $\Gamma_2(b_k^2)$  are given by

$$\Gamma_1(b_k^2) = b_k^2(D_1 + D_0) - (\partial_{\mathbf{v}}g + \partial_{\mathbf{u}}f), \quad (9.32)$$

$$\Gamma_2(b_k^2) = a_2b_k^4 + a_1b_k^2 + a_0, \quad (9.33)$$

with

$$\begin{aligned} a_2 &= D_0D_1, \\ a_1 &= -\left(D_1\partial_{\mathbf{u}}f + D_0\partial_{\mathbf{v}}g + \partial_{\mathbf{B}_1^\top\mathbf{u}}g \partial_{\mathbf{B}_1\mathbf{v}}f\right), \\ a_0 &= \partial_{\mathbf{u}}f \partial_{\mathbf{v}}g. \end{aligned} \quad (9.34)$$

Since both the leading coefficient of Eq.(9.31) and  $\Gamma_1(b_k^2)$  are positive, the existence of a solution with positive real part requires that  $\Gamma_2(b_k^2) < 0$  for some  $k$ <sup>2</sup>. Let us observe that  $\Gamma_2(b_k^2)$  given by Eq. (9.33) is a parabola in  $b_k^2$  with positive concavity,  $a_2 = D_0D_1 > 0$ , and positive constant term,  $a_0 = \partial_{\mathbf{u}}f \partial_{\mathbf{v}}g > 0$ . Therefore, to satisfy the condition  $\Gamma_2(b_k) < 0$  with a real  $b_k$ , a necessary condition is

$$D_0\partial_{\mathbf{v}}g + D_1\partial_{\mathbf{u}}f + \partial_{\mathbf{B}_1^\top\mathbf{u}}g \partial_{\mathbf{B}_1\mathbf{v}}f > 0. \quad (9.35)$$

By using these conditions we can guarantee that  $\Gamma_2(b_k^2) < 0$  if the minimum of the parabola is negative. A straightforward computation returns the condition

$$\frac{\left(D_0\partial_{\mathbf{v}}g + D_1\partial_{\mathbf{u}}f + \partial_{\mathbf{B}_1^\top\mathbf{u}}g \partial_{\mathbf{B}_1\mathbf{v}}f\right)^2}{4D_0D_1} > \partial_{\mathbf{u}}f \partial_{\mathbf{v}}g. \quad (9.36)$$

Let us observe that differently from the classical Turing framework, such condition depends on the diffusive coefficients separately and not on their ratio.

In conclusion, we have hence found the conditions for the onset of Turing instability for topological signals whose dynamics is described by Eq. (9.19), namely the stability of the homogeneous solution given by Eq. (9.23) and the existence of at least one unstable mode according to Eqs. (9.35) and (9.36). Moreover the roots of Eq. (9.31)

<sup>2</sup>This is a consequence of the Descartes' rule of signs for polynomials.

are given by  $\lambda_{1,2} = -\Gamma_1 \pm \sqrt{\Gamma_1^2 - 4\Gamma_2}$ , but  $\Gamma_1 > 0$  and  $\Gamma_2 < 0$ , and thus  $\lambda_{1,2}$  are real numbers. Consequently, the corresponding patterns are stationary.

Let us now note that as expected, when the topological signals on nodes and links are not coupled by the Dirac reaction term, i.e., when

$$F(\Phi, \mathcal{D}\Phi) = F(\Phi) = \begin{pmatrix} f(\mathbf{u}) \\ g(\mathbf{v}) \end{pmatrix}, \quad (9.37)$$

we can never have Turing patterns. In fact in this case we would have  $\partial_{\mathbf{B}_1^\top \mathbf{u}} g = 0$ ,  $\partial_{\mathbf{B}_1 \mathbf{v}} f = 0$  and Eq. (9.35) cannot be satisfied together with Eq. (9.23). A major result of this study is that the Turing instability of the topological signals of a network will be never localized only on nodes or only on links but will always involve both nodes and links signals. Moreover, we also obtain that if the original signals  $\Phi = (\mathbf{u}, \mathbf{v})^\top$  display a Turing pattern, the projected dynamics of  $\mathcal{D}\Phi = (\mathbf{B}_1 \mathbf{v}, \mathbf{B}_1^\top \mathbf{u})^\top$  also does.

### 9.1.2 Numerical results on a benchmark network

The aim of this section is to validate the above results with a numerical study. To focus on the novelty of the framework and to remove unnecessary complicated features, we will build a toy model with cubic nonlinearities to test our theory (the square lattice implementation can be found in Appendix B). By keeping the same notation as before, i.e.,  $\mathbf{u}$  is the signal on the nodes and  $\mathbf{v}$  that on the links, the equations of our model read

$$\begin{aligned} \dot{\mathbf{u}} &= -a\mathbf{u} - b\mathbf{u}^3 + c\mathbf{B}_1 \mathbf{v} - D_0 \mathbf{L}_0 \mathbf{u}, \\ \dot{\mathbf{v}} &= -\alpha\mathbf{v} - \beta\mathbf{v}^3 + \gamma\mathbf{B}_1^\top \mathbf{u} - D_1 \mathbf{L}_1 \mathbf{v}, \end{aligned} \quad (9.38)$$

where  $a, b, c, \alpha, \beta, \gamma$  are non-negative real parameters and each non linear function is to be intended component-wise.

System (9.38) admits  $(u_0 \mathbf{h}, v_0 \mathbf{h}) = (0, 0)$  as equilibrium point. By computing the Jacobian of the system evaluated at this point and removing the space contribution due to  $\mathbf{h}$ , we get

$$\mathbf{J}_0 = \begin{pmatrix} \partial_{\mathbf{u}} f & \partial_{\mathbf{B}_1 \mathbf{v}} f \\ \partial_{\mathbf{B}_1^\top \mathbf{u}} g & \partial_{\mathbf{v}} g \end{pmatrix} = \begin{pmatrix} -a & c \\ \gamma & -\alpha \end{pmatrix}.$$

The system exhibits a Turing instability if the above parameters satisfy the conditions (9.23), (9.35) and (9.36), that we now rewrite

$$a > 0 \quad \alpha > 0, \quad c\gamma > \alpha D_0 + a D_1, \quad (9.39)$$

$$(c\gamma - \alpha D_0 - a D_1)^2 > 4 D_0 D_1 a \alpha, \quad (9.40)$$

and the simplicial complex is such that  $\mathbf{h} \in \ker \mathbf{L}_1$ . A simple example of a 1-dimensional simplicial complex satisfying the latter condition is provided by a network of 16 nodes, whose nodes degrees are even and with closed loops. Note that the latter is chosen to be a subset of a square lattice. The Turing pattern associated to this dynamics are evident when one monitors nodes as well as link signals. In Fig. 9.2.a the nodes and links are colored according to the asymptotic concentration of respectively  $u$  and  $v$  and we can thus have a geometric view of the emerging pattern. On the other hand a dynamical view is presented in Fig. 9.2.c – d where we report the nodes

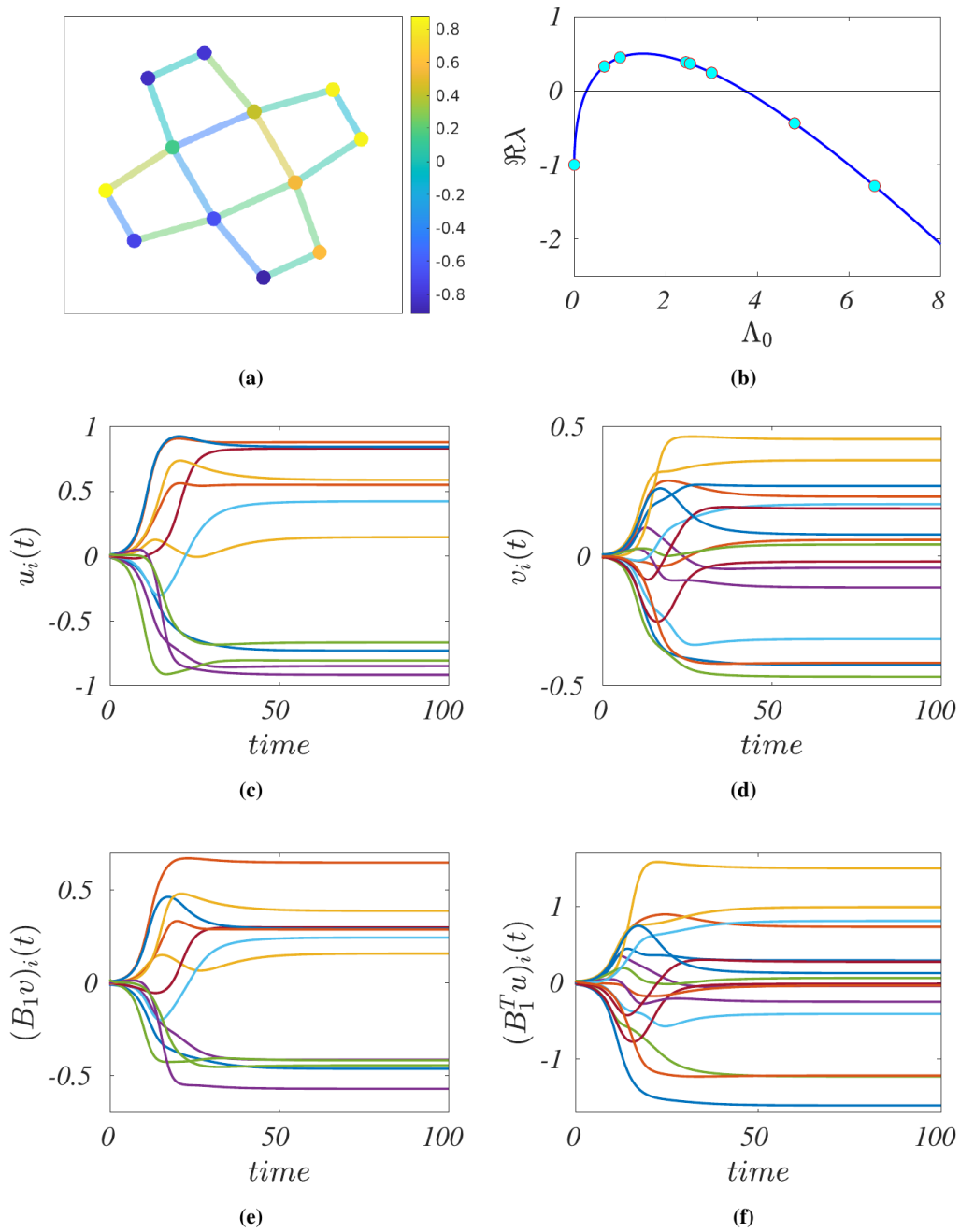


Figure 9.2: *a)* Turing patterns for the species on the nodes and on the links described by model (9.38) on a network satisfying the conditions for the homogeneous equilibrium; *b)* dispersion relation: in blue we depict the continuous curve, computed as a function of the continuous parameter  $b_k^2$ , while the cyan dots are the actual dispersion relation, where now the onset of the Turing instability is a function of the (real) spectrum of  $\mathbf{L}_0$ ; in panels *c)* and *d)* we depict time series of the two species  $u$  and  $v$  on the nodes and the links, respectively, while panels *e)* and *f)* show the time series of the projection of the two species with the action of the boundary operator  $B_1$ . The parameters are  $a = \alpha = b = \beta = \gamma = D_0 = D_1 = 1$  and  $c = 6$ ; the initial perturbation is  $\sim 10^{-2}$ .

concentration,  $u_i(t)$ , and link concentration,  $v_i(t)$ , as a function of time. From this figure one can clearly appreciate the onset of the instability at short time, pushing the initial conditions far from the equilibrium state  $(u_0, v_0) = (0, 0)$ , and the stationary asymptotic behavior of the solution. Interestingly we observe that the projected dynamics also display a Turing pattern (see Fig.9.2.e and Fig.9.2.f). In Fig. 9.2b we show the dispersion relation and we emphasize the unstable modes triggering the instability (dots).

### 9.1.3 Dirac cross-diffusion term

We now consider the dynamics including the Dirac cross-diffusion terms. In particular we will cover the linear cross-diffusion case. Topological signals on nodes and links can be coupled by a linear cross-diffusion term, leading to the reaction-diffusion dynamics

$$\dot{\Phi} = F(\Phi, \mathcal{D}\Phi) - \tilde{\gamma}\mathcal{D}\Phi - \gamma\mathcal{L}\Phi, \quad (9.41)$$

where the dynamical state of the network is captured by the vector  $\Phi = (\mathbf{u}, \mathbf{v})^\top$ . The diagonal  $(N_0 + N_1) \times (N_0 + N_1)$  matrix  $\tilde{\gamma}$  of cross-diffusion coefficients is here chosen to have block structure

$$\tilde{\gamma} = \begin{pmatrix} D_{01} & 0 \\ 0 & D_{10} \end{pmatrix}. \quad (9.42)$$

In particular the coupled dynamics of the topological signals  $u$  and  $v$  can be re-written as

$$\begin{aligned} \frac{d\mathbf{u}}{dt} &= f(\mathbf{u}, \mathbf{B}_1\mathbf{v}) - D_{01}\mathbf{B}_1\mathbf{v} - D_0\mathbf{L}_0\mathbf{u}, \\ \frac{d\mathbf{v}}{dt} &= g(\mathbf{v}, \mathbf{B}_1^\top\mathbf{u}) - D_{10}\mathbf{B}_1^\top\mathbf{u} - D_1\mathbf{L}_1\mathbf{v}. \end{aligned} \quad (9.43)$$

It is pretty straightforward to prove that system (9.41) can be mapped onto (9.16) and thus results from the previous section can be used to derive the conditions under which the reaction-diffusion dynamics with the linear cross-diffusion term displays Turing patterns. We notice that by putting

$$F(\Phi, \mathcal{D}\Phi) - \tilde{\gamma}\mathcal{D}\Phi = \tilde{F}(\Phi, \mathcal{D}\Phi), \quad (9.44)$$

Eq.(9.41) reduces to Eq.(9.16) with Dirac reaction term given by  $\tilde{F}(\Phi, \mathcal{D}\Phi)$ , i.e. it reduces to

$$\dot{\Phi} = \tilde{F}(\Phi, \mathcal{D}\Phi) - \gamma\mathcal{L}\Phi. \quad (9.45)$$

It follows that the conditions for the onset of the Turing instability can be trivially obtained directly from Eq. (9.35) and Eq. (9.36) by making the substitutions

$$\partial_{\mathbf{B}_1\mathbf{v}}f \rightarrow \partial_{\mathbf{B}_1\mathbf{v}}f - D_{01}, \quad \partial_{\mathbf{B}_1^\top\mathbf{u}}g \rightarrow \partial_{\mathbf{B}_1^\top\mathbf{u}}g - D_{10}. \quad (9.46)$$

This allows us to obtain that in the case with linear Dirac cross-diffusion terms we can observe the onset of the Turing instability when in addition to Eq.(9.23), the following two conditions are satisfied:

$$\begin{aligned} A &= D_0\partial_{\mathbf{v}}g + D_1\partial_{\mathbf{u}}f + (\partial_{\mathbf{B}_1\mathbf{v}}f - D_{01})(\partial_{\mathbf{B}_1^\top\mathbf{u}}g - D_{10}) > 0, \\ A^2 &> 4D_0D_1\partial_{\mathbf{u}}f\partial_{\mathbf{v}}g. \end{aligned} \quad (9.47)$$

Let us stress a major consequence of these conditions, i.e., the cross-diffusion term is the driver for the instability. Indeed the cross-diffusion term enforced through the Dirac operator allows the onset of Turing patterns also in situations where patterns can never emerge if we silence cross-diffusion. In particular we can observe Turing patterns in presence of Dirac-type crossed-diffusion patterns, also when the reaction term only depends on  $\Phi$  but not on  $\mathcal{D}\Phi$ , i.e.,

$$F(\Phi, \mathcal{D}\Phi) = F(\Phi) = \begin{pmatrix} f(u) \\ g(v) \end{pmatrix}, \quad (9.48)$$

as long as Eq. (9.23) and Eqs. (9.1.3) hold which can occur as long as  $D_{01}D_{10} > 0$ . Let us recall that, as discussed in the previous section, under the latter assumption (9.48) Turing patterns cannot develop in absence of linear cross-diffusion terms.

## 9.2 Conclusions

In this Chapter exploited the fundamental concepts of Algebraic topology in order to formulate a reaction-diffusion dynamics of topological signals defined on nodes, links, and higher-order simplices of simplicial complexes. In this framework, each species of reactants lives on simplices of a given dimension, for instance in a simplicial complex of dimension  $d = 2$  one would consider three species living on nodes, links and triangles respectively. Species associated to simplices of different dimension can be coupled thanks to the Dirac operator which projects a signal defined on  $n$ -dimensional simplices either one dimension up or one dimension down. In the proposed reaction-diffusion dynamics, the coupling can then be enforced either by a Dirac reaction term or/and Dirac cross-diffusion terms. After discussing the general framework valid for simplicial complexes of arbitrary dimension, we focus on the reaction-diffusion dynamics of topological signals defined on networks, i.e., coupling the dynamics between links and nodes, and we establish conditions for the onset of the Turing instability. The latter conditions are derived when signals of different dimension are only coupled with the Dirac reaction term, as well as when they are also coupled by a linear Dirac cross-diffusion term. We have found that the Turing patterns arising from the reaction-diffusion dynamics of topological signals are never localized only on nodes or links of the network. Instead they always involve both node and link signals. Moreover, the projection of the link signals on the nodes, and the projection of the node signals onto the links are shown to also display a Turing pattern. We also observe that when the reaction term does not depend on the projected signal, the Turing pattern can be observed only in presence of a linear Dirac cross-diffusion term. Our results are validated on a small toy model for the reaction-diffusion of topological signal on a network, and on simulations of square lattices with periodic boundary conditions.

After examining the behavior of reaction-diffusion systems in complex structures like simplicial complexes, we are now turning our attention to synchronization dynamics. Both topics are influenced by the complex topologies they're situated in, from simple nodes and links to more elaborate structures like faces and cell complexes. Our earlier research highlighted the role of the Dirac operator in how topological signals interact and spread across the network. This operator serves as a conduit for



signals to traverse and interact across different dimensions of the network.

As we proceed, we will shift our focus from Turing patterns to the dynamics of synchronization in these topological signals. Moving from a situation where the eigenvectors of the coupling operators are set unstable to one where stability is prioritized. While these multi-dimensional signals are increasingly discussed in scientific literature, our understanding of their collective behavior is still in its early stages. This upcoming Chapter aims to integrate topology and non-linear dynamics to explore the conditions necessary for synchronization in these signals.

Specifically, simplicial complexes can pose challenges for the synchronization of signals in odd dimensions, while cell complexes offer avenues to mitigate these issues. This highlights the dual role that topology can play in either hindering or facilitating synchronization processes.

Central to this discussion is the significance of the spectral properties of network-defined operators like the adjacency matrix and the Laplacian. As we delve further, these spectral elements not only provide a theoretical foundation but also have a direct impact on the behavior of these systems. For instance, in machine learning algorithms like multi-layer perceptrons (MLPs), it is the spectral structure of the adjacency matrix that informs feature extraction. Similarly, in the dynamics of simplicial complexes, the spectrum of the Dirac operator plays a critical role.



# Chapter 10

## Global Synchronization in Simplicial Complexes

Synchronization is a widespread phenomenon at the root of several biological rhythms or human made technological systems [168], [169]. Synchronization refers to the spontaneous ability of coupled oscillators to operate at unison and thus exhibit a coherent collective behavior. Global synchronization is the resulting phenomenon where all oscillators behave in the same way. Traditionally synchronization has been studied when identical [170], [171] or non-identical oscillators [172], [173] are defined on the nodes of a network and are coupled by the network links. However, to capture the function of many complex systems, e.g., brain networks [174], [175], social networks [176] and protein interaction networks [177], it is important to go beyond pairwise interactions and consider higher-order interactions [178] between two or more nodes instead. Lately, synchronization of identical and non-identical oscillators defined on the nodes of higher-order networks has been a field of intense research activity. Global synchronization of identical oscillators have been first formulated for special topologies (a  $p$ -regular hypergraph) [179] and for a peculiar Laplace operator obtained from the hyper-adjacency matrix [180] while recently a general and comprehensive theoretical framework has been proposed in [181] to study dynamical systems defined on hypergraph with heterogeneous hyperedges size distribution, the latter influencing also the Laplace matrix. Partial synchronization of non-identical nodes oscillators has been investigated using a variation of the Kuramoto model leading to explosive transitions [118].

Recently the formulation of higher-order topological Kuramoto model [108], [182], [183] has demonstrated that topological signals of any dimension can synchronize leading to either continuous or to explosive synchronization transitions. These results concern partial synchronization while an important question is whether global synchronization of topological signals can ever be achieved.

The aim of this Chapter is to determine the topological and dynamical conditions under which global topological synchronization of identical topological oscillators can be observed. Relying on the use of higher-order Laplacian matrices and the extension of the Master Stability Function (MSF) to simplicial and cell complexes dynamics, we will be able to tackle this problem emphasizing the difference existing among the two frameworks.

Anticipating on our results we can state that on simplicial complex we observe topological obstruction: given a simplicial complex of dimension  $K$ , if the topologi-

cal signal is defined on an odd-dimensional simplex of dimension  $k < K$  then, global synchronization is not possible. On the other hand if the simplex has an even dimension, then we can have global synchronization provided the simplex is *balanced* (see hereafter) and the model parameters allow for it. Interestingly we show that cell complexes can overcome topological obstruction and some topologies can sustain global synchronization of signals of any dimension.

## 10.1 Dynamical Framework

Let us now consider a topological  $k$ -dimensional signal encoded in a  $k$  dimensional cochain  $\mathbf{x} : C_k \rightarrow \mathbb{R}^d$  which assigns to every chain  $C_k$  (linear combination of  $k$ -simplices) values on  $\mathbb{R}^d$ . The  $k$ -topological signal has elements  $\mathbf{x}_i = \mathbf{x}(\sigma_k^{(i)}) = (x_i^1, \dots, x_i^d)$  defined on the  $i$ -th oriented  $k$ -simplex,  $\sigma_k^{(i)}$ , as we have already said in the precedent Chapter, according to the properties of the  $k$ -cochains we have  $\mathbf{x}(-\sigma_k^{(i)}) = -\mathbf{x}(\sigma_k^{(i)})$ , being the discrete analogous of differential forms on manifolds. Let us assume the value of the topological signal on every simplex  $i$  follows the same dynamics and evolves according to  $\dot{\mathbf{x}}_i = \mathbf{f}(\mathbf{x}_i)$ , for some odd nonlinear function  $\mathbf{f} : \mathbb{R}^d \rightarrow \mathbb{R}^d$ .

Assume now the  $k$ -simplex to belong to a  $K$ -simplicial complex,  $K \geq k$ , and assume the existence of a diffusive-like nonlinear interaction among adjacent simplices of the same dimension:

$$\frac{d\mathbf{x}_i}{dt} = \mathbf{f}(\mathbf{x}_i) - \sum_{j=1}^{N_k} L_k(i, j) \mathbf{h}(\mathbf{x}_j) \quad \forall i = 1, \dots, N_k, \quad (10.1)$$

where  $\mathbf{h} : \mathbb{R}^d \rightarrow \mathbb{R}^d$  is some odd non linear coupling function. This equation generalizes the dynamics of identical oscillators anchored to each node [170] to the scenario in which identical oscillators are defined on higher-dimensional simplices or cells. Please note that requiring odd functions  $\mathbf{f}(\mathbf{x}_i)$  and  $\mathbf{h}(\mathbf{x}_i)$  is necessary for higher-order topological signals with  $k > 0$  in order to ensure invariance under change of orientation of each simplex  $i$  (see Chapter 8, where topological signals are introduced). For node dynamics ( $k = 0$ ) the existence of a global synchronized state is automatically determined by the properties of the graph Laplacian whose kernel is spanned by  $\mathbf{u} = (1, \dots, 1)^\top$ , indeed we have  $\beta_0 = 1$ , one connected component. The stability of this global synchronized state is instead determined by the celebrated Master Stability Function (MSF) [170], [171].

Given the growing interest in topological signals, a key question is how these classic results of nonlinear dynamics on networks extend to nonlinear dynamics of topological signals on simplicial complexes. Anticipating our results, we will show that the topology and the combinatorics of the higher-order Laplacian will not always ensure the existence of a globally synchronized state, and moreover since the dimension of the kernel of  $\mathbf{L}_k$  can be bigger than one, also the MSF will differ from the standard case.

Let us then fix a reference stable solution  $\mathbf{s}(t)$  of the uncoupled system,  $\dot{\mathbf{x}}_i = \mathbf{f}(\mathbf{x}_i)$ . We are interested in determining the conditions under which the state having each simplex  $i$  either in the state  $\mathbf{x}_i = \mathbf{s}(t)$  or in  $\mathbf{x}_i = -\mathbf{s}(t)$  is also a stable solution of the coupled system (10.1). Namely the latter exhibits a *global synchronous* behavior in which all simplices display the same dynamics of the isolated simplices when we

account for differences of sign, determined by their orientation. To give an intuition of this result, take the case of link signals indicating fluxes: saying that a flux is  $J$  on the link  $[i, j]$  oriented from  $i$  to  $j$  is the same as saying that the flux is  $-J$  on the link  $[i, j]$  oriented from  $j$  to  $i$ . Therefore global synchronization is observed where the dynamics of each simplex is the same, when factoring out the sign due to the choice of orientation.

Let us now introduce the vector  $\mathbf{v} = (v_1, \dots, v_{N_k})^\top \in \{-1, 1\}^{N_k}$  such that the globally synchronized state is given by  $\mathbf{x}_i = v_i \mathbf{s}(t)$ . If we turn our attention to the case of interacting topological signals coupled by the Hodge Laplacian operator whose dynamics is dictated by Eq. (10.1), a *necessary condition* to observe global synchronization, is that  $\sum_j L_k(i, j) \mathbf{v}_j = 0$ , namely the 'homogeneous' signal is in the kernel. Note that since the degeneracy of the zero eigenvalue of  $\mathbf{L}_k$  is given by the  $k$ -th Betti number, it follows that the Hodge Laplacian  $\mathbf{L}_k$ , when  $k > 0$ , can admit more than one eigenvectors of this form. However we are not guaranteed that an arbitrary simplicial or cell complex will have at least one such eigenvectors. In order to give a concrete example of such topologies we mention that a  $d$ -dimensional regular cell complex which tessellates a torus has  $\binom{d}{k}$  such eigenvectors while any simplicial or cell complex without any  $k$ -dimensional hole will have  $k$ -th Betti number  $\beta_k = 0$  and hence any such eigenvector. This is a property that is in strike contrast with what happens for  $k = 0$  where any connected network has one and only one constant eigenvector  $(1, \dots, 1)^\top$  in its kernel. Note that these combinatorial constraints have also topological consequences, as we will see, indeed they imply that global synchronization is allowed on topologies structures having holes that span the entire structure as hypersphere and tori.

Let us recall that  $\ker \mathbf{L}_k = \ker \mathbf{B}_k \cap \ker \mathbf{B}_{k+1}^\top$ , thus the  $\sum_j L_k(i, j) \mathbf{v}_j = 0$  condition ultimately returns to require  $\mathbf{v}^\top \mathbf{B}_{k+1} = 0$  (condition **i**) and  $\mathbf{B}_k \mathbf{v} = 0$  (condition **ii**).

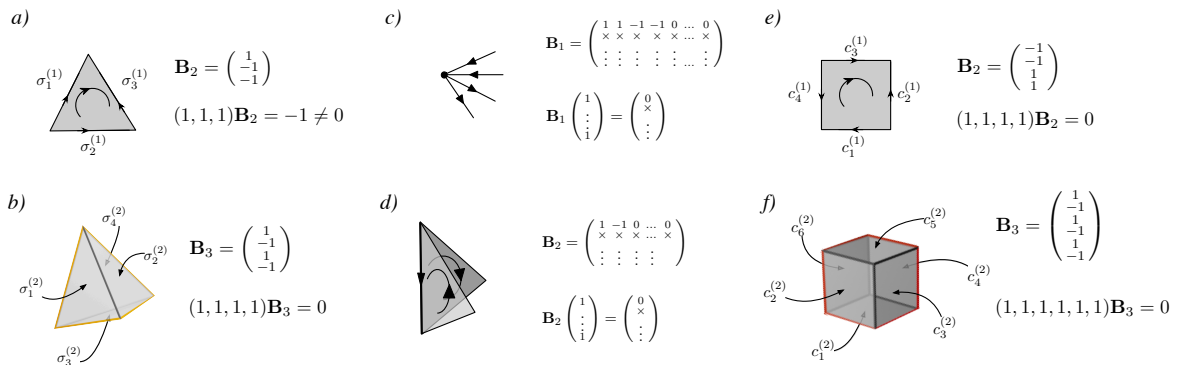


Figure 10.1: Schematic description of conditions **i**) (panels a,b,e,f) and **ii**) (panels c and d) for topological signals defined on 1-dimensional cells (links, panels in top row) and 2-dimensional cells (triangles or squares, panels in bottom row) in which we assume that there is an orientation such that  $\mathbf{w} = \mathbf{u} = (1, 1, \dots)^\top$ . In the case of simplicial complexes (panels a, d) condition **i**) cannot be satisfied for signals defined on 1-dimensional simplices while in the case of cell complexes (panels e, f) condition **i**) it can be satisfied. Condition **ii**) can be satisfied on simplicial and cell complexes as long as the simplices are balanced (see panels c, d) for the simplicial complex case).

The first condition has a striking consequence. If  $k$  is an odd number, because

any  $(k + 1)$ -simplex contains an odd number of  $k$ -faces, then condition **i)** cannot be ever satisfied. On the contrary if  $k$  is even, then any  $(k + 1)$ -simplex contains an even number of  $k$ -faces condition **i)** can be realised (see Fig. 10.1a-b). On the other hand condition **ii)** can be satisfied by imposing a suitable condition of the  $(k - 1)$ -faces of the  $k$ -simplex, which we call *balanced condition*. In particular if  $\mathbf{v} = \mathbf{u} = (1, \dots, 1)^\top$  this condition can be satisfied requiring every  $(k - 1)$ -face to be adjacent to an even number of  $k$ -simplices and moreover to be oriented coherently with half of them (see Fig. 10.1c-d) (See Appendix A and following sections for further details).

Therefore for even values of  $k$ , global synchronization can be achieved while if  $k$  is odd we observe, as long as  $\mathbf{v}^\top \mathbf{B}_{k+1} \neq \mathbf{0}$ , a *topological obstruction* to the onset of global synchronization. Interestingly for  $K$ -dimensional signals having  $\mathbf{B}_{K+1} = \mathbf{0}$ , only the balanced condition remains (i.e., condition **ii)**) which is automatically satisfied for the vector  $\mathbf{v} = \mathbf{u}$  if the simplicial complex is a closed manifold without boundary. Hence  $K$ -dimensional topological signals defined on closed  $K$  dimensional manifolds can always achieve global synchronization for arbitrary value of  $K$ .

A similar derivation can be generalized and extended to topological signals defined on the  $k$ -dimensional cells of cell complexes. In particular the conditions to achieve global synchronization on a cell complex are unchanged and given again by condition **i)** and **ii)**. However the combinatorics of cell complexes is different from the one of simplicial complexes. Take for instance a cell complex whose network skeleton is formed by a  $d$ -dimensional square lattice with periodic boundary conditions, i.e., a regular tessellation of  $d$ -dimensional torus. Then every cell of dimension  $k + 1 > 0$  has a even number of  $k$ -dimensional faces therefore condition **i)** can be satisfied also if  $k$  is odd (see Fig. 10.1e-f). This implies that on cell complexes we can overcome topological obstruction. Until now we have focused on the combinatorial implication of the conditions **i)** and **ii)**. Now that we have characterized the conditions for the onset of global synchronization we shall focus on the construction of such discrete manifolds. We will start with the more straightforward one and then move to the more elaborate, leaving the cell complex as last.

## 10.2 Construction of eulerian discrete manifolds

In this section we show how to construct a simplicial or cell complex capable of hosting a globally synchronized state. The example shown are all related to a more general analysis of the problem that is not shown in this Chapter but it is presented in Appendix A. In the latter we relate the topological problem to a graph problem showing how the proprieties of  $\mathbf{B}_k$  and  $\mathbf{B}_{k+1}$  of a given simplicial complex can be transferred to the analysis of a flow over a graph.

### 10.2.1 2-simplicial complex

The aim of this section is to build a 2-simplicial complex satisfying conditions **i)** and **ii)** for  $k = 2$  and  $k = 0$ . In this case the simplex is formed by triangles connected among them to pave a 2-torus. We could have used the orientation given by the node labels, but again we decided to introduce an alternative orientation, following a *bottom up* approach, aimed at directly obtaining that  $\mathbf{u} = (1, \dots, 1)^\top$  lies in  $\ker \mathbf{L}_2$ . Also in this case this choice does not change the topological properties of the simplicial complex

with respect to those of the same simplicial complex oriented by using the node labels orientation.

Consider the case of topological signals defined on 2-simplices, i.e., the faces of the 2-simplicial complex. Conditions **ii)** becomes thus  $\mathbf{B}_2\mathbf{u} = 0$ , while condition **i)** is automatically satisfied because  $\mathbf{B}_3 = 0$ , being the 3-simplices not allowed. The basic element is thus an oriented triangle, i.e., a 2-simplex  $A = [acb]$  whose links are oriented as  $[ab]$ ,  $[bc]$  and  $[ca]$  (see Fig. 10.2a). Consider now a second triangle  $B = [bcd]$  sharing the link  $[bc]$  with the previous one, and whose links are oriented as  $[bc]$ ,  $[cd]$  and  $[db]$ . Because of the chosen orientations the shared link determine a  $+1$  and a  $-1$  entries in the matrix  $\mathbf{B}_2$ , namely in the product  $\mathbf{B}_2\mathbf{u}$  they contribute with a  $0$  (see Fig. 10.2b). We can then repeat such construction by alternating the two kinds of triangles above defined and eventually identify among them the “vertical left” and “vertical right” sides (red lines in Fig. 10.2c) and also the “horizontal top” and “horizontal bottom” sides (blue lines in Fig. 10.2c). In this way we have obtained a 2-torus whose surface is paved with triangles and such that by construction we have  $\mathbf{B}_2\mathbf{u} = 0$ . Let us observe that because nodes are not oriented no special construction is required for the case  $k = 0$  except each node to have an even degree, condition that holds true in the present case, indeed  $k_i = 6 \forall i$ , and they are “correctly” oriented.

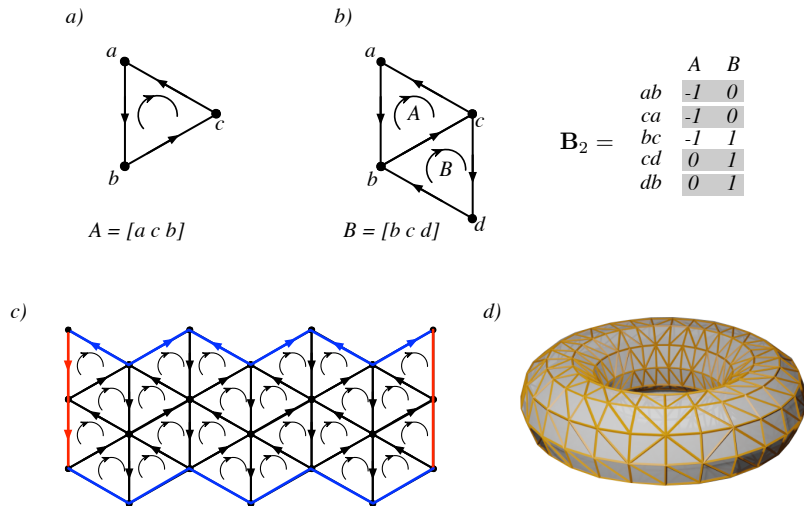


Figure 10.2: **A 2-torus paved with triangles.** We consider 2-torus whose surface is paved with oriented triangles (panel a)) organized in such a way that once they share a common link then the latter is coherent with the orientation of one triangle and incoherent with the second; they contribute thus to the matrix  $\mathbf{B}_2$  with entries  $+1$  and  $-1$  (panel b)). Such construction is repeated in the longitudinal and vertical direction and eventually including periodic boundary conditions (panel c)).

### 10.2.2 3-simplicial complex

The aim of this section is to show how to build a 3-simplicial complex,  $\mathcal{P}$ , satisfying conditions **i)** and **ii)** for  $k = 2$  and  $k = 0$ , and thus admitting a synchronous manifold for topological signals defined on faces or nodes. The nodes of this simplicial complex are placed on a 2-dimensional square grid with periodic boundary conditions. However each four nodes of any (imaginary) square placquette of this grid are belonging to a

distinct tetrahedron. As it can be directly shown this structure admits an eigenvector  $\mathbf{v}$  with elements  $|v_i| = 1$  for any orientation induced by the node labels, as long as  $k \in \{0, 2\}$ .

This interesting structure can be obtained by looking at topologies that satisfy conditions **i)** and **ii)** for an orientation such that  $\mathbf{v} = \mathbf{u} = (1, \dots, 1)^\top$  will lie in  $\ker \mathbf{L}_2$  (and of course in  $\ker \mathbf{L}_0$ ). One can show that the resulting oriented simplicial complex has the the same topological properties of the simplicial complex oriented by using the node label order, the reason being that their basis are related by an invertible linear transformation.

For sake of concreteness we will consider the case of topological signals defined on 2-simplices, belonging to a 3-simplicial complex. The case  $k = 0$  being associated to a standard network synchronization phenomenon fits in the standard global synchronisation phenomenon on networks. Conditions **i)** and **ii)** translate thus into  $\mathbf{B}_3^\top \mathbf{u} = 0$  and  $\mathbf{B}_2 \mathbf{u} = 0$ . The sought simplicial complex will thus be obtained by assembling together nodes, links, triangles and tetrahedra. Let us notice that once such simplicial complex has been built we show that he can support global synchronization also for topological signals defined on 0-simplex, i.e., on nodes, while this will not be the case for topological signals defined on 1-simplex, links, or 3-simplex, tetrahedra.

Let us start by considering a tetrahedron (see Fig. 10.3a) and let us fix an orientation for its faces,  $A$ ,  $B$ ,  $C$  and  $D$ . For sake of concreteness, we define a face to be positively oriented if one circulates between its vertices in a anticlockwise manner, and negatively oriented on the opposite (see Fig. 10.3b). To better appreciate the relative orientations of the faces we show the same tetrahedron but “flattened”, in this way it is clear the presence of two positively oriented ( $A$  and  $C$ ) and two negatively oriented ( $B$  and  $D$ ) faces, which implies that condition **i)** is satisfied, i.e.,  $\mathbf{B}_3^\top \mathbf{u} = 0$ .

By labelling the vertices with  $a$ ,  $b$ ,  $c$  and  $d$ , we can assign an orientation to the links (Fig. 10.3b); together with the faces orientation we can compute the incidence matrix  $\mathbf{B}_2$  (see Fig. 10.3c). From the latter we can appreciate the existence of four links (rows shaded in light grey in the incidence matrix, also coloured in orange in Fig. 10.3a) that are coherently oriented with two faces, thus in the product  $\mathbf{B}_2 \mathbf{u}$  they return a  $+2$ . There are also two links (white rows in the incidence matrix) that are not coherently oriented with the two faces to which they are incident, hence in the product  $\mathbf{B}_2 \mathbf{u}$  they contribute with a  $0$ .

The idea to build the sought 3-simplicial complex satisfying the constraints  $\mathbf{B}_2 \mathbf{u} = 0$  and  $\mathbf{B}_3^\top \mathbf{u} = 0$ , is to add another oriented tetrahedron with an opposite orientation for its faces with respect to the previous one, but with the same orientation of the links (see Fig. 10.4a). This “opposite” tetrahedron will have two positively oriented and two negatively oriented faces and thus it satisfies again  $\mathbf{B}_3^\top \mathbf{u} = 0$ , moreover its incidence matrix  $\mathbf{B}_2$  will be the opposite of the previous one (panel b) of Fig. 10.4). This means that we can select an edge in the first tetrahedron (say  $ad$ ) and a second edge in the opposite tetrahedron (say  $a'd'$ ) and “glue” them (see Fig. 10.4c). In this way this edge will be incident with four faces, and it will be coherently oriented with two of them and non coherently oriented with the remaining two. Hence the row of the incidence matrix of the “glued” tetrahedron,  $\mathbf{B}_2^{(\text{glue})}$ , will contain two “ $+1$ ” and two “ $-1$ ”, the remaining entries being “ $0$ ” (see Fig. 10.4d). In conclusion this link will contribute to  $0$  in the product  $\mathbf{B}_2^{(\text{glue})} \mathbf{u}$ . As we have shown the construction of this simplicial complex has been realized as a mixture of the *top down* and *bottom up*



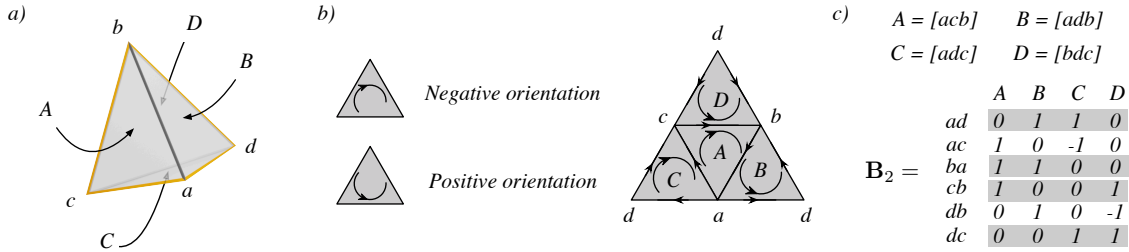


Figure 10.3: **An oriented tetrahedron.** We consider an oriented tetrahedron, i.e., 3-simplex, both in a 3D view (panel a)) and in a “flatten” 2D one (panel b)). Given the orientation of the faces and of the links proposed in panel b), we can compute the incidence matrix  $\mathbf{B}_2$  (panel c)). We can observe the presence of four links whose contribution to  $\mathbf{B}_2 \mathbf{u}$  is non zero, they correspond to the grey rows in the matrix and to the orange links in panel a).

approach.

By construction the 3-simplicial complex obtained by gluing the two opposite tetrahedra contains a link,  $(ad)$  that is incident with four faces and it is coherently oriented with two of them and incoherently with the other two (see Fig. 10.5a). Moreover such simplicial complex contains six links each one belonging to two faces and coherently oriented with both of them, hence they will contribute with a non zero entry in the product  $\mathbf{B}_2 \mathbf{u}$ . In panel a) of Fig. 10.5 they are colored in blue in the case of the first considered tetrahedron and in red for the second one; let us for short say that such links have the “wrong property”. To tackle this issue we repeat the construction show on Fig. 10.4 by alternating “positively” and “negatively” oriented tetrahedra along the two opposite sides of the square obtained by “flattening” each tetrahedron and considering only the sides with the “wrong property” (see Fig. 10.5b). The resulting structure resemble to a “waffle” (see Fig. 10.5c) to which we imposed periodic boundary conditions (see Fig. 10.5c). Eventually taking  $L$  squares along the longitudinal direction and  $M$  squares along the latitudinal direction, we obtain the sought 3-simplicial complex that by construction satisfies  $\mathbf{B}_2 \mathbf{u} = 0$  and  $\mathbf{B}_3^\top \mathbf{u} = 0$ . It can thus support a synchronous manifold for topological signals defined on 2-simplex but not on the 1-simplex; indeed  $\mathbf{B}_2^\top \mathbf{u} \neq 0$ , nor the 3-simplex because  $\mathbf{B}_3 \mathbf{u} \neq 0$ . Let us observe that in this case  $\mathbf{B}_4^\top \mathbf{u} = 0$ , being the tetrahedron the simplex with the largest dimension and thus  $\mathbf{B}_4 = 0$ , however we do not have global synchronization of 3-topological signals because the used simplex is not a closed manifold without boundary. Now that we have realized the construction of a simplicial complex capable of hosting a globally synchronized state we will move to the cell-complex case.

### 10.2.3 3-cell complex: 3D-torus

In the following we will build a 3-cell complex made of nodes, links, squares (i.e., 2-cells) and cubes (i.e., 3-cells) capable to support global synchronization for topological signals of any dimensions,  $k = 0, 1, 2, 3$ .

Let us consider a cube formed by six 2-cells, the squares (see Fig. 10.6a), and we can orient the faces to have three positive and three negative contributions to  $\mathbf{B}_3^\top \mathbf{u}$  is in such a way the latter sums to zero (see panels b)). It is important for the following

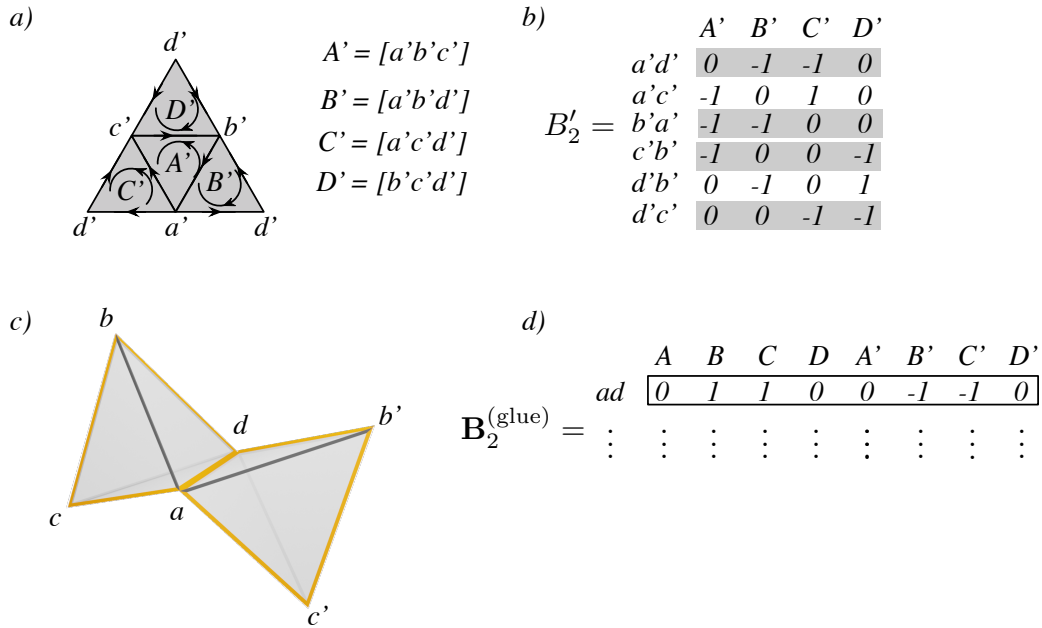


Figure 10.4: **An opposite oriented tetrahedron.** We show a tetrahedron whose faces have an opposite orientation with respect to the ones of Fig. 10.3, while the tetrahedron and its edges are oriented in the same way in both cases (panels a) and b)). In panel c) we show the “gluing” process of these two tetrahedra along two selected links,  $ad$  and  $a'd'$ , in such a way the shared link is now incident with four faces and its is coherently oriented with two of them and non coherently with the two other ones (panels c) and d)).

construction to incoherently orient opposite faces (see panels c)-d) and e)), this is because once we will “glue” together several cubes, the shared faces should have the right orientation to ensure  $\mathbf{B}_2\mathbf{B}_3 = 0$ .

We then orient the links according to a left-right on the horizontal direction, bottom-up in the vertical one and front-back in the transversal direction (see again panels a) and b)). Once faces and links have been oriented we can compute the matrices  $\mathbf{B}_k$ ,  $k = 1, \dots, 3$  and we can observe that six links exist, three of which contributing with “+2” (solid blue line) or “-2” (dashed red line) to  $\mathbf{B}_2\mathbf{u}$ .

Once we have obtained the oriented cube, we can “glue” together several copies of the same cube along the three directions (see Fig. 10.7). Because of the orientations given to the faces (see Fig. 10.6), the face shared by each couple of cubes will have the right orientation. Moreover this gluing process ensures also that  $\mathbf{B}_2\mathbf{u} = 0$ ; indeed as it can be observed from Fig. 10.7 the unique link incident with four cubes, hence eight faces, has a zero contribution to  $\mathbf{B}_2\mathbf{u}$ . By considering for instance the configuration shown in panel a), the vertical link common to the four cubes A, B, C and D will contribute with +2 (it has been coloured in blue according to the scheme presented in Fig. 10.6) because of the two faces of the B cube, with -2 (it has been coloured in red according to the scheme presented in Fig. 10.6) because of the two faces of the D cube and 0 (it has been coloured in black according to the scheme presented in Fig. 10.6) to the faces in the cubes A and C. The same analysis can be done for the constructions in the vertical direction (panel b)) and in the transversal one (panel c)). Finally by

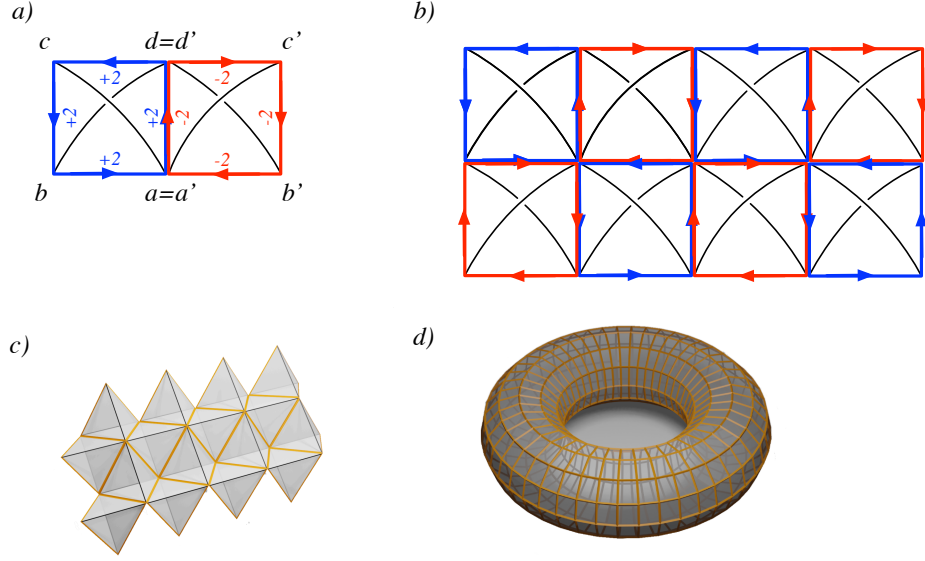


Figure 10.5: **The 3-simplicial complex  $\mathcal{P}$ .** We schematically show the construction of the 3-simplicial complex obtained by gluing together opposite tetrahedra and then by identifying the opposite sides of the square composed by the tetrahedron links contributing with a non zero value to  $\mathbf{B}_2\mathbf{u}$  (the edges in blue and red in panels a-b) or in yellow in panel c)).

imposing periodic boundary conditions in the three directions we obtain a 3D torus with the required properties. Let us observe that the links orientation together with the periodic boundary conditions ensure  $\mathbf{B}_1\mathbf{u} = 0$ .

### 10.3 Master Stability Equation for Topological Signals

Let us now assume the reference solution  $\mathbf{s}(t)$  to be also a solution of the coupled system (10.1), then by introducing the distance from the reference orbit,  $\delta\mathbf{x}_i = \mathbf{x}_i - \mathbf{s}(t)$ , we can derive its time evolution by linearising Eq. (10.1):

$$\frac{d\delta\mathbf{x}_i}{dt} = \mathbf{J}_f(\mathbf{s})\delta\mathbf{x}_i - \sum_{j=1}^{N_k} L_k(i, j)\mathbf{J}_h(\mathbf{s})\delta\mathbf{x}_j \quad \forall i = 1, \dots, N_k \quad (10.2)$$

being  $\mathbf{J}_f(\mathbf{s})$  (resp.  $\mathbf{J}_h(\mathbf{s})$ ) the Jacobian of the function  $\mathbf{f}$  (resp.  $\mathbf{h}$ ) evaluated on the reference solution.

The matrix  $\mathbf{L}_k$  being symmetric, it admits an orthonormal basis,  $\phi_k^{(\alpha)}$ , associated to eigenvalues  $\Lambda_k^{(\alpha)}$ ,  $\alpha = 1, \dots, N_k$ , namely  $\mathbf{L}_k\phi_k^{(\alpha)} = \Lambda_k^{(\alpha)}\phi_k^{(\alpha)}$ . In particular, since we work under the assumption that the simplicial complex is balanced,  $\phi_k^{(1)} \sim (1, \dots, 1)^\top \in \mathbb{R}^{N_k}$ ,  $\Lambda_k^{(\alpha)} = 0$  for  $1 \leq \alpha \leq \beta_k$  and  $\Lambda_k^{(\alpha)} > 0$  for all  $\alpha > \beta_k$ .

Let us decompose the deviation vectors  $\delta\mathbf{x}_i$  onto this eigenbasis:  $\delta\mathbf{x}_i = \sum_{\alpha} \delta\mathbf{x}^{(\alpha)}(\phi_k^{(\alpha)})_i$ . Then linearising the dynamical equation, we obtain

$$\frac{d\delta\mathbf{x}^{(\alpha)}}{dt} = [\mathbf{J}_f(\mathbf{s}) - \Lambda_k^{(\alpha)}\mathbf{J}_h(\mathbf{s})] \delta\mathbf{x}^{(\alpha)} \quad \forall \alpha = 1, \dots, N_k \quad (10.3)$$

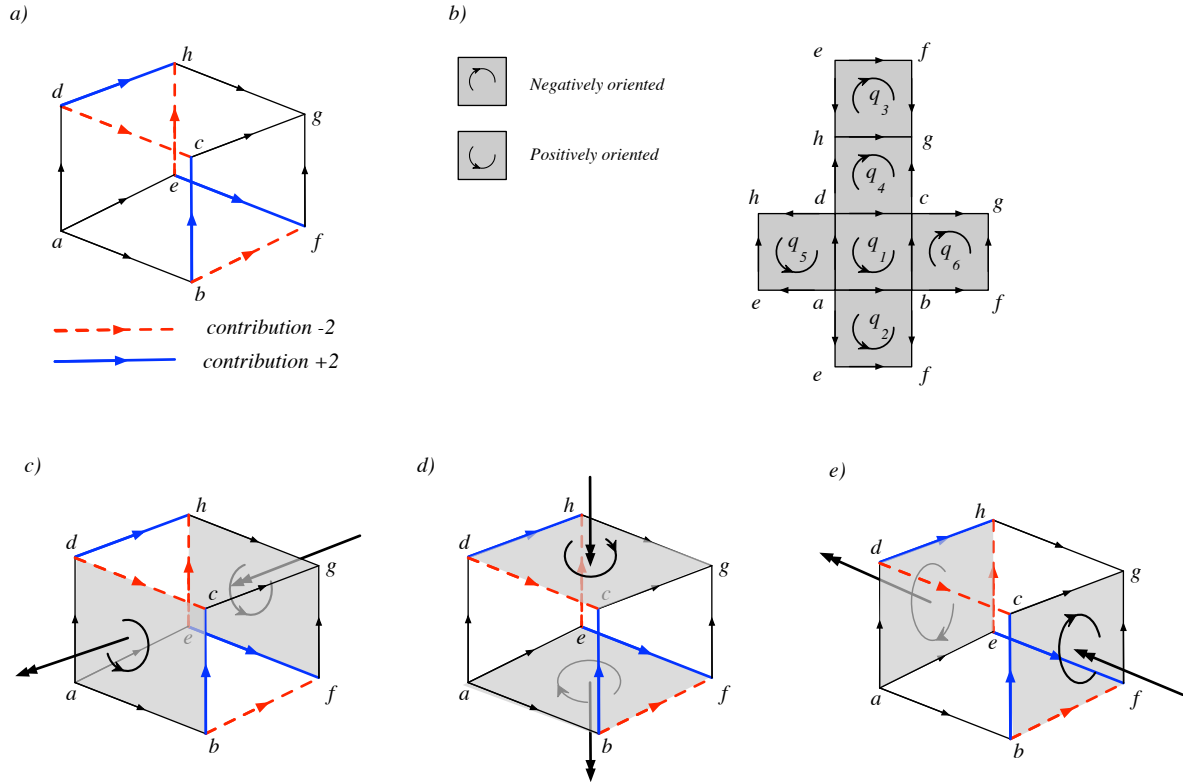


Figure 10.6: **An oriented cube.** We show an oriented cube, both in a 3D view (panel a)) then in a “flatten” 2D one (panel b)). Given an orientation of the faces and of the links (see panels a) and b)), we can compute the incidence matrix  $\mathbf{B}_2$  and identify the link whose contribution to  $\mathbf{B}_2\mathbf{u}$  is not zero. This allows to emphasise 6 links forming a closed loop (thicker dashed red and solid blue lines in panel a)) in the cube. Let us observe that opposite faces of the cube must be incoherently oriented (see panels c)-d) and e)).

Perturbations aligned with the kernel do not change the stability of the uncoupled system, therefore only the perturbations orthogonal to the kernel can modify the stability of the reference solution. This observation, in the realm of simplicial complexes is far more relevant with respect to the graph case, as a kernel with dimension larger than one is very easily obtainable. This is the MSF in the framework of simplicial synchronization of topological signals. It is a linear, in general non autonomous, ODE parametrized by the eigenvalues  $\Lambda_k^{(\alpha)}$ , allowing to infer the stability character of the reference solution by looking at its spectrum.

### 10.3.1 Simplicial Stuart-Landau model

As an application of the general theory above introduced, let us consider the Stuart-Landau (SL) model [184]–[186] defined for topological signals of dimension  $k$  and  $d = 2$ . For  $k = 0$  the model describes a nonlinear oscillator anchored at each node, while for  $k = 1$  it can describe an oscillating flux associated to an edge linking two nodes. More precisely let us define  $w_j = x_j^1 + ix_j^2$  and let us consider the “local reaction” function  $f(\mathbf{x}) = f(w) = \sigma w - \beta w|w|^2$ , where  $\sigma = \sigma_{\Re} + i\sigma_{\Im}$  and  $\beta = \beta_{\Re} + i\beta_{\Im}$  are complex control parameters. We can prove that the uncoupled dynamics

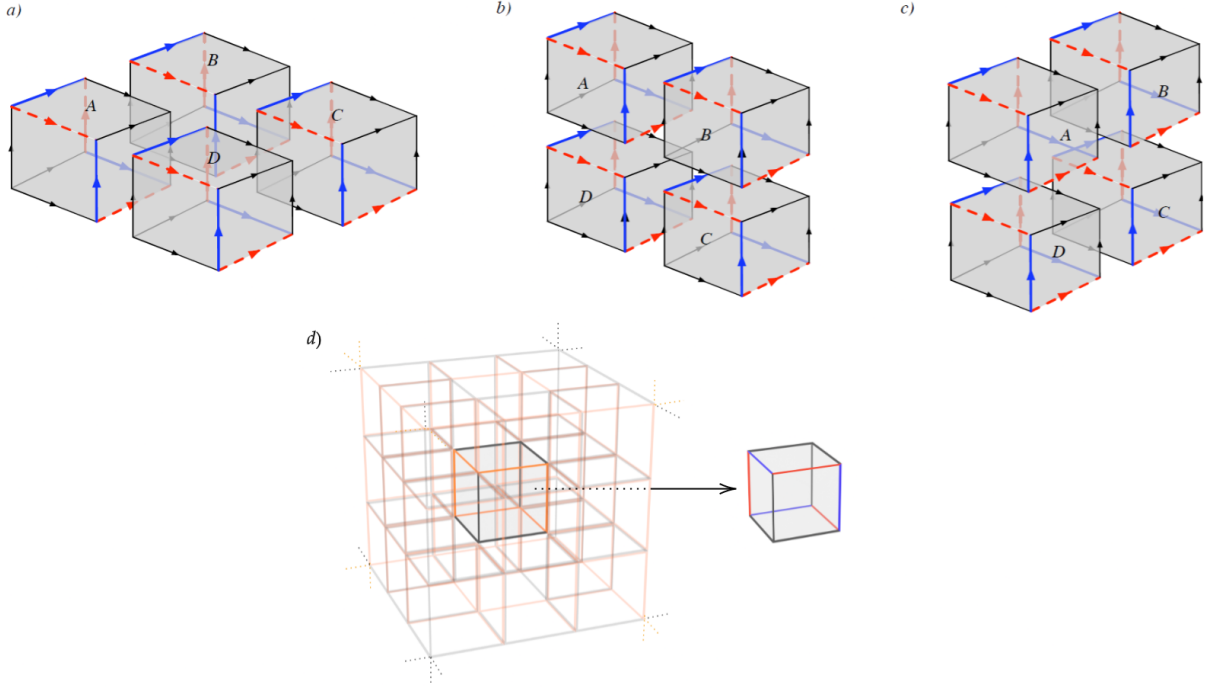


Figure 10.7: **Gluing together four oriented cubes along the three dimensions.** We show how to glue together four cubes in the horizontal (panel a)), vertical (panel b)) and vertical transversal (panel c)) plane. We can observe that in all cases the link common to eight faces (and four cubes) returns a null contribution to  $\mathbf{B}_1 \mathbf{u}$ , indeed it contributes with 0 to two couples of faces (thin black line), with +2 to a couple of faces (blue thick line) and with  $-2$  to the remaining couple of faces (dashed red line). The final construction is schematically depicted in panel d). The cube is repeated along each one of the three spatial dimensions and periodic boundary conditions are implemented connecting opposite faces of the cubic array (represented with a lower opacity). The basic element of the construction is shown in the centre. As in the other figures, the non zero contributing edges are shown in orange and in dark grey all the others. The edges coloured with the same color index as in panels a)-c) are also shown for orientation reference.

$\dot{w}_j = f(w_j)$  in each simplex  $j$  admits a limit cycle solution  $\hat{z}(t) = \sqrt{\sigma_{\Re}/\beta_{\Re}} e^{i\omega t}$ , where  $\omega = \sigma_{\Im} - \beta_{\Im}\sigma_{\Re}/\beta_{\Re}$ , that is stable provided  $\sigma_{\Re} > 0$  and  $\beta_{\Re} > 0$ , conditions that we hereby assume. We now consider the coupled dynamics Eq. (10.1) with nonlinear coupling function  $h(\mathbf{x}) = h(w) = \mu w |w|^{m-1}$ , where  $m$  is a positive integer and  $\mu = \mu_{\Re} + i\mu_{\Im}$  a complex parameter that sets the coupling strength. We study the stability of the reference limit cycle solution  $\hat{z}(t)$  (see Appendix C) and we prove that the system can globally synchronize, i.e., the dispersion relation is negative, only if the model parameters do satisfy  $\mu_{\Re} + \mu_{\Im}\beta_{\Im}/\beta_{\Re} > 0$ , and the simplicial complex is such that  $\mathbf{u} \in \ker \mathbf{L}_k$ . To measure global synchronization we compute the (generalised) Kuramoto order parameter  $R(t) = \left| \sum_j \rho_j(t) e^{i\theta_j(t)} \right| / N_k$ , where  $w_j(t) = \rho_j(t) e^{i\theta_j(t)}$  is the polar form of the complex signal. Then  $R(t) \rightarrow 1$  testifies the existence of phase and amplitude synchronization. Results shown in Fig. 10.8 provide numerical evidence of our theoretical predictions. In Fig. 10.8a-c we show the results obtained by

studying the SL model defined on top of a designed balanced 3-simplicial complex (see Section 10.2.2). The model parameters have been set to values allowing for a negative dispersion relation and indeed once the complex amplitudes are defined on 2-faces, i.e., triangles, the system globally synchronizes (see Fig. 10.8b). On the other hand, once the SL oscillators are defined on links the system cannot globally synchronize (see Fig. 10.8c). In Fig. 10.8d-f we provide an example of a cell complex which overcomes topological obstruction: a 3D square lattice with periodic boundary conditions. Such cell complex is made of nodes, link, squares and cubes (see Section 10.2.3). In this case case global synchronization can be achieved for signals of every dimension (see Fig. 10.8e-f for global synchronization of links and squares).

## 10.4 Conclusions

In this Chapter we have studied global synchronization of identical topological oscillators on simplicial or cell complexes. We have found that global synchronization of topological signals cannot be observed on arbitrary simplicial or cell complexes but that only some special higher-order network topologies can sustain such a dynamical state. This is in stark contrast with the corresponding scenario in networks where global synchronization can be observed in every network topology given proper dynamical conditions. By combining topology, and in particular the spectral properties of higher-order Laplacians, to nonlinear dynamics techniques such as the MSF, we have identified the topological and dynamical conditions under which identical topological oscillators can achieve global synchronization on simplicial or cell complexes. We have proved that global synchronization of odd dimensional topological signals is obstructed in simplicial complexes. This topological obstruction implies that on a  $K$ -dimensional simplicial complex we can never observe global synchronization of odd dimensional topological signals of dimension  $d < K$ . However, such obstruction is not present in cell complexes. In particular we show evidence that in specific topologies such as the  $d$ -dimensional square lattice with periodic boundary conditions global synchronization of topological signal of any dimension can be observed. These results significantly enrich our understanding of the relation between higher-order network topology and dynamics revealing collective phenomena of topological signals. Our study is relevant, for its inherent simplicity, to a wide spectrum of applications (neuroscience/biology/-social sciences) where many-body interactions involve signals defined on face or link in the framework of condensed matter [187].

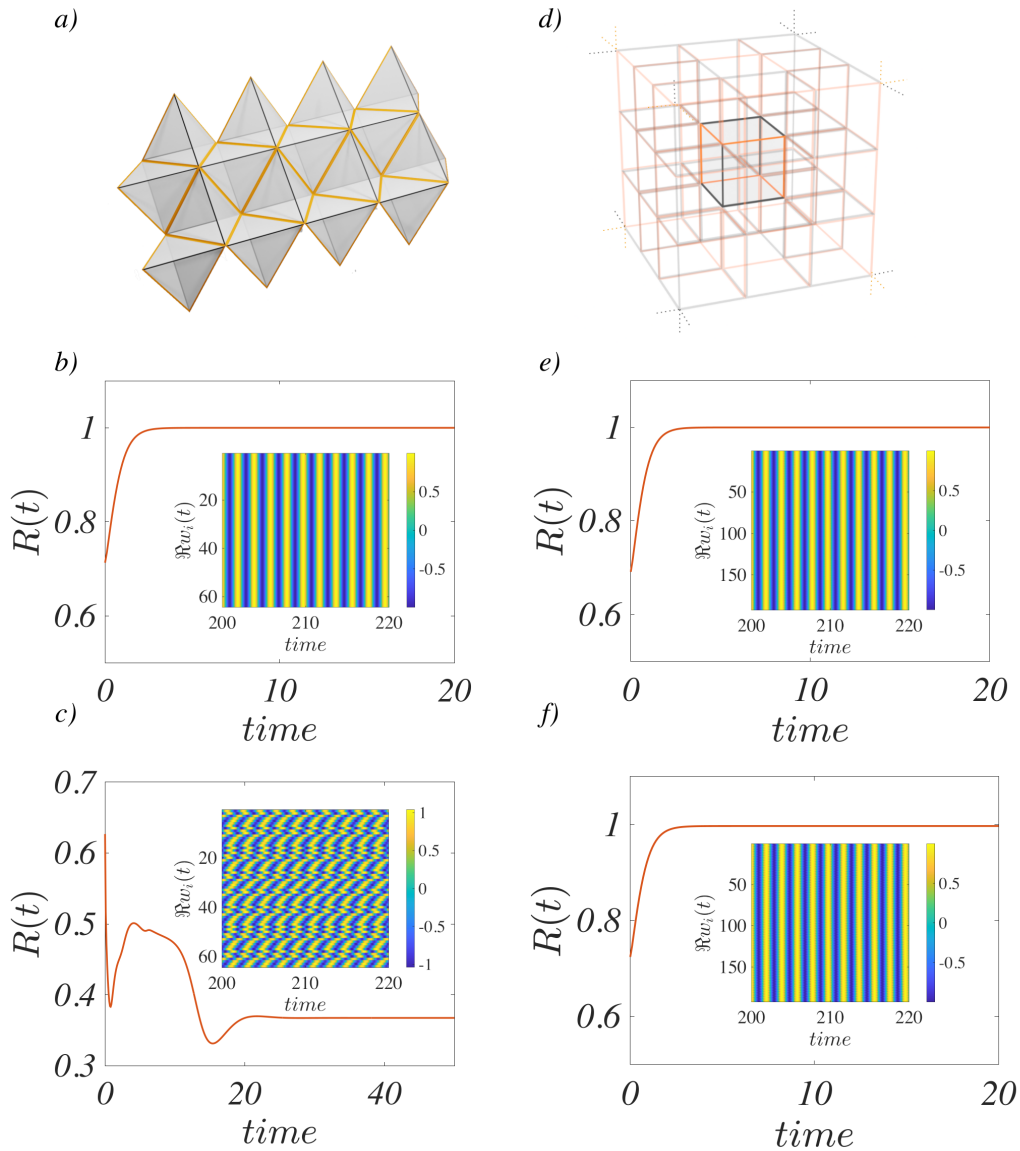


Figure 10.8: The Kuramoto order parameter  $R$  is plotted versus time  $t$  for the Stuart-Landau model of topological oscillators of the balanced simplicial and cell complexes represented in panels a) and d) respectively. Panels b) and c) refer respectively to the order parameter of triangles and links of the simplicial complex in panel a). Panels e) and f) refer respectively to the order parameter of the squares and links of the cell complex in panel d). The insets display the dynamical time series of the topological signals. It is clear that while on the links of the simplicial complex, the oscillators do not globally synchronize, the ones of the cell complex do support synchronization. The model parameters are  $\sigma = 1.0 + 4.3i$ ,  $\beta = 1.0 + 1.1i$ ,  $\mu = 1.0 - 0.5i$  and  $m = 3$  ensuring the negativity of the dispersion relation (see Appendix C).





# Chapter 11

## Conclusions and outcomes

As we reach the end of this thesis, it is apt to reflect on the intertwined pathways we have traversed. We began with an exploration into the spectral properties of neural networks and culminated in the investigation of signal dynamics within simplicial complexes. Although these two domains appear disparate at first glance, both investigations demonstrate the undeniable utility of spectral methods as analytical and optimization tools in the landscape of theoretical physics and machine learning.

In the first part of this research ventured into the territory of neural network interpretability and efficiency via the introduction of a novel concept: the Spectral Parametrization. We have shown that by translating neural network weights into the spectral domain, specifically the eigenvalues and eigenvectors of the adjacency matrix, one can gain a remarkable level of insight. Via this clever yet simple parametrization the optimization process changes and the eventual structure can be interpreted much more easily: the eigenvectors components are the features learned and the eigenvalues their respective relevance. Thanks to this understanding pragmatic algorithms for network slimming and spectral regularization have been developed. These algorithms not only compact the networks but also maintain their performance, addressing one of the central challenges in machine learning today. Moreover imposing an opportune regularization on the eigenvalues a much more feature oriented network structure appears, revealing a processing core reminiscent of the complexity of the wanted task. The invariance of this core, made by only the relevant neurons, tested against different initialization structures, let us speculate that the majority of the employed architecture can be slimmed and that the proposed regularization, mathematically and empirically solid, could be of great applicative interest.

We hope that with this research we have paved the way for a proper neural network analysis as only the relevant information is present in the structure. Indeed, the optimization process works synergically with the parametrization and the regularization exploring preferentially feature oriented and compact structures.

The spectral formulation and the parallelism with its relevance in network dynamics, where it is a crucial aspect for predicting the evolution, enables us to connect with the world of dynamical systems: we have created a context where "learning" means steering the dynamics stemming from an initial state (the input) towards a given attractor. This research line opens up a plethora of scenarios where the tools of

dynamical systems are efficiently and beneficially applied to understand the learning process.

Pivoting from this bridge between those areas we embarked on a comprehensive analysis of pattern formation and synchronization in simplicial complexes topological signals. Through the lens of the Dirac operator, we unravel the relationships that tie together topological signals dynamics and spectral properties. The discovered relationships have implications for understanding the behavior of complex systems, from neuronal networks to social interactions, governed by the same mathematical principles. For the first time we have been able to understand the contexts where topological signals of arbitrary dimension can synchronize and form organized interdimensional patterns, culminating with the development of a simplicial master stability function. The field of topological signal is still at its infancy and its beauty and mathematical rigour have been widely recognised and explored in this thesis. We sincerely hope that with the theoretical advances presented this powerful mathematical framework could soon unravel its true potential in relevant applications such as neuronal dynamics and likely, as we have briefly said, in condense matter. Indeed, the latter, where operators have been related to links and faces of discrete structures, seems a very promising unexplored territory for topological signals application, unravelling in a very natural way the recognised interplay between the lattice topology and the phenomenology.

Regarding our contribution to deep learning we would like to stress that, nowadays, a substantial number of political and social challenges have emerged due to the increasing effectiveness of neural networks. Unfortunately, their widespread adoption is not matched by a corresponding understanding of the underlying mechanisms or, importantly, their post-training content. While there is a growing array of models and techniques capable of explaining the operation of simpler neural networks, effective tools for understanding state-of-the-art architectures are still lacking. We believe this thesis can contribute to addressing this issue through a judicious blend of theory and computational methods, creating a technique to simplify these complex structures for easier analysis. Hopefully, by achieving this, we may one day render the learned content of neural networks interpretable, transforming them from merely a form of 'black-box oracle' in the hands of private corporations to an instrument for cultural enrichment.

# Appendix A

## Construction of a simplex satisfying $\mathbf{L}_k \mathbf{u} = 0$

### A.1 Introduction

The problem of finding a simplicial complex capable of supporting an homogeneous signal that lies in the the kernel of a boundary operator of a certain dimension is a mixture of topology and combinatorial can be tackled with a formalism we are about to explain.

The core idea behind it is to represent the action of a  $k$  boundary operators in terms of divergence of a fictitious vector field in a network made by three different types of nodes: one representing  $k$ -simplex and another  $k \pm 1$ .

Such vector field will represent how a homogeneous signal (on a given basis we are about to define) rebounds on the upper and lower faces  $\sigma_{k\pm 1}$  after the action of a boundary or coboundary operator.

The whole procedure we are about to explain has similarities similar to the one at page 50 of [188], which describes how to construct the dual complex. Let us also observe that in [188] the dual and primal complex are considered separately whereas we will represent them to be part of the same structure.

### A.2 Graph representation of Boundary operator $\mathbf{B}_k$

Let us start with the definition of the action of a boundary operator  $\mathbf{B}_k$  that acts on a  $k$  simplex and returns a  $k - 1$  simplex. In the following we proceed firstly by illustrating how to construct a bipartite network representation of the simplex where nodes are  $k$ -simplices and upper (or lower) faces and the way  $\mathbf{B}_k$  acts on the latter for every  $k$ . Then we will show how to exploit such representation to construct a simplicial complex whose Hodge Laplacian  $\mathbf{L}_k$  has the propriety that  $\mathbf{L}_k(1, \dots, 1) = (0, \dots, 0)$ . The construction process shown is valid for every simplicial complex of any order but, for simplicity we will show the construction for the case of a tetrahedron  $P$ , represented in Figure A.1 (a).

We focus on  $k = 2$  and therefore  $S_k = T$ , the space of triangles and  $S_{k-1} = e$ , the space of edges. In Figure (b) a flat version of the tetrahedron  $P$  is shown and the

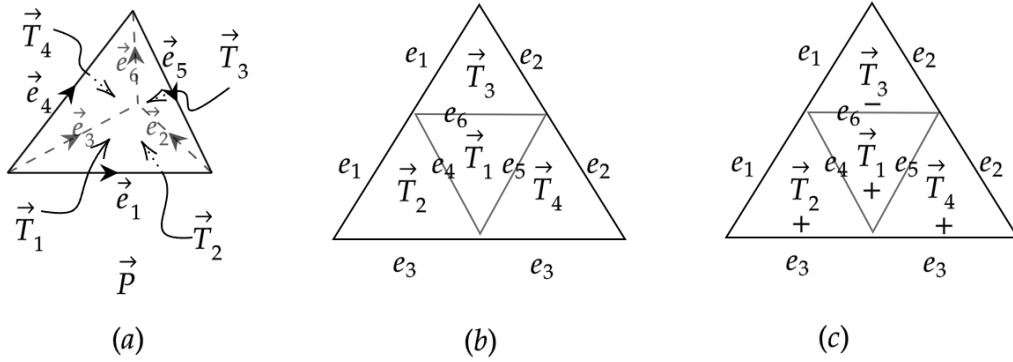


Figure A.1: This figure depicts a tetrahedron in three distinct representations. In subfigure (a), the 3-dimensional structure is shown with oriented edges indicated by arrows on the links, while the triangles are unoriented but labeled. Subfigure (b) presents a flattened version of the tetrahedron for clarity. In subfigure (c), the structure is again displayed, but the triangles are oriented arbitrarily. The symbols + and - are used, at this level, to describe the triangles that are coherently or incoherently oriented.

triangles have been labelled. The first and only thing, at this level, that we need to set, is the relative orientation of every triangle. Such orientation is fundamental and will be represented with a sign +, -. Intuitively as soon as we have defined a fixed  $T$  to be *positively oriented* and marked with +, every other triangle marked with + will be oriented *in the same way* and every one with - *in the opposite way*.

Defining the relative orientation of every triangle accounts for the choice of base in which the homogeneous vector  $\mathbf{h}$  is written in; a homogeneous signal means that it has exactly the same direction/orientation as every  $T$  in the basis chosen.

Let us orient them as shown in Figure (c), meaning that triangles 2,1,4 are oriented in the same way and every one of them is opposed to  $T_3$ .

We shall now describe how to represent a simplicial complex that has been described by a list of oriented simplices. More specifically we will focus on two dimensions at a time, namely  $(k, k + 1)$  and  $(k, k - 1)$ . The bipartite graph we are about to show is just a schematization of the interaction between  $\sigma_k$  by means of  $\sigma_{k-1}$ .

This could be done representing every  $\sigma_k = T$  with a white circle and every  $\sigma_{k-1} = e$  with a filled circle. Then every triangle is connected with every edge it contains. The resulting graph is represented in Figure A.2 (a). From this graph we can easily retrieve the description of every simplex in terms of its bounding  $\sigma_{k-1}$  and therefore the action of the  $k$ -boundary operator  $\mathbf{B}_k$ . The core idea is that if  $\sigma_k^{(1)}$  and  $\sigma_k^{(2)}$  share the same bounding  $\sigma_{k-1}$  and are oriented in the same way (+, + or -, -) then  $\sigma_{k-1} \subset \sigma_k^{(1)}$  and  $-\sigma_{k-1} \subset \sigma_k^{(2)}$  or, equivalently,  $-\sigma_{k-1} \subset \sigma_k^{(1)}$  and  $\sigma_{k-1} \subset \sigma_k^{(2)}$ . This is, basically, a *top down* way of coherently infer the orientation of a  $k - 1$  simplex.

Of course if they are oriented in opposite way then  $\sigma_{k-1} \subset \sigma_k^{(1)}, \sigma_k^{(2)}$ . Such rule can be exemplified in terms of the construction rule presented in Figure A.2 (b) leading to the graph in Figure A.3 (a). If two simplices  $\sigma_k^{(1)}, \sigma_k^{(2)}$  are oriented in the same way then when we apply the boundary operator to the homogeneous signal  $\mathbf{h} = \sigma_k^{(1)} + 1\sigma_k^{(2)}$  it will result in a mutual compensation on the shared  $\sigma_{k-1}$ , represented by the filled

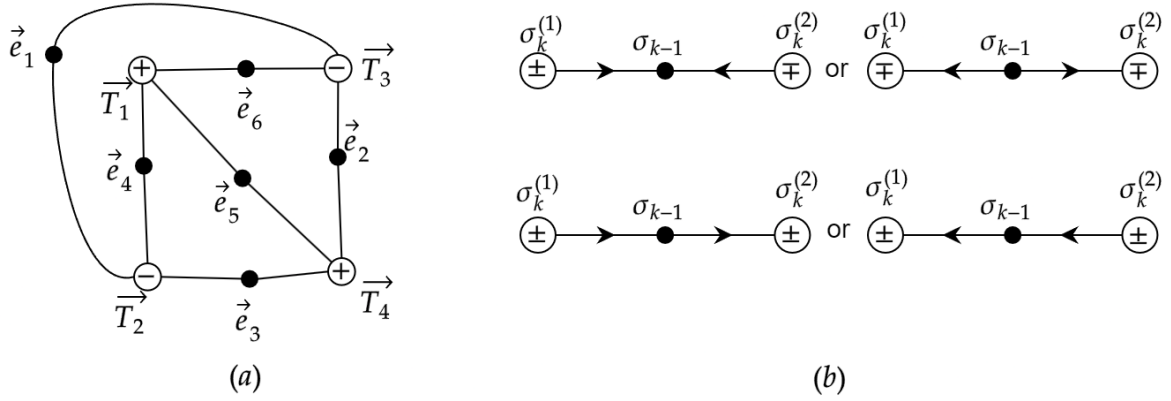


Figure A.2: In subfigure (a) we show a bipartite graph representation of  $P$  where only the triangles are oriented and represented as a white circle. The edges are black filled circles. Subfigure (b) schematically show the way of inferring the orientation (and therefore the action of the boundary operator  $\mathbf{B}_k$ ) of a shared  $\sigma_{k-1}$  when two  $\sigma_k$  are coherently or incoherently oriented.

circle. For example if  $\sigma_{k-1}$  is present inside the  $\sigma_k^{(1)}$  and  $-\sigma_{k-1}$  in  $\sigma_k^{(2)}$  and they are oriented in the same way the boundary on  $\sigma_{k-1}$  will be the sum of +1 contribute from  $\sigma_k^{(1)}$  (an arrow from the white node  $\sigma_k^{(1)}$  to the filled black node) and -1 contribute from  $\sigma_k^{(2)}$  (a reversed arrow from the white node  $\sigma_k^{(2)}$  to the filled black node). This situation is shown in the bottom left diagram of Figure A.2 (b).

Clearly if the two simplices are oriented in the opposite way both  $\sigma_k^{(1)}$  and  $\sigma_k^{(2)}$  will contribute with +1 (diagram top left) or -1 (top right) depending the way the  $\sigma_{k-1}$  simplex appears in their respective list of bounding simplices.

Using this formalism we can see at a glance the links with "non zero boundary": the filled nodes with non zero divergence with respect to the black arrows inserted after fixing the orientation of each  $\sigma_k$ .

The reader can thus appreciate that this construction formally describes the action of the boundary operator  $\mathbf{B}_k$  as a propriety of a vector field on a graph. This results in a much easier understanding of the proprieties of a given simplex and transfers the algebraic propriety of  $\mathbf{B}_k$  into proprieties of a given flow in a bipartite graph.

Indeed, for a more formal description, let  $e_j^*$  be the dual element of  $e_j$ . This is the linear operator that returns the  $j$ -th component of the chain. When considering a filled point corresponding to  $\sigma_{k-1}^{(j)} = e_j$ , it has an equal number of incoming and outgoing arrows. This implies that  $e_j^*(\mathbf{B}_2\mathbf{h}) = 0$ , meaning that the component on the  $j$ -th simplex is zero. On the other hand, if  $e_j$  has  $n_i$  incoming arrows and  $n_o$  outgoing arrows, then  $e_j^*(\mathbf{B}_2\mathbf{h}) = n_i - n_o$ .

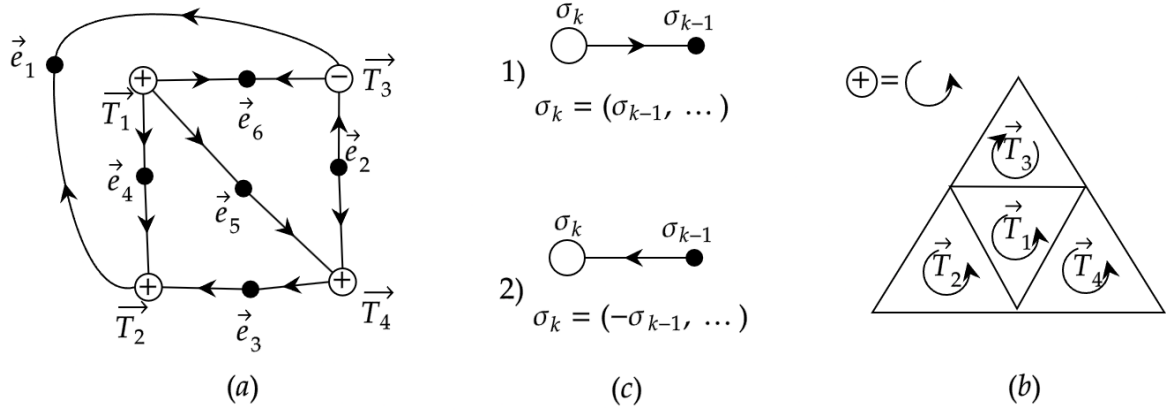


Figure A.3: Subfigure (a) shows the representation of the full bipartite graph of  $P$  with the vector field in each link that shows the action of the  $\mathbf{B}_2$  operator. Subfigure (b) shows the rule that needs to be followed in order to obtain the list of  $k - 1$  simplex that belongs to every  $k$  simplex in the simplicial complex once the arrows are set.

### A.3 Top-down construction of $P$

Once the orientation of every  $T$  is given we can infer the mutual orientation of every bounding  $e$ . The representation via listing every  $\sigma_k \in S_k$  can be obtained<sup>1</sup> with the rule in Figure A.3 (b), leading to:

$$T_1 = (-e_4, e_5, -e_6) \qquad T_2 = (e_4, e_1, e_3) \qquad (\text{A.1})$$

$$T_3 = (e_1, e_2, -e_6) \qquad T_4 = (-e_3, -e_5, e_2) \qquad (\text{A.2})$$

The simplex (the tetrahedron  $P$  in our example) can therefore be visualized after imposing an absolute geometrical orientation. For the sake of visualization we shall set  $\oplus$  equal to *counter clockwise* as in A.3 (c). The list representation consequently induce the orientation of every edge represented as an arrow in Figure A.4 (a).

Let us now summarize the steps that we have followed. First of all we have set an arbitrary orientation of every  $T$ . Then we have inferred the orientation of every  $e$  so that it can be a well defined bounding simplex i.e a (sub)face of a given  $T$ ; this has been done following the rule in Figure A.2 (b). Under the hood this sets the basis of both spaces:  $S_2 = T$  and  $S_1 = e$ . The base on  $T$  is the one where the homogeneous signal stands for our oriented triangles all taken one time, the base on  $e$  is the one induced by the neighbourhood definition of  $B_2$  applied to every  $T_j \in T$ .

We therefore have, in Figure A.3 all the information to understand the action of the boundary operator  $\mathbf{B}_2$  on a given signal we just need to transfer, according to the arrows, the activity from the white nodes to the black ones (we will come again to this point later).

<sup>1</sup>The representation via listing can only be obtained after having "correctly oriented" to ensure that we are constructing simplexes, namely satisfying  $\mathbf{B}_{k-1} \mathbf{B}_k = 0$

**Change of basis**

Such formalism is able to account also for a change in the base of  $S_k$  or  $S_{k-1}$ . An orientation flip of either  $\sigma_k$  or  $\sigma_{k-1}$  can be represented in this formalism as in Figure A.4 (b), i.e., with a change in the orientation of every arrow around an empty ( $\sigma_k$ ) or filled ( $\sigma_{k-1}$ ) circle. Such effect is schematically shown in panel (b) of figure A.4. If

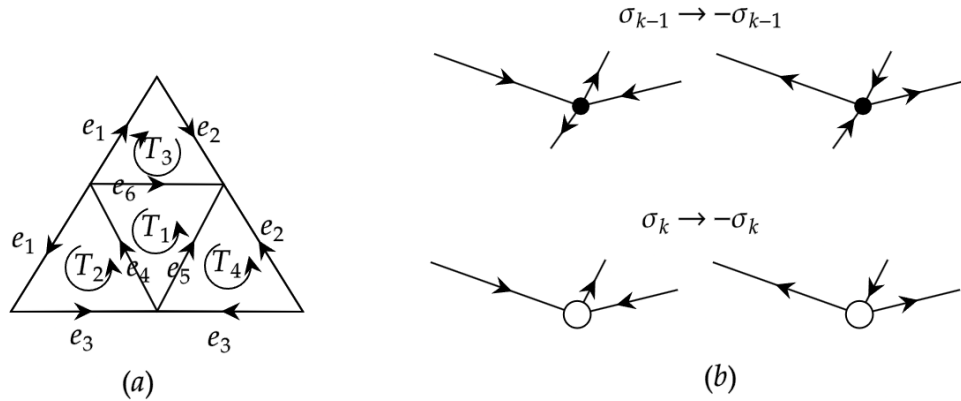


Figure A.4: On panel (a) representation of the flattened tetrahedron  $P$  expliciting all the orientation induced by our construction. On panel (b) the effect on the arrows of every node (i.e. the Boundary Operator action) when a given simplex is flipped.

we change  $\sigma_k^{(i)} \rightarrow -\sigma_k^{(i)}$  the zero boundary signal will therefore be the one such that  $\sigma_k^{*(i)}(u') = -1$ .

**A.3.1  $\mathbf{B}_{k+1}^\top$**

If we consider also the  $S_{k+1}$  faces by starting from the base definition at level  $S_k$ , the relative orientation has, again, to be taken into account. We shall start from the construction of the bipartite graph in the same way as before, having, this time, filled circle for  $S_k$  and empty one to denote  $S_{k+1}$ . The relative orientation of every  $S_{k+1}$  will be represented as  $+$  or  $-$  inside every circle. This time we will be interested to infer the listed representation of  $P$  once we have fixed the orientation of every  $T$  and arbitrarily fixed the one of  $P$ . Now we cannot employ the same top-down approach as before because the neighbourhood representation of  $\mathbf{B}_k^\top$  is less trivial. Therefore, we will proceed by setting the orientation of the level  $S_3 = P$  and then construct the operator  $\mathbf{B}_2^\top$  action (the arrows' orientations on the bipartite graph).

Let us therefore assume that the tetrahedron  $P$  is oriented  $+$  and represent the graph in Figure A.5 (a). Now we shall apply basically the same idea as before, schematically represented in Figure A.5 (b) and implemented in (c). As can be noticed the rule is the opposite as before (Figure A.3 (b)) and this is due to the fact that we are interested in the action of  $\mathbf{B}_{k+1}^\top$ , the transpose of the boundary operator <sup>2</sup>. The action of  $\mathbf{B}_{k+1}$  can be shown simply flipping the arrows but, of course, leaving the rule unchanged.

<sup>2</sup>The signal should be intended as starting from the filled circle here and from the empty in the precedent setting

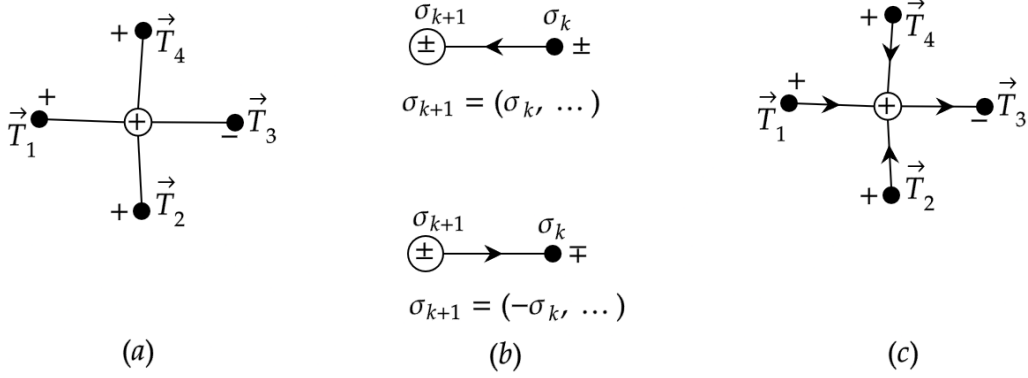


Figure A.5: Panel (a) shows the orientation of 2 and 3 simplices, connected according to the tetrahedron's boundary. In (b) we show the construction rule for this case and in (c) the resulting  $B_2$  effect.

This will lead exactly to the same representation as before (it is just a matter of book-keeping). Thanks to that rule we can infer the via list representation of  $P$ , namely  $P = (T_1, T_2, -T_3, T_4)$ .

Of course with this method we obtain  $\mathbf{B}_k \mathbf{B}_{k+1} = 0$  and therefore the correct representation of Boundary and Coboundary operators. We point out that, without this construction rule, it is very hard to properly construct the simplex without starting to set the orientation of the largest dimensional simplices.

## A.4 Construction of *eulerian* simplicial complex

In the following we will construct a Simplicial complex made by  $S_k$  and  $S_{k+1}$  simplex capable of supporting an homogeneous signal  $\mathbf{h}$  with the following properties:

- $\sigma_k^{*(j)}(\mathbf{h}) = T_j^*(\mathbf{h}) = 1 \quad \forall j$
- $\mathbf{B}_k \mathbf{h} = \mathbf{B}_2 \mathbf{h} = 0$
- $\mathbf{B}_{k+1}^\top \mathbf{h} = \mathbf{B}_3^\top \mathbf{h} = 0$

Recalling that  $\ker \mathbf{L}_k = \ker \mathbf{B}_k \cap \ker \mathbf{B}_{k+1}^\top$  this implies  $\mathbf{L}_k \mathbf{h} = 0$  making it possible for the simplicial complex to host a globally synchronized state.

Let us start with the construction of the base  $\{T_j\}$  that encodes the signal orientation. In order for every tetrahedron in our simplicial complex to fulfil the third condition we have to set up the graph on the left in Figure A.6. Employing such relative orientation, indeed, the empty circle is *divergence-less* (according to the fictitious flow that represents the Boundary Operator) independently of its orientation, as could be seen by the 2 diagrams (corresponding to two different via list representation) on the right. Such base will ensure the third condition but rebounds non trivially on the structure of  $\mathbf{B}_2$  as we shall see in the following.

In Figure A.7 (a), no matter which base we use for the space of edges, we can see how such base induces four non 0 components on the vector  $\mathbf{B}_2 \mathbf{h}$ . Those will be



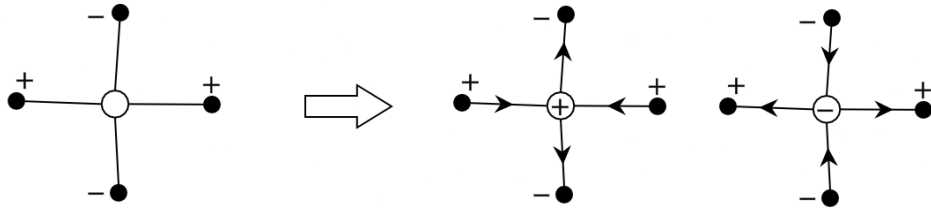


Figure A.6: Examples of how the orientation of a given tetrahedron (white circle) rebounds on the boundary operator, once the triangles are oriented, i.e. the basis  $\{T_j\}$  is fixed (black circle). The resulting divergence in the white circle is always 0.

$e_j^*(\mathbf{B}_2\mathbf{h})$  with  $j = 1, 2, 3, 4$ , whose corresponding nodes in the graph are marked in red. It is important to notice, however, that  $e_j^*(\mathbf{B}_2\mathbf{h}) = \pm 1$  and we have the freedom of choosing either  $+1$  or  $-1$  based on the way we orient  $e_j$ , rule defined by Figure A.4 (b). Let us focus on the sub graph containing only the portion with non 0 components,

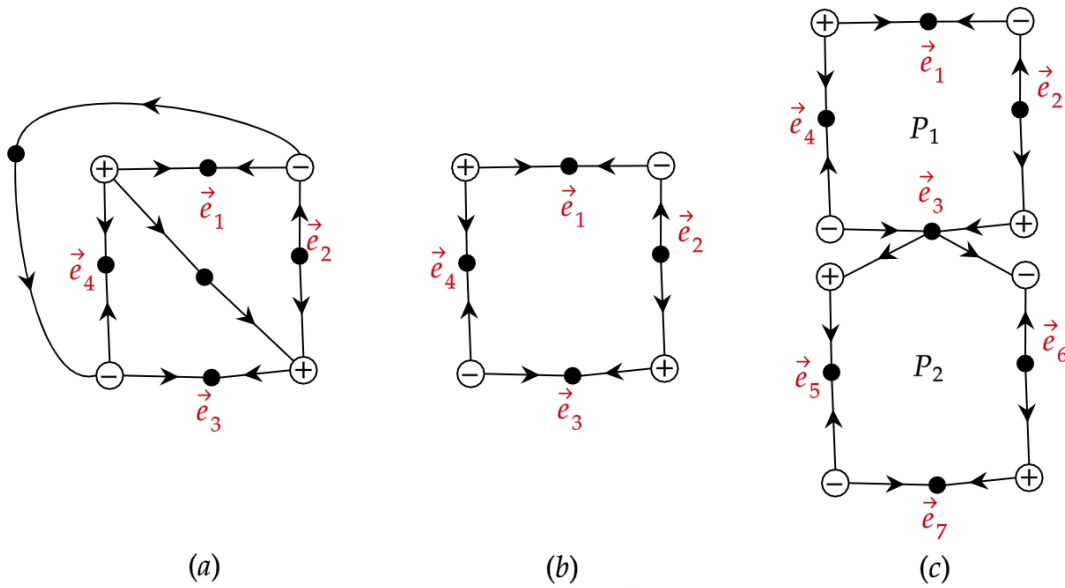


Figure A.7: (a): The boundary operator action on a tetrahedron once the basis of  $S_{1,2}$  is set. In (b) the sub graph containing the edges with non zero divergence is shown and in (c) it is presented how the divergence of the fictitious flux can be set to zero glueing together different graph.

illustrated in Figure A.7 (b) and lets imagine to attach another tetrahedron via edge  $e_3$  as in (c). Thanks to that operation we have solved the problem on  $e_3$  (indeed we are altering the topological proprieties by glueing together tetrahedron) but added non zero  $e_{5,6,7}$ . However the same procedure can be repeated and the resulting lattice-like structure resembles a  $2D$  torus.

By doing this we can compensate every non zero component in every edge in the same way as the one illustrated in A.8.

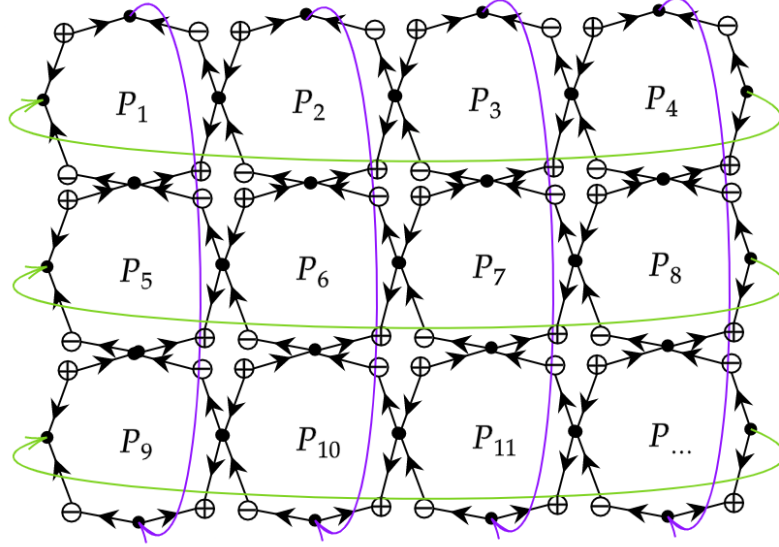


Figure A.8: Representation of the graph after several tetrahedrons are glued together in order to make the fictitious flux divergence free in every black circle.

#### A.4.1 Non-homogeneous vector

Working with topological signal on  $k$ -simplicial complex we should also take into account the fact that a signal such that  $\sigma_k^{*(i)}(\mathbf{h}) = 1$ , like  $(1, 1, 1 \dots 1, 1, 1)$  is, for our purpose, equivalent to  $\mathbf{h}'$  such that  $\sigma_k^{*(i)}(\mathbf{h}') = -1$  for  $j$  like  $(-1, -1, 1 \dots 1, 1, 1)$ . However the analysis of the second signal in a given simplicial complex can be reframed in terms of the analysis of the first signal after a change of basis such that  $\sigma_k^{(j)} \rightarrow -\sigma_k^{(j)}$  for all  $j \in J$ . As an example lets consider the analysis for the case of the tetrahedron. If we take the orientation of every  $T_i = +$  the signal  $(1, 1, 1, 1)$  no longer stays in the kernel of  $\mathbf{B}_{k+1}^\top$  but  $(1, 1, -1, -1)$  does. However it is exactly the same as considering  $T_1 = T_2 = +$  and  $T_3 = T_4 = -$  and with the signal  $(1, 1, 1, 1)$ . By shifting the focus on the latter the consideration are easier and can be rephrased in terms of divergence in every node: the white ones when accounting for  $\mathbf{B}_{k+1}$  and the filled ones for  $\mathbf{B}_k$ .

### A.5 Higher dimension

The above reasoning scheme could be extended and generalized to higher dimension. Because a  $S_k$  simplex is made by  $k + 1$   $S_{k-1}$  simplices, in order for the homogeneous vector to be in the kernel of  $\mathbf{B}_{k+1}^\top$ , we must restrict to even  $k$  to be able to satisfy the required property. Figure A.5 (a) shows, indeed, how every  $S_{k+1}$  simplex has an even number of incoming nodes and therefore, choosing an opportune base (i.e. orientation) in which the signal  $\mathbf{h}$  (homogeneous vector) is written in, it is possible to have a 0 coboundary for all  $S_{k+1}$ .

Let us then consider an even  $k$ . This imply that every  $S_{k+1}$  in our simplicial complex will have  $k + 2$  faces  $S_{k-1}$  simplices (i.e. the graphical representation will have  $k + 2$  nodes). The base will be such that the orientation of each  $S_k$  will have a number of  $+$  oriented faces equal to the number of  $-$  oriented ones.

When passing to the structure of  $\mathbf{B}_k$  (like diagram in Figure A.3 (a)) having fixed

simplices the base of  $S_k$  we can calculate the number of "non zero boundary"  $S_{k-1}$ , namely the non 0 divergence filled node of the associated graph.

Being every  $S_k$  connected with every other and being their orientation either + or - we can reason as follows: the total number of links in such graph is  $\binom{k+2}{2}$ . From that number we have to remove all the zero-divergence  $S_{k-1}$  simplex, namely the simplexes connecting two equally oriented  $S_k$ . This is exactly the number of filled nodes in the associated graph with zero divergence.

The number of such  $S_{k-1}$  filled nodes can be easily calculated as the number of connection between every + oriented  $S_k$  plus the number of connection between every - oriented  $S_k$ . Being equal in number this is just  $\binom{(k+2)/2}{2} \times 2$ . Therefore the number of non zero divergence filled nodes (or non zero boundary  $S_{k-1}$ ) is, for an even k-simplex

$$\tau_k = \binom{k+2}{2} - \binom{(k+2)/2}{2} \times 2 \quad (\text{A.3})$$

Which is equal to

$$\tau_k = \frac{(k+1)(k+2)}{2} - \frac{k+2}{2} \frac{k}{2} = \frac{(k+2)^2}{4} \quad (\text{A.4})$$

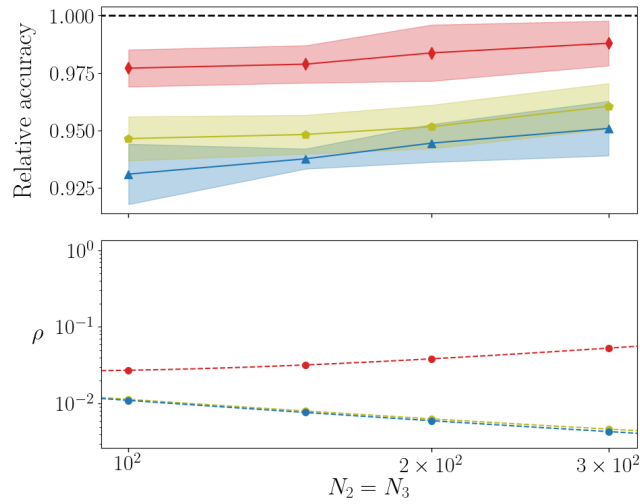
For the case of  $k = 2$  we get  $\tau_2 = 4$ , the right value already exploited in the construction of the torus. However the number with the next  $k$ , namely  $k = 4$  is  $\tau_4 = 9$ . We therefore have to construct an higher dimensional tessellation to compensate that. Moreover the resulting graph, after the removal of the zero boundary connections, is very intricate, making the job pretty tough. We speculate that mapping this problem into a graph colouring scheme could solve the issue and we reserve this aspect further investigation.

## A.6 Graph formulation of the problem

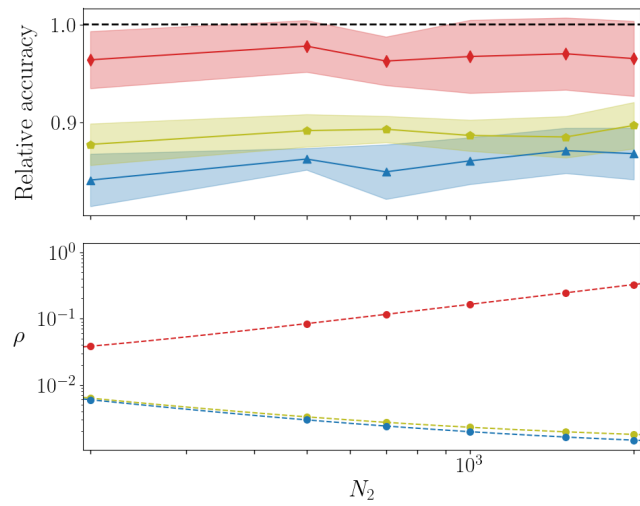
We are now in the position to state in a different way our problem of finding a simplicial or cell complex capable of hosting global synchronization. We conjecture that *a k-dimensional topological signal can globally synchronize if the corresponding graph schematization of k, k + 1 and k - 1, k are eulerian graph.*

## A.7 Results in deep networks: Sparsity and SVD

In this Appendix we will test the setting of a multi-layered architecture by generalising beyond the case study  $\ell = 3$  that we have already explored. More specifically, we have trained according to different modalities a four-layer ( $\ell = 4$ ) deep neural network, by modulating  $N_2 = N_3$  over a finite window. In A.1a the results for the S-SVD and S-QR are shown and confirm that the *S-QR* strategy yields performance that are comparable to those reached with conventional learning approaches, but relying on a much smaller set of trainable parameters. As usual, the size of the incoming and outgoing layers are set by the specificity of the examined datasets. In Fig. A.2 the effect of the imposed sparsity on the classification accuracy is displayed for both conventional and *S-QR* method. Similar conclusions can be reached for MNIST (not shown) and CIFAR-10 A.1b.



(a) MLP - F-MNIST



(b) MLP - CIFAR-10

Figure A.1: Comparative analysis of different datasets with various configurations. The relative accuracy as obtained by training a four layer network with  $N_2 = N_3$  via three different strategies presented in Chapter ??.

Results for the sparsification of a MLP for the F-MNIST dataset.

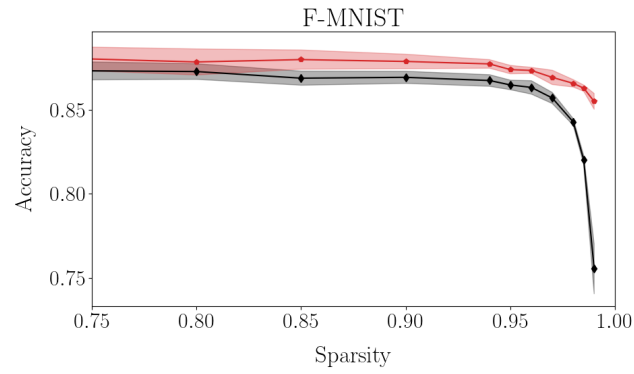


Figure A.2: **MLP: training a sparse network.** The accuracy of the trained network against the degree of imposed sparsity. Black diamonds refer to the usual training in direct space, while red pentagons refer to the  $S$ - $QR$  method. The analysis is carried out for F-MNIST. Here,  $N_2 = N_3 = 500$ .

# Appendix B

## Square Lattice with periodic boundary conditions

For this case of interest, where the simplicial complex is a  $d$ -dimensional square lattice with periodic boundary conditions (p.b.c.), interesting phenomena occur. For a rectangular portion of a  $d$ -dimensional square lattice with linear size  $L_m$  in the direction  $m$ , the eigenvalues and the eigenvectors of the graph Laplacians  $\mathbf{L}_0$  and  $\mathbf{L}_1$  can be easily computed [91]. Indeed the eigenvectors of  $\mathbf{L}_0$  are the Fourier modes of the lattice associated with wave number  $\mathbf{q} = (q_1, q_2, \dots, q_m, \dots, q_d)$  and the eigenvalues of  $\mathbf{L}_0$  can be expressed as

$$\Lambda_0 = 4 \sum_{m=1}^d \sin^2(q_m/2). \quad (\text{B.1})$$

The periodic boundary conditions impose

$$q_m = \frac{2\pi}{L_m} \hat{n}_m \quad \hat{n}_m = 0, 1, 2 \dots L_m - 1. \quad (\text{B.2})$$

The analysis that has been carried out for the case presented in the main text will let us conclude that as soon as an eigenvalue returns a positive dispersion law, i.e.,  $\lambda(b_k^2) > 0$ , the corresponding eigenspace will be constituted by one periodic eigenvector that spans the nodes and one, again periodic, that spans the links. Consequently, as in the general case, the arising instability cannot be confined to the space of nodes or links, here too.

To be concrete, we have numerically analyzed the dynamical system (9.38) defined on a  $4 \times 4$  2-dimensional lattice with p.b.c.. The results are depicted in Fig. B.1: on the left column, panels *a*), *c*) and *e*), refer to the case where there is single unstable mode, while on the right column (panels *b*), *d*) and *f*)), multiple unstable modes are allowed. Let us first observe that the critical mode, i.e., the one associated to the largest value of the dispersion relation, is the same for both parameter configurations; we also remark that to each mode, except for the 0-th one, there are associated several linearly independent eigenvectors, four vectors in the case of a  $4 \times 4$  lattice with p.b.c.. When only one mode is unstable (see panel *c*) in Fig. B.1), we observe that the signal on the nodes exhibits a (horizontal) striped-like pattern and the signals on the links are non-zero only when the link connects nodes with different signals values (Fig. B.1*a*). Such ordered structure is destroyed when multiple modes are unstable (Fig. B.1*b, d*). When there is a single unstable mode, the stationary pattern is a linear combination

of the 4 eigenvectors associated to such mode (Fig. B.1e); remarkably this continues to be true when more than one mode is unstable (Fig. B.1f).

Let us conclude by observing that the former result is a slight generalization of the one we can find in [144], where authors showed that in the case of a unique unstable and non-degenerate mode, the patterns can be described by such eigenvector, despite the fact that they are the reflex of a nonlinear process. Here we have shown that the same result holds true if the unique critical eigenvalue possesses a high-dimensional subspace spanned by several eigenvectors and even in the case of multiple unstable modes.



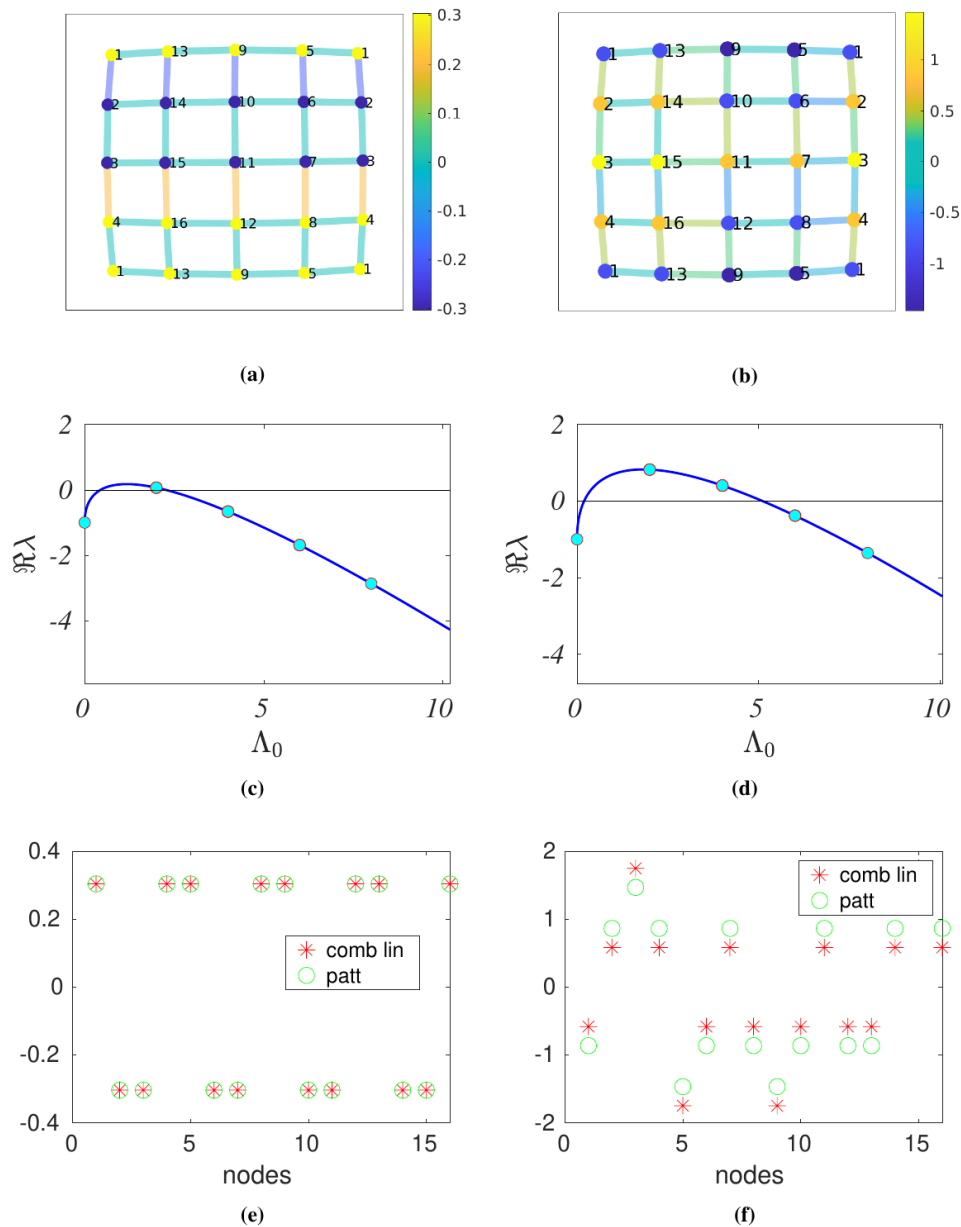


Figure B.1: Model (9.38) on a  $4 \times 4$  2-dimensional lattice with p.b.c. The periodicity of lattice, shown in panels *a*) and *b*), is represented by adding one column and one row, so that the displayed nodes are 25, but effectively they are 16. Panels on the left show the case where only one mode contributes to the instability, while those on the right where multiple modes are unstable, as shown by the dispersion laws in panels *c*) and *d*), respectively. When only one mode is unstable, the nodes' pattern is striped-like, while the signal on the links is non-zero only when the given link connects two nodes with different signals, as shown in panel *a*); on the other hand, when multiple modes are unstable, such regular structure is lost, as we can see in panel *b*). Panels *e*) and *f*) show a comparison of the nodes' pattern with a linear combination of the eigenvector associated to the critical mode(s), showing a good accordance. The model parameters are  $a = \alpha = b = \beta = \gamma = D_0 = D_1 = 1$ ;  $c = 4.7$  for panels *a*), *c*), *e*), while  $c = 7.3$  for panels *b*), *d*), *f*); the initial perturbation is  $\sim 10^{-2}$ .



# Appendix C

## Analysis of the Stuart-Landau model

The aim of this section is to provide the interested reader more details about the global synchronization of identical Stuart-Landau systems defined on top of simplicial and cell complexes introduced in the main text.

Let us thus assume to deal with a  $k$  dimensional topological signal associated to a nonlinear SL oscillator, whose time evolution is given by

$$\frac{dw}{dt} = \sigma w - \beta w |w|^2,$$

where  $\sigma = \sigma_{\Re} + i\sigma_{\Im}$  and  $\beta = \beta_{\Re} + i\beta_{\Im}$  are complex model parameters. One can easily prove that the former system admits a limit cycle solution  $\hat{z}(t) = \sqrt{\sigma_{\Re}/\beta_{\Re}} e^{i\omega t}$ , where  $\omega = \sigma_{\Im} - \beta_{\Im}\sigma_{\Re}/\beta_{\Re}$ , is the signal frequency. Such solution is stable provided  $\sigma_{\Re} > 0$  and  $\beta_{\Re} > 0$ , conditions that we hereby assume.

We now consider  $N_k$  SL topological oscillators and we couple them using the the  $(k-1)$  and  $(k+1)$ -faces of a simplicial complex. Let also assume the coupling to be given by the nonlinear function  $h(w) = \mu w |w|^{m-1}$ , where  $m$  is a positive integer and  $\mu = \mu_{\Re} + i\mu_{\Im}$  is a complex parameter defining the interaction strength. Observe that such coupling has been recently introduced and studied in the framework on synchronization on time varying networks in [189]. In conclusion we are interested in studying the system

$$\frac{dw_i}{dt} = \sigma w_i - \beta w_i |w_i|^2 + \mu \sum_{j=1}^{N_k} L_k(i, j) w_j |w_j|^{m-1} \quad \forall i = 1, \dots, N_k. \quad (\text{C.1})$$

We are now interested in studying the stability of the reference limit cycle solution  $\hat{z}(t)$ . If this condition is realised, then the system shows global synchronization. To achieve this goal we introduce real “small” functions  $\rho_j(t)$  and  $\theta_j(t)$  and rewrite  $w_j(t)$  as follows

$$w_j(t) = \hat{z}(t)(1 + \rho_j(t))e^{i\theta_j(t)}, \quad (\text{C.2})$$

where  $\rho_j(t)$  and  $\theta_j(t)$  are real valued functions. We now insert the previous expression in the coupled Eq. (C.1). By using the expression for  $\hat{z}(t)$  and by expanding the

resulting equation up to the first order in  $\rho_j$  and  $\theta_j$ , we eventually obtain

$$\begin{cases} \frac{d\rho_j}{dt} = -2\sigma_{\Re}\rho_j - \left(\frac{\sigma_{\Re}}{\beta_{\Re}}\right)^{\frac{m-1}{2}} \sum_{\ell=1}^{N_k} L_k(j, \ell) (m\mu_{\Re}\rho_{\ell} - \mu_{\Im}\theta_{\ell}) \\ \frac{d\theta_j}{dt} = -2\beta_{\Im}\frac{\sigma_{\Re}}{\beta_{\Re}}\rho_j - \left(\frac{\sigma_{\Re}}{\beta_{\Re}}\right)^{\frac{m-1}{2}} \sum_{\ell=1}^{N_k} L_k(j, \ell) (m\mu_{\Im}\rho_{\ell} + \mu_{\Re}\theta_{\ell}) . \end{cases} \quad (\text{C.3})$$

We can then decompose  $\rho_j(t)$  and  $\theta_j(t)$  on the orthonormal eigenbasis  $\phi_k^{(\alpha)}$ ,  $\alpha = 1, \dots, N_k$ , of the Laplace matrix  $\mathbf{L}_k$ :

$$\rho_j = \sum_{\alpha} \hat{\rho}_{\alpha} \phi_k^{(\alpha)}(j) \text{ and } \theta_j = \sum_{\alpha} \hat{\theta}_{\alpha} \phi_k^{(\alpha)}(j), \quad (\text{C.4})$$

to eventually obtain

$$\begin{cases} \frac{d\hat{\rho}_{\alpha}}{dt} = -2\sigma_{\Re}\hat{\rho}_{\alpha} - \left(\frac{\sigma_{\Re}}{\beta_{\Re}}\right)^{\frac{m-1}{2}} \Lambda_k^{(\alpha)} (m\mu_{\Re}\hat{\rho}_{\alpha} - \mu_{\Im}\hat{\theta}_{\alpha}) \\ \frac{d\hat{\theta}_{\alpha}}{dt} = -2\beta_{\Im}\frac{\sigma_{\Re}}{\beta_{\Re}}\hat{\rho}_{\alpha} - \left(\frac{\sigma_{\Re}}{\beta_{\Re}}\right)^{\frac{m-1}{2}} \Lambda_k^{(\alpha)} (m\mu_{\Im}\hat{\rho}_{\alpha} + \mu_{\Re}\hat{\theta}_{\alpha}) . \end{cases} \quad (\text{C.5})$$

Let us observe that Eq. (C.5) is autonomous, hence one can compute its eigenvalues and define the largest real part of the latter ones, say  $\lambda$ , named in the literature *dispersion relation*. One can thus conclude that if  $\lambda < 0$  the reference solution is stable and hence the system globally synchronizes. The same reasoning can be done whenever the generic MSF (10.3) is autonomous.

Let us also observe that a similar conclusion can be obtained if  $\mathbf{s}(t)$  is a periodic solution by resorting to Floquet analysis; calling again  $\lambda$  the largest real part of the Floquet eigenvalues we can show that if  $\lambda < 0$  then the reference solution is stable and the system globally synchronizes. In the general case, one has to (numerically) compute the Lyapunov exponent of (10.3) and infer about the stability of the reference solution using the Lyapunov theory. Let us observe that to stress the dependence on the simplex eigenvalues we will also write  $\lambda_{\alpha} = \lambda(\Lambda_k^{(\alpha)})$ .

Back to Eq. (C.5) one can infer the stability of the reference solution and thus of the global simplicial synchronization by studying if the perturbations  $\rho_j$  and  $\theta_j$  fade away, or equivalently if their projections  $\hat{\rho}_{\alpha}$  and  $\hat{\theta}_{\alpha}$  vanish. Sufficient conditions are obtained by assuming an exponential behavior, namely  $\hat{\rho}_{\alpha} \sim \tilde{\rho}_{\alpha} e^{\lambda_{\alpha} t}$  and  $\hat{\theta}_{\alpha} \sim \tilde{\theta}_{\alpha} e^{\lambda_{\alpha} t}$  with time-independent  $\tilde{\rho}_{\alpha}$  and  $\tilde{\theta}_{\alpha}$ . Inserting this ansatz into (C.5) and imposing that  $(\tilde{\rho}_{\alpha}, \tilde{\theta}_{\alpha}) \neq (0, 0)$ , one gets the following equation for  $\lambda_{\alpha}$

$$\lambda_{\alpha}^2 + \lambda_{\alpha} \left( \left(\frac{\sigma_{\Re}}{\beta_{\Re}}\right)^{\frac{m-1}{2}} \mu_{\Re} \Lambda_k^{(\alpha)} (m+1) + 2\sigma_{\Re} \right) + m \left(\frac{\sigma_{\Re}}{\beta_{\Re}}\right)^{m-1} \left(\Lambda_k^{(\alpha)}\right)^2 (\mu_{\Re}^2 + \mu_{\Im}^2) + \quad (\text{C.6})$$

$$+ 2\Lambda_k^{(\alpha)} \left(\frac{\sigma_{\Re}}{\beta_{\Re}}\right)^{\frac{m-1}{2}} \left( \mu_{\Re}\sigma_{\Re} + \mu_{\Im}\beta_{\Im}\frac{\sigma_{\Re}}{\beta_{\Re}} \right) = 0. \quad (\text{C.7})$$

We eventually define the *dispersion relation* (or maximum Floquet exponent)  $\lambda = \max_{\alpha} \Re \lambda_{\alpha}$ . Let us observe that  $\lambda_1 = 0$ , that is  $\lambda$  vanishes if evaluated on  $\Lambda_k^{(1)} = 0$ ; this

is because the reference solution is a limit cycle. By considering then the behavior of  $\lambda_\alpha$  for  $\Lambda_k^{(\alpha)}$  close to zero, one can develop the root  $\lambda_\alpha$  as follows

$$\begin{aligned} \lambda_\alpha &= \frac{1}{2} \left[ - \left( \frac{\sigma_{\mathfrak{R}}}{\beta_{\mathfrak{R}}} \right) \mu_{\mathfrak{R}} \Lambda_k^{(\alpha)} (m+1) \right. \\ &\quad \left. - 2\sigma_{\mathfrak{R}} + 2\sigma_{\mathfrak{R}} \left( 1 + \frac{1}{2} \left( \frac{\sigma_{\mathfrak{R}}}{\beta_{\mathfrak{R}}} \right)^{(m-1)/2} \Lambda_k^{(\alpha)} \left( \frac{\mu_{\mathfrak{R}}}{\sigma_{\mathfrak{R}}} (m-1) - 2 \frac{\beta_{\mathfrak{S}} \mu_{\mathfrak{S}}}{\beta_{\mathfrak{R}} \sigma_{\mathfrak{R}}} \right) \right) + \dots \right] \\ &= -\Lambda_k^{(\alpha)} \left( \frac{\sigma_{\mathfrak{R}}}{\beta_{\mathfrak{R}}} \right)^{(m-1)/2} \left( \mu_{\mathfrak{R}} + \frac{\beta_{\mathfrak{S}} \mu_{\mathfrak{S}}}{\beta_{\mathfrak{R}}} \right) + \dots \end{aligned}$$

Hence there exists an interval of values for  $\Lambda_k^{(\alpha)}$  such that  $\lambda_\alpha > 0$ , namely the global synchronization cannot be achieved, if and only if

$$\mu_{\mathfrak{R}} + \mu_{\mathfrak{S}} \frac{\beta_{\mathfrak{S}}}{\beta_{\mathfrak{R}}} < 0.$$

The above presented theory is applied to topological SL signals defined on top of the 3-simplicial complex and the 3-cell complex previously introduced. The model parameters have been set to some generic values allowing for a negative dispersion relation (see Fig. C.1(b-c) for the simplicial complex and Fig. C.1(e-f) for the cell complex). However once the complex amplitudes are defined on 2-faces, i.e., triangles or squares, the system globally synchronizes as we can appreciate from the inset in panel b) and e), while SL oscillators defined on links have a different behavior if we are dealing with a simplicial complex where they cannot globally synchronize (see panel c)) or a cell complex where global synchronization is achieved (see panel f)). Those different behaviors result from the fact that  $\mathbf{u} \in \ker \mathbf{L}_1$  for the cell complex while  $\mathbf{u} \notin \ker \mathbf{L}_1$  for the simplicial complex, in the latter case synchronization cannot be achieved because of the presence of the topological obstruction.

Let us conclude this section by showing the dispersion relation for the topological SL signals defined on the 2-torus paved with triangles. By assuming the same model parameters as in Fig. C.1 the numerical results presented in Fig. C.2 confirm our theory, global synchronization can be achieved only for topological signals defined on nodes, i.e.,  $k = 0$  simplices (see left panel), or triangles, i.e.,  $k = 2$  simplices (see right panel). Indeed we have  $\mathbf{u} \in \ker \mathbf{L}_0$  and  $\mathbf{u} \in \ker \mathbf{L}_2$ ; on the other hand  $\mathbf{u} \notin \ker \mathbf{L}_1$  and thus topological signals defined on links cannot globally synchronize (see middle panel).

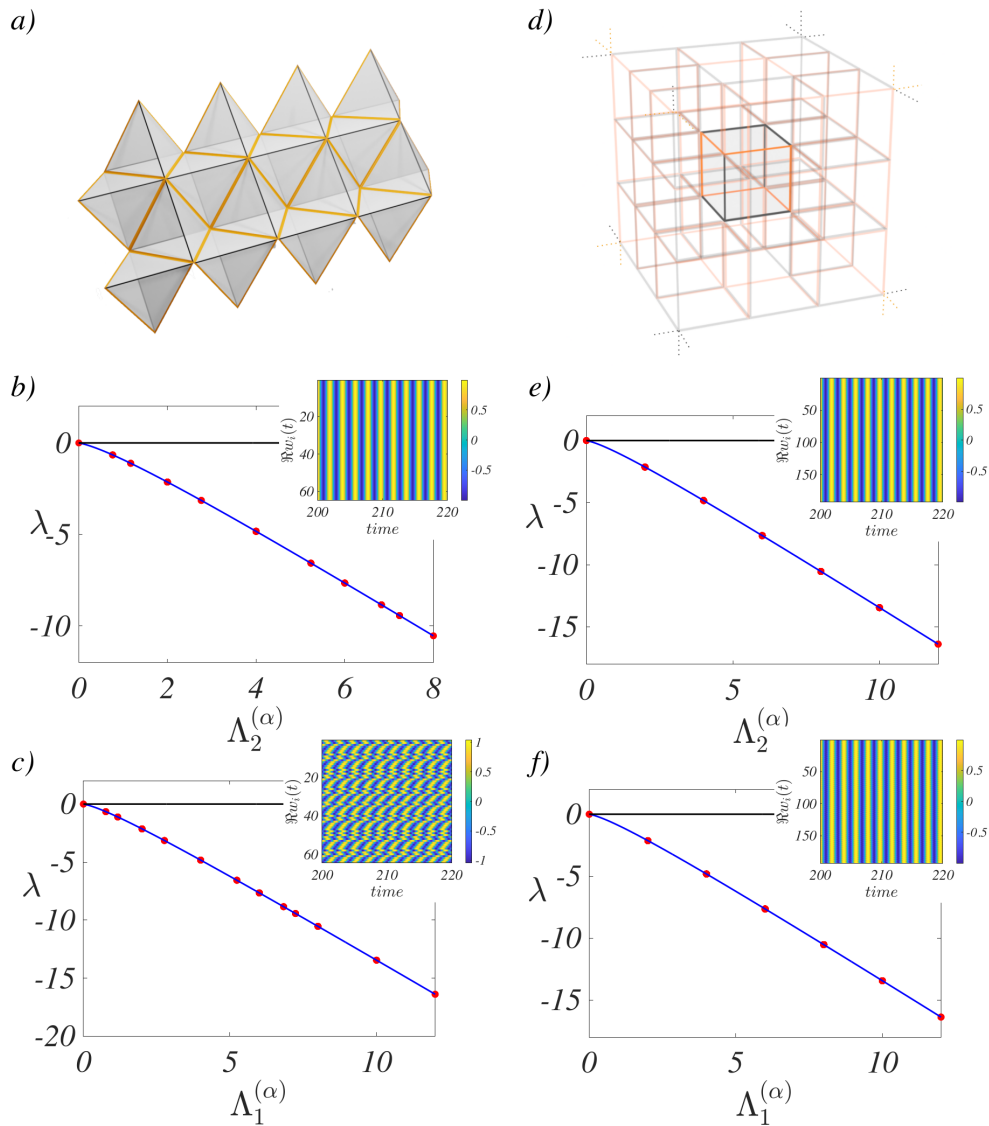


Figure C.1: **Dispersion relation for topological Stuart-Landau model.** The left panels refer to the 3-simplicial complex schematically represented in panel a). In panel b) we report the dispersion relation  $\lambda$  as a function of the eigenvalues  $\Lambda_2^{(\alpha)}$  in the case of topological complex amplitudes defined on 2-faces, i.e., triangles, while as a function of  $\Lambda_1^{(\alpha)}$  in panel c) dealing with signals defined on links. Similar functions are reported in the right panels in the case of 3-cell complex schematically represented in panel d). The model parameters have been fixed to some generic values,  $\sigma = 1.0 + 4.3i$ ,  $\beta = 1.0 + 1.1i$ ,  $\mu = 1.0 - 0.5i$  and  $m = 3$ , and are the same used to obtain the results reported in Fig. 10.8.

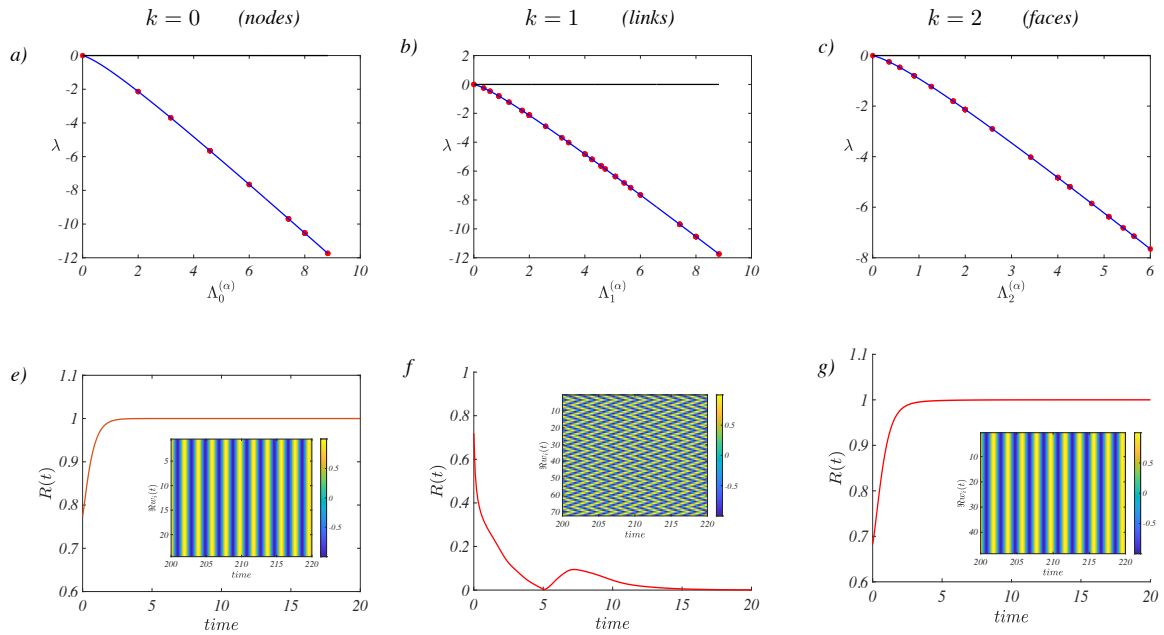


Figure C.2: **Dispersion relation for topological Stuart-Landau model (II).** We consider the 2-torus paved with triangles. The left panel refers to topological signals defined on nodes, i.e., 0-simplices, the middle panel to the case of links, i.e., 1-simplices and the right panel to the case of faces, i.e., 2-simplices panels. Top panels show the dispersion relation and we can observe that in all cases the latter is negative except for the zero value associated to  $\Lambda_k^{(1)} = 0$ ,  $k = 0, 1, 2$ . Bottom panels present the (generalized) order parameters while the inset report the real part of the signals. The model parameters have been fixed to some generic values,  $\sigma = 1.0 + 4.3i$ ,  $\beta = 1.0 + 1.1i$ ,  $\mu = 1.0 - 0.5i$  and  $m = 3$ , and are the same used to obtain the results reported in Fig. 10.8.





# Bibliography

- [1] C. M. Bishop, *Pattern Recognition and Machine Learning*, English, 1st ed. 2006. Corr. 2nd printing 2011 edition. New York: Springer, Apr. 2011, ISBN: 978-0-387-31073-2.
- [2] T. M. Cover and J. A. Thomas, *Elements of information theory* (Wiley series in telecommunications). New York: Wiley, 1991, ISBN: 978-0-471-06259-2.
- [3] T. Hastie, R. Tibshirani, and J. Friedman, *The elements of statistical learning: data mining, inference, and prediction*. Springer Science and Business Media, 2009.
- [4] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.
- [5] A. Graves, A.-r. Mohamed, and G. Hinton, “Speech recognition with deep recurrent neural networks,” in *2013 IEEE international conference on acoustics, speech and signal processing*, IEEE, 2013, pp. 6645–6649.
- [6] N. Sebe, I. Cohen, A. Garg, and T. S. Huang, *Machine learning in computer vision*. Springer Science and Business Media, 2005, vol. 29.
- [7] S. Grigorescu, B. Trasnea, T. Cocias, and G. Macesanu, “A survey of deep learning techniques for autonomous driving,” *Journal of Field Robotics*, vol. 37, no. 3, pp. 362–386, 2020.
- [8] M. Chen, S. Mao, and Y. Liu, “Big data: A survey,” *Mobile networks and applications*, vol. 19, no. 2, pp. 171–209, 2014.
- [9] F. Rosenblatt, “Principles of neurodynamics. perceptrons and the theory of brain mechanisms,” Cornell Aeronautical Lab Inc Buffalo NY, Tech. Rep., 1961.
- [10] E. Meyers and L. Wolf, “Using biologically inspired features for face processing,” *International Journal of Computer Vision*, vol. 76, no. 1, pp. 93–104, 2008.
- [11] L. Caponetti, C. A. Buscicchio, and G. Castellano, “Biologically inspired emotion recognition from speech,” *EURASIP journal on Advances in Signal Processing*, vol. 2011, no. 1, p. 24, 2011.
- [12] F. Rosenblatt, “The perceptron: A probabilistic model for information storage and organization in the brain.,” *Psychological review*, vol. 65, no. 6, p. 386, 1958.
- [13] Y. Bengio, P. Lamblin, D. Popovici, and H. Larochelle, “Greedy layer-wise training of deep networks,” in *Advances in neural information processing systems*, 2007, pp. 153–160.

- [14] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016, <http://www.deeplearningbook.org>.
- [15] G. E. Hinton, S. Osindero, and Y.-W. Teh, “A fast learning algorithm for deep belief nets,” *Neural computation*, vol. 18, no. 7, pp. 1527–1554, 2006.
- [16] D. E. Rumelhart, G. E. Hinton, R. J. Williams, *et al.*, “Learning representations by back-propagating errors,” *Cognitive modeling*, vol. 5, no. 3, p. 1, 1988.
- [17] N. Xie, G. Ras, M. van Gerven, and D. Doran, “Explainable deep learning: A field guide for the uninitiated,” *arXiv preprint arXiv:2004.14545*, 2020.
- [18] A. Vaswani, N. Shazeer, N. Parmar, *et al.*, “Attention is all you need,” *Advances in neural information processing systems*, pp. 5998–6008, 2017.
- [19] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, “Bert: Pre-training of deep bidirectional transformers for language understanding,” *arXiv preprint arXiv:1810.04805*, 2018.
- [20] T. B. Brown, B. Mann, N. Ryder, *et al.*, “Language models are few-shot learners,” *arXiv preprint arXiv:2005.14165*, 2020.
- [21] D. Erhan, A. Courville, and Y. Bengio, “Understanding representations learned in deep architectures,” *Department dInformatique et Recherche Operationnelle, University of Montreal, QC, Canada, Tech. Rep*, vol. 1355, p. 1, 2010.
- [22] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning*. MIT press, 2016.
- [23] L. Giambagli, L. Buffoni, T. Carletti, W. Nocentini, and D. Fanelli, “Machine learning in spectral domain,” *Nature Communications*, vol. 12, no. 1, p. 1330, 2021.
- [24] L. Chicchi, L. Giambagli, L. Buffoni, T. Carletti, M. Ciavarella, and D. Fanelli, “Training of sparse and dense deep neural networks: Fewer parameters, same performance,” *Physical Review E*, vol. 104, no. 5, p. 054312, 2021.
- [25] D. J. Surmeier and R. Foehring, “A mechanism for homeostatic plasticity,” *Nature neuroscience*, vol. 7, no. 7, pp. 691–692, 2004.
- [26] S. d’Ascoli, L. Sagun, G. Biroli, and J. Bruna, “Finding the needle in the haystack with convolutions: On the benefits of architectural bias,” *Advances in Neural Information Processing Systems*, vol. 32, 2019.
- [27] Martin Abadi, Ashish Agarwal, Paul Barham, *et al.*, *TensorFlow: Large-scale machine learning on heterogeneous systems*, Software available from tensorflow.org, 2015. [Online]. Available: <https://www.tensorflow.org/>.
- [28] D. Kingma and J. Ba, “Adam: A method for stochastic optimization,” *arXiv preprint arXiv:1412.6980*, 2014.
- [29] `\NoCaseChange{https://github.com/Bufioni/spectral\_learning}`.
- [30] Y. LeCun, “The mnist database of handwritten digits,” <http://yann.lecun.com/exdb/mnist/>, 1998.
- [31] X. Glorot and Y. Bengio, “Understanding the difficulty of training deep feed-forward neural networks,” *PMLR*, pp. 249–256, Mar. 2010. [Online]. Available: <https://proceedings.mlr.press/v9/glorot10a.html>.

- [32] A. Saxe, J. L. McClelland, and S. Ganguli, *Exact solutions to the nonlinear dynamics of learning in deep linear neural networks*, Dec. 2013. [Online]. Available: [https://openreview.net/forum?id=\\_wzZwKpTDF\\_9C](https://openreview.net/forum?id=_wzZwKpTDF_9C).
- [33] H. Xiao, K. Rasul, and R. Vollgraf, “Fashion-mnist: A novel image dataset for benchmarking machine learning algorithms,” *arXiv preprint arXiv:1708.07747*, 2017.
- [34] A. Krizhevsky and G. Hinton, “Learning multiple layers of features from tiny images,” University of Toronto, Technical Report, 2009.
- [35] M. Denil, B. Shakibi, L. Dinh, M. Ranzato, and N. de Freitas, “Predicting parameters in deep learning,” *neural information processing systems*, vol. 26, pp. 2148–2156, Dec. 2013. DOI: <https://doi.org/10.14288/1.0165555>. [Online]. Available: <https://open.library.ubc.ca/soa/cIRcle/collections/ubctheses/24/items/1.0165555>.
- [36] H. Yang, M. Tang, W. Wen, *et al.*, “Learning low-rank deep neural networks via singular vector orthogonality regularization and singular value sparsification,” in *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2020, pp. 2899–2908. DOI: [10.1109/CVPRW50498.2020.00347](https://doi.org/10.1109/CVPRW50498.2020.00347).
- [37] M. Gabri e, A. Manoel, C. Luneau, *et al.*, “Entropy and mutual information in models of deep neural networks,” *Journal of Statistical Mechanics: Theory and Experiment*, vol. 2019, no. 12, p. 124 014, Dec. 2019. DOI: [10.1088/1742-5468/ab3430](https://doi.org/10.1088/1742-5468/ab3430). [Online]. Available: <https://doi.org/10.1088%5C%2F1742-5468%5C%2Fab3430>.
- [38] J. Frankle and M. Carbin, “The lottery ticket hypothesis: Finding sparse, trainable neural networks,” *arXiv preprint arXiv:1803.03635*, 2018.
- [39] J. O. Neill, “An overview of neural network compression,” *arXiv preprint arXiv:2006.03669*, 2020.
- [40] Y. Cheng, D. Wang, P. Zhou, and T. Zhang, “A survey of model compression and acceleration for deep neural networks,” *arXiv preprint arXiv:1710.09282*, 2017.
- [41] Y. LeCun, B. Boser, J. S. Denker, *et al.*, “Backpropagation applied to handwritten zip code recognition,” *Neural computation*, vol. 1, no. 4, pp. 541–551, 1989.
- [42] S. Bai, J. Z. Kolter, and V. Koltun, “Deep equilibrium models,” *Conference Paper NeurIPS*, 2019.
- [43] D. Zhang, H. Wang, M. Figueiredo, and L. Balzano, “Learning to share: Simultaneous parameter tying and sparsification in deep learning,” in *Conference Paper ICLR*, 2018.
- [44] J. Chang and J. Sha, “Prune deep neural networks with the modified  $L_{-}\{1/2\}$  penalty,” *IEEE Access*, vol. 7, pp. 2273–2280, 2018.
- [45] P. Molchanov, S. Tyree, T. Karras, T. Aila, and J. Kautz, “Pruning convolutional neural networks for resource efficient inference,” *arXiv preprint arXiv:1611.06440*, 2016.

- [46] P. de Jorge, A. Sanyal, H. S. Behl, P. H. Torr, G. Rogez, and P. K. Dokania, “Progressive skeletonization: Trimming more fat from a network at initialization,” *Conference Paper ICLR*, 2021.
- [47] G. Hinton, O. Vinyals, and J. Dean, “Distilling the knowledge in a neural network,” *arXiv preprint arXiv:1503.02531*, 2015.
- [48] A. Polino, R. Pascanu, and D. Alistarh, “Model compression via distillation and quantization,” *arXiv preprint arXiv:1802.05668*, 2018.
- [49] S. I. Mirzadeh, M. Farajtabar, A. Li, N. Levine, A. Matsukawa, and H. Ghasemzadeh, “Improved knowledge distillation via teacher assistant,” in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, 2020, pp. 5191–5198.
- [50] M. Masana, J. van de Weijer, L. Herranz, A. D. Bagdanov, and J. M. Alvarez, “Domain-adaptive deep network compression,” in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 4289–4297.
- [51] A. Novikov, D. Podoprikin, A. Osokin, and D. Vetrov, “Tensorizing neural networks,” *arXiv preprint arXiv:1509.06569*, 2015.
- [52] I. V. Oseledets, “Tensor-train decomposition,” *SIAM Journal on Scientific Computing*, vol. 33, no. 5, pp. 2295–2317, 2011.
- [53] X. Yu, T. Liu, X. Wang, and D. Tao, “On compressing deep models by low rank and sparse decomposition,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 7370–7379.
- [54] P. Stock, A. Joulin, R. Gribonval, B. Graham, and H. Jégou, “And the bit goes down: Revisiting the quantization of neural networks,” *arXiv preprint arXiv:1907.05686*, 2019.
- [55] R. Banner, Y. Nahshan, E. Hoffer, and D. Soudry, “Post-training 4-bit quantization of convolution networks for rapid-deployment,” *Conference Paper NeurIPS*, 2018.
- [56] T. He, Y. Fan, Y. Qian, T. Tan, and K. Yu, “Reshaping deep neural network for fast decoding by node-pruning,” in *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2014, pp. 245–249. DOI: 10.1109/ICASSP.2014.6853595.
- [57] X. Wang, F. Yu, L. Dunlap, *et al.*, “Deep mixture of experts via shallow embedding,” in *Uncertainty in Artificial Intelligence*, PMLR, 2020, pp. 552–562.
- [58] X. Wang, F. Yu, Z.-Y. Dou, T. Darrell, and J. E. Gonzalez, “Skipnet: Learning dynamic routing in convolutional networks,” in *Proceedings of the European Conference on Computer Vision (ECCV)*, Sep. 2018.
- [59] E. Bengio, P.-L. Bacon, J. Pineau, and D. Precup, “Conditional computation in neural networks for faster models,” *Conference Paper ICLR*, 2016.
- [60] R. Yu, A. Li, C.-F. Chen, *et al.*, “Nisp: Pruning networks using neuron importance score propagation,” *Thecvf.com*, pp. 9194–9203, 2018. [Online]. Available: [https://openaccess.thecvf.com/content\\_cvpr\\_2018/html/Yu\\_NISP\\_Pruning\\_Networks\\_CVPR\\_2018\\_paper.html](https://openaccess.thecvf.com/content_cvpr_2018/html/Yu_NISP_Pruning_Networks_CVPR_2018_paper.html).

- [61] P. Molchanov, A. Mallya, S. Tyree, I. Frosio, and J. Kautz, “Importance estimation for neural network pruning,” *Proceedings of 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019.
- [62] J. D. Yann LeCun and S. Solla, “Optimal brain damage,” *Advances in Neural Information Processing Systems*, 1989.
- [63] Z. Liu, M. Sun, T. Zhou, G. Huang, and T. Darrell, “Rethinking the value of network pruning,” in *International Conference on Learning Representations*, 2019. [Online]. Available: <https://openreview.net/forum?id=rJlnB3C5Ym>.
- [64] L. Giambagli, L. Buffoni, T. Carletti, W. Nocentini, and D. Fanelli, “Machine learning in spectral domain,” *Nature communications*, vol. 12, no. 1, pp. 1–9, 2021.
- [65] A. Krizhevsky, G. Hinton, *et al.*, “Learning multiple layers of features from tiny images,” Citeseer, 2009.
- [66] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, “Mobilenetv2: Inverted residuals and linear bottlenecks,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2018.
- [67] N. Komodakis and S. Zagoruyko, “Paying more attention to attention: Improving the performance of convolutional neural networks via attention transfer,” *Hal.science*, Jun. 2017. DOI: <https://hal-enpc.archives-ouvertes.fr/hal-01832769>. [Online]. Available: <https://hal.science/hal-01832769/>.
- [68] W. Park, D. Kim, Y. Lu, and M. Cho, “Relational knowledge distillation,” *Thecvf.com*, pp. 3967–3976, 2019. [Online]. Available: [https://openaccess.thecvf.com/content\\_CVPR\\_2019/html/Park\\_Relational\\_Knowledge\\_Distillation\\_CVPR\\_2019\\_paper.html](https://openaccess.thecvf.com/content_CVPR_2019/html/Park_Relational_Knowledge_Distillation_CVPR_2019_paper.html).
- [69] L. Beyer, X. Zhai, A. Royer, L. Markeeva, R. Anil, and A. Kolesnikov, *Knowledge distillation: A good teacher is patient and consistent*. [Online]. Available: [https://openaccess.thecvf.com/content/CVPR2022/papers/Beyer\\_Knowledge\\_Distillation\\_A\\_Good\\_Teacher\\_Is\\_Patient\\_and\\_Consistent\\_CVPR\\_2022\\_paper.pdf](https://openaccess.thecvf.com/content/CVPR2022/papers/Beyer_Knowledge_Distillation_A_Good_Teacher_Is_Patient_and_Consistent_CVPR_2022_paper.pdf).
- [70] S. Goldt, M. S. Advani, A. M. Saxe, F. Krzakala, and L. Zdeborová, “Dynamics of stochastic gradient descent for two-layer neural networks in the teacher–student setup\*,” *Journal of Statistical Mechanics: Theory and Experiment*, vol. 2020, no. 12, p. 124 010, Dec. 2020. DOI: <https://doi.org/10.1088/1742-5468/abc61e>. [Online]. Available: [https://proceedings.neurips.cc/paper\\_files/paper/2019/file/cab070d53bd0d200746fb852a922064a-Paper.pdf](https://proceedings.neurips.cc/paper_files/paper/2019/file/cab070d53bd0d200746fb852a922064a-Paper.pdf).
- [71] H. S. Seung, H. Sompolinsky, and N. Tishby, “Statistical mechanics of learning from examples,” vol. 45, no. 8, pp. 6056–6091, Apr. 1992. DOI: <https://doi.org/10.1103/physreva.45.6056>. [Online]. Available: <https://journals.aps.org/prabstract/10.1103/PhysRevA.45.6056>.

- [72] F. Krzakala, F. Ricci-Tersenghi, L. Zdeborová, R. Zecchina, E. W. Tramel, and L. F. Cugliandolo, *Statistical Physics, Optimization, Inference, and Message-Passing Algorithms*. Dec. 2015. DOI: <https://doi.org/10.1093/acprof:oso/9780198743736.001.0001>. [Online]. Available: <https://academic.oup.com/book/26783>.
- [73] G. Hinton, O. Vinyals, and J. Dean, *Distilling the knowledge in a neural network*, 2015. [Online]. Available: <https://research.google/pubs/pub44873/>.
- [74] D. Saad and S. A. Solla, “On-line learning in soft committee machines,” *Physical Review E*, vol. 52, no. 4, pp. 4225–4243, Oct. 1995. DOI: <https://doi.org/10.1103/physreve.52.4225>. [Online]. Available: <https://journals.aps.org/pre/abstract/10.1103/PhysRevE.52.4225>.
- [75] B. Aubin, A. Maillard, J. Barbier, F. Krzakala, N. Macris, and L. Zdeborová, “The committee machine: Computational to statistical gaps in learning a two-layers neural network,” *Journal of Statistical Mechanics: Theory and Experiment*, vol. 2019, no. 12, p. 124023, Dec. 2019. DOI: <https://doi.org/10.1088/1742-5468/ab43d2>. [Online]. Available: <https://iopscience.iop.org/article/10.1088/1742-5468/ab43d2/meta>.
- [76] J. Frankle and M. Carbin, “The lottery ticket hypothesis: Finding sparse, trainable neural networks,” *ICLR Conference*, Dec. 2018. [Online]. Available: <https://openreview.net/forum?id=rJl-b3RcF7>.
- [77] S.-l. library, *California housing dataset*, [https://scikit-learn.org/stable/datasets/toy\\_dataset.html](https://scikit-learn.org/stable/datasets/toy_dataset.html), As per the current release.
- [78] D. J. Gauthier, E. Bollt, A. Griffith, and W. A. Barbosa, “Next generation reservoir computing,” *Nature communications*, vol. 12, no. 1, p. 5564, 2021.
- [79] G. Tanaka, T. Yamane, J. B. Héroux, *et al.*, “Recent advances in physical reservoir computing: A review,” *Neural Networks*, vol. 115, pp. 100–123, 2019.
- [80] W. Maass, T. Natschläger, and H. Markram, “Real-time computing without stable states: A new framework for neural computation based on perturbations,” *Neural computation*, vol. 14, no. 11, pp. 2531–2560, 2002.
- [81] M. McCloskey and N. J. Cohen, “Catastrophic interference in connectionist networks: The sequential learning problem,” in *Psychology of learning and motivation*, vol. 24, Elsevier, 1989, pp. 109–165.
- [82] S. Lewandowsky and S.-C. Li, “Catastrophic interference in neural networks: Causes, solutions, and data,” in *Interference and inhibition in cognition*, Elsevier, 1995, pp. 329–361.
- [83] R. Kemker, M. McClure, A. Abitino, T. Hayes, and C. Kanan, “Measuring catastrophic forgetting in neural networks,” in *Proceedings of the AAAI conference on artificial intelligence*, vol. 32, 2018.
- [84] J. Kirkpatrick, R. Pascanu, N. Rabinowitz, *et al.*, “Overcoming catastrophic forgetting in neural networks,” *Proceedings of the national academy of sciences*, vol. 114, no. 13, pp. 3521–3526, 2017.

- [85] I. J. Goodfellow, M. Mirza, D. Xiao, A. Courville, and Y. Bengio, “An empirical investigation of catastrophic forgetting in gradient-based neural networks,” *arXiv preprint arXiv:1312.6211*, 2013.
- [86] X. Li, Y. Zhou, T. Wu, R. Socher, and C. Xiong, “Learn to grow: A continual structure learning framework for overcoming catastrophic forgetting,” in *International Conference on Machine Learning*, PMLR, 2019, pp. 3925–3934.
- [87] F. Chollet *et al.*, “Keras: The python deep learning library,” *Astrophysics source code library*, ascl–1806, 2018.
- [88] A. Sherstinsky, “Fundamentals of recurrent neural network (rnn) and long short-term memory (lstm) network,” *Physica D: Nonlinear Phenomena*, vol. 404, p. 132 306, 2020.
- [89] Y. Goldberg, *Neural network methods for natural language processing*. Springer Nature, 2022.
- [90] L. R. Medsker and L. Jain, “Recurrent neural networks,” *Design and Applications*, vol. 5, no. 64-67, p. 2, 2001.
- [91] G. Bianconi, *Higher-Order Networks: An introduction to simplicial complexes*. Cambridge University Press, 2021.
- [92] M. Nakahara, *Geometry, topology and physics*. CRC Press, 2003.
- [93] L.-H. Lim, “Hodge laplacians on graphs,” *SIAM Review*, vol. 62, no. 3, pp. 685–715, Jan. 2020. DOI: 10.1137/18m1223101. [Online]. Available: <https://doi.org/10.1137/18m1223101>.
- [94] M.-L. Linne, J. Aćimović, A. Saudargiene, and T. Manninen, “Neuron–glia interactions and brain circuits,” in *Computational Modelling of the Brain*, Springer, 2022, pp. 87–103.
- [95] J. Faskowitz, R. F. Betzel, and O. Sporns, “Edges in brain networks: Contributions to models of structure and function,” *Network Neuroscience*, vol. 6, no. 1, pp. 1–28, 2022.
- [96] A. Santoro, F. Battiston, G. Petri, and E. Amico, “Unveiling the higher-order organization of multivariate time series,” *arXiv preprint arXiv:2203.10702*, 2022.
- [97] E. Katifori, G. J. Szöllösi, and M. O. Magnasco, “Damage and fluctuations induce loops in optimal transport networks,” *Physical review letters*, vol. 104, no. 4, p. 048 704, 2010.
- [98] J. W. Rocks, A. J. Liu, and E. Katifori, “Hidden topological structure of flow network functionality,” *Physical Review Letters*, vol. 126, no. 2, p. 028 102, 2021.
- [99] D. Witthaut, F. Hellmann, J. Kurths, S. Kettemann, H. Meyer-Ortmanns, and M. Timme, “Collective nonlinear dynamics and self-organization in decentralized power grids,” *Reviews of Modern Physics*, vol. 94, no. 1, p. 015 005, 2022.
- [100] S. Barbarossa and S. Sardellitti, “Topological signal processing over simplicial complexes,” *IEEE Transactions on Signal Processing*, vol. 68, pp. 2992–3007, 2020.
- [101] S. Sardellitti and S. Barbarossa, “Topological signal representation and processing over cell complexes,” *arXiv preprint arXiv:2201.08993*, 2022.

- [102] M. T. Schaub and S. Segarra, “Flow smoothing and denoising: Graph signal processing in the edge-space,” in *2018 IEEE Global Conference on Signal and Information Processing (GlobalSIP)*, IEEE, 2018, pp. 735–739.
- [103] M. T. Schaub, Y. Zhu, J.-B. Seby, T. M. Roddenberry, and S. Segarra, “Signal processing on higher-order networks: Livin’ on the edge... and beyond,” *Signal Processing*, vol. 187, p. 108 149, 2021.
- [104] A. P. Millán, J. J. Torres, and G. Bianconi, “Explosive higher-order Kuramoto dynamics on simplicial complexes,” *Physical Review Letters*, vol. 124, no. 21, p. 218 301, 2020.
- [105] A. P. Millán, J. G. Restrepo, J. J. Torres, and G. Bianconi, “Geometry, topology and simplicial synchronization,” in *Higher-Order Systems*, Springer, 2022, pp. 269–299.
- [106] J. J. Torres and G. Bianconi, “Simplicial complexes: Higher-order spectral dimension and dynamics,” *Journal of Physics: Complexity*, vol. 1, no. 1, p. 015 002, 2020.
- [107] R. Ghorbanchian, J. G. Restrepo, J. J. Torres, and G. Bianconi, “Higher-order simplicial synchronization of coupled topological signals,” *Communications Physics*, vol. 4, no. 1, pp. 1–13, 2021.
- [108] L. Calmon, J. G. Restrepo, J. J. Torres, and G. Bianconi, “Topological synchronization: Explosive transition and rhythmic phase,” *preprint arXiv:2107.05107*, 2021.
- [109] A. Arnaudon, R. L. Peach, G. Petri, and P. Expert, “Connecting hodge and Sakaguchi-Kuramoto: A mathematical framework for coupled oscillators on simplicial complexes,” *arXiv preprint arXiv:2111.11073*, 2021.
- [110] L. DeVille, “Consensus on simplicial complexes: Results on stability and synchronization,” *Chaos: An Interdisciplinary Journal of Nonlinear Science*, vol. 31, no. 2, p. 023 137, 2021.
- [111] M. Reitz and G. Bianconi, “The higher-order spectrum of simplicial complexes: A renormalization group approach,” *Journal of Physics A: Mathematical and Theoretical*, vol. 53, no. 29, p. 295 001, 2020.
- [112] C. Ziegler, P. S. Skardal, H. Dutta, and D. Taylor, “Balanced Hodge Laplacians optimize consensus dynamics over simplicial complexes,” *Chaos: An Interdisciplinary Journal of Nonlinear Science*, vol. 32, no. 2, p. 023 128, 2022.
- [113] M. T. Schaub, A. R. Benson, P. Horn, G. Lippner, and A. Jadbabaie, “Random walks on simplicial complexes and the normalized Hodge 1-Laplacian,” *SIAM Review*, vol. 62, no. 2, pp. 353–391, 2020.
- [114] C. Bodnar, F. Frasca, Y. Wang, *et al.*, “Weisfeiler and lehman go topological: Message passing simplicial networks,” in *International Conference on Machine Learning*, PMLR, 2021, pp. 1026–1037.
- [115] S. Ebli, M. Defferrard, and G. Spreemann, “Simplicial neural networks,” *arXiv preprint arXiv:2010.03633*, 2020.



- [116] T. M. Roddenberry and S. Segarra, “Hodgenet: Graph neural networks for edge data,” in *2019 53rd Asilomar Conference on Signals, Systems, and Computers*, IEEE, 2019, pp. 220–224.
- [117] M. Hajij, K. Istvan, and G. Zamzmi, “Cell complex neural networks,” *arXiv preprint arXiv:2010.00743*, 2020.
- [118] P. S. Skardal and A. Arenas, “Abrupt desynchronization and extensive multistability in globally coupled oscillator simplexes,” *Physical Review Letters*, vol. 122, no. 24, p. 248 301, 2019.
- [119] P. S. Skardal and A. Arenas, “Higher order interactions in complex networks of phase oscillators promote abrupt synchronization switching,” *Communications Physics*, vol. 3, no. 1, pp. 1–6, 2020.
- [120] L. V. Gambuzza, F. Di Patti, L. Gallo, *et al.*, “Stability of synchronization in simplicial complexes,” *Nature Communications*, vol. 12, no. 1, pp. 1–13, 2021.
- [121] K. Kovalenko, X. Dai, K. Alfaro-Bittner, A. Raigorodskii, M. Perc, and S. Boccaletti, “Contrarians synchronize beyond the limit of pairwise interactions,” *Physical Review Letters*, vol. 127, no. 25, p. 258 301, 2021.
- [122] U. Alvarez-Rodriguez, F. Battiston, G. F. de Arruda, Y. Moreno, M. Perc, and V. Latora, “Evolutionary dynamics of higher-order interactions in social networks,” *Nature Human Behaviour*, vol. 5, no. 5, pp. 586–595, 2021.
- [123] Y. Lee, J. Lee, S. M. Oh, D. Lee, and B. Kahng, “Homological percolation transitions in growing simplicial complexes,” *Chaos: An Interdisciplinary Journal of Nonlinear Science*, vol. 31, no. 4, p. 041 102, 2021.
- [124] T. Carletti, F. Battiston, G. Cencetti, and D. Fanelli, “Random walks on hypergraphs,” *Physical Review E*, vol. 101, no. 2, p. 022 308, 2020.
- [125] M. Lucas, G. Cencetti, and F. Battiston, “Multiorder laplacian for synchronization in higher-order networks,” *Physical Review Research*, vol. 2, no. 3, p. 033 410, 2020.
- [126] Y. Tang, D. Shi, and L. Lü, “Optimizing higher-order network topology for synchronization of coupled phase oscillators,” *Communications Physics*, vol. 5, no. 1, pp. 1–12, 2022.
- [127] Y. Zhang, V. Latora, and A. E. Motter, “Unified treatment of synchronization patterns in generalized networks with higher-order, multilayer, and temporal interactions,” *Communications Physics*, vol. 4, no. 1, pp. 1–9, 2021.
- [128] M. Chutani, B. Tadić, and N. Gupte, “Hysteresis and synchronization processes of kuramoto oscillators on high-dimensional simplicial complexes with competing simplex-encoded couplings,” *Physical Review E*, vol. 104, no. 3, p. 034 206, 2021.
- [129] D. Horak and J. Jost, “Spectra of combinatorial Laplace operators on simplicial complexes,” *Advances in Mathematics*, vol. 244, pp. 303–336, 2013.
- [130] L.-H. Lim, “Hodge Laplacians on graphs,” *Siam Review*, vol. 62, no. 3, pp. 685–715, 2020.

- [131] G. Bianconi, “The topological Dirac equation of networks and simplicial complexes,” *Journal of Physics: Complexity*, vol. 2, no. 3, p. 035 022, 2021.
- [132] S. Lloyd, S. Garnerone, and P. Zanardi, “Quantum algorithms for topological and geometric analysis of data,” *Nature communications*, vol. 7, no. 1, pp. 1–7, 2016.
- [133] B. Ameneyro, V. Maroulas, and G. Siopsis, “Quantum persistent homology,” *arXiv preprint arXiv:2202.12965*, 2022.
- [134] O. Knill, “The Dirac operator of a graph,” *arXiv preprint arXiv:1306.2166*, 2013.
- [135] D. Mulder and G. Bianconi, “Network geometry and complexity,” *Journal of Statistical Physics*, vol. 173, no. 3, pp. 783–805, 2018.
- [136] E. Steinitz, “Beiträge zur analysis situs,” *Sitz-Ber. Berlin Math. Ges*, vol. 7, pp. 29–49, 1908.
- [137] R. Klette, “Cell complexes through time,” in *Vision Geometry IX. Int. Soc. for Opt. and Photon.*, vol. 4117, 2000, pp. 134–145.
- [138] A. Hatcher, *Algebraic topology*. Cambridge University Press, 2005.
- [139] L. Grady and J. Polimeni, *Discrete calculus: Applied analysis on graphs for computational science*. Sprin. Sci. and Busin. Media, 2010.
- [140] F. Baccini, F. Geraci, and G. Bianconi, “Weighted simplicial complexes and their representation power of higher-order network data and topology,” *Physical Review E*, vol. 106, no. 3, p. 034 319, 2022.
- [141] I. Prigogine and G. Nicolis, “Symmetry breaking instabilities in dissipative systems,” *J. Chem. Phys.*, vol. 46, p. 3542, 1967.
- [142] A. Pikovsky, J. Kurths, and M. Rosenblum, *Synchronization: a universal concept in nonlinear sciences*. Cambridge university press, 2001, vol. 12.
- [143] A. M. Turing, “The chemical basis of morphogenesis,” *Phil. Trans. R. Soc. Lond. B*, vol. 237, p. 37, 1952.
- [144] H. Nakao and A. S. Mikhailov, “Turing patterns in network-organized activator-inhibitor systems,” *Nature Physics*, vol. 6, p. 544, 2010.
- [145] R. Pastor-Satorras and A. Vespignani, “Patterns of complexity,” *Nature Physics*, vol. 6, p. 480, 2010.
- [146] Y. Kuramoto, “Self-entrainment of a population of coupled non-linear oscillators,” in *International Symposium on Mathematical Problems in Theoretical Physics*, H. Araki, Ed., Berlin, Heidelberg: Springer Berlin Heidelberg, 1975, pp. 420–422.
- [147] S. H. Strogatz, “From Kuramoto to crawford: Exploring the onset of synchronization in populations of coupled oscillators,” *Physica D: Nonlinear Phenomena*, vol. 143, no. 1-4, pp. 1–20, 2000.
- [148] A. Arenas, A. Diaz-Guilera, J. Kurths, Y. Moreno, and C. Zhou, “Synchronization in complex networks,” *Phys. Rep.*, vol. 469, no. 3, pp. 93–153, 2008. DOI: 10.1016/j.physrep.2008.09.002.

- [149] S. Boccaletti, A. N. Pisarchik, C. I. Del Genio, and A. Amann, *Synchronization: from coupled systems to complex networks*. Cambridge University Press, 2018.
- [150] T. Carletti and D. Fanelli, “Theory of synchronisation and pattern formation on time varying networks,” *Chaos, Solitons and Fractals*, vol. 159, p. 112180, 2022. DOI: <https://doi.org/10.1016/j.chaos.2022.112180>.
- [151] A.-L. Barabási, *Network science*. Cambridge university press, 2016.
- [152] M. E. Newman, *Networks: An Introduction*. Oxford: Oxford University Press, 2010.
- [153] V. Latora, V. Nicosia, and G. Russo, *Complex Networks: Principles, Methods and Applications*. Cambridge University Press, 2017.
- [154] S. Boccaletti, V. Latora, Y. Moreno, M. Chavez, and D.-U. Hwang, “Complex networks: Structure and dynamics,” *Physics Reports*, vol. 424, no. 4-5, pp. 175–308, 2006.
- [155] J. D. Murray, *Mathematical biology II: Spatial models and biomedical applications*. Springer-Verlag, 2001.
- [156] A. Gierer and H. Meinhardt, “A theory of biological pattern formation,” *Kybernetik*, vol. 12, p. 30, 1972.
- [157] H. G. Othmer and L. E. Scriven, “Instability and dynamic pattern in cellular networks,” *J. Theor. Biol.*, vol. 32, p. 507, 1971.
- [158] M. Asllani, J. D. Challenger, F. S. Pavone, L. Sacconi, and D. Fanelli, “The theory of pattern formation on directed networks,” *Nature Communication*, vol. 5, no. 4517, 2014.
- [159] M. Asllani, D. M. Busiello, T. Carletti, D. Fanelli, and G. Planchon, “Turing patterns in multiplex networks,” *Phys. Rev. E*, vol. 90, p. 042814, 4 2014.
- [160] J. Petit, B. Lauwens, D. Fanelli, and T. Carletti, “Theory of Turing patterns on time varying networks,” *Phys. Rev. Letters*, vol. 119, p. 148301, 2017.
- [161] R. Muolo, M. Asllani, D. Fanelli, P. K. Maini, and T. Carletti, “Patterns of non-normality in networked systems,” *Journal of Theoretical Biology*, vol. 480, p. 81, 2019.
- [162] D. Fanelli, C. Cianci, and F. Di Patti, “Turing instabilities in reaction-diffusion systems with cross diffusion,” *Eur. Phys. J. B*, vol. 86, p. 142, 2013.
- [163] D. Busiello, G. Planchon, M. Asllani, T. Carletti, and D. Fanelli, “Pattern formation for reactive species undergoing anisotropic diffusion,” *Eur. Phys. J. B*, vol. 88, p. 222, 2015.
- [164] T. Carletti, D. Fanelli, and S. Nicoletti, “Dynamical systems on hypergraphs,” *Journal of Physics: Complexity*, vol. 1, no. 3, p. 035006, 2020.
- [165] R. Muolo, L. Gallo, V. Latora, M. Frasca, and T. Carletti, “Turing patterns in systems with high-order interaction,” *arXiv preprint arXiv:2207.03985*, 2022.

- [166] L. Giambagli, L. Calmon, R. Muolo, T. Carletti, and G. Bianconi, “Diffusion-driven instability of topological signals coupled by the dirac operator,” *Phys. Rev. E*, vol. 106, p. 064314, 6 Dec. 2022. DOI: 10.1103/PhysRevE.106.064314. [Online]. Available: <https://link.aps.org/doi/10.1103/PhysRevE.106.064314>.
- [167] T. Carletti and R. Muolo, “Finite propagation enhances Turing patterns in reaction–diffusion networked systems,” *Journal of Physics: Complexity*, vol. 2, no. 4, p. 045004, 2021.
- [168] A. Pikovsky, M. Rosenblum, and J. Kurths, *Synchronization*. Cambridge University Press, Cambridge, UK, 2001.
- [169] A. Arenas, A. Diaz-Guilera, J. Kurths, Y. Moreno, and C. Zhou, “Synchronization in complex networks,” *Physics Reports*, vol. 469, no. 3, p. 93, 2008.
- [170] L. M. Pecora and T. L. Carroll, “Master stability functions for synchronized coupled systems,” *Phys. Rev. Lett.*, vol. 80, no. 10, p. 2109, 1998.
- [171] L. M. Pecora, T. L. Carroll, G. A. Johnson, D. J. Mar, and J. F. Heagy, “Fundamentals of synchronization in chaotic systems, concepts, and applications,” *Chaos: An Interdisciplinary Journal of Nonlinear Science*, vol. 7, no. 4, p. 520, 1997.
- [172] Y. Kuramoto, *Chemical oscillations, waves, and turbulence*. Springer-Verlag, New York, 1984.
- [173] F. A. Rodrigues, T. K. D. Peron, P. Ji, and J. Kurths, “The Kuramoto model in complex networks,” *Physics Reports*, vol. 610, pp. 1–98, 2016.
- [174] C. Giusti, R. Ghrist, and D. S. Bassett, “Two’s company, three (or more) is a simplex,” *Journal of Computational Neuroscience*, vol. 41, no. 1, pp. 1–14, 2016.
- [175] M. W. Reimann, M. Nolte, M. Scolamiero, *et al.*, “Cliques of neurons bound into cavities provide a missing link between structure and function,” *Frontiers in Computational Neuroscience*, p. 48, 2017.
- [176] A. Patania, G. Petri, and F. Vaccarino, “The shape of collaborations,” *EPJ Data Sci.*, vol. 6, no. 1, p. 18, 2017.
- [177] E. Estrada and G. J. Ross, “Centralities in simplicial complexes. applications to protein interaction networks,” *J. Theor. Biol.*, vol. 438, p. 46, 2018.
- [178] F. Battiston, E. Amico, A. Barrat, *et al.*, “The physics of higher-order interactions in complex systems,” *Nature Physics*, vol. 17, no. 10, pp. 1093–1098, 2021.
- [179] A. Krawiecki, “Chaotic synchronization on complex hypergraphs,” *Chaos, Solitons and Fractals*, vol. 65, p. 44, 2014.
- [180] R. Mulas, C. Kuehn, and J. Jost, “Coupled dynamics on hypergraphs: Master stability of steady states and synchronization,” *Phys. Rev. E*, vol. 101, p. 062313, 2020.
- [181] T. Carletti, D. Fanelli, and S. Nicoletti, “Dynamical systems on hypergraphs,” *Journal of Physics Complexity*, vol. 1, p. 035006, 2020.

- [182] A. P. Millán, J. J. Torres, and G. Bianconi, “Explosive higher-order Kuramoto dynamics on simplicial complexes,” *Phys. Rev. Lett.*, vol. 124, p. 218 301, 21 2020.
- [183] R. Ghorbanchian, J. G. Restrepo, J. J. Torres, and G. Bianconi, “Higher-order simplicial synchronization of coupled topological signals,” *Communications Physics*, vol. 120, p. 1, 4 2021.
- [184] A. van Harten, “On the validity of the ginzburg-landau equation,” *J. Nonlinear Sci.*, vol. 1, p. 397, 1991.
- [185] I. Aranson and L. Kramer, “The world of the complex ginzburg-landau equation,” *Reviews of Modern Physics*, vol. 74, p. 99, 2002.
- [186] V. Garca-Morales and K. Krischer, “The complex ginzburg-landau equation: An introduction,” *Contem. Phys.*, vol. 53, p. 79, 2012.
- [187] A. Kitaev, “Fault-tolerant quantum computation by anyons,” *Annals of Physics*, vol. 303, no. 1, pp. 2–30, Jan. 2003. DOI: [https://doi.org/10.1016/s0003-4916\(02\)00018-0](https://doi.org/10.1016/s0003-4916(02)00018-0). [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0003491602000180>.
- [188] L. Grady and J. Polimeni, *Discrete Calculus: Applied Analysis on Graphs for Computational Science*. Springer London, 2010, ISBN: 9781849962902. [Online]. Available: <https://books.google.it/books?id=E3-OSVSPbU0C>.
- [189] T. Carletti and D. Fanelli, “Theory of synchronisation and pattern formation on time varying networks,” *Chaos, Solitons and Fractals*, vol. 159, p. 112 180, 2022.