Politecnico di Torino

Memory capacity of neuromorphic self-oganizing nanowire networks

Tesi di Laurea Magistrale in Nanotechnologies for ICTs

Author: **Alberto Sivera**

Student ID: s290313
Advisor: Prof. Carlo Ricciardi
Co-advisors: Gianluca Milano
Academic Year: 2022-23

Abstract

Neuromorphic computing emerged as a promising candidate for future computing technology oriented to overcome the limitations imposed by the end of Moore's law and by the traditional von Neumann architecture, which is becoming less and less suitable for many artificial intelligence (AI) applications. Given their ease of production and their biological neural network-like topology, neuromorphic self-organized networks of nanowires with memristive junctions are currently being explored as a very promising neuromorphic system that can be use to develop AI-specific hardware. The recent demonstration of the computing capabilities of these nanowire networks, including recognition of spatio-temporal patterns and time-series prediction, has grown the interest in finding a way to assess the information processing capability of the networks and its dependencies on system physical parameters. A possible way to do this is to use the Memory Capacity (MC) benchmark test, already used in the framework of reservoir computing to evaluate the computing capability of the reservoirs. In this work, by means of simulations based on a random graph model of the nanowire networks, it is first discussed a way to properly measure MC, and then the dependence of MC on several system parameters such as network density, network topology, input signal amplitude, and number of reading electrodes. Further, the application of a voltage bias to the nanowire networks, and the particular positioning of input-output electrodes are explored as possible solutions to increase the MC of the nanowire networks. The aim of the simulations and measurements carried out in this work is to provide a deeper insight of the relations that exists between system parameters and system performances, in order to support a more complex experimental activity towards the realization of neuromorphic hardware specifically designed for AI applications based on neuromorphic self-organized nanowire networks.

Keywords: neuromorphic, nanowire networks, memory capacity, in materia reservoir computing.

Contents

Abstract **i**

Contents **iii**

1 Introduction **1**

1.1 Neuromorphic self-organizing nanowire networks . . . 3

1.1.1 Device fabrication . . . 5

1.1.2 Physics of memristive junctions . . . 7

1.1.3 Modeling of memristive junctions . . . 8

1.1.4 Random graph modeling of nanowire networks . . . 10

1.1.5 Modified voltage node analysis . . . 11

1.1.6 Electrical stimulation of nanowire networks . . . 13

1.2 Reservoir computing . . . 15

1.2.1 PRC with Ag-PVP NWNs . . . 18

1.2.2 Readout module . . . 18

1.2.3 Memory capacity . . . 19

2 Results **21**

2.1 Memory capacity computation algorithm . . . 23

2.1.1 Training phase . . . 25

2.1.2 Test phase . . . 28

2.1.3 Forgetting curve and signal prediction . . . 29

2.2 Memory capacity measurement conditions . . . 31

2.2.1 Network initialization and number of considered delays . . . 32

2.2.2 Signal length and training signal . . . 33

2.2.3 Dependence of optimal signal length on signal amplitude and on number of nanowires . . . 35

2.2.4 Dependence of MC on test signal . . . 40

2.3 Effects of system parameters on Memory capacity 41
2.3.1 Signal amplitude 41
2.3.2 Network density 44
2.3.3 Percentage of reading nodes 46
2.3.4 Signal bias 49
2.3.5 I/O Electrodes configuration 52

3 Discussion **57**

Bibliography **59**

List of Figures **67**

Acknowledgements **69**

1 | Introduction

The approaching of Big-data era and the idea of internet of things (IoT) have exponentially increased the development of AI systems based software-implemented artificial neural networks (ANNs) aimed to carry out data-centric tasks. However, conventional electronics onto which these AI systems are implemented is now showing its limits, both on the device and on the architecture levels. From the device point of view, the further increase in performance of MOSFET transistors will be limited by the incoming end of the Moore's law, and the leakage currents are becoming a serious issue for what concerns the power consumption. On the architecture level, the constant data shuttling between the information processing and memory units in the von Neumann architecture significantly limits the speed and energy efficiency (the 'von Neumann bottleneck'), making this architecture inefficient in handling AI tasks. Therefore, in order to meet the increasingly critical requirements in terms of processing speed and power efficiency imposed by practical applications of AI, new computing technologies are needed.
In this framework, neuromorphic computing technologies are emerging as very promising candidates to develop hardware components that are specifically designed for AI applications [1, 2]. Specifically, neuromorphic computers are non-von Neumann computers whose structure and function are inspired by biological neural networks and that are composed of "neurons" and "synapses". For comparison, while von Neumann computers are composed of separate processing and memory units, in neuromorphic computers, on the other hand, both processing and memory are governed by the neurons and the synapses, implementing in this way the in-memory computing paradigm. Programs in neuromorphic computers are defined by the structure of the neural network and its parameters, rather than by explicit instructions as in a von Neumann computer [3].

For what concerns real implementations of neuromorphic computers, several solutions have been proposed and neuromorphic chips based on different technologies have been developed and are now objects of active research. Comprehensive reviews of neuromorphic technologies and their application can be found in [4, 5]. Many of these technologies possess the capability to be integrated at high-density yet still being energy-efficient. This

would allow not only for the improvement of currently implemented AI systems in terms of speed and energy consumption but also for application of AI technology in power-constrained environments, such as edge computing and IoT [4].
As depicted in figure 1.1, neuromorphic chips can be roughly classified into two categories: the CMOS-based systems, and the memristive systems. The neuromorphic feature that is implemented in CMOS-based systems [6–10] is the co-location of memory and processing units. This minimal neuromorphic feature alone has dramatically improved power efficiency in training various artificial neural network (ANN) models [5]. On the other hand, memristive systems tries to emulate also the synapses and the topology of the biological neural networks. The former are implemented by both memristive crossbar arrays [11] and by self-assembled networks, while the latter only by self-assembled networks.

This work focuses on the self-assembled networks, in particular, on neuromorphic self-organizing networks of PVP-coated silver nanowires with memristive junctions (Ag-PVP NWNs). Among the low-dimensional neuromorphic nanomaterials that have been explored [12–16], self-organizing nanowire networks are particularly interesting as their self-assembly naturally embeds neural network-like circuitry into their structure, with neural-like dynamics emerging from recurrent feedback loops and memristive cross-point junctions [17]. Given their characteristics, self-organizing nanowire networks are particularly suitable to be used as reservoirs in the framework of physical reservoir computing (PRC) [18–23], and recently Milano et al. [20] experimentally demonstrated that this kind of systems can perform in materia computing of prediction of chaotic time series and classification of handwritten digit images, reaching an accuracy of 90.4% on the MNIST test set.
In this thesis work, memory properties of neuromorphic self-organizing Ag-PVP NWNs used as reservoir are studied by means of simulations. The memory properties of the system are evaluated through the memory capacity (MC) benchmark test introduced by Jaeger in [24] for evaluating the short-term memory (STM) of software-implemented echo state networks (ESN) and used in other studies dealing with biological neural networks [25] and self-organizing nanowire networks [21, 22]. The aim of the thesis is to explore the memory performances of the nanowire networks and how MC of the networks depends on the system's physical parameters such as network density, network topology, input signal amplitude, and number of reading electrodes. Further, the application of a voltage bias to the nanowire networks, and the particular positioning of input-output electrodes are explored as possible solutions to increase the MC of the nanowire networks.

The thesis is structured as follow. The first chapter gives an introduction to the neuromorphic self-organizing Ag-PVP NWNs, explaining their physics and how they are modeled so that they can be simulated, and than it presents the concept of PRC and how nanowire networks can be used as reservoir. The second chapter presents the results of the simulations that have been carried out in this thesis work in order to asses how MC of the networks has to be measured, and how it depends on system physical parameters. Finally, a discussion of the results is reported.

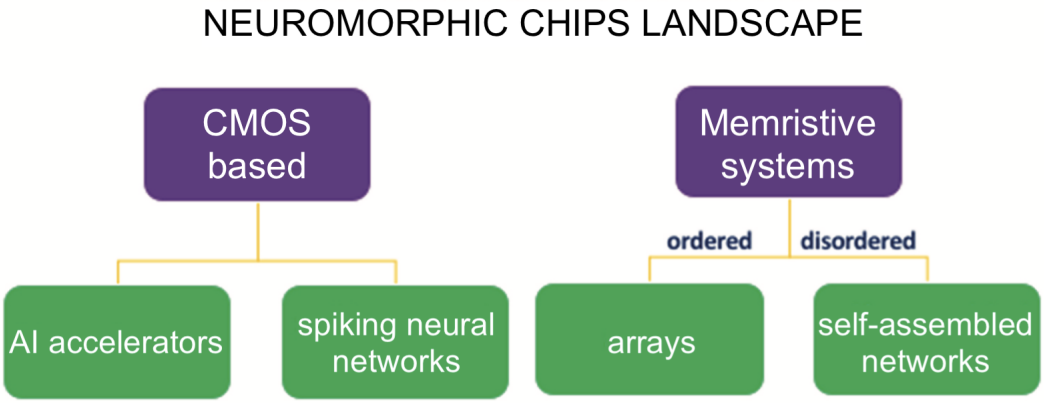


Figure 1.1: Schematic outline of the neuromorphic systems landscape, broadly subdivided by CMOS-based neuromorphic chips and beyond-CMOS memristive devices, the latter further subdivided into ordered systems (fabricated by top-down methods) and disordered systems (synthesised by bottom-up methods). Adapted from [17].

1.1. Neuromorphic self-organizing nanowire networks

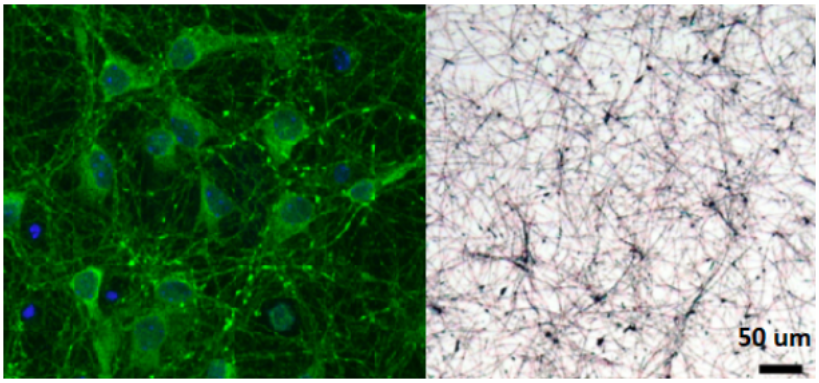


Figure 1.2: **Left:** Human brain, fluorescence imaging. **Right:** Self-organizing memristive nanowire network, SEM image. Adapted from [26].

Neuromorphic self-organizing nanowire networks are neuromorphic devices which are particularly interesting because collective electrical responses emerge from the complex network structure, drawing strong similarities to the structure–function relation in biological neural networks formed by bottom-up self-assembly [27]. This enables nanowire networks to perform learning tasks intrinsically, without requiring the software implementation of an algorithmic artificial neural network learning model. Varying the spatio-temporal input signals (i.e. delivered via different contact electrodes and with time-varying amplitudes) can form new electrical transmission routes, in a manner analogous to synaptogenetic learning [28, 29].
In particular, this work deals with Ag-PVP NWNs, in which the intersection points between nanowires present a memristive behaviour due to Ag+ ion migration across the metal-insultaor-metal (MIM) junction which lead to the formation and dissolution of an Ag conductive filament between the cores of intersecting nanowires, modulating in this way the junction conductivity (figure 1.3) [17]. Ag-PVP NWNs embed neural network-like circuitry into their structure, with neural-like dynamics emerging from recurrent feedback loops and memristive cross-point junctions. The latter have a synapse-like nonlinear electrical response, owing to the ability to replicate diverse brain-like functionalities such as short-term and long-term potentiation (STP/LTP) and spike timing-depen- dent plasticity (STDP), which are key to learning abilities [30–33].

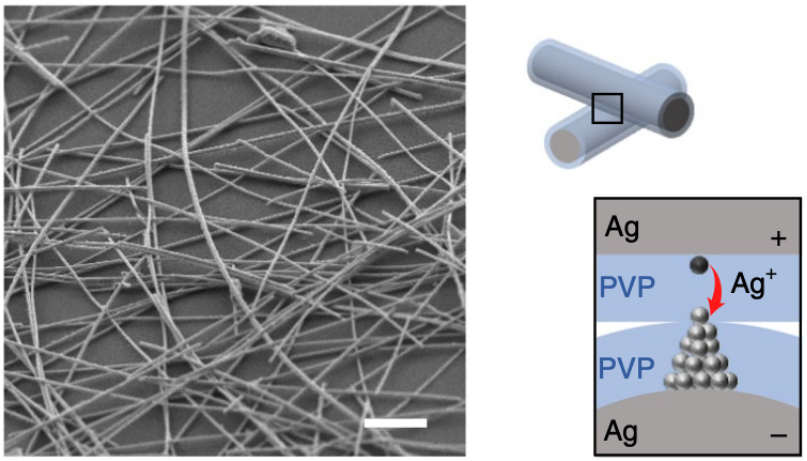


Figure 1.3: **Left:** Scanning electron microscopy (SEM) image of a highly interconnected memristive NW network reservoir (scale bar, 2 μm). **Right:** Schematic representation of the resistive switching mechanism occurring at the NW junctions, where the conductivity can be modulated by the formation/rupture of a metallic Ag conductive path across the NW shell layer, under the action of the applied electric field. Adapted from [20].

As studied by Diaz-Alvarez et al. [34], these kind of structures exhibit complex dynamics

as collective memory response, resilience and adaptation behavior as in biological neuronal systems. Another important property of nanowire networks has been highlighted by Gomes da Rocha et al. [35]: during potentiation, the network is capable of self select the most energy efficient path connection among electrodes, as presented in figure 1.4. Moreover, this behavior leads to conductance plateaus which are stable over a certain range in current compliance. Stated otherwise, inputs can be mapped in the network internal state by means of conductive paths. These results have generated great attraction concerning neuromorphic computing using Ag-PVP NWNs systems, especially as reservoir in PRC framework. This work deals with the assessment of memory properties of Ag-PVP NWNs that emerges from the random recurrent connections between memory elements (the memristors that form at the intersection between Ag-PVP nanowires).
In the following, insights on the device fabrication, modelling and electrical simulation are provided.

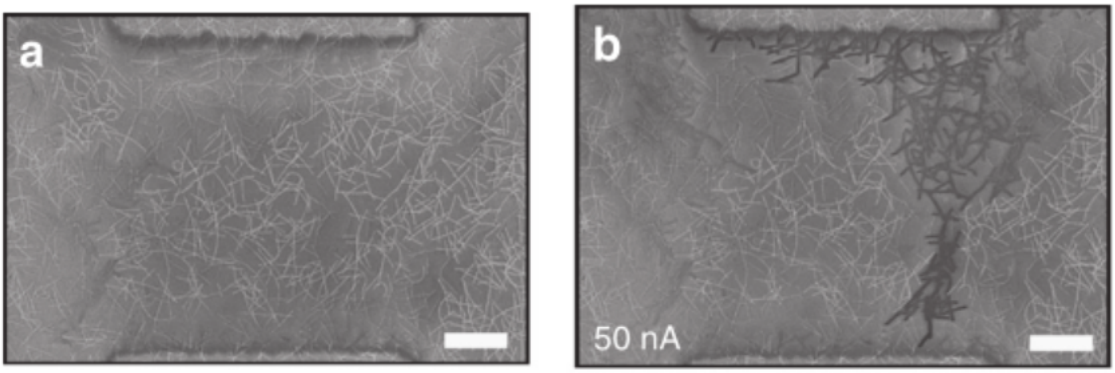


Figure 1.4: Winner takes all phenomenon in a $10\mu m$x$10\mu m$ Ag nanowire network. Reprinted from. **Left:** Unperturbed nanowire network, SEM image (scale bar: 2 μm). **Right:** Stimulated network with current compliance IC = 50 nA, SEM image (scale bar: 2 μm). Adapted from [35].

1.1.1. Device fabrication

The simulations carried out in this work refer to a device made up of Silver nanowires covered by a thin layer ($1 \div 2$ nm) of Polyvinylpyrrolidone (PVP), an insulating polymer. PVP is the residue of the nanowire synthesis, since it acts as a surfactant to obtain high aspect ratio structures. The presence of an insulating polymer, however, is fundamental to obtain a resistive switching behavior providing a metal-insulator-metal structure. The fabrication [26] of Ag-PVP NWNs is performed by drop-casting Silver nanowires in alcohol (IPA) suspension on a SiO2 substrate. The areal mass density (AMD) of the deposited nanowires can be controlled by tuning the mass ratio between suspended nanowire and

IPA. In particular, lower AMD can be obtained with higher levels of IPA, as shown in figure 1.5. Nanowires are purchased in alcohol suspension: if a lower AMD is desired, dilution in IPA is increased. Subsequently, gold pads (electrodes to access the network) are realized along the periphery of the network through sputtering with the use of a shadow mask. A zoomed view of pads, which are however disposed in a different way from that simulated in this work, is highlighted in figure 1.6. It is important to remark that there is no need for cleanroom facilities or lithographic processes, making the process simple and cheap.

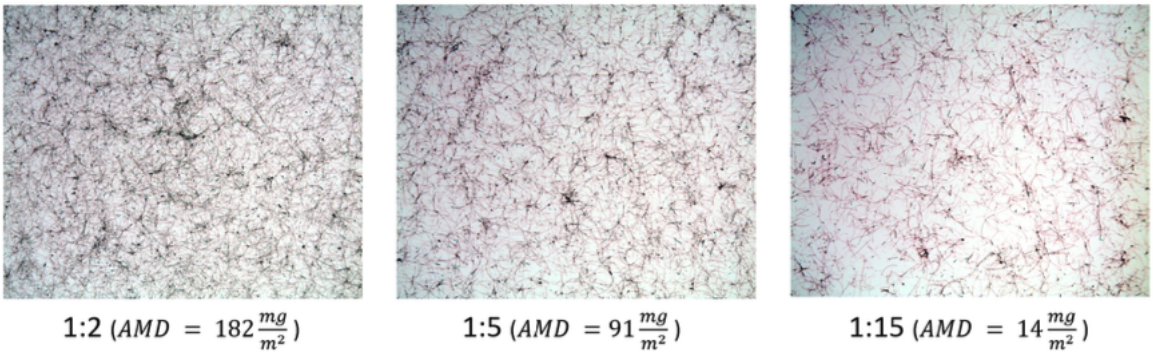


Figure 1.5: Nanowire density controlled by the ratio Ag NW: IPA.

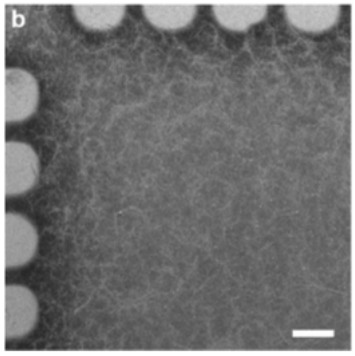


Figure 1.6: Gold electrodes deposited through sputtering. SEM image, scale bar 250 μm. Adapted from [26].

For what concerns the simulations performed in this thesis, the process of device fabrication, in particular the dispersion of of nanowires on the substrate, is here simulated by randomly generating the nanowire positions and orientations from uniform distributions, and by tuning the density of nanowires by choosing their number with the simulation parameter `NWs`. The process of electrodes patterning is not simulated in a realistic manner. The electrode area is chosen arbitrarily and all the nanowires intersecting the electrode area are considered to be electrically connected with a short circuit with the electrode,

which is then linked to the voltage generator or to ground trough a 82 Ω resistor, as described in section 1.1.6.

1.1.2. Physics of memristive junctions

The physiscs of memristive MIM junctions that forms at Ag-PVP nanowires intersection points is based on the resistive switching phenomena, which is a physical phenomena where a dielectric can changes its resistance state under the action of a strong electric field or current.
Resistive switching memory has been observed in a range of functional materials, including phase-change memory, ferroelectric and ferromagnetic materials [4]. The resistance change can be attributed to nanoscale geometric confinement of the material, in some cases modifying its band structure and associated tunnel barrier, but in general, the large surface-area-to-volume ratio dramatically enhances bias-catalysed redox reactions that couple electronic and ionic transport [36, 37]. Coupling of ionic and electronic transport mechanisms can also cause rapid switching between high and low resistance states (HRS/LRS) in electrolytic materials, which are electrically insulating and ionically conducting. The LRS is attributed to the formation of a conductive filament across a biased MIM junction (figure 1.7). First recognized in atomic switch devices [31, 38–40], this mechanism mimics biological synaptic dynamics due to neurotransmitter molecules and is thought to be responsible for the fast resistive switching that occurs at the nanoscale MIM cross-point junctions. Studies have also found resistive switching in Ag-PVP nanowire systems [34, 41]. This may be attributed to the amorphous phase of PVP, a polymer electrolyte. In Ag-PVP systems, conductive filament formation and dissolution are attributed to electrochemical metallisation (ECM).
When a voltage bias is applied, metal cations migrate from the anode nanowire, across the MIM junction, to the cathode nanowire, where electrochemical reduction results in the formation of an atomic metal nano-filament (figure 1.7 left panel). The filament can grow both in length and, when a complete bridge is formed between the two silver cores, also in width (figure 1.7 right panel). The growth of the filament increases the junction conductance, mimicking the potentiation of biological synapses.After the filament formation, if the voltage bias is no longer applied, the filament spontaneously dissolves (relaxes) due to surface diffusion forces [42], mimicking the depression of biological synapses and implementing in this way the fading memory concept that is important for PRC. Similar effects are obtained if a polarization opposite to that of formation is applied [41]. Further, it has been shown in [41] that the change in junction conductance and relaxation time can be modulated by changing the voltage amplitude, with higher voltages resulting

in larger changes of conductance and longer relaxation times. In addition, paired pulse facilitation phenomena similar to those occurring in biological neural networks are observed in Ag-PVP-Ag junctions [41]. All these characteristics of Ag-PVP-Ag memristive junctions contribute to making Ag-PVP NWNs very interesting systems in neuromorphic computing field.

Finally, the way in which the conductance of each junction changes during filament growth and relaxation is nonlinear, and this make Ag-PVP NWNs suitable to be used as reservoir in PRC framework. In fact, the coupling of nonlinear junction dynamics to the complex network topology via Kirchhoff's Laws results in rich recurrent dynamics that can be exploited for reservoir computing applications [4, 18].

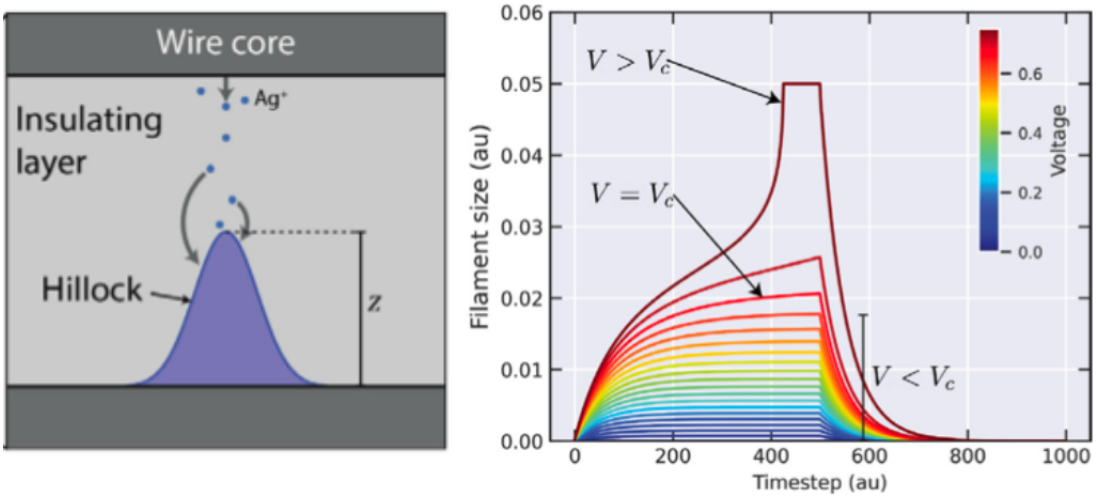


Figure 1.7: **Left:** Illustration of a possible growth mechanism that drives the dynamics within the junction between two silver nanowires. As an electrical potential difference occurs between two contacting wires, silver ions migrate into the junction, forming a filament or hillock. **Right:** Conductive filament thickness as a function of the applied electrical potential difference. Adapted from [21].

1.1.3. Modeling of memristive junctions

The emergent network dynamics, driven out of equilibrium by input voltage signals, depends on the evolution of each single memristive edge in the network. In this work edges dynamics are modelled by a potentiation-depression rate balance equation proposed by Enrique Miranda et al. in [43]:

$$\frac{\mathrm{d}g}{\mathrm{d}t} = k_p(1-g) + k_d g \tag{1.1}$$

with g the normalized conductance and k_p, k_d the potentiation and depression coefficient respectively, defined as a function of input voltage only:

$$k_p = k_{p_0} e^{\eta_p V} \tag{1.2}$$

$$k_d = k_{d_0} e^{-\eta_d V} \tag{1.3}$$

Equations 1.2 and 1.3 derive from the physical description of the ionic diffusion [44–46], making the model a physics-based one. Defining the conductance range of each edge as $[g_{min}, g_{max}]$, the current flowing into it will follow the first Ohm's law:

$$I(t) = [g_{min}(1 - g(t)) + g_{max}(g(t))]V(t) \tag{1.4}$$

The low computational cost of the model is linked to the analyticity of the solution, which can be casted into a recursive fashion on a discretized time domain, emphasizing the memory property of the memristive edges:

$$g(t) = \frac{k_p}{k_p + k_d} \left\{ 1 - \left[1 - \left(1 + \frac{k_p}{k_d} \right) g(t - \Delta t) \right] e^{-(k_p + k_d)\Delta t} \right\} \tag{1.5}$$

In this context, the model consists of 7 parameters (k_{p_0} , k_{d_0} , η_p, η_d, g_0, g_{min}, g_{max}) that can be retrieved from interpolation of experimental measurements. The values of these paramenters are provided in [20]: $k_{p_0} = 2.56\,\mu s^{-1}$, $k_{d_0} = 64,9\,s^{-1}$, $\eta_p = 34,9\,V^{-1}$, $\eta_d = 5,59\,V^{-1}$, $g_{min} = 1,01\,mS$, $g_{max} = 2,72\,mS$. The parameter g_0 is the initial conductance value of the memristive element from which the recursive solution (equation 1.5) starts. Since the initial conductance distribution across the physical nanowire network is not known, g_0 is assumed to be identical for each edge. The g_0 value for each edge was defined as the value such that the experimental pristine conductance measurement of the network (i.e. the effective conductance in between two points of the network) across two terminal is reproduced through data interpolation [47]. This approach is further corroborated by the fact that the effect of initial state of the network vanishes after few iterations of the model solution.

1.1.4. Random graph modeling of nanowire networks

After having discussed how the single junctions are modeled, it is now reported how the networks is modeled in order to carry out simulations.
Ag-PVP NWNs are modeled in this work by means of random graph modeling, which is used also in other similar studies such as [22, 48, 49].
Random graph modeling of NW network is done by assuming the NWs as 1D objects distributed over a two dimensional plane. The random NW distribution over a two dimensional plane was done using Loeffler et al method [49]. Here, the mean length of NW is taken as 40 μm and standard deviation is considered as 14 μm. The NW network is mapped as a graph where each NW node is represented by a dot and the junctions or edges are represented by link. The modeling, analysis and simulation are done on the Python 3.9.12 platform using version 2.6.3 of *NetworkX* package. An example of random graph models of NWNs is shown in figure 1.8, where are depicted graphs of NWNs with different densities (obtained by increasing the number of nanowires). Each blue dot (node) in this graph represents a Ag-PVP nanowire. The number of nodes (nanowire) in the network is indicated as N. The black edges that link nodes represent the memristive junctions which occur at nanowire-nanowire intersection. It is important to notice that, in general, not all the nanowires are connected in a single component. The number of connected and isolated nodes depend on the density of NWs [50]. The normalized node density D is calculated with the following equation.

$$D = N\frac{L^2}{S} \tag{1.6}$$

Here, N is the number of nanowires, L represents the mean length of nanowires and the area of the plane is expressed by S.
There is a threshold value of density after which most of the nodes gets connected with the largest connected component (figure 1.8). This value of density is known as percolation critical density (D_c) and its value is $D_c \sim 5.63$ for two-dimensional networks of identical nanowires [51]. For real systems the value of D_c depends on both the finite system size S and the stick length distribution [52]. However, in this work $D_c = 5.63$ is assumed as the critical value of reference and only networks with D greater that this value are considered in order to have a connected component between the input and output electrodes in almost all the random realizations of the network [50]. In particular, networks with a number of nanowires greater than 150 (corresponding to $D \sim 8$) are considered.

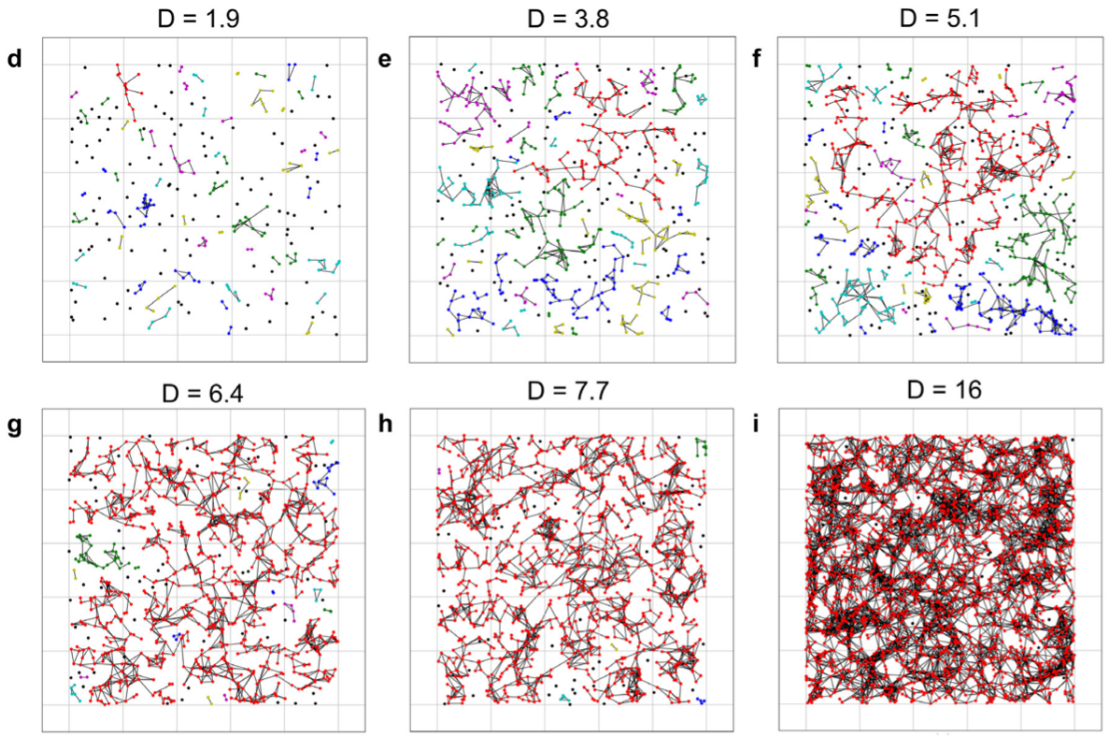


Figure 1.8: Emergence of a giant connected component. **(a)** Number of connected components as a function of the NW network normalized density. **(b)** Fraction of isolated nodes as a function of the normalized network density. **(c)** Fraction of nodes in the largest connected components as a function of the normalized network density, showing a phase transition at a critical percolation density of $D_c \sim 5.63$. **(d-i)** Graph representation of modeled NW networks with increasing densities, showing the emergence of a giant component by increasing D. Nodes belonging to the largest component are depicted in red. Adapted from [50].

1.1.5. Modified voltage node analysis

Once defined the graph that represents the nanowire network, the electrical simulation of the network needed to compute the conduction evolution of its edges (see section 1.1.6 for details) has been simulated by implementing the modified voltage node analysis (MVNA) algorithm [53]. The idea is to place voltage signal generators between stimulated electrodes and solve the electrical circuit of a grid of resistances. Moreover, at each computational time step, each edge resistance value needs to be updated according to equation 1.1 depending on the voltage drop across the particular memristive edge. In this scenario each memristive junction in the network is considered as a resistor of a certain resistance value for each time step. The algorithm consists in solving a linear system of equations, which in case of independent current and voltage sources results to be:

$$Ax = z \tag{1.7}$$

Considering a graph with N nodes and M sources, the matrices definition is:

$$A = \begin{bmatrix} G & B \\ C & D \end{bmatrix} \tag{1.8}$$

$$x = \begin{bmatrix} v \\ j \end{bmatrix} \tag{1.9}$$

$$z = \begin{bmatrix} i \\ e \end{bmatrix} \tag{1.10}$$

where:

- G is a $N \times N$ matrix containing along the diagonal the sum of elements con- ductance connected to a each node and off-diagonal elements are the negative value of element conductance connected to each pair of nodes
- B is a $N \times M$ matrix containing 0, 1, -1 values corresponding to the presence of a source between two nodes and in which orientation
- C is a $M \times N$ matrix corresponding to the transpose of B
- D is a $M \times M$ matrix full of zeros in case of independent sources
- v is a $N \times 1$ matrix with each element corresponding to node voltages
- j is a $M \times 1$ matrix where each entry is the current flowing through each voltage source
- i is a $N \times 1$ matrix with each entry equal to the sum of currents through each element connected to a certain node
- e is a $M \times 1$ matrix corresponding to independent voltage sources

The system implements the Kirchoff current law at each node of the graph, while introducing additional equations for each source of the circuit.
Estimation of computation time to perform MVNA in a single time step as a function of the number of nodes in the graph (i.e. the number of nanowires in the network) has been carried out and the results are reported in figure 1.9, which shows a proportionality of the

computation time to $N^{1.9}$. Dependence of computation time on a power of N is expected since the algorithm bottleneck is linked to the matrix inversion operation, performed by means of Python function *numpy.linalg.inv()*. Moreover, the total computational time of the simulation increases linearly with the considered number simulation time-steps, since a fixed dimension matrix needs to be inverted for N_t times, with N_t the number of time-steps. In the simulations performed in this work, N_t corresponds to the total length of the input signal which is given by the sum of the values of the simulation parameters `cut`, `duration`, and `num_lags`, which are discussed in the following.

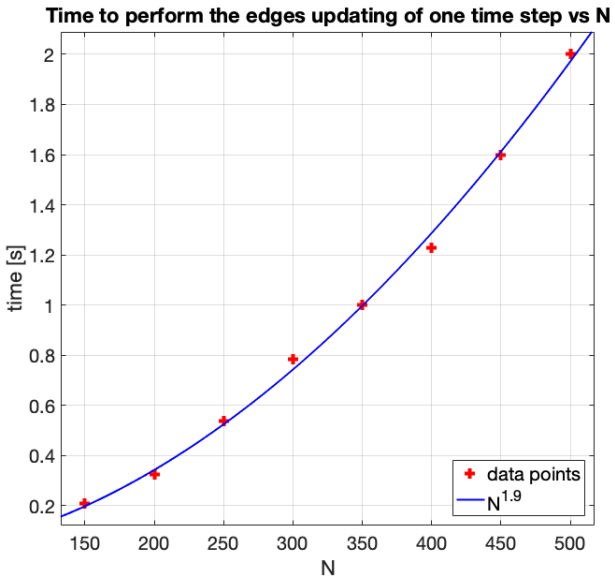


Figure 1.9: Computation time to perform the edges conductance update of a single time step [s] vs the number of nodes in the graph N (nuber of nanowires in the network). The computation time is proportional to $N^{1.9}$.

1.1.6. Electrical stimulation of nanowire networks

The simulations carried out in this work consist in obtaining the evolution in time of the edges electrical conductivity and of nodes potential when the network is electrically stimulated. This is done by using the junction model presented in section 1.1.3 and modified voltage node analysis (section 1.1.5).
The electrical scheme through which the real networks are electrically stimulated is depicted in figure 1.10, and it has been implemented in the simulation python code.

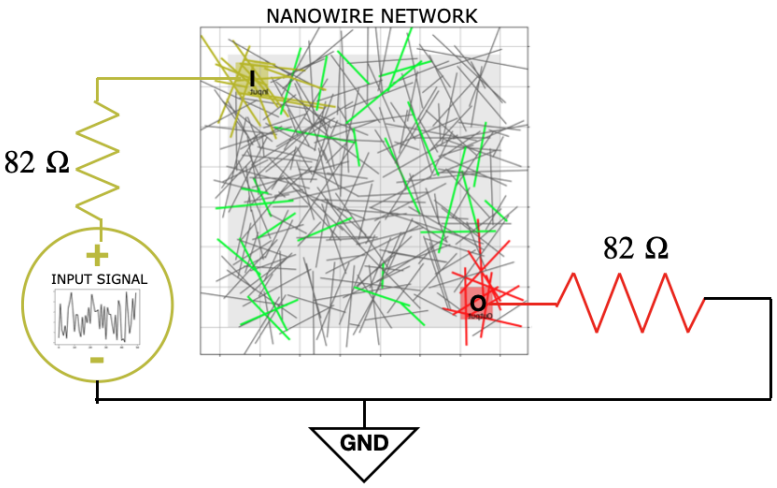


Figure 1.10: Electrical scheme for nanowire network electrical stimulation.

The electrical stimulation of nanowire networks is simulated by means of the algorithm represented in figure 1.11. The algorithm consist in evaluating, at each time step of the application of an input signal, the voltage of all the nanowires in the network by means of MVNA, and than updating the conductivity of memristive junctions using 1.1 so that MVNA at the following time step can provide new voltages of the nanowires. In other words, the algorithm consists in solving at each time step a resistive network in which the resistances of resistors depend on the previously applied electrical stimuli.

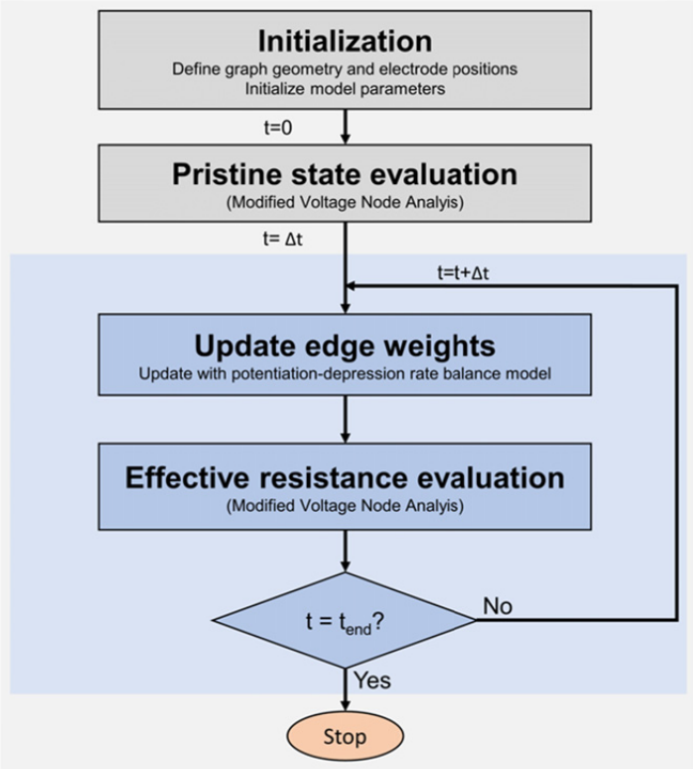


Figure 1.11: Algorithm for modelling the memristive emergent dynamics of the network. After initialization and pristine state evaluation, graph edges are updated after each time step according to the potentiation-depression rate balance model during electrical stimulation (blue box). Adapted from [47].

1.2. Reservoir computing

After the overview on Ag-PVP NWNs concerning their physical behaviour and modeling given in the previous section, it is here introduced the concept of physical reservoir computing (PRC), which constitutes the framework into which Ag-PVP NWNs are applied in this thesis work.

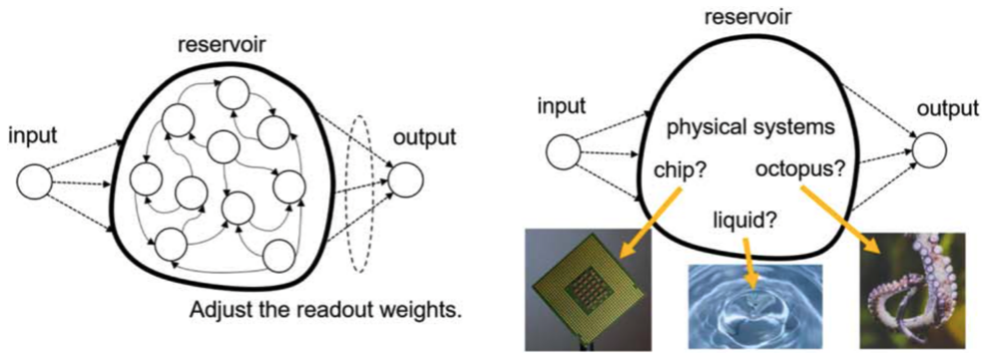


Figure 1.12: **Left:** Typical ESN setting, a representative model in the RC framework. The reservoir is an RNN often equipped with a nonlinear activation function. Only the readout part is trained to the target function. **Right:** Physical reservoir computing (PRC), which exploits the physical dynamics as a reservoir. Adapted from [18].

The reservoir computing (RC) paradigm is a framework for recurrent neural network (RNN) training that was born to simplify the training of RNN. In fact, RNN presents recursive connections that if on one hand they make the network enough articulated so that a variety of complex tasks can be performed, on the other hand they also make the training process very slow and energy consuming when the network degree of complexity increases.

RNNs basically refer to a couple of training algorithms: backpropagation through time (BPTT) [54, 55] and real-time recurrent learning (RTRL) [56, 57]. The former works by unfolding the RNN in time and training it as if it was a forward neural network (FNN) with a backpropagation method [56], showing issues in long-term dependencies learning. The latter shows advantage in online learning, but with a too high computational cost. For this reason, the introduction of a reservoir, seen as a black box, could mimic the recurrent network in a more rapid, efficient and natural way. The first approaches in RC were independently developed by Jaeger et al. [58] and Maass et al. [59]. The former dealt with the echo state networks (ESN), while the latter with the liquid state machine (LSM).

The competitive advantage of RC approach relies in the training operation, which acts only on the readout function (one-layer neural network) which takes information from the reservoir and translate them into problem solution by a linear transformation. The great advantage of this computing approach relies on the lower computational and energy cost to perform training. Moreover, in principle, a given reservoir may be equipped with several readout functions to have a general purpose system exploiting the same reservoir [18, 60].

The basic working principle of RC is to use the reservoir to nonlinearly map the input

signal into a higher-dimensional state space in which the signal is represented and better processed or classified by the linear readout function [61]. This is achieved through the use of a large number of reservoir nodes that are connected to each other through the recurrent nonlinear dynamics of the reservoir [61]. In order to perform RC, the reservoir should satisfy three important requirements [18, 61]:

- its elements must be able to store information
- it must be made up of independent units exhibiting a non-linear behavior
- it must exhibit the separation property
- it has to be designed such that the effect of an input on the reservoir must vanish after a certain time (fading memory [59])

The first is fundamental to ensure a memory effect to highlight input time corre- lations, the second is essential to solve complex tasks, the third is important to map different inputs on different reservoir states and the latter ensures a memory of recent past and not of distant past. The latter is also known as echo state memory [58]. Recent trends on RC outline how, despite it was born to deal with temporal pattern recognition, it can be exploited with many other machine learning problems by properly transforming the input data into temporal pattern. The relevant applications of RC are linked to spoken digit recognition [59], human activity recognition [60], handwritten digit recognition [61], waveform classification [62], sine-wave generation [63] and so on.
For what concern the reservoir implementation, it can be implemented both in software or by a dynamical physical system. In the former case, are normally used RNN with nonlinear activation function of the neurons (figure 1.12 right panel). Instead, in the latter case, the reservoir can be any dynamical system (figure 1.12 left panel) that satisfies the previous cited requirements. In the case the reservoir is a physical system, the RC framework is called "physiscal reservoir computin" or "in materia computing" [18]. Literature is available concerning mechanical [62], electronic [63], photonic [64], spintronic [65] and biological [66] reservoir type and a review can be found in [18]. In this thesis, the interest is in using the Ag-PVP NWNs as reservoirs for PRC, as it has been done in [19–22, 67], with the aim of assessing their memory properties.

1.2.1. PRC with Ag-PVP NWNs

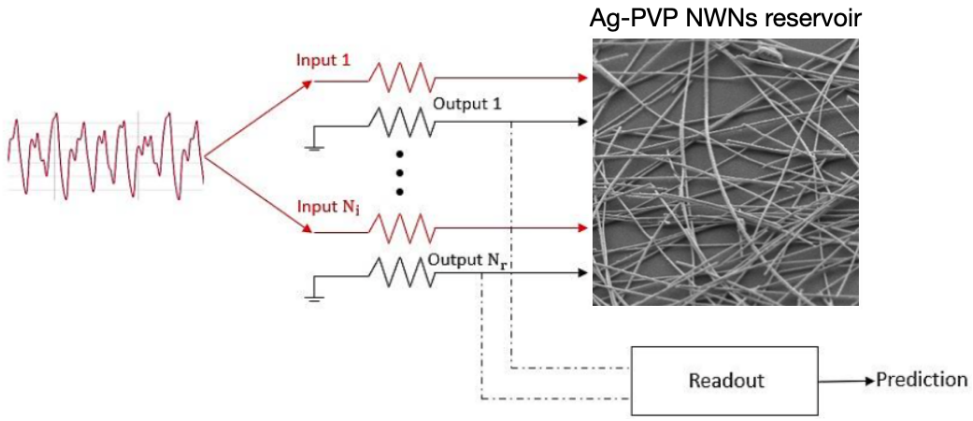


Figure 1.13: Ag-PVP NWNs used as reservoir in PRC framework.

According to the definition of RC, self-assembly nanowires network represents a good candidate for the reservoir. The memory effect and the non-linearity behavior are guaranteed by the memristive nature of the Ag-PVP nanowires cross-point junctions and by recurrent connections among them. Moreover, the separation property and fading memory effect have been investigated and proven in [23].
The way the Ag-PVP NWNs are used as reservoir is depicted in figure 1.13 and it consists in sending an electrical input signal by applying a voltage to the networks. The network would, then, be potentiated depending on the particular input structure, exploiting the physical phenomena discussed in sections 1.1 and 1.1.2. Therefore, each input produces different network evolution according to its spatio-temporal structure. In this way, the input is nonlinearly mapped into a higher-dimensional reservoir state, which is defined in this work as the collection of readout nodes voltages. Once the input is sent, the voltages of readout nodes are red by the readout module, which linearly combine them to give an output (prediction of delayed input signal in the case of this work).

1.2.2. Readout module

The readout module is the part of the PRC system which reads the reservoir state and performs a transformation of it to return the desired output for what it has been trained for. Usually, in PRC framework, the transformation of the reservoir state performed by the readout module is a linear one [18]. The readout module is the only part of the PRC system that has to be trained and the training can be simply performed by a linear regression which returns the weights of the readout layer [18, 20, 21, 25].
Several implementations are possible for the readout layer. An interesting one is that used

in [20], which consisting in a crossbar array of $metal - TaO_x - metal$ memristive junctions (figure 1.14). In this way, the whole system composed by reservoir and readout module is a memristive system. This kind of readout function is able to physically perform the linear transformation of the reservoir state since the current at a single junction depends on the voltage and on the conductance of that junction, and it is calculated by the Ohm's law ($I = GV$). Moreover, the current at a single column follows Kirchhoff's law where the current at a column, I_j is represented by the following equation 1.11. If the voltages V_i are the voltages of the readout nodes and their collection represents the reservoir state, then the current I_j are a linear transformation of the reservoir state.

$$I_j = \sum_i V_i G_{ij} \tag{1.11}$$

In this work, the readout layer consists simply in a linear combination of the voltages of the readout nodes, and no real devices acting as readout layer are simulated.

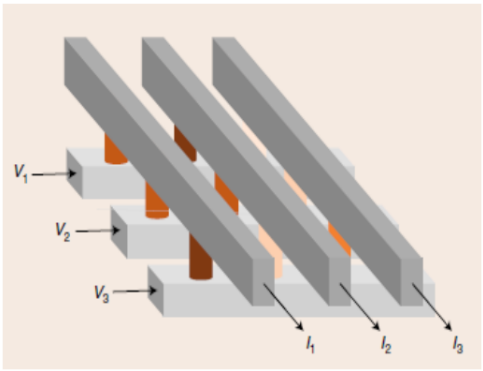


Figure 1.14: Schematic of the implementation of the readout function using a crossbar array memristive device. Different shades of orange represents the different conductance state of the memristive materials.

1.2.3. Memory capacity

As explained in previous sections Ag-PVP NWNs presents many brain-like properties emerging from the memristive behaviour of nanowires cross-point junctions and from recurrent connections among them which originates from the self-assembling of the networks. This work is aimed to assessing one of these brain-like properties of the Ag-PVP NWNs: the short term memory (STM) property he memory properties of Ag-NWNs. In order to this, it is used the benchmark test of memory capacity (MC) introduced by

Jaeger in [24] to assess STM of software-implemented ESN. In literature, MC benchmark test has already been used also for evaluating STM of other recurrent networks such as biological neural networks [25] and also for Ag-PVP NWNs [21, 22].
The MC benchmark test consists in training a neural network (in the case of PRC the readout layer only) to generate at its output units a delayed version of a single-channel input. For a certain delay, the squared correlation coefficients between the trained network prediction of the delayed input and the actual delayed input (target) is called MCk, and it is computed as expressed in equation 1.13. The sum over k of the MCk coefficients is the MC of the network (equation 1.13).

$$MC_k = \frac{cov^2\left(\nu(t-k),\ y_k(t)\right)}{\sigma^2\left(\nu(t)\right)\sigma^2\left(y_k(t)\right)} \tag{1.12}$$

$$MC = \sum_{k=1}^{\infty} MC_k \tag{1.13}$$

Once the readout module is trained to predict delayed versions of the input signal, the ability of the readout nodes to represent past inputs in their current activation states (i.e. the MC), is proportional to the performance of the nanowire network in the memory task i.e in its ability to encode past input history in its ongoing dynamics. The magnitude of the memory capacity is then proportional to the ability of the reservoir's activation states to encode both past and present input stimuli [25].

2 | Results

This simulative work deals with the measurements of the MC of neuromorphic self-organizing nanowire networks and with the exploration of the dependence of this quantity on the system parameters.

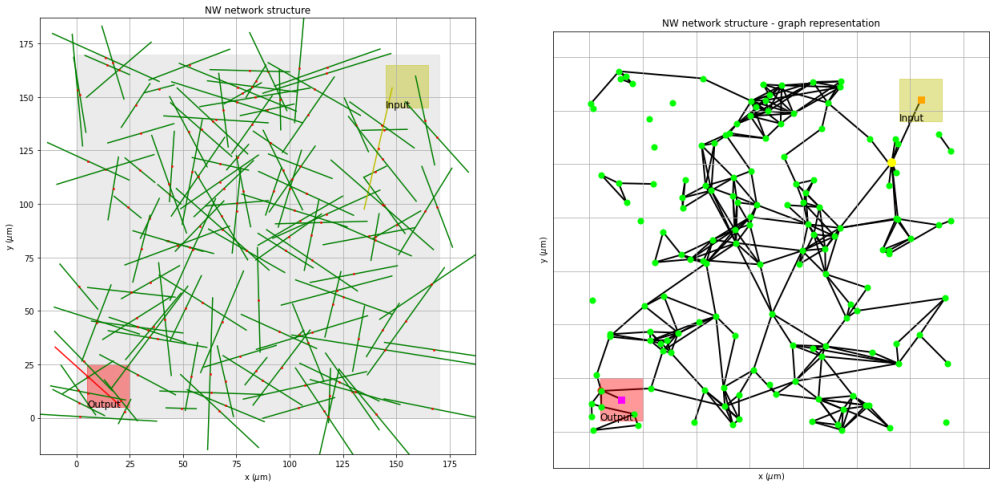
In this chapter it is first presented the algorithm that has been developed to compute the MC of the networks, then a discussion on the MC measurement conditions is carried out, and finally the results of the simulations aimed to find the dependencies of MC on system parameters are reported.

For all simulations, the dimensions of the deposition area are kept fixed at 170×170 μm^2. The wire lengths are normally distributed with mean 40 μm and standard deviation 14 μm. In order to model the self-assembly of the network, the x and y positions of the centers of the nanowires on the deposition area are distributed according to a uniform distribution $U([0, 170])$, and the orientation of the nanowires is given by an angle $\theta \sim U([0, 2\pi])$.

In figure 2.1 is reported a schematic of an example of a simulated nanowire system and its graph representation. In figure it is possible to see the input electrode (yellow square) which is linked to the voltage generator which generated the input signals which are fed into the network, the output electrode (red square) which is grounded, and the nanowires (nodes in the graph representation) that have different colors according to their function. In particular, yellow nanowires (nodes) are linked to input electrodes, and they will be referred to as "input nodes". Red nanowires (nodes) are linked to output electrodes, and they will be referred to as "output nodes". Green nanowires (nodes) are reading nanowires, which can be thought to be linked to reading electrodes (which are not grounded), their voltages take part in the network state $\mathbf{x}(t)$, and they will be referred to as "reading nodes". Gray nanowires (blue nodes) are nanowires whose voltage does not take part in the network state, and in practice they can be thought as nanowire which

are not accessible by a reading electrode (see [23] for a similar nanowire organization and nomenclature).
The state of the network $\mathbf{x}(t) = (V_1(t), V_2(t), ..., V_{Nout}(t))$ is constituted by the collection of the voltages of input nodes, reading nodes, and output nodes, and $Nout$ is the sum of the number of input nodes, the number of reading nodes, and the number of output nodes. All the nodes involved in the network state are referred to as "readout nodes".
In this work, the reading nodes are always all the nodes that are neither input nor output nodes, with the exception of section 2.3.3 in which the number of reading nodes is variable. In this way the state of the network is constituted by the voltages of all the nanowires in the network. This hypothesis should lead to theoretical maximum values of MC of the networks [21], and this is confirmed by the results obtained in section 2.3.3.

The electrode configuration is that reported in figure 2.1 in all simulations until section 2.3.5, in which other electrode configurations are tried. The electrode configuration in figure 2.1 is called "diagonal" and it is used in other studies such as [34, 50]. The electrodes have been implemented in simulations as explained in section 1.1.1, and the electrical scheme that is simulated is that reported in section 1.1.6. In all simulations, all input voltage values are positive, which means that the network junctions are operating in a strictly unipolar regime, as in [21].



(a) (b)

Figure 2.1: lorem **(a) :** Nanowire network 150NWs aggiungi densità. **(b) :** Graph representation of the nanowire network in (a).

2.1. Memory capacity computation algorithm

The computation of MC of the nanowire networks is performed following the approach used in [21, 22, 25, 68], and it consists in a training phase of the system followed by a test phase in which the MC is evaluated.

For what concerns the training phase, the readout module (discussed in section 1.2.2) is trained to approximate a delayed version of the input signal at different time-lags. More precisely, there are as many linear units in the readout module as there are time-lags considered for the evaluation of the MC, and each linear unit is independently trained from one another.
In order to perform the training of the readout module an input signal $\nu(t) \sim U([V_{min}, V_{max}])$ is applied to the network through a set of input nodes, and simultaneously the state of the network $\mathbf{x}(t)$, consisting of the voltages of all the readout nodes, is recorded in time. The input signal is of the type depicted in figure 2.2 and it is constituted of three parts. The first part is `cut` points long and it is not used to compute MC but to initialize the system so that any transients (see fig 2.9) or dependencies on initial conditions are removed. The second part is `duration` points long, and it is the portion of the signal used for MCk computation at a certain delay k. The third part is `num_lags` points long and it is the part in which the signal used for MCk computation is shifted into when the delay k increases. `cut`, `duration` and `num_lags` are user-defined parameters.
The recorded time series of networks states is then used to train a linear regression model to reproduce the same input signal at different time lags (see the section 2.1.1 for details).

Finally, the MC is evaluated through a test phase (see the section 2.1.2 for details) in which another input signal of the same kind of the training signal is fed into the nanowire network and predictions of the delayed input are performed by the network with the trained readout module.

The above described procedure to compute the MC of the nanowire networks is performed following the algorithm which is here reported:

- **Training phase**

    1. Feed the network with an input signal $\nu(t)$ generated by sampling from a uniform distribution, and collect the states of the network during time (network history) by reading the voltage at readout electrodes.

2. Discard the first `cut` points of the network history that are used only to initialize the network, so that possible dependence from the initial conditions are removed and transients are extinguished.

3. Collect a number `num_lags` of delayed input signals $\nu(t-k)$ (targets). The length of the targets is `duration` and the targets are taken considering $\nu(t)$ from $end - \texttt{duration} - k$ to $end - k$ (end is the ending point of the signal).

4. Training of the weights of the readout module: The network history and targets collected in the previous steps are used as data set for fitting a linear model whose coefficients will be used as weights of the readout function of the network.

- **Test phase**

1. Feed the network with an input signal $\nu(t)$ (different from that used during the training phase) generated by sampling from a uniform distribution with the same support of that used in training phase, and collect the network history by reading the voltage at readout electrodes.

2. Discard the first `cut` points of the network history that are used only to initialize the network, so that possible dependence from the initial conditions are removed and transients are extinguished.

3. Collect a number `num_lags` of delayed input signals $\nu(t-k)$ (targets). The length of the targets is `duration` and the targets are taken considering $\nu(t)$ from $end - \texttt{duration} - k$ to $end - k$ (end is the ending point of the signal).

4. Perform a prediction of the targets at every considered delay k using the network history produced by the test input signal applied at point 1 of test phase and the readout module with the trained weights obtained during the training phase.

5. Compute MCk for every considered delay k (there are `num_lags` considered delays) using the targets $\nu(t-k)$ and their prediction computed at point 4 of test phase.

6. Compute MC as sum over k of MCk, as expressed in equation 1.13. The value of MC obtained in this way will be referred to as "MC_test" or simply "MC"

7. Compute MC repeating points from 4 to 6 using the training network histories and the training targets. The value of MC obtained in this way will be referred to as "MC_train".

This algorithm has been implemented in python.

Point 7 of test phase is performed to assess the capability of the network to generalize to new test dataset. In particular, when MC_test is similar to MC_train it means that the trained model is not overfitted and it has a good generalization capability. In this condition, the obtained MC_test values are considered to be reliable and actually indicative of the short term memory property of the network. In section 2.2 it is shown that this conditions is achieved for certain values of `cut`, `duration` and `num_lags`, and it depends on the number of nanowires and on the amplitude of the applied input signal.

In the following sections, the training phase and the test phase are discussed more in detail.

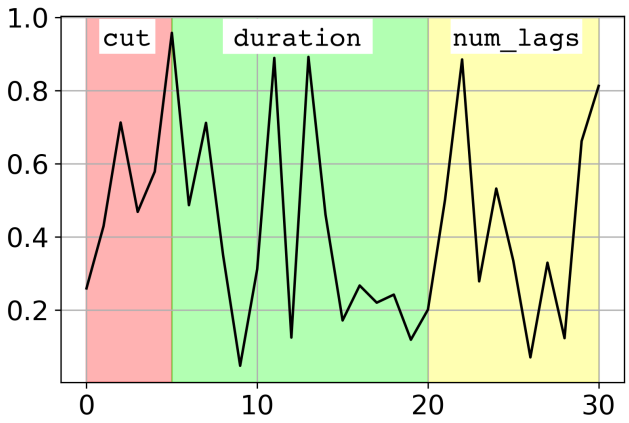


Figure 2.2: Schematic of applied input signal, $V_{min} = 0\,V$, $V_{max} = 1\,V$. The input signal is constituted of three parts: the first part (red) is `cut` points long and it is used to initialize the system; the second part (green) is `duration` points long, and it is the portion of the signal used for MCk computation at a certain delay k; the third part (yellow) is `num_lags` points long and it is the part in which the signal used for MCk computation is shifted into when the delay k increases. In this figure `cut` = 5, `duration` = 15, `num_lags` = 10.

2.1.1. Training phase

As explained in section 1.2.2, in RC framework the aim of the training phase is to train the readout module to extract from the reservoir state a desired output that is useful for the task that has to be performed. In other words, The role of the readout module is to approximate in the best way possible the task-specific target signal by reading the

reservoir state [25]. To do this, the readout module must be trained. In the present work, a linear readout module is used i.e. the readout module implements a linear regression model. This is the simplest and more common choice for the readout module in RC framework [18, 60].

In the specific case of MC task, the readout module has to reproduce delayed versions of the input signal from the reservoir state. In order to do this, in the training phase, the readout module is trained to approximate a delayed version of the input signal at different time-lags k. More precisely, as it is possible to see in figure 2.3, there are as many linear units in the readout module as there are time-lags (delays) considered for the evaluation of the MC, and each linear unit is independently trained from one another. Before the training phase, the weights of every linear units are unknown. Every linear unit of the readout module implements a linear regression model and it is fitted using the least squares approach, as it is usually done in RC framework [20–22, 68, 69] . The linear regression models of the readout module are fitted to a dataset consisting of network histories and delayed versions of the input (targets). This fitting procedure produces the weights $\mathbf{w}_k$ of the linear units of the readout functions i.e. the coefficients of the linear regressions, and it is equivalent to minimize the root mean square error (RMSE) between the targets and the predictions performed by the trained network [21].

$$RMSE = \sqrt{\frac{(\mathbf{y}_k - \hat{\mathbf{y}}_k)^2}{L}} \tag{2.1}$$

Where L is the length of the input sequence (equals to `duration` in this work). $\mathbf{y}_k$ is the target at delay k (i.e. the delayed version of the input at delay k $\nu(t-k)$), it is a vector with length equals to `duration`, and it is also referred to as $y_k(t)$. $\hat{\mathbf{y}}_k = \mathbf{w}_k\mathbf{X}$ is the prediction of the target at delay k performed by the trained network, and it is a vector with length equals to `duration`. $\mathbf{X}$ is a matrix with dimensions `duration` $\times\, Nout$ containing the network state history recorded during the application of the input signal, and it is also referred to as $\mathbf{x}(t)$. $\mathbf{w}_k$ are the weights of the linear unit relative to the k-th time-lag, and it is a vector with length equals to $Nout$.

The practical way in which the described training phase is performed is explained in the previous section 2.1, and it is depicted in figures 2.4 and 2.5.

The software implementation of the above described training procedure is inspired by that of [68].

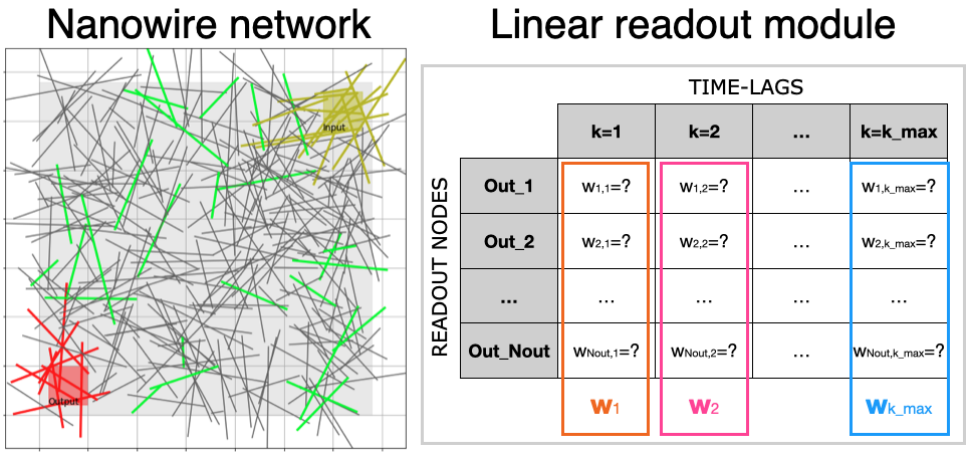


Figure 2.3: Nanowire network and readout function before training phase. The weights of the readout function are not yet determined.

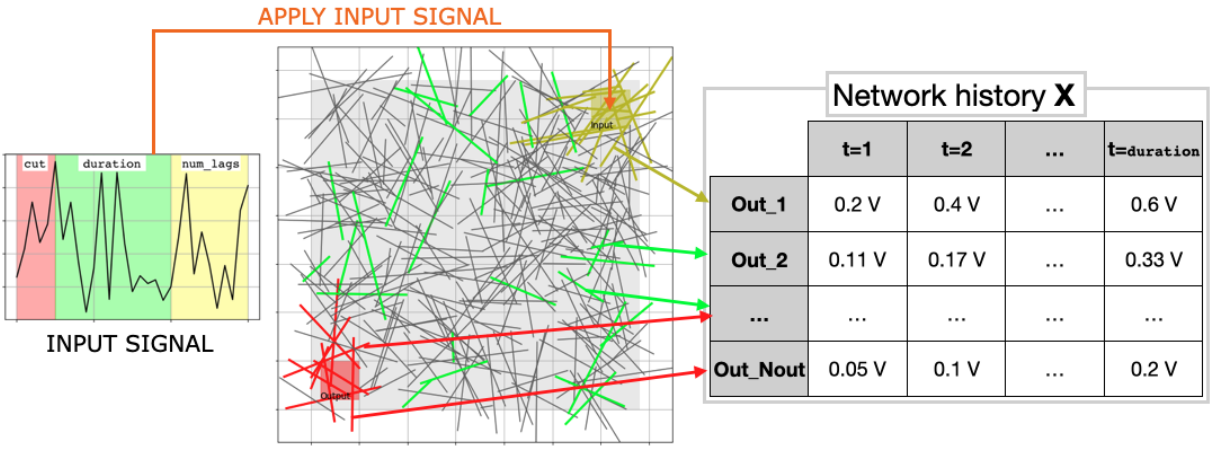


Figure 2.4: Schematic of the collection of network history duting the application of the input signal.

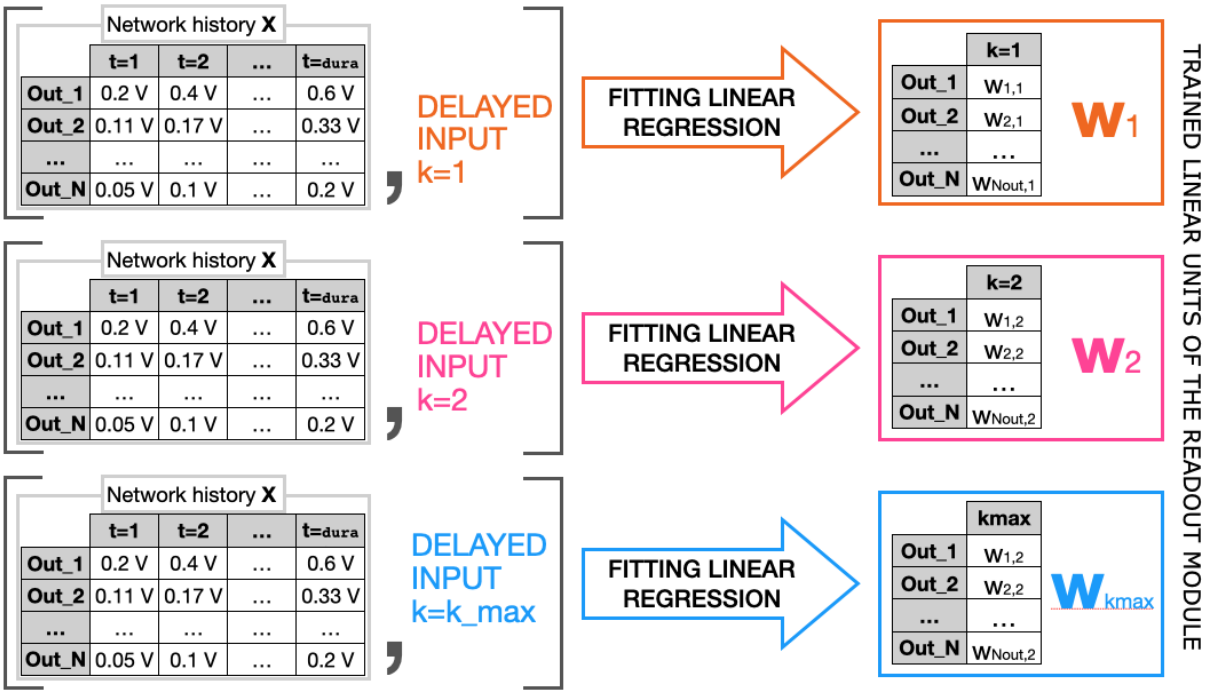


Figure 2.5: Schematic of the linear regressions performed to train the linear units and to get in this way the readout module wight.

2.1.2. Test phase

After the training phase, the system is composed by a nanowire network (the reservoir) and a readout module that is now trained to produce delayed version of the input signal. So now when an input signal is fed into the system, the output (target prediction $\hat{\mathbf{y}}_k$) will be an approximation of a delayed version of the input (the delay k depends on the linear unit considered for the readout, see fig 2.6). Since the readout module implements a linear regression, its output will be a linear combination of the voltages of the readout nodes within the reservoir (fig 2.6).

Then, by feeding the trained network with a test input signal, predictions of delayed versions of the input signal (target) are computed by the network. Therefore, MCk at each k is computed inserting targets predictions and targets in equation 1.12, and then MC is computed using 1.13.

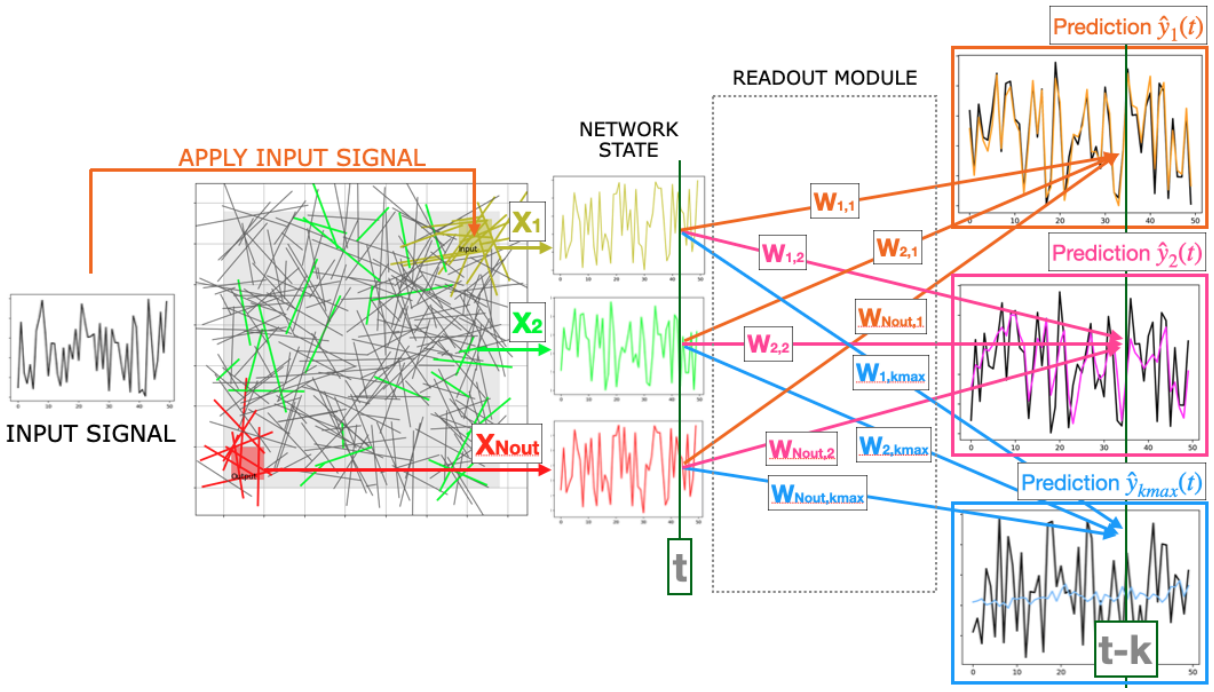


Figure 2.6: Schematic of the test phase, in which an input signal is applied to the trained systems which produces in output delayed versions of the input signal.

2.1.3. Forgetting curve and signal prediction

As a result of the test phase, MCk for every considered time-lag k is obtained. If MCk is plotted against k, the obtained curve is called forgetting curve [24] and it can be seen as a profile of the memory properties of the studied system. In fact, MCk values close to 1 means that the trained network is able to reproduce well the delayed input signal at delay k, and so that inside the reservoir at time k there is still information about the input signal at time $t - k$.

According to several studies, neuromorphic nanowire networks present short therm memory properties due to memristive junctions and recurrent loops [17, 41], so they should produce forgetting curves that have high (close to 1) values of MCk for smaller k and than decay quite rapidly to reach MCk $= 0$ for higher k. This is confirmed by the shape of the obtained forgetting curves, an example of which is reported in figure 2.7, which shows that the nanowire networks under study actually exhibit short term memory. Forgetting curves of similar shape have been obtained also in other studies dealing with software-implemented ESN [69], even if in that case a longer "plateaux" of MCk values close to 1 is present for small k. In software-implemented ESN this "plateaux" can reach also higher

values of k, such as in [24, 70].

As stated by equation 1.13, the area under the forgetting curve is the MC of the system. The absolute values of MC obtained in this work are quite consistent with that obtained in [21], where very similar systems are simulated, but very different from that obtained in [22], probably due to the physical model used to model the memristive junctions. However, this work deals with the dependencies of MC on the system parameters rather than the absolute values of MC.

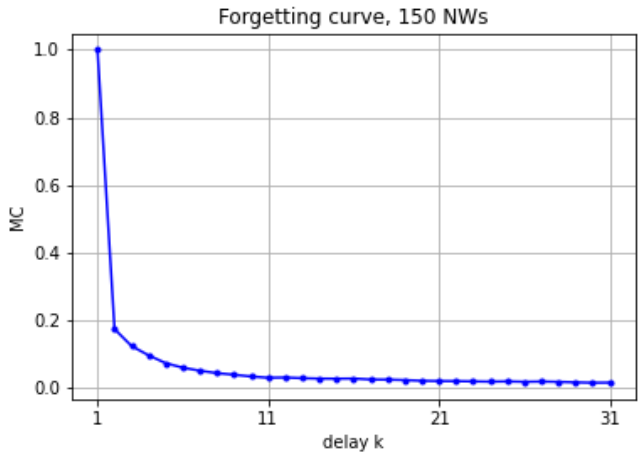


Figure 2.7: Forgetting curve of the network reported in figure 2.1, when a signal $\nu(t) \sim U([0V, 0.5V])$ is applied.

Figure 2.8 shows that the prediction of the target signal (delayed input) worsens as the time-lag k increases, accordingly to the decreasing of MCk with increasing k in the forgetting curve. This shows that, at high time-lags k, the readout module is not able to reconstruct the input signal at time $t-k$ by a linear combination of the readout nodes voltages at time t. This means that in the reservoir there is no longer information of the input signal at time $t-k$, and so that the network has no memory of a distant past. On the other hand, in the reservoir there is information of a (very) recent past and, in fact, the readout module is able to reconstruct the input signal at time $t-k$ if k is small. This demonstrate the fading memory property of the nanowire network, which is a fundamental property that a good reservoir must have [18, 60].

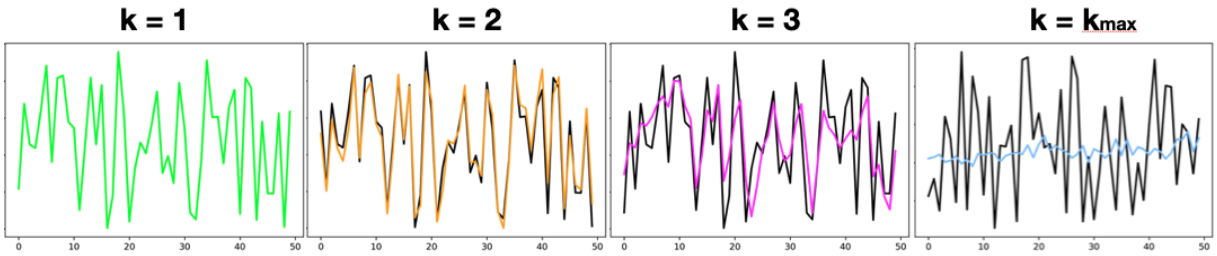


Figure 2.8: Predictions vs targets for increasing delay k from left to right. Inreasing the delay k, the prediction of the delayd input worsens.

2.2. Memory capacity measurement conditions

In many works in which MC of neuromorphic nanowire network [21, 22], or of software-implemented ESN [24, 68, 69], or of biological neural networks [25] is computed, very little attention is devoted to the measurement conditions of MC, and to how they depends on the system parameters. Usually, what is done in these studies, is to report the chosen length of the input signal used for training and test without providing further explanations regarding this choice. The aim of this section is to provide a deeper insight on MC measurement conditions and on their dependencies on physical system parameters, topics that are not so much addressed by the literature.

Here, with "measurement conditions", are meant the simulation parameters that are not physical parameters of the system, but are user-defined parameters related to the MC task and, in particular, to the input signal used to compute MC. These parameters are `cut`, `duration`, `num_lags`.
In this section it is showed that MC depends on these parameters, in particular on `duration`, and this dependency is affected especially by the number of nanowire in the network and by the amplitude of the applied input signal. Therefore, before investigating the MC dependencies on physical parameters of the system, it is necessary to identify the MC measurement conditions (that are, in theory, specific for each particular physical parameters setup) in order to obtain reliable MC values, which are truly indicative of the memory properties of the networks and not influenced by user-defined parameters related to the practical implementation of the MC task. Only in this way is it possible to obtain any MC dependencies on the physical parameters of the system.

2.2.1. Network initialization and number of considered delays

`cut` and `num_lags` parameters respectively deals with network initialization and the considered delays for MC computation.

For what concerns `cut` parameter, its functions are that of removing any dependencies on initial conditions and of removing from the network history $\mathbf{X}$ the transitory regime that occurs when an input signal is applied (fig. 2.9) and that is not object of study of this work. Therefore, `cut` must be set in such a way MC is computed in a stationary regime. As it is possible to see in fig 2.9, since 0.1 V is the smallest signal amplitude under study in this work and the transitory regime is shorter for increasing input signal voltage, this requirement is largely satisfied for every input signal voltage if `cut` = 200. This value of `cut` parameters is used in all the simulations presented in this work.

For what concerns the number of considered time-lags for the MC computation, they are defined by the parameter `num_lags`. Even if in equation 1.13 the summation is over an infinite number of terms, this is practically impossible to implement and so a finite number of time-lags equals to `num_lags` is considered in the summation. In order to make MC not to be dependent on the considered number of time-lags, `num_lags` = 200 is chosen for all the simulations. This value ensures that all the significant MCk terms in the summation are taken into account, and increasing `num_lags` above 200 does not alter the obtained MC value, since the terms added in the summation would be in the order of 10^{-5} and so they are not significant.

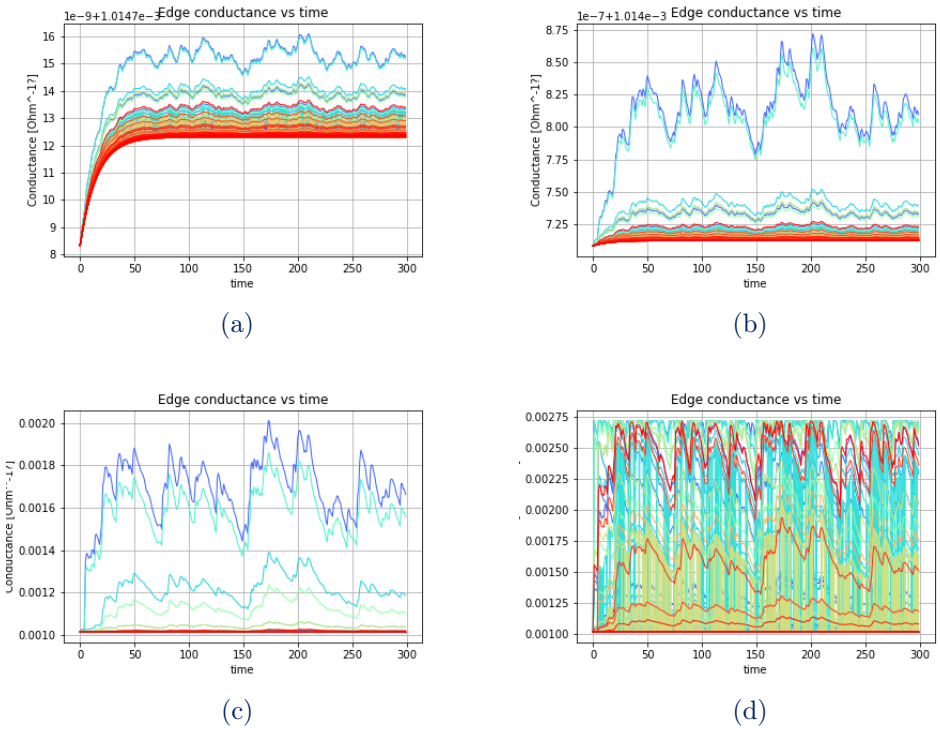


Figure 2.9: Transient at different signal amplitude. In the figures are shown the evolution of the edges conductance during the first 300 points of the applied signal. It is possible to see that at time 200 transient are extinguished at every signal amplitude. **(a) :** Signal amplitude 0.1 V. **(b) :** Signal amplitude 0.5 V. **(c) :** Signal amplitude 2 V. **(d) :** Signal amplitude 5 V.

2.2.2. Signal length and training signal

`duration` is the parameter that mostly affects the obtained value of MC because it is involved in the training of the readout module, as it is possible to see from figure 2.10, which shows the learning curve of the system. The system is composed by the nanowire network (the reservoir) and the readout module, but actually, as explained in 1.2.2, only the readout module is trained. Even if the MC is expected to be a reservoir-related quantity which should not depend on the readout module, figure 2.10 shows that MC actually depends on the input signal length and so on the training of the readout module. This happens because the readout module has to be trained in order to compute the input signal at time $t - k$ from the reservoir state at time t, so the better the readout module is trained, the better it approximates the target signal and higher is MC.
Therefore, in order to get a value of MC that is truly indicative of the memory properties of the network, the readout must be trained in the best way possible so that the obtained

value of MC is no longer dependent on the readout training. To evaluate the goodness of the training of the readout module, learning curves are used, which are a method widely used in machine learning framework.
Once the readout module is sufficiently well trained to approximate delayed versions of the input, the performance of the system in the MC task depends only on the memory properties of the nanowire network. As it is explained in the following, this happens above a certain value of `duration` which depends on the particular physical parameters of the system.

The learning curves reported in figures 2.10, 2.12, and 2.13 are composed of two curves each: the curve made of red boxes is relative to the MC values computed using in the test phase the same signal used to perform the training, while the one made of blue boxes is relative to the MC values computed using in the test phase a new test signal, which is different from that used in the training phase. The boxes represent the distribution of MC values obtained using ten different seed for the generation of the training signal, for each value of `duration`. Therefore, these learning curves show not only the goodness of the training of the readout module, but also the dependency of MC on the particular signal used for the training.
All the training curves reported in this section present the same behaviour and two main facts can be notices as `duration` increases. The first is that MC increases and the over-fitting of the linear model decreases, therefore, the training of the readout module gets better for increasing `duration`. The second is that the spreading of MC values obtained with different training signals decreases, this means that the dependency of MC on the particular training signal used for the computation decreases for increasing `duration`. These two facts both play in favour of obtaining an MC value which is truly indicative of the memory properties of the nanowire network and which is not dependent on the parameters of the particular training procedure.

For the reasons described above, ideally an infinitely long input signal should be used to obtain the MC value that perfectly reflects the memory properties of the nanowire networks. However, it is not practically possible to use an infinitely long input signal because of time and computational resources, so a minimum value of `duration` above which the obtained MC value is considered significant and reliable must be found. The aim of the next section 2.2.3 is that of finding that minimal value of `duration` for physical parameters setups of the system which are of interest for this work.

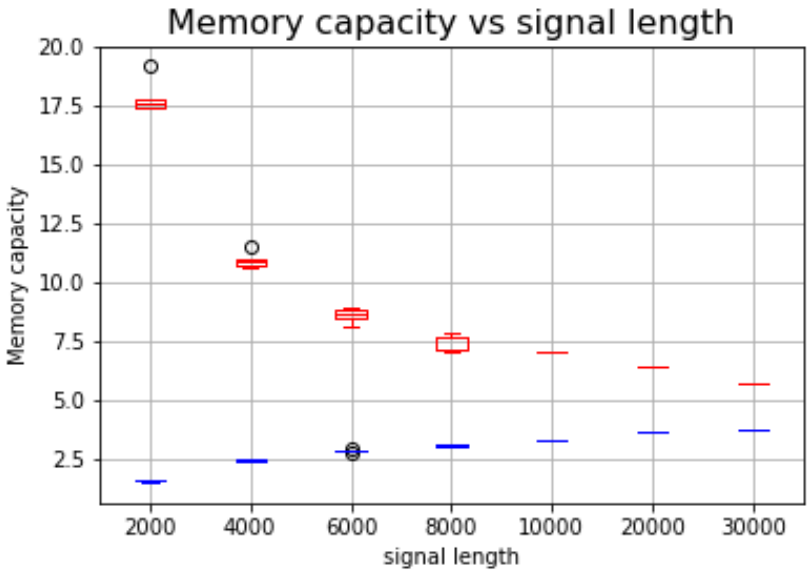


Figure 2.10: Learning curve of 300 NWs network, $V_{max} = 4\ V$.

2.2.3. Dependence of optimal signal length on signal amplitude and on number of nanowires

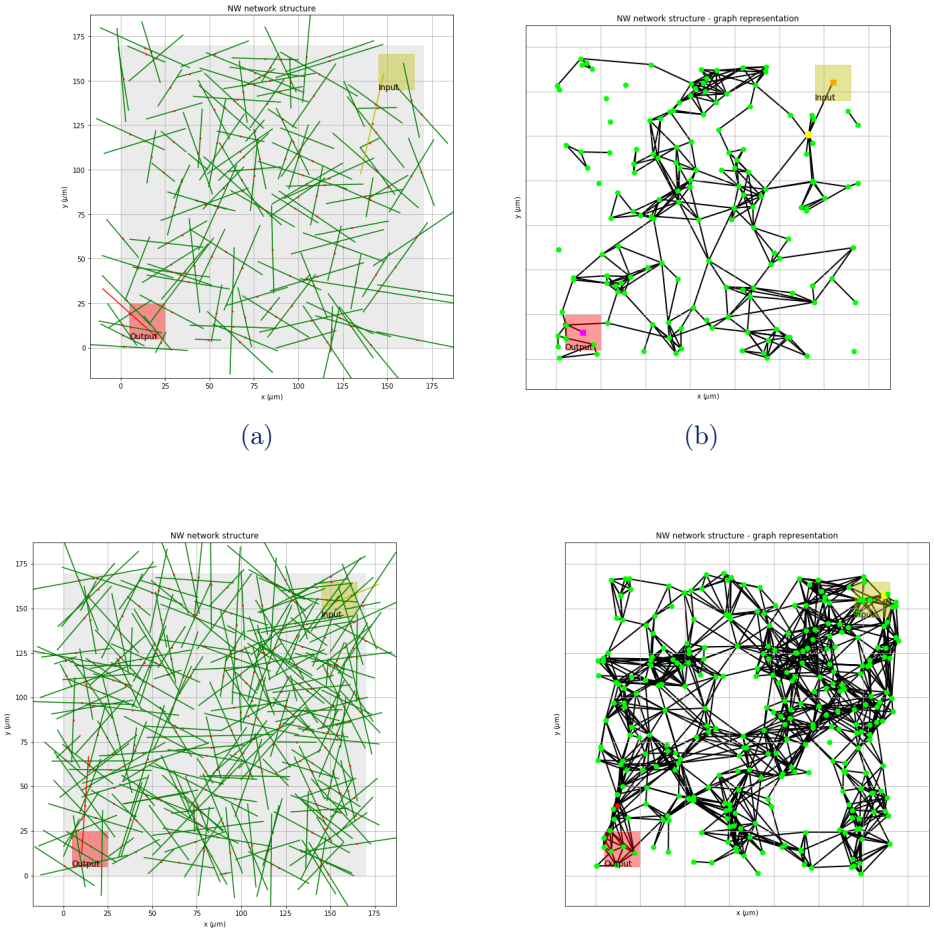
Computing the learning curves of systems with different parameters setups, it has been found that they are quite dependent in particular on the number of nanowires (NWs), and on the applied voltage (fig. 2.12, 2.13). This is probably because these two parameters play important roles in the network dynamics and in the activation of a smaller or greater number of memristive junctions in the network [22].

It is in the interest of this work to show the dependencies of MC on system parameters such as the amplitude of the applied signal, the number of nanowire in the network, the number of reading nodes, and the position of the electrodes. However, as has been explained before, it is important to find adequate values of `duration` to obtain significant values of MC and any of its dependency on system parameters.

Therefore, since the learning curves are dependent on the number of nanowires and on the applied voltage, an optimal value of `duration` which leads to significant values of MC and, at the same time, to not too long computation time and not too high memory requirements, must be found for every value of number of nanowires and applied voltage of interest.

In particular, the values of number of nanowires of interest for this study ranges from 150 NWs to 300 NWs. The inferior limit of this range has been dictated by the necessity to have a connected component in the network between the two electrodes positioned in the

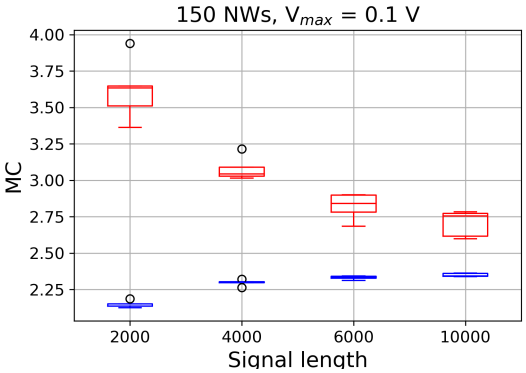
top-right corner and in the bootm-left corner of the simulation domain. In fact, 150 NWs corresponds to a normalized density of nodes of 8.3, which is well above the critical percolation density $D_c \sim 5$, so there exist a connected component between the two electrodes for most of the network realizations [50, 71]. This is consistent with experimental studies where densities of wires are typically well above the percolation threshold [17, 21]. On the other hand, the upper limit of 300 NWs has been imposed by the simulations times that would have been too long to simulate networks with a higher number of nanowires (see section 1.1.5), and by the high memory requirements that emerged from the need for storing the network history $\mathbf{X}$.
For what concerns the value of applied voltage of interest, if an input signal of the type $\nu(t) \sim U([0, V_{max}])$ is considered, the study has been limited to $V_{max} = 0.1\,V,\ 0.5\,V,\ 1\,V,\ 2\,V$. Higher values of V_{max} are not taken into account because they would have required a too high value of `duration` to obtain a significant value of MC. In fact, as it can be seen from figure 2.10, at $V_{max} = 4\,V$, `duration` $= 20000$ is needed to obtain a significant value of MC, which is a too long signal length to perform a consistent number of simulations. Actually, few simulations were performed at $V_{max} = 4\,V$, and the MC values are reported in section 2.3.1.
The following figures 2.12, 2.13 shows the learning curves obtained from systems reported in figure 2.11 when they are stimulated with an input signal of the type $\nu(t) \sim U([0, V_{max}]),\ V_{max} = 0.1\,V,\ 0.5\,V,\ 1\,V,\ 2\,V$. From these learning curves a value of `duration` $=$ 10000 has been chosen to perform all the following simulations reported in section 2.3. The chosen value of `duration` allows for a good training of the readout module, a small dependence of MC on the particular training signal, and reasonable simulation time and memory consumption.



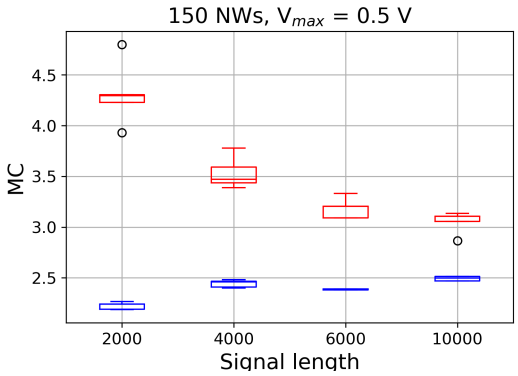
(a) (b)

(c) (d)

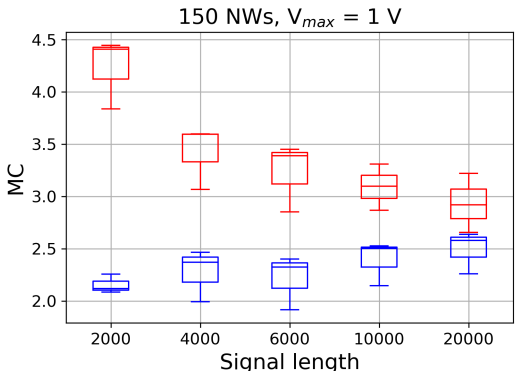
Figure 2.11: NWNs used to obtain the learning curves reported in figures 2.12 and 2.13 **(a) :** Nanowire network with 150 NWs. **(b) :** Nanowire network with 300 NWs.



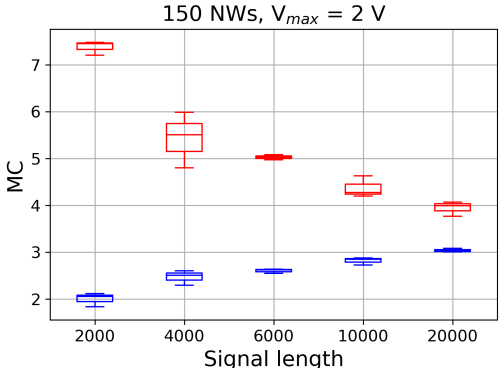
(a)



(b)

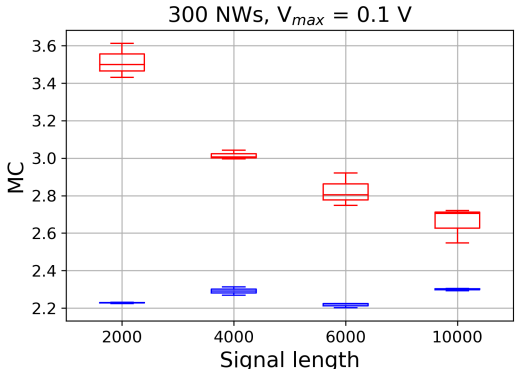


(c)

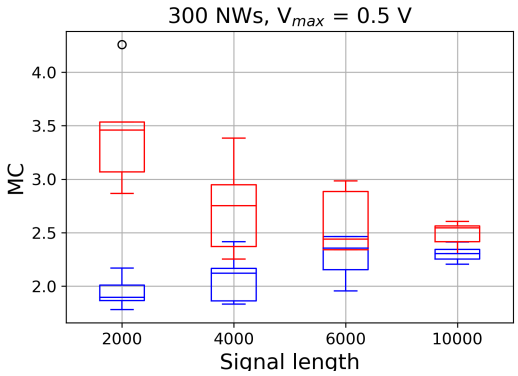


(d)

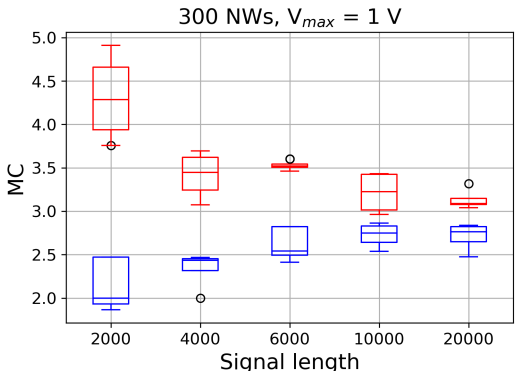
Figure 2.12: lorem **(a) :** Nanowire network 150NWs aggiungi densità. **(b) :** Graph representation of the nanowire network in (a).



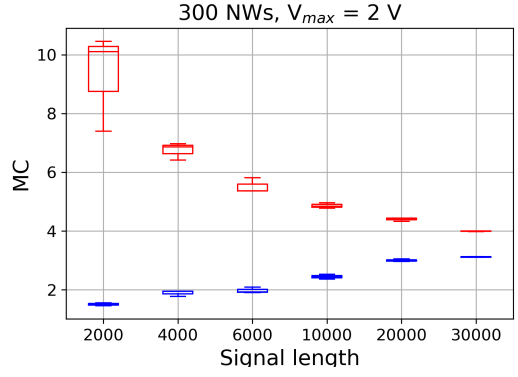
(a)



(b)



(c)



(d)

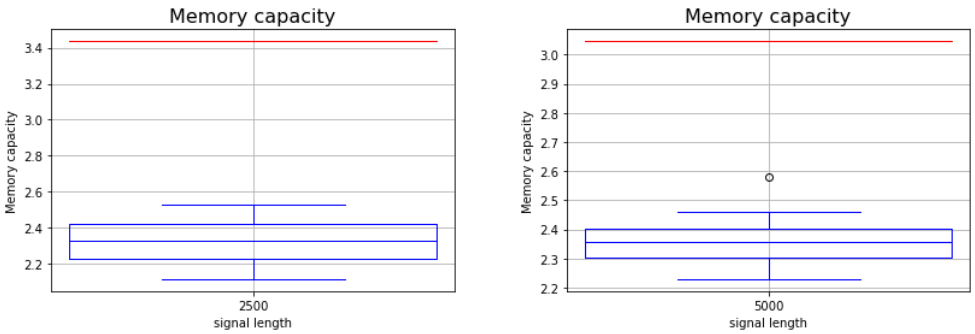
Figure 2.13: lorem **(a) :** Nanowire network 150NWs aggiungi densità. **(b) :** Graph representation of the nanowire network in (a).

2.2.4. Dependence of MC on test signal

After having assessed in the previous sections how the particular training signal affects the MC values, it is here studied the effect of the particular test signal on MC.

Figure 2.14 is the result of a simulation in which the seed for the generation of the training signal is fixed for every considered `duration` values, while ten different seeds (the same for every `duration`) are used to generate ten different test signals. In this way, the box plots represents the distributions of MC values obtained for the same training signal and different test signals. Therefore, figure 2.14 shows that there exist a dependence of MC on the particular test signal used to compute it, and that this dependence decreases for increasing signal length, since the spread of the distribution of MC decreases as `duration` increases.

This result means that choosing `duration` in order to have a good training of the readout module, i.e. increasing `duration` as much as it is compatible with the computational and time resources, has also a positive effect on the dependency of MC on the particular test signal, which is decreased. However, to have a better value of MC, it is possible to use more than one test signal and take an average of the MC values obtained with each different test signal. The developed code allows to do it, but for reasons of time, another strategy has been chosen: to fix the seed of the test signal for all the simulations so as to eliminate the dependence of MC on the particular test signal. This strategy allows to perform a smaller number of simulations and therefore to save time.



(a) (b)

Figure 2.14: Dependence on Test signal. **(a) :** duration=2500. **(b) :** duration=5000.

2.3. Effects of system parameters on Memory capacity

In this section, the dependencies of MC on several system physical parameters are explored. The system parameters taken into account are the number of nanowires (or the network density), the signal amplitude, the presence of a bias signal, the number of the readout nodes, and the electrode configuration.

For all simulations (except where specified), the MC has been evaluated using the values of the MC-task parameters `cut` = 200, `duration` = 10000, `num_lags` = 200 that has been found to be good for obtaining significant and reliable values of MC. Since at `duration` = 10000 the dependence of MC on the particular train and test signals is quite small, a single training and test signals is used for the evaluation of MC in all simulations. To further remove any dependency of the results on the particular training and test signals used, the seed for the generation of these signals is fixed in all simulations.
Furthermore, in all simulations (except for that in section 2.3.5), input and output electrodes are assumed to be linked to a single nanowire that intersect the electrode area, and they are positioned respectively in the top-right corner and bottom-right corner of the simulation domain.

2.3.1. Signal amplitude

The first considered physical parameter is the amplitude of the input signal. In particular, in this section, with "signal amplitude" it is indicated the value V_{max} of an input signal of the type $\nu(t) \sim U([0, V_{max}])$.
Five networks made of 150 NWs and five networks made of 300 NWs with different topology are here considered, and input signals with V_{max} ranging from 0.1 V to 4 V are applied to the networks. Different network topologies have been obtained by using different seeds for the random positioning of the nanowires. Figure 2.15 shows the results of the simulations. The solid lines in figure represent the means of MC values obtained for different networks realizations (topology), while the shading ranges from the minimum to the maximum values of MC obtained for a certain V_{max} and different network topology.

The obtained results shows that MC increases with increasing V_{max} both in the case of networks with 150 NWs and in the case of networks with 300 NWs. Networks with 300 NWs present values of MC which are slightly higher that that of networks with 150 NWs

in almost all the V_{max} range considered, except for V_{max} smaller than 1 V. The difference between MC of 150 NWs networks and MC of 300 NWs networks increases for increasing V_{max}. Further, the spread of MC values around the mean value is slightly bigger for 300 NWs networks, indicating a greater dependence of MC on topology for 300 NWs networks rather than for 150 NWs networks.

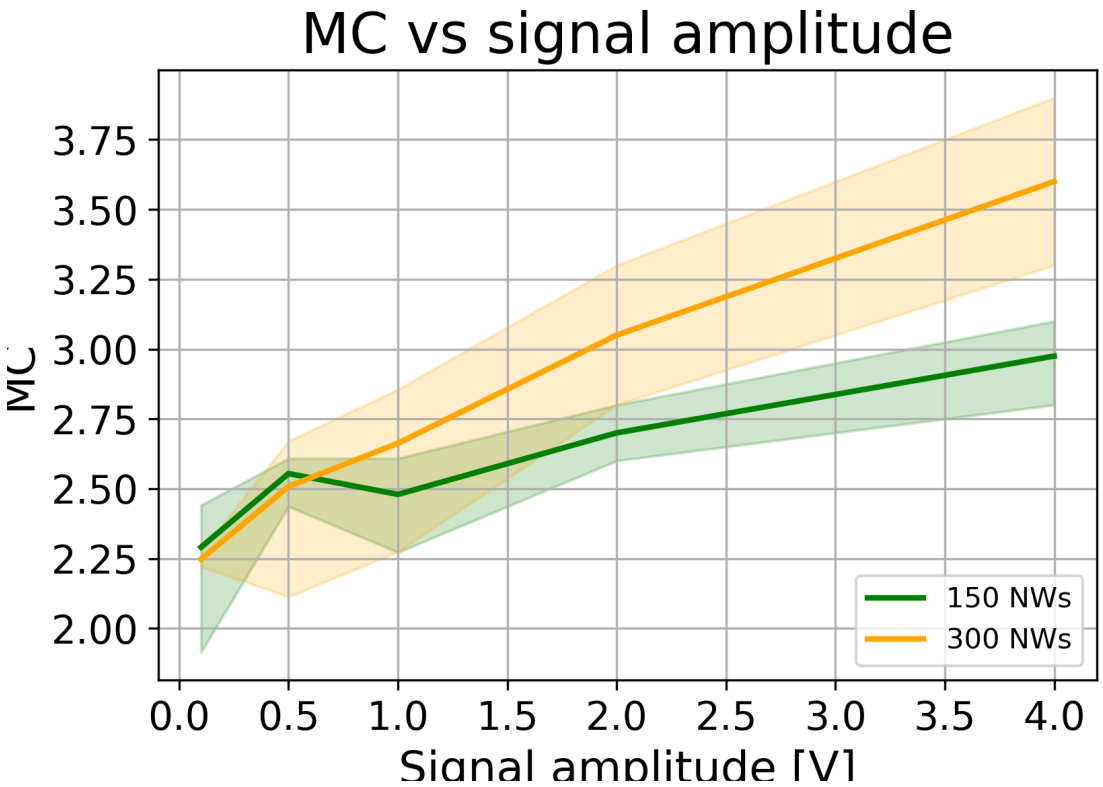


Figure 2.15: MC vs signal amplitude. red=350NWs. blue=150NWs.

The increase of MC with V_{max} is consistent to what has been found in [21, 22], and it is associated with the formation of a Winner-takes-all (WTA) conductance paths in the networks [22], as it can be seen in figures 2.16, 2.17 which display maps of conductance of the memristive junctions. In particular, a WTA path starts forming from $V_{max} = 0.5\ V$ for the 150 NWs (figure 2.16), and from $V_{max} = 1.0\ V$ for the 300 NWs (figure 2.17). The emergence of a WTA after a certain applied voltage is a typical emergent behaviour of electrically stimulated nanowire networks [34].

However, the increase of MC with V_{max} happens in a different way to that reported in [21, 22]. [22] reports a sudden increase in MC task performance once a WTA path is formed. However this is not observed both in this work and in [21]. On the other hand, the difference between this work and [21] is that here a saturation of MC at around 2 V is not observed. The saturation of MC reported in [21] is probably due to not optimal MC

measurement conditions. In fact, the training curves reported in section 2.2 shows that at higher voltages a good training of the readout module requires a longer training with respect to that required at lower voltages. It is probable that in [21] the training phase is not oprimal and so an underestimation of MC at higher voltages occurred.
Finally, figures 2.16, 2.17 could also explain why the difference of MC between 150 and 300 NWs networks is smaller at lower voltages. This may happen because at lower voltages, both networks are in a state of low activation. In this state, no conductance paths are formed between source and drain, but paths are beginning to form from each electrode to its closest neighbouring nodes [22]. In this condition, the effectively involved parts of the networks are very similar in the two cases of 150 and 300 NWs, and so performance in MC task are similar. On the other hand, increasing V_{max}, a WTA conductance path is formed, a greater number of junctions is activated, and in this condition the networks with 300 NWs seems to perform slightly better than the networks with 150 NWs.

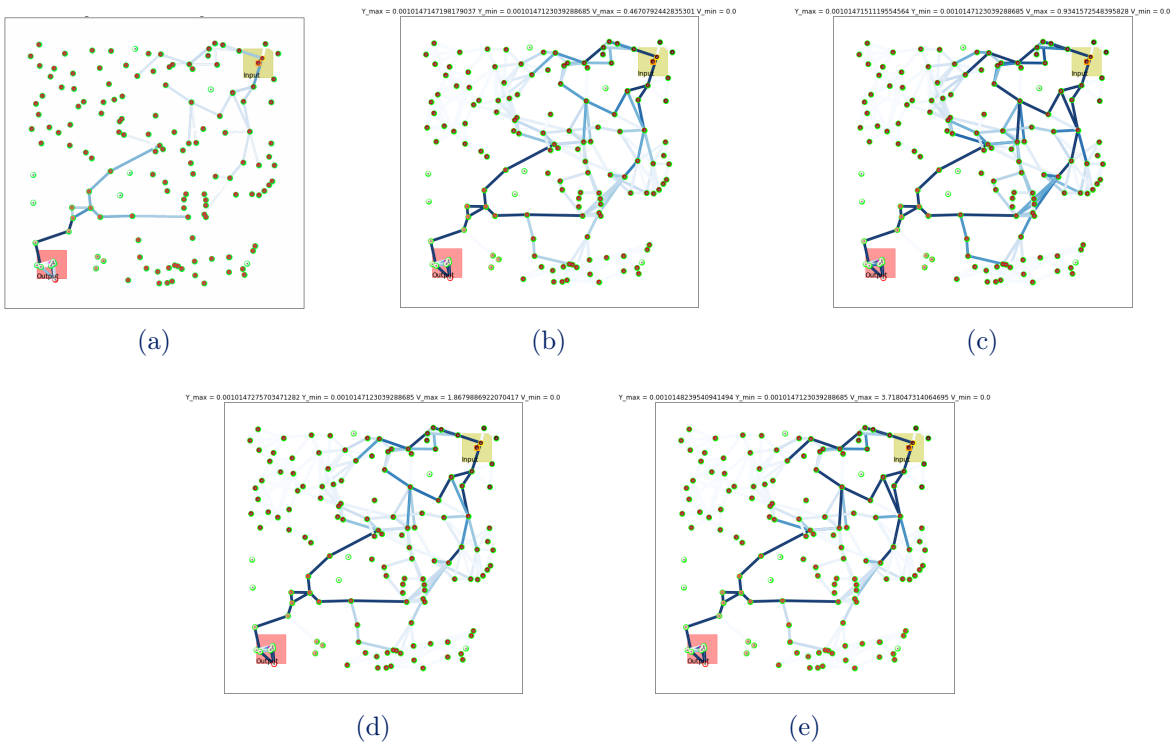


Figure 2.16: Edges conductance map in 150 NWs networks for increasing voltages.

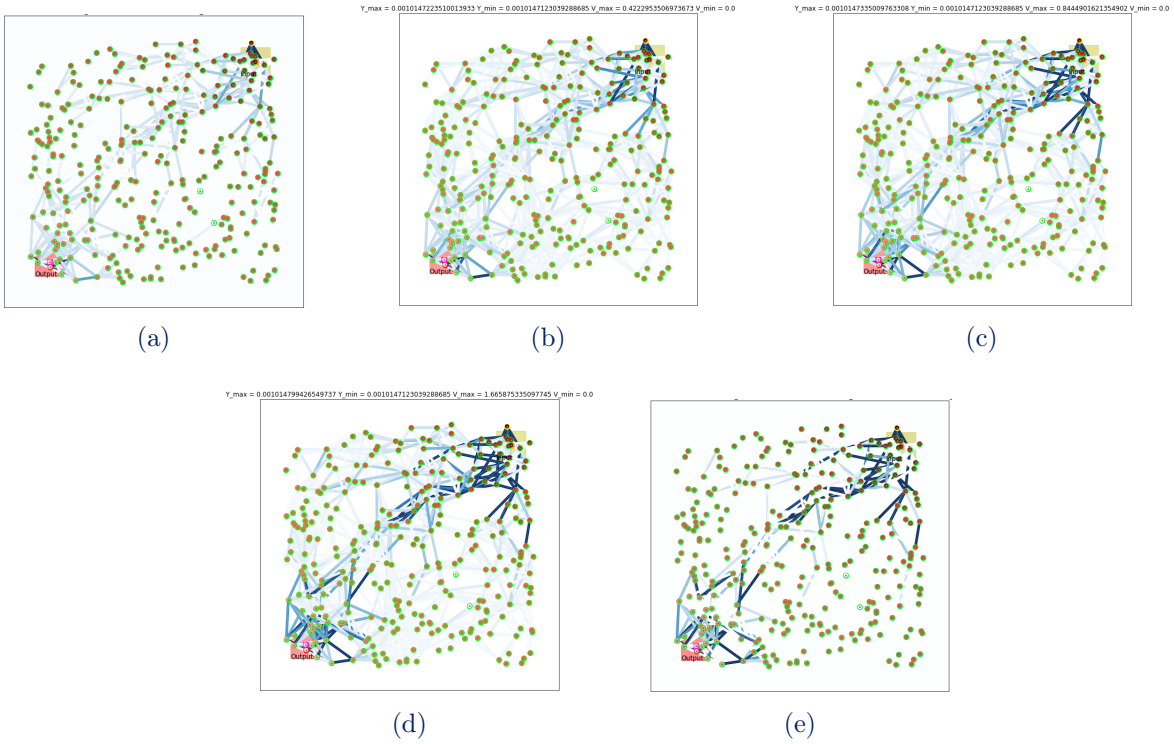


Figure 2.17: Edges conductance map in 300 NWs networks for increasing voltages.

2.3.2. Network density

It is now explored the dependency of MC on the number of nanowires (nodes) in the network, which is proportional to the normalized density of nodes D (simply referred to as "density"). The direct proportionality relation between the number of nanowires and the density of the network is given by equation 1.6.
Since the memory properties of the nanowire networks emerge as a result of internal memory of memristive junctions and feedback input from previous time steps via recurrent connections [41] and after the onset of percolation an increase of the NW density results in an increase of the network connectivity [50], it is expected a dependence of MC on the number of nanowires in the network. Further, several studies on nanowire networks [22] and on ESN [68, 69] have found dependencies of MC on the topological properties of the networks, i.e. on how the nodes are organized (linked) in the networks.
Therefore, here are reported the results of simulations in which not only the number of nanowires has been varied, but also different realizations of the network that differs for their topology are taken into account.

Figure 2.18 shows the MC of networks with different number of nanowires, for different

values of voltage V_{max}. The applied signal is of the type $\nu(t) \sim U([0, V_{max}])$. Different curves (different colors) correspond to a different V_{max}. The boxes represent the spread of MC value due to a different network topology. To obtain a different topology, a different seed for the random generation of the network has been used. Every box is the result of ten network realizations.

The obtained results shows that at $V_{max} = 0.1\ V$, MC is constant for all the considered number of nanowires ad there is almost no effect of the network topology on MC. This is probably due to the fact that at $V_{max} = 0.1\ V$, only the junctions very close to the electrodes are activated, so the major part of the network is not involved in the activation and almost all the networks perform the same in MC task in this condition. At $V_{max} = 0.5\ V$, MC remains quite constant for all the considered number of nanowires but the spread of MC value due to topology increases due to the fact that greater portions of the networks are activated with respect to the case of $V_{max} = 0.1\ V$. Finally, at $V_{max} = 1.0\ V$, a value of V_{max} for which WTA paths are quite formed, the spread of MC value due to topology increases considerably (especially for higher number of nanowires), indicating that when a great part of the networks junctions are activated the topology of the network has a considerable effect on MC. However, also in this case, the mean MC value remains constant for all the considered number of nanowires.
Therefore, the number of nanowires does not affect significantly the MC of the networks, which remains almost constant in the range of number of nanowires considered, for all the V_{max}. This is consistent with other studies on nanowire networks [21], and also with studies on software-implemented ESN [69]. However, the dependency on the particular realization of the network seems to be higher for higher number of nanowires, when the applied tension V_{max} is sufficient to create WTA conductance paths. Consequently, from a practical point of view, it seems better to work with networks with the smallest number of nanowires possible. Here in particular, 150 NWs, corresponding to a normalized density of nodes of 8.3, would be the best choice. In fact, a network with a smaller number of nanowires provides almost the same MC of networks with a larger number of nanowires, while it requires less material resources, and it gives a more controllable MC value without the need for adding further components to the system to control the topology of the networks, and so their MC.

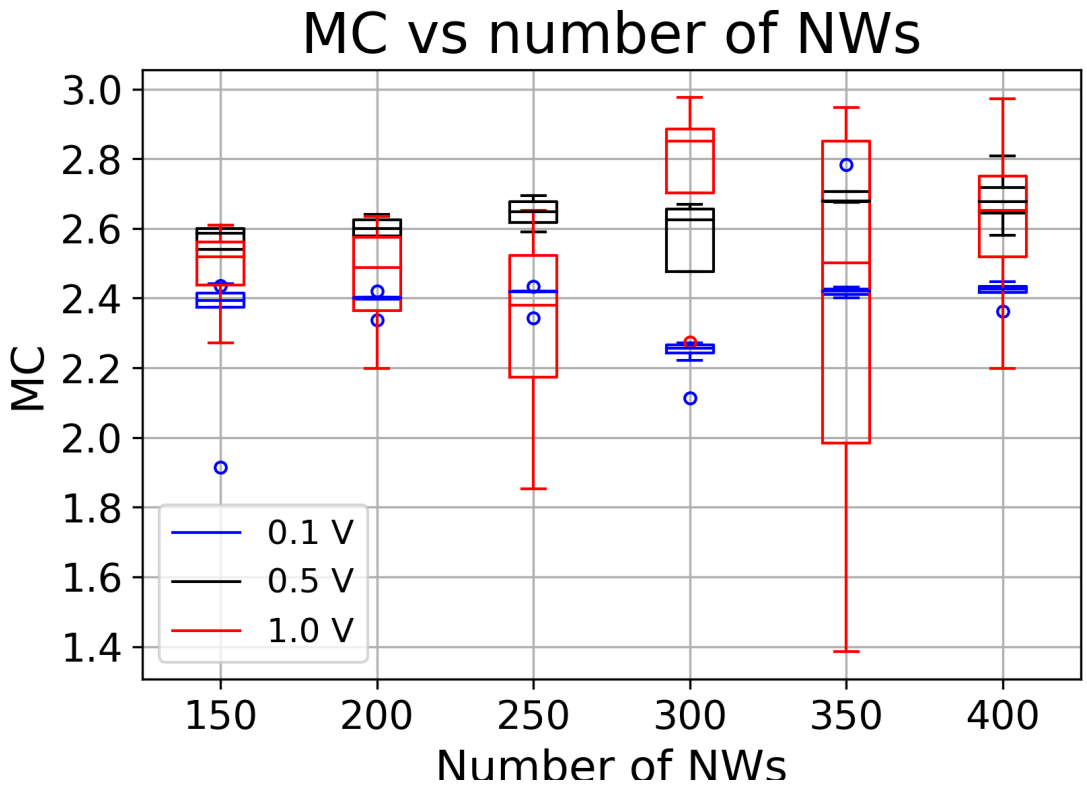


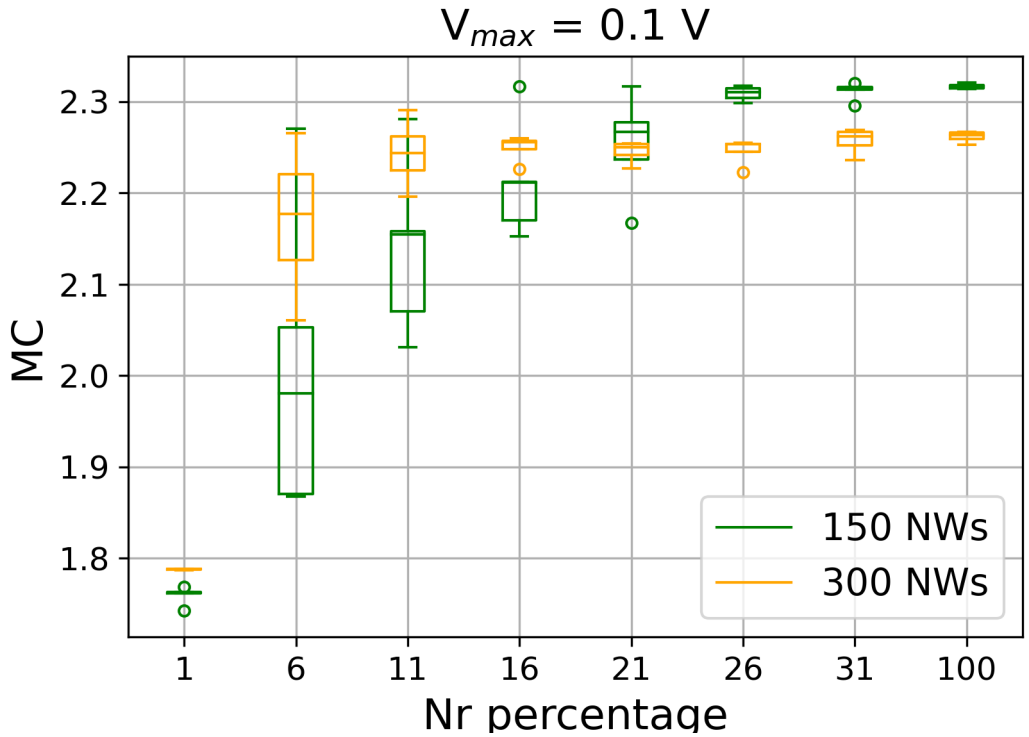
Figure 2.18: MC vs number of nanowires.

2.3.3. Percentage of reading nodes

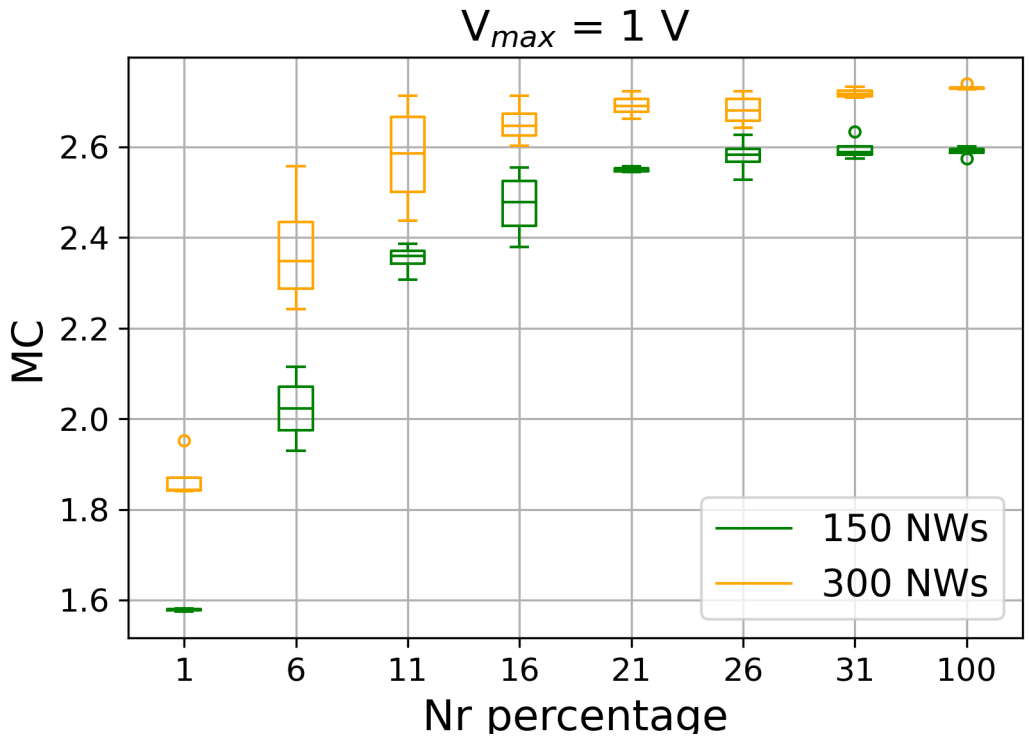
For what concern the dependence of MC on the percentage of reading nodes (Nr), single realizations (topology) of a 150 and a 300 NWs networks are considered. The networks are excited with an input signal $\nu(t) \sim U([0, V_{max}])$, and $V_{max} = 0.1\,V,\ 1.0V$ are considered, in order to study the systems both in a regime where no WTA conductance paths are formed, and in a regime in which WTA paths exist. The simulations are performed for Nr in a range from 1% to 31%, and for $Nr = 100\%$, for which in theory maximum values of MC should be obtained [21]. For each value of Nr, ten different reading nodes configurations have been simulated, and the results are reported in the box plots of figure 2.19.

The results of the simulations show that, in all the considered simulation conditions, MC increases for increasing Nr up to a certain value and then it saturates. The value at which MC saturates corresponds to the value of MC obtained for $Nr = 100\%$, confirming this value to be the maximum value of MC when the number of reading nodes is varied. In figure 2.19, it is also possible to see that 150 and 300 NWs networks reach MC saturation values for different values of Nr. In particular, this happens at $Nr = 21$ for the 150 NWs

network, and at $Nr = 11$ for the 300 NWs network. These two values both corresponds to about 30 reading nodes. Then, from a practical point of view, it should be sufficient to use about 30 nodes of the network as reading nodes in order to obtain MC values that are very close that the theoretical maximum values of MC at $Nr = 100\%$. However, considering more reading nodes could be useful to decrease the variability of MC due to the reading nodes particular configuration (positioning). In fact, it is possible to observe that there is a spread of MC values due to the different positions of reading nodes, for a certain value of Nr. This spread is high at low Nr values and it decreases for increasing Nr. Once Nr is sufficient to have the maximum MC value, increasing Nr leads to a decrease of the variability of MC due to particular reading node positioning.
Finally, it must be noted that increasing Nr corresponds to an increase of the dimensionality of the network output. This means that increasing Nr enriches the reservoir state that is red by the readout module. If this enrichment is performed in an effective way, i.e. if the dimensionality of the network output is effectively increased, then the training and the prediction capability of the readout module are improved. This explain also because there is variability of MC due to the particular reading node positioning. In fact, if the reading nodes are positioned in the network in such a way their activation state (voltage) is very similar, they do not provide different information to the readout module, and it is like there are less reading nodes (the dimensionality of the output is not effectively increased). To have an increase in MC, the reading nodes have to be placed in such a way they provide an effective increase in the dimensionality of the output. This is consistent with the principles of reservoir computing, in which the performance of the system are linked to the dimensionality of the reservoir's output and the role of the reservoir is, in fact, that of mapping the input signal in an higher-dimensional reservoir state [18, 63].
Based on this idea that an higher dimensionality of the network output leads to better performance in MC task, two solutions are proposed in the following sections to increase the reservoir state's effective dimensionality. In fact, if increasing Nr is useful to read an higher dimensional reservoir state, it does not incraese the effective dimensionality of the reservoir state itself, which is determined by physical characteristics of the reservoir and by the type of the applied input signal.



(a)



(b)

Figure 2.19: MC vs Nr. **Top:** $V_{max} = 0.1\ V$. **Bottom:** $V_{max} = 1.0\ V$.

2.3.4. Signal bias

In this section it is proposed a solution to increase the effective dimensionality of the reservoir state by adding a constant bias signal to the so far used input signal $\nu(t) \sim U([0, V_{max}])$, obtaining in this way an input signal of the type $\nu(t) \sim U([B, V_{max} + B])$, where B is the value in V of the bias.
The reasons for applying a bias signal to increase MC comes from the observation that a signal of the type $\nu(t) \sim U([0, V_{max}])$ produces a reservoir state $\mathbf{X}(t)$ which has a low effective dimensionality when $nu(t)$ is close to 0 V (figure 2.20), and from the results reported in [72], in which input signals with larger random values lead to higher MC. Further, a bias signal should lead to a greater activation of the network, which has been shown to be related to an increase of MC in section 2.3.1.

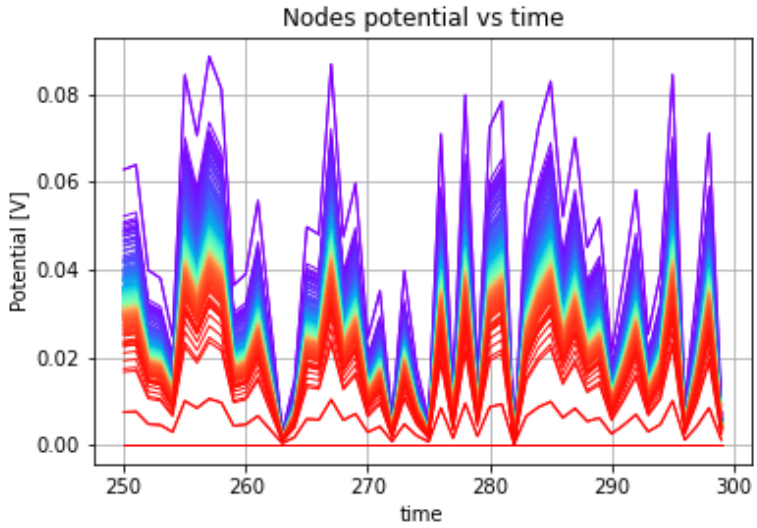


Figure 2.20: Reservoir state with no bias applied.

Figure 2.21 shows the results of simulations in which a signal $\nu(t) \sim U([B, V_{max} + B])$ is applied to a 150 NWs network and to a 300 NWs network, for $V_{max} = 0.1\,V,\ 1.0\,V$, and for different values of B. In figure, the horizontal lines represent the reference values of MC obtained without the application of a bias signal.
It can be observed that the application of a bias increases considerably the performance of the networks in the MC task when B is similar or greater than V_{max}. This is true both for the 150 NWs network and for the 300 NWs network. In the case of 150 NWs network, the application of a bias $B = 2\,V$ increases MC of about three times both at $V_{max} = 0.1\,V$ and at $V_{max} = 1.0\,V$.
The reason for this increase can be found precisely in the increase of the effective dimen-

sionality of the reservoir state and in a greater activation of the network with respect to the case of no bias signal applied. Figure 2.22 shows the difference in the network states when a bias is applied or not, both for 150 NWs and for 300 NWs network. It can be noticed that when a bias is applied, the network state remains effectively high-dimensional at all time steps. On the other hand, when no bias is applied, the network states is not effectively high-dimensional at all time steps. This happens because when no bias is applied, very few (or zero) voltage drops on the nanowire network when the input signal $\nu(t)$ is very close to zero (or zero), resulting in moments in which the voltage is almost the same (or the same and equal to zero) for every node and so the network state $\mathbf{X}(t)$ is not effectively high-dimensional. This is not the case when a bias signal is applied, since on the nanowire network there is always a voltage drop when a bias is applied.
Finally, figure 2.23 shows that the network activation is greater when a bias signal is applied. In particular, even if the bias is small (0.1 V), the emergence of a WTA path can already be observed both in 150 NWs and in 300 NWs networks, even at $V_{max} = 0.1\,V$.

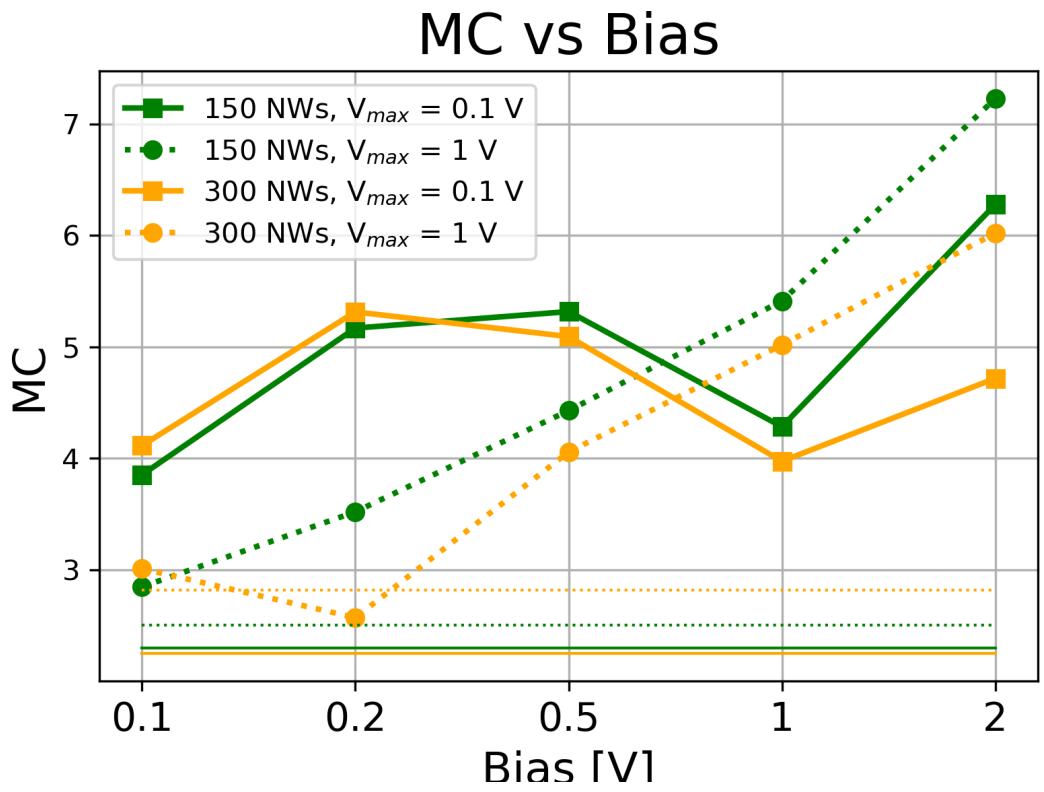


Figure 2.21: MC vs Bias.

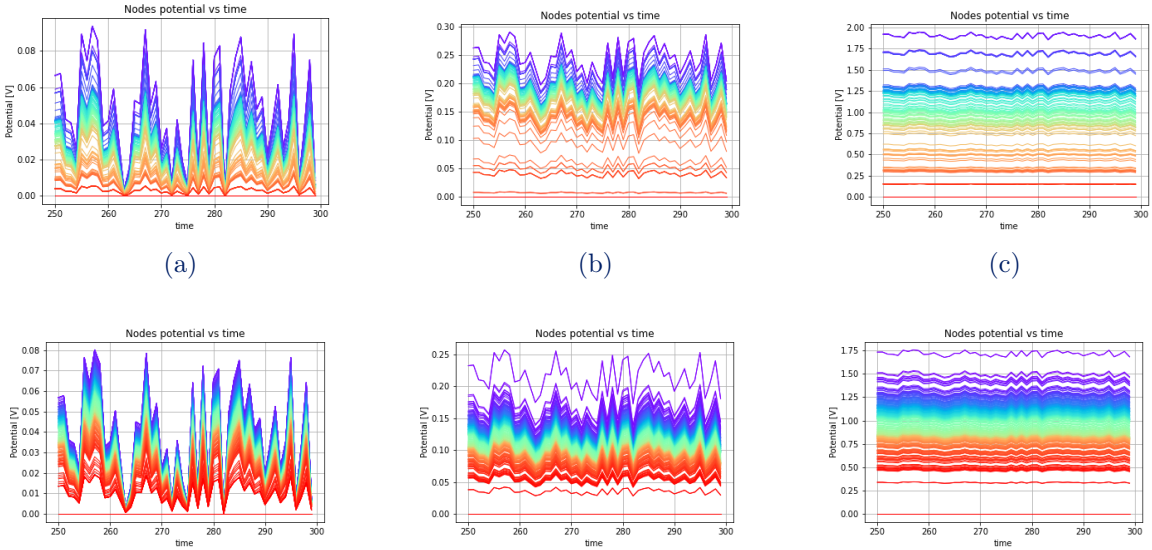


Figure 2.22: Network states in time with diffent bias applied: from left to right B=0, B=0.2V, B=2V. Top row: 150 NWs network. Bottom row: 300 NWs network.

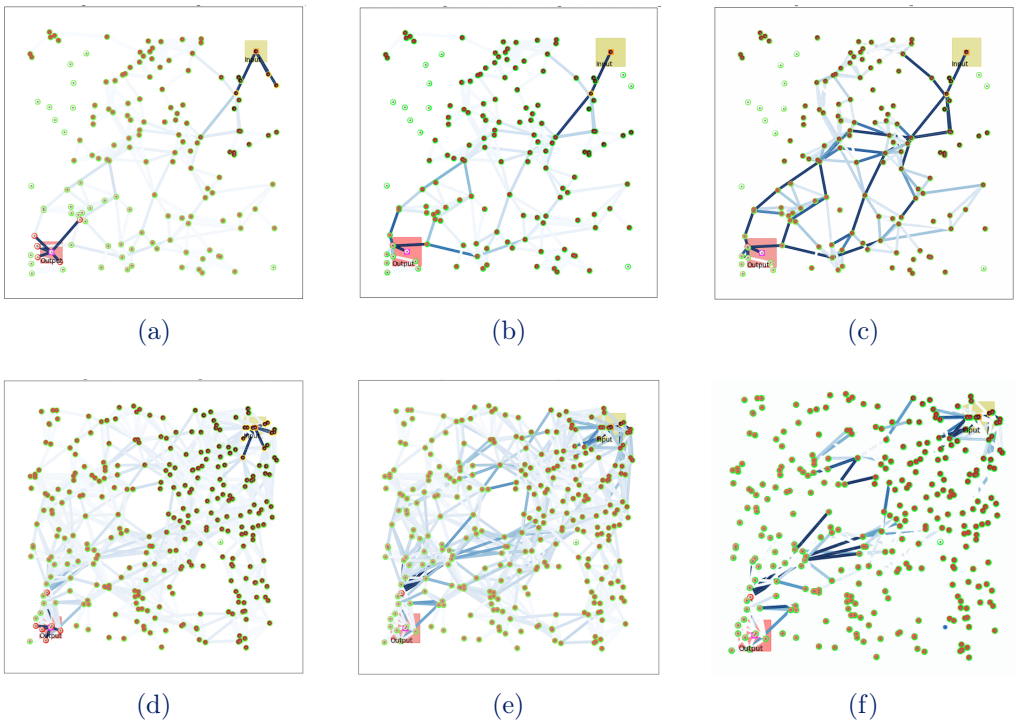
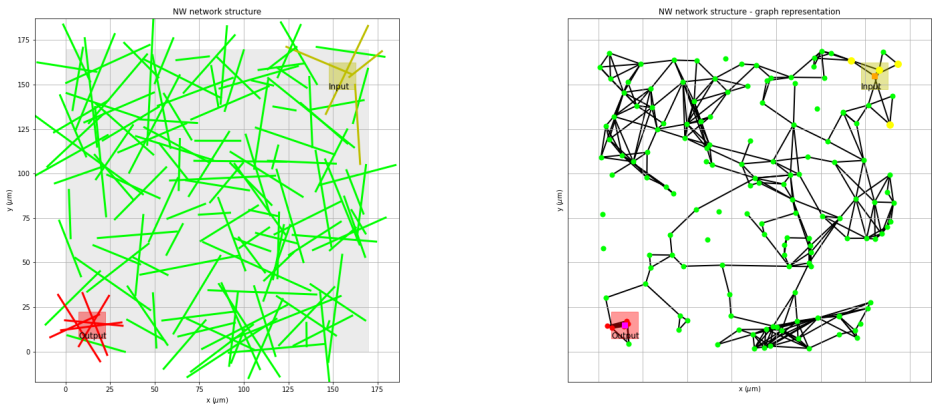


Figure 2.23: Conductance maps of 3000 NWs network with $V_{max} = 0.1\ V$ and increasing bias applied going from (a) to (f).

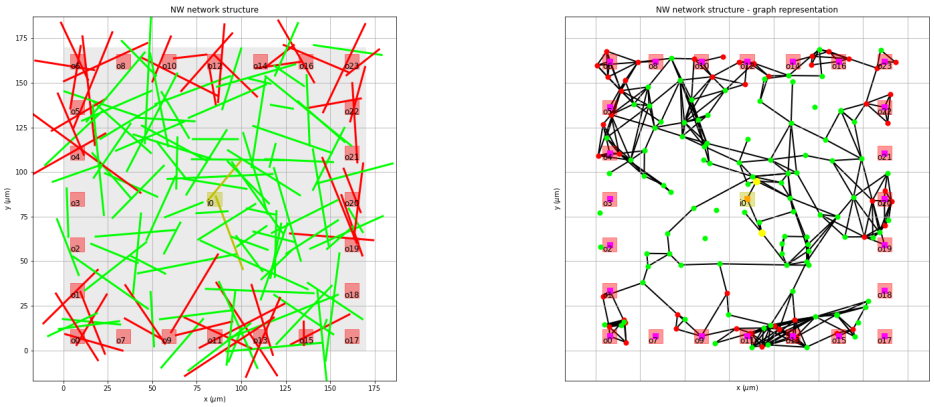
2.3.5. I/O Electrodes configuration

The second proposed solution to increase the effective dimensionality of the reservoir state deals with the Input/Output (I/O) electrodes configuration. Different electrodes configurations should affect the effective dimensionality of the reservoir state by affecting the voltage drop distribution across the nanowire network and the formation of conductance paths. The simulated I/O electrodes configurations are shown in figures 2.24, 2.25, 2.26, 2.27, and for each configuration, five networks realizations with 150 NWs and five networks realizations with 300 NWs are considered. Further, all the networks have been simulated in presence or in absence of a bias signal, and the considered values of V_{max} are 0.1 V and 1. The input signal is again of the type $\nu(t) \sim U([B, V_{max} + B])$.
The MC values obtained in the simulations are reported in figure 2.28, which shows a comparison between the MC task performance of network with different I/O electrodes configurations. In particular, the configuration "diagRef" is that used so far in all the previous simulation of this work, and it serves as reference electrode configuration in order to measure the possible advantage of using a different electrodes configuration. The boxes in figure represent the variability of MC due to different topology of the networks.
Both in 150 NWs networks and in 300 NWs networks, the effect of a different I/O electrodes configuration is greater in presence of a bias signal. In addition, both in presence or in absence of a bias signal, networks with 150 NWs seem to derive a greater benefit in MC performance from electrodes configurations which are different from "diagRef" with respect to 300 NWs networks.



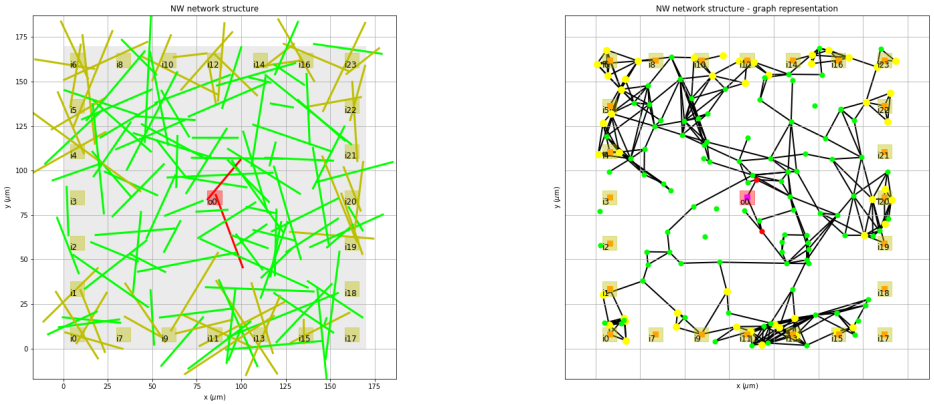
(a) (b)

Figure 2.24: "diagRef" electrodes configuration.



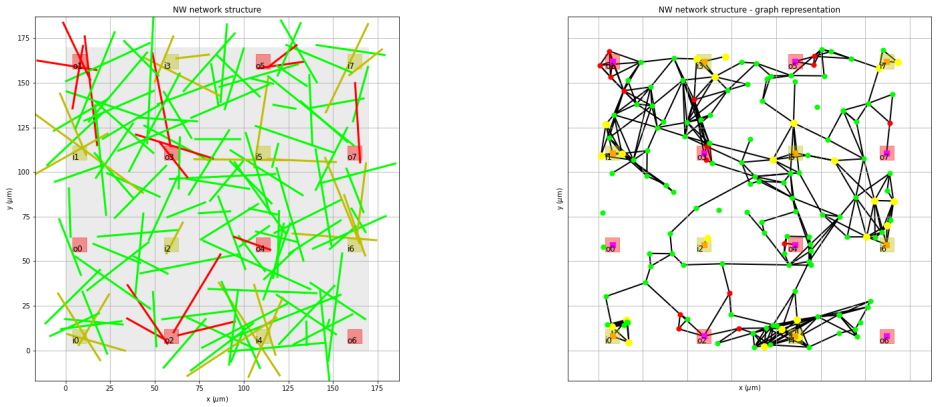
(a) (b)

Figure 2.25: "square" electrodes configuration.



(a) (b)

Figure 2.26: "squareInv" electrodes configuration.



(a) (b)

Figure 2.27: "chessboard" electrodes configuration.

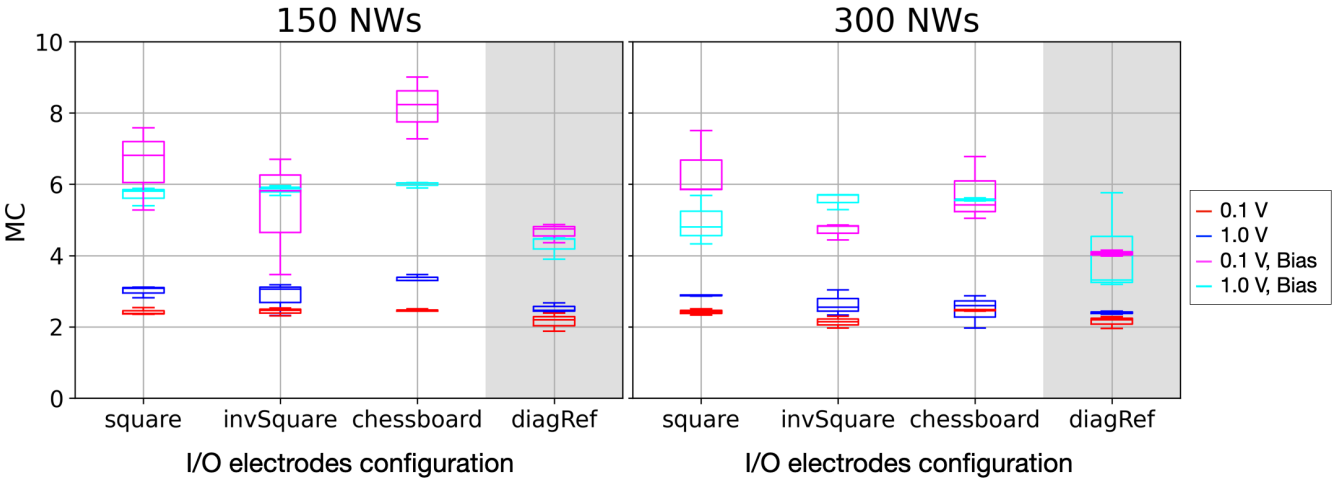
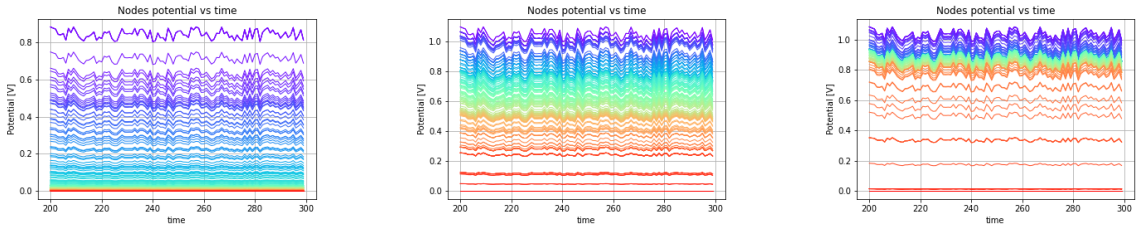


Figure 2.28: MC vs I/O configuration.

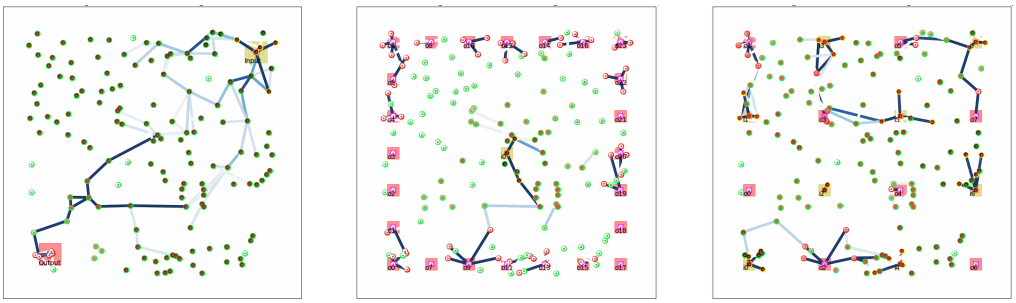
The I/O electrodes configuration that seems to perform better than the others are the "square" and the "chessboard" configurations, both for 150 NWs and for 300 NWs networks. In particular, the best performance are reached by 150 NWs networks with "chessboard" configuration and applied bias signal. The "chessboard" configuration and the bias signal allows to reach a value of MC that exceeds 9, and so it is more than four times larger that the value obtained with "diagRef" configuration and no bias applied, for the same network topology and the same $V_{max} = 0.1\,V$. The good performance of this configuration have to be searched in the effective dimensionality of the reservoir state and in the activation of the networks junction. The latter can be seen in figure 2.30, and

the former in figure 2.29, which show that the "chessboard" configuration produces the network state in which the voltage of the readout nodes are more uniformly spread across the voltage interval $[0, V_{max} + B]$.
In figure **??**, are also reported the forgetting curves obtained for "square" (blue curves) and the "chessboard" (red curves) configurations, compared with that obtained with "diagRef" configuration (black curve). It is possible to see that networks with "square" and "chessboard" configurations have a longer fading memory with respect to that with "diagRef" configuration.



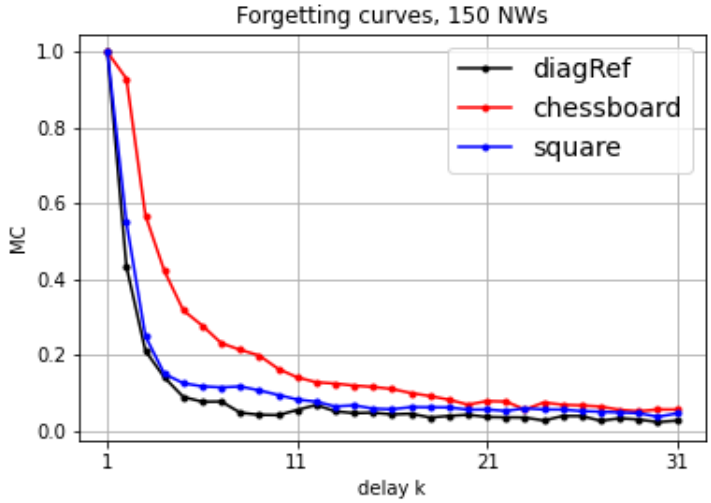
(a) (b) (c)

Figure 2.29: Network state corresponding to (a):"chessboard", (b):"square", (c): "diagRef".

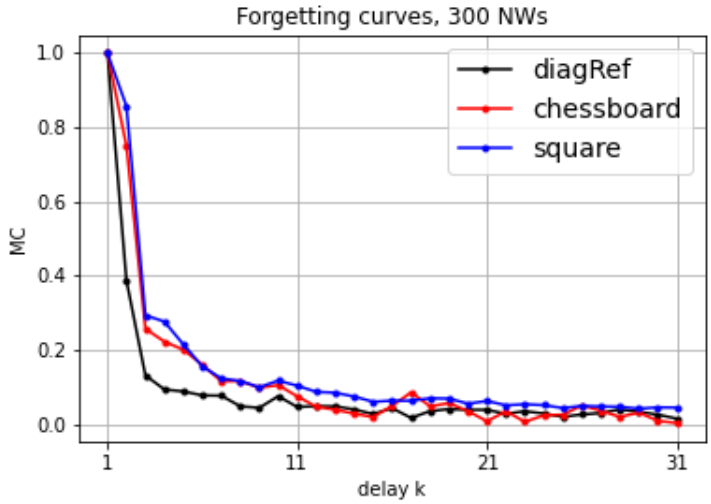


(a) (b) (c)

Figure 2.30: Conductance maps corresponding to (a):"chessboard", (b):"square", (c): "diagRef".



(a)



(b)

Figure 2.31: Forgetting curves for different I/O electrodes configurations. At $V_{max} = 0.1\ V$ and $B = 1\ V$.

3 | Discussion

The main results achieved in this work are the definition of MC measurement conditions that allow to obtain reliable and significant values of MC from simulations, the dependence of MC on the amplitude of the applied electrical signal, on the number of nanowires in the network, and on the number of readout nodes, and finally, two solutions to increase MC of self-organizing nanowire networks.

For what concern the definition of MC measurement conditions, that is a topic to which very little attention is devoted in the literature, the major findings in this thesis is that MC has a non negligible dependency on the the signals used to compute it. In particular MC depends on the length of the training and test signals and on the particular training and test signals. These dependencies have been found from the learning curves of the system which show that the variability of MC due to the particular training and test signals decreases for increasing signal length, and that in order to obtain a value of MC which is truly indicative of the memory properties of the nanowire networks and not influenced by the training of the readout module, signals longer than a certain minimum value must be used. This minimum value is in general dependent on the particular system, and it can be found by plotting its learning curve as a function of the signal length.

After having found suitable conditions for measuring MC of the networks, the dependencies of MC on the amplitude of the applied electrical signal and on the number of nanowires in the network have been studied. The results showed that increasing the amplitude of the applied input signal produces an increasing of MC which is accompanied by the formation of WTA conductance path in the network. The increase of MC with the signal amplitude is consistent with the results reported in [21, 22], in which similar systems are studied but a different model for the memristive junctions is used. Similarities with [21] are found also in the dependence of MC on the number of nanowires in the network, which is found to be a parameter that has almost no effect on MC. However, with respect to [21, 22], some differences in how MC increases with increasing signal amplitude have been found in this work. [22] reports a sudden increase in MC task performance once a WTA path is formed which is not found here, probably because the explored range of signal amplitudes considered here is limited to 4 V, while in [22] the networks are stimulated with input

signal with amplitude up to 10 V. The limitation to 4 V considered here is imposed by the too long minimal signal length that would have been necessary to be used in order to get significant values of MC, and by the consequent too long computation times and too high memory requirements.On the other hand, in [21] is reported a saturation of MC at around 2 V which is not observed here. This is probably due to not optimal MC measurement conditions in [21].

Concerning the number of readout nodes, simulations showed that MC increases with increasing number of readout nodes up to a certain upper bound given by the value of MC obtained by reading the network state from all the wires in the network. This is consistent with what was found in [21], where however a different electrode configuration was used. In this work, the upper bound of MC is reached for about 30 readout nodes both in 150 NWs and in 300 NWs networks. This result has important practical implications, because it means that it would be sufficient to measure the voltage by means of electrodes only from a limited number of wires in the network to obtain the same performance that would be obtained by measuring the voltage from all the wires in the network, which would be practically impossible. Further, a dependence on the particular positioning on readout nodes is found, but it becomes negligible when the number of readout nodes exceed the value of which the upper bound of MC is achieved.

The last results reported in this thesis are two solutions oriented to increase the performance of the nanowire networks in the MC task. Both the solutions are based on the idea of increasing the effective dimensionality of the reservoir state. The first solution is the application of a bias signal superimposed on the random signal used in the MC task. This solution alone is showed to be able to increase MC of the networks up to three times with respect to the case in which no bias is applied. The second solution is changing I/O electrodes configuration. In particular, the configurations named "chessboard" and "square" (figures 2.27 and 2.25) are proved to give better performance in MC task with respect to the "diagRef" (figure 2.24) configuration of reference. Further, the combination of "chessboard" configuration with the application of a bias is proved to increase more than four times the MC of a 150 NWs network.

Finally, the software developed in this work can be easily readjusted to carry out simulation of neuromorphic self-organizing nanowire networks with custom electrodes configuration performing other tasks different from the MC one. This could be extremely useful to assess the impact of particular positioning of input, output and reading electrodes on the performance of the networks in real application tasks such as image recognition, chaotic time series forecasting, and speech recognition.

Bibliography

[1] Kaushik Roy, Akhilesh Jaiswal, and Priyadarshini Panda. Towards spike-based machine intelligence with neuromorphic computing. *Nature*, 575(7784):607–617, 2019.

[2] Abu Sebastian, Manuel Le Gallo, Riduan Khaddam-Aljameh, and Evangelos Eleftheriou. Memory devices and applications for in-memory computing. *Nature Nanotechnology*, 15(7):529–544, 2020.

[3] Catherine D. Schuman, Shruti R. Kulkarni, Maryam Parsa, J. Parker Mitchell, Prasanna Date, and Bill Kay. Opportunities for neuromorphic computing algorithms and applications. *Nature Computational Science*, 2(1):10–19, 2022.

[4] Jiadi Zhu, Teng Zhang, Yuchao Yang, and Ru Huang. A comprehensive review on emerging artificial neuromorphic devices. *Applied Physics Reviews*, 7(1):011312, 2020.

[5] Wenqiang Zhang, Bin Gao, Jianshi Tang, Peng Yao, Shimeng Yu, Meng-Fan Chang, Hoi-Jun Yoo, He Qian, and Huaqiang Wu. Neuro-inspired computing chips. *Nature Electronics*, 3(7):371–382, 2020.

[6] Giacomo Indiveri, Bernabe Linares-Barranco, Tara J. Hamilton, and et al. Neuromorphic silicon neuron circuits. *Frontiers in Neuroscience*, 5, 2011.

[7] Thomas Pfeil, Andreas Grübl, Sebastian Jeltsch, and et al. Six networks on a universal neuromorphic computing substrate. *Frontiers in Neuroscience*, 7, 2013.

[8] Paul A. Merolla, John V. Arthur, Rodrigo Alvarez-Icaza, and et al. A million spiking-neuron integrated circuit with a scalable communication network and interface. *Science*, 345(6197):668–673, 2014.

[9] Steve Furber. Large-scale neuromorphic computing systems. *Journal of Neural Engineering*, 13(5):051001, 2016.

[10] Tobias Wunderlich, Andreas F. Kungl, Eric Muller, and et al. Demonstrating advantages of neuromorphic computation: a pilot study. *Frontiers in Neuroscience*, 13, 2019.

[11] Qiangfei Xia and J. Joshua Yang. Memristive crossbar arrays for brain-inspired computing. *Nature Materials*, 18(4):309–323, April 2019.

[12] Vinod K Sangwan and Mark C Hersam. Neuromorphic nanoelectronic materials. *Nat Nanotechnol*, 15:7, 2020.

[13] Srimoyee K Bose, Caleb P Lawrence, Zheng Liu, and et al. Evolution of a designless nanoparticle network into reconfigurable boolean logic. *Nat Nanotechnol*, 10:12, 2015.

[14] Tianran Chen, John Van Gelder, Bartholomeus Van De Ven, and et al. Classification with a disordered dopant-atom network in silicon. *Nature*, 577:7790, 2020.

[15] Amy M Shen, Chih-Liang Chen, Keunwoo Kim, and et al. Analog neuromorphic module based on carbon nanotube synapses. *ACS Nano*, 7:6117–6122, Jul 2013.

[16] Hiroyuki Tanaka, Megumi Akai-Kasaya, Ahmad Termeh-Yousefi, and et al. A molecular neuromorphic computing device using single-walled carbon nanotube network. *Sci Rep*, 9, 2019.

[17] Zdenka Kuncic and Tomonobu Nakayama. Neuromorphic nanowire networks: principles, progress and future prospects for neuro-inspired information processing. *Advances in Physics: X*, 6(1):1894234, 2021.

[18] Kohei Nakajima. Physical reservoir computing—an introductory perspective. *Japanese Journal of Applied Physics*, 59(6):060501, 2020.

[19] Zdenka Kuncic, Omid Kavehei, Ruomin Zhu, Alon Loeffler, Kaiwei Fu, Joel Hochstetter, Mike Li, James M. Shine, Adrian Diaz-Alvarez, Adam Stieg, James Gimzewski, and Tomonobu Nakayama. Neuromorphic information processing with nanowire networks. In *2020 IEEE International Symposium on Circuits and Systems (ISCAS)*, pages 1–5, 2020.

[20] Gianluca Milano, Giacomo Pedretti, Kevin Montano, Saverio Ricci, Shahin Hashemkhani, Luca Boarinol, Daniele Ielmini, and Carlo Ricciardi. In materia reservoir computing with a fully memristive architecture based on self-organizing nanowire networks. *Nature Materials*, 21:195–202, Oct 2021.

[21] R.K. Daniels, J.B. Mallinson, Z.E. Heywood, P.J. Bones, M.D. Arnold, and S.A. Brown. Reservoir computing with 3d nanowire networks. *Neural Networks*, 154:122–130, 2022.

[22] Alon Loeffler et al. Modularity and multitasking in neuro-memristive reservoir networks. *Neuromorphic Computing and Engineering*, 1(1):014003, 2021.

[23] Laura E. Suarez, Jude D. Kendall, and Juan C. Nino. Evaluation of the computational capabilities of a memristive random network (mn3) under the context of reservoir computing. *Neural Networks*, 107:244–255, 2018.

[24] Herbert Jaeger. Short term memory in echo state networks. *Gert Westermann (Ed.) GMD Report 152, ISSN 0930-0305*, 2002.

[25] Laura E. Suárez, Blake A Richards, Guillaume Lajoie, and Bratislav Misic. Learning function from structure in neuromorphic networks. *Nature Machine Intelligence*, 3:771–786, 2021.

[26] Gianluca Milano, Giacomo Pedretti, M. Fretto, Luca Boarino, Fabio Benfenati, D. Ielmini, Ilia Valov, and Carlo Ricciardi. Self-organizing memristive nanowire networks with structural plasticity emulate biological neuronal circuits. *Nature Communications*, 10(1):2393, 2019.

[27] Christopher W Lynn and Danielle S Bassett. The physics of brain network structure, function and control. *Nature Reviews Physics*, 1(5):277–290, 2019.

[28] Karen Zito and Karel Svoboda. Activity-dependent synaptogenesis in the adult mammalian cortex. *Neuron*, 35(6):1015–1017, Sep 2002.

[29] Yuwei Cui, Subutai Ahmad, and Jeff Hawkins. Continuous online sequence learning with an unsupervised neural network model. *Neural Computation*, 28(12):2474–2504, Sep 2016.

[30] Adnan Mehonic, Abu Sebastian, Bipin Rajendran, and et al. Memristors—from in-memory computing, deep learning acceleration, and spiking neural networks to the future of neuromorphic and bio-inspired computing. *Advanced Intelligent Systems*, 2(8):2000085, 2020.

[31] Takahisa Ohno, Tsuyoshi Hasegawa, Tohru Tsuruoka, and et al. Short-term plasticity and long-term potentiation mimicked in single inorganic synapses. *Nature Materials*, 10(8):591–595, 2011.

[32] Teresa Serrano-Gotarredona, Timothée Masquelier, Themistoklis Prodromakis, and et al. Stdp and stdp variations with memristors for spiking neuromorphic learning systems. *Frontiers in Neuroscience*, 7:2, 2013.

[33] Selena La Barbera, Dominique Vuillaume, and Fabien Alibart. Filamentary switching: synaptic plasticity through device volatility. *ACS Nano*, 9(2):941–949, 2015.

[34] Adrian Diaz-Alvarez, Rintaro Higuchi, Paula Sanz-Leon, Ido Marcus, Yoshitaka Shin-

gaya, Adam Stieg, James Gimzewski, Zdenka Kuncic, and Tomonobu Nakayama. Emergent dynamics of neuromorphic nanowire networks. *Scientific Reports*, 9(1):1–13, 2019.

[35] Claudia Gomes da Rocha, Mauro Ferreira, John Boland, Hugh Manning, and Subhajit Biswas. Emergence of winner-takes-all connectivity paths in random nanowire networks. *Nature Communications*, 9(1):1–9, 2018.

[36] Massimiliano Di Ventra and Yuriy V Pershin. On the physical properties of memristive, memcapacitive and meminductive systems. *Nanotechnology*, 24(25):255201, 2013.

[37] Yuriy V Pershin and Massimiliano Di Ventra. Memory effects in complex materials and nanoscale systems. *Advances in Physics*, 60(2):145–227, 2011.

[38] Kazuya Terabe, Tomonobu Nakayama, Tatsuo Hasegawa, and et al. Formation and disappearance of a nanoscale silver cluster realized by solid electrochemical reaction. *Journal of Applied Physics*, 91(10):10110–10114, 2002.

[39] Kazuya Terabe, Tatsuo Hasegawa, Tomonobu Nakayama, and et al. Quantized conductance atomic switch. *Nature*, 433(7021):47–50, 2005.

[40] Karthik Krishnan, Tohru Tsuruoka, Cedric Mannequin, and et al. Mechanism for conducting filament growth in self-assembled polymer thin films for redox-based atomic switches. *Advanced Materials*, 28(3):640–648, 2016.

[41] Gianluca Milano, Giacomo Pedretti, Matteo Fretto, Luca Boarino, Fabio Benfenati, Daniele Ielmini, Ilia Valov, and Carlo Ricciardi. Brain-inspired structural plasticity through reweighting and rewiring in multi-terminal self-organizing memristive nanowire networks. *Advanced Intelligent Systems*, 2(8):2000096, 2020.

[42] Wei Wang, Ming Wang, Elia Ambrosi, Alessandro Bricalli, Mario Laudato, Zhong Sun, Xiaodong Chen, and Daniele Ielmini. Surface diffusion-limited lifetime of silver and copper nanofilaments in resistive switching devices. *Nature Communications*, 10(1):81, 2019.

[43] E. Miranda, G. Milano, and C. Ricciardi. Modeling of short-term synaptic plasticity effects in zno nanowire-based memristors using a potentiation-depression rate balance equation. *IEEE Transactions on Nanotechnology*, 19(4):609–612, 2020.

[44] Z. Wang, S. Joshi, S. E. Savel'ev, H. Jiang, R. Midya, P. Lin, M. Hu, N. Ge, J. P. Strachan, Z. Li, Q. Wu, M. Barnell, G. Li, H. L. Xin, R. S. Williams, Q. Xia, and J. J.

Yang. Memristors with diffusive dynamics as synaptic emulators for neuromorphic computing. *Nature Materials*, 16(1):101–108, 2017.

[45] S. Menzel, S. Tappertzhofen, R. Waser, and I. Valov. Switching kinetics of electrochemical metallization memory cells. *Physical Chemistry Chemical Physics*, 15(18):6945–6952, 2013.

[46] A. Rodriguez-Fernandez, C. Cagli, J. Sune, and E. Miranda. Switching voltage and time statistics of filamentary conductive paths in hfo2-based reram devices. *IEEE Electron Device Letters*, 39(5):656–659, 2018.

[47] Kevin Montano, Gianluca Milano, and Carlo Ricciardi. Grid-graph modeling of emergent neuromorphic dynamics and heterosynaptic plasticity in memristive nanonetworks. *Neuromorphic Computing and Engineering*, 2(1):014007, 2022.

[48] Ruomin Zhu, Joel Hochstetter, Alon Loeffler, Adrian Diaz-Alvarez, Adam Stieg, James Gimzewski, Tomonobu Nakayama, and Zdenka Kuncic. Harnessing adaptive dynamics in neuro-memristive nanowire networks for transfer learning. In *2020 International Conference on Rebooting Computing (ICRC)*, pages 102–106, 2020.

[49] Alexander Loeffler, Nicolò Zagni, Stefano Zordan, Fabien Alibart, and Dmitri B. Strukov. Topological properties of neuromorphic nanowire networks. *Frontiers in Neuroscience*, 14:184, Mar 2020.

[50] Gianluca Milano, Enrique Miranda, and Carlo Ricciardi. Connectome of memristive nanowire networks through graph theory. *Neural Networks*, 150:137–148, 2022.

[51] José Luis Mietta, Ricardo Martín Negri, and Pablo Ignacio Tamborenea. Numerical simulations of stick percolation: Application to the study of structured magnetorheological elastomers. *The Journal of Physical Chemistry C*, 118(35):20594–20604, 2014.

[52] Yuri Yu Tarasevich and Andrei V. Eserkepov. Percolation of sticks: Effect of stick alignment and length dispersity. *Physical Review E*, 98(6):1–6, 2018.

[53] C. W. Ho, A. Ruehli, and P. Brennan. The modified nodal approach to network analysis. *IEEE Transactions on Circuits and Systems*, 22(6):504–509, 1975.

[54] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning internal representations by error propagation. Technical report, California Univ. San Diego La Jolla Inst. for Cognitive Science, 1985.

[55] Paul J Werbos. Backpropagation through time: what it does and how to do it. *Proceedings of the IEEE*, 78(10):1550–1560, 1990.

[56] Ronald J Williams and David Zipser. A learning algorithm for continually running fully recurrent neural networks. *Neural Computation*, 1(2):270–280, 1989.

[57] Kenji Doya. Recurrent networks: supervised learning. In *The handbook of brain theory and neural networks*, pages 796–800. MIT Press, 1998.

[58] Herbert Jaeger. The "echo state" approach to analysing and training recurrent neural networks. Technical Report GMD Report 148, German National Research Center for Information Technology, 2001.

[59] Wolfgang Maass, Thomas Natschläger, and Henry Markram. Real-time computing without stable states: a new framework for neural computation based on perturbations. *Neural Computation*, 14(11):2531–2560, 2002.

[60] Benjamin Schrauwen, David Verstraeten, and Jan Campenhout. An overview of reservoir computing: Theory, applications and implementations. In *Proceedings of the 15th European Symposium on Artificial Neural Networks*, pages 471–482, 2007.

[61] L. Appeltant, M.C. Soriano, G. Van der Sande, J. Danckaert, S. Massar, J. Dambre, B. Schrauwen, C.R. Mirasso, and I. Fischer. Information processing using a single dynamical node as complex system. *Nature Communications*, 2(1):468, 2011.

[62] Helmut Hauser, Auke Jan Ijspeert, Rudolf M Füchslin, Rolf Pfeifer, and Wolfgang Maass. Towards a theoretical foundation for morphological computation with compliant bodies. *Biological Cybernetics*, 105(5-6):355–370, 2011.

[63] Lennert Appeltant, Miguel C Soriano, Guy Van der Sande, Jan Danckaert, Serge Massar, Joni Dambre, Benjamin Schrauwen, Claudio R Mirasso, and Ingo Fischer. Information processing using a single dynamical node as complex system. *Nature Communications*, 2:468, 2011.

[64] Kristof Vandoorne, Wouter Dierckx, Benjamin Schrauwen, David Verstraeten, Roel Baets, Peter Bienstman, and Jan Van Campenhout. Toward optical signal processing using photonic reservoir computing. *Optics express*, 16(14):11182–11192, 2008.

[65] Jacobo Torrejon, Mathieu Riou, Flavio Abreu Araujo, Sumito Tsunegi, Guru Khalsa, Damien Querlioz, Pierre Bortolotti, Vincent Cros, Kay Yakushiji, Akio Fukushima, et al. Neuromorphic computing with nanoscale spintronic oscillators. *Nature*, 547(7664):428–431, 2017.

[66] Dean V Buonomano and Wolfgang Maass. State-dependent computations: spatiotemporal processing in cortical networks. *Nature Reviews Neuroscience*, 10(2):113–125, 2009.

[67] Rui Zhu, Alon Loeffler, Joel Hochstetter, and et al. Mnist classification using neuromorphic nanowire networks. In *International Conference on Neuromorphic Systems*, 2021.

[68] Nathaniel Rodriguez, Eduardo Izquierdo, and Yong-Yeol Ahn. Optimal modularity and memory capacity of neural reservoirs. *Network Neuroscience*, 3(2):551–566, 2019.

[69] Xinyu Han, Yi Zhao, and Michael Small. Revisiting the memory capacity in reservoir computing of directed acyclic network. *Chaos*, 31(3):033106, 2021.

[70] Igor Farkaš, Radomír Bosák, and Peter Gergeľ. Computational analysis of memory capacity in echo state networks. *Neural Networks*, 83:109–120, 2016.

[71] Dietrich Stauffer and Ammon Aharony. *Introduction To Percolation Theory: Second Edition*. Taylor & Francis, 2 edition, 1992.

[72] Peter Barančok and Igor Farkaš. Memory capacity of input-driven echo state networks at the edge of chaos. In Stefan Wermter, Cornelius Weber, Włodzisław Duch, Timo Honkela, Petia Koprinkova-Hristova, Sven Magg, Günther Palm, and Alessandro E. P. Villa, editors, *Artificial Neural Networks and Machine Learning – ICANN 2014*, pages 41–48, Cham, 2014. Springer International Publishing.

List of Figures

1.1 Schematic outline of the neuromorphic systems landscape. 3
1.2 Human brain vs memristive nanowire network. 3
1.3 Ag-PVP-Ag junction. 4
1.4 WTA SEM image. 5
1.5 Ag-PVP NWNs with different AMD. 6
1.6 Gold pads SEM image. 6
1.7 MIM junction in Ag-PVP nanowires intersections. 8
1.8 Graphs of NWNs with different densities. 11
1.9 MVNA computation time vs number of nanowires. 13
1.10 Electrical scheme for nanowire network stimulation. 14
1.11 Algorithm for modelling the memristive emergent dynamics of the network. 15
1.12 PRC schematic. 16
1.13 PRC with Ag-PVP NWNs. 18
1.14 Crossbar array memristive readout module. 19

2.1 Schematic of an example of simulated nanowire system 22
2.2 Schematic of applied input signal. 25
2.3 Schematic of system before training. 27
2.4 Schematic of the first part of training phase. 27
2.5 Schematic of the second part of training phase. 28
2.6 [Schematic of the test phase. 29
2.7 Example of forgetting curve. 30
2.8 Predictions vs targets. 31
2.9 Transient at different signal amplitude. 33
2.10 Learning curve, 300 NWs, 4V. 35
2.11 Schematic of an example of simulated nanowire system 37
2.12 Schematic of an example of simulated nanowire system 38
2.13 Schematic of an example of simulated nanowire system 39
2.14 Dependence on Test signal. 40
2.15 MC vs signal amplitude. 42

2.16 WTA in 150 NWs networks. 43
2.17 WTA in 300 NWs networks 44
2.18 MC vs number of nanowires. 46
2.19 MC vs Nr. 48
2.20 Reservoir state with no bias. 49
2.21 MC vs Bias. 50
2.22 Network states in time with bias 51
2.23 WTA and Bias 51
2.24 "diagRef" electrodes configuration. 52
2.25 "square" electrodes configuration. 53
2.26 "squareInv" electrodes configuration. 53
2.27 "chessboard" electrodes configuration. 54
2.28 MC vs I/O configuration 54
2.29 Network state for different configuration. 55
2.30 Conductance maps for different configuration. 55
2.31 Forgetting curves for different I/O electrodes configurations. 56

Acknowledgements

A special thank goes to Prof. Carlo Ricciardi, for the trust in my work and for having introduced me to the fascinating subject of neuromorphic computing.

I am also strongly grateful to my co-supervisor Dr. Gianluca Milano, who assisted me throughout this thesis providing me useful advices. Working with you in the last months was a great occasion.