TOWARDS THE CONSTRUCTION OF A CHEMICAL COMPUTER

BY

AMAHURY J. L. DIAZ

BS. in Physics, Universidad Nacional Autónoma de México, 2023

THESIS

Submitted in partial fulfillment of the requirements for
the degree of Master of Science in Systems Science
in the Graduate School of
Binghamton University
State University of New York
2024

Accepted in partial fulfillment of the requirements for
the degree of Master of Science in Systems Science
in the Graduate School of
Binghamton University
State University of New York
2024

November 25th, 2024

Carlos Gershenson-Garcia, Faculty Advisor
School of Systems Science and Industrial Engineering, Binghamton University

Hiroki Sayama, Member
School of Systems Science and Industrial Engineering, Binghamton University

Luis M. Rocha, Member
School of Systems Science and Industrial Engineering, Binghamton University

Abstract

Recently, it has been shown that there is a parallelism between certain chemical reactions and different types of automata. In particular, it was shown that Belousov-Zhabotinsky chemical oscillators are capable of universal computation (at the bounded linear automaton level), allowing the elucidation of the first non-hybrid programmable chemical computer. However, so far no one has studied the computational capability of two or more coupled chemical automata. By transforming the problem of accepting/rejecting inputs by a chemical Turing machine into a minimization problem, it is possible to use time series forecasting as a benchmark to quantify the difficulty for a configuration of chemical oscillators to accept such a sequence. This thesis shows that, in both the deterministic and stochastic cases, configurations composed of two chemical automata are more efficient than a single chemical automaton in forecasting complex time series, which implies that at least two chemical automata are necessary to accept complex computable languages. The above gives way to show that a chain of n strongly coupled chemical automata are more efficient at processing complex computable languages than a chain of $n-1$ strongly coupled chemical automata. Given the above, different possibilities are explored to exploit this topology (among

other more generalized ones) for the transfer and manipulation of information, which allows the elucidation of a non-hybrid and programmable chemical computer. The impacts of this research on problems such as the search for cellular supremacy in biocomputing and the exploration of algorithmic networks in the resource-bounded case are also inspected.

Dedication

To my family, who have always supported me to the best of their ability.
To Hao Duxiao (郝读晓), who on countless occasions offered me unconditional emotional support since I came to study in New York, thus becoming my confidant. Thanks to you my perspective on love has changed and additionally I was able to find an enormous development in myself, in my life. 我爱你.

Acknowledgements

I am grateful to Prof. Juan Perez-Mercader, senior research fellow at Harvard University and external faculty at the Santa Fe Institute, for his valuable comments during the elaboration of this thesis. Of course none of this would have been possible without the valuable guidance of Prof. Carlos Gershenson, who has constantly supported me academically and whom I consider a friend more than an advisor.

List of Figures

Contents

Chapter 1

Introduction

1.1 The importance of unconventional computing

Since the last century, economic globalization and technological growth have brought with it a severe exploitation of natural resources, directly modifying the ecological dynamics, leading to the consequences of what we now perceive as climate change [Crowley, 2000]. Nowadays Artificial intelligence (AI) uses a significant amount of energy, and its consumption is expected to increase in the coming years [Leffer, 2023]. If in the next century we wish to maintain a relationship with technology similar to the one we observe today, it is necessary to find a sustainable computing alternative that will allow us to do so.

As has been mentioned in [Kaack et al., 2022], the developing of more green artificial intelligence (AI) technologies is fundamental in order to fight climate change. Unconventional computing architectures are outstanding candidates to accomplish this task, since they can often operate on orders of magnitude less power than traditional computing systems [Lu et al., 2024]. Nowadays a plethora of unconventional hardware has been explored [Ziegler, 2020]. However, for the purpose of implementing such non-von Neumann structures, we first need to flourish an information-processing paradigm which allows us create algorithms compatible with such decentralized computational organizations.

Despite the diverse variety of substrates that can be used in unconventional computation, today it is still difficult to elucidate a programmable unconventional computing structure. To understand this in more detail, think about your favorite programming tool right now. No matter what language you are using, it is almost trivial to write a function that outputs the first ten positive integers. To achieve such functionality using a unconventional structure, it is necessary to manipulate the flow of information in it. This is not trivial, and so far our efforts have led to digitally hybridized structures [Kim et al., 2016], inefficient algorithms [Schuman et al., 2017] and complicated implementations [Xu et al., 2023].

This leaves us with an interesting question: what is the most suitable substrate for the purpose of manipulating information in the components of unconventional architectures?

1.2 A brief introduction to chemical computation

Chemical Computation (or simply "Chemputation") is one of many approaches to unconventional computing. It is based on the idea of using a semi-solid chemical soup as a reactor to process and store information. This concept should not be confused with *computational chemistry*, which is a branch of chemistry that uses computer simulations to help solve chemical problems. The history and state of the art in chemical computation are not that intricate, so a brief summary is given below.

It all began in 1989, when Kuhnert and colleagues demonstrated that it is possible to use chemical waves to process images [Kuhnert et al., 1989]. A little over a decade later, Andrew Adamatzky (one of the leading exponents in the field) demonstrated that it is possible to implement simple logic gates using reaction-diffusion processes [Adamatzky and Costello, 2002]. One of the major limitations of the Adamatzky approach is the dependence on wave speed. Since reaction-diffusion processes are usually slow, this makes it impossible to compute efficiently.

It was not until 2014 that an international team led by the Swiss Federal Laboratories for Materials Science and Technology materialized a chemical computer based on surface tension calculations derived from the Marangoni effect using an acidic gel to find the most efficient route between two points [Suzuno et al., 2014]. The next year a Stanford team created a computer based on magnetic fields and water droplets infused with magnetic nanoparticles [Katsikis et al., 2015]. Both proposals are designed to solve specific problems, and given their architecture, are difficult to implement on a

large scale.

In 2019 a team from Harvard demonstrated that the Belousov–Zhabotinsky reaction is Turing complete in the linear bounded automata subclass [Dueñas-Díez and Pérez-Mercader, 2019]. Such a system is capable of recognizing Chomsky type-1, type-2 and type-3 languages using Gibbs free energy considerations. A year later a group at the University of Glasgow claimed to have developed the first programmable chemical computer capable of using memory and pattern recognition [Parrilla-Gutierrez et al., 2020]. However, this architecture turned out to be a hybrid, with the digital part being fundamental to its operation.

We can then look back to the work of [Dueñas-Díez and Pérez-Mercader, 2019] as the major attempt to build a non-hybrid programmable chemical computer so far. One of the major achievements of this research is the characterization of the chemical oscillators as automata. Since they can be described in these terms, BZ oscillators are therefore a good prospect as components of a programmable unconventional architecture. Notwithstanding, so far no one has explored the computational capabilities of coupled chemical oscillators, which is necessary to build a network made of these entities.

In [Wolpert, 2001] the definition of computational capability is grounded in the framework of physical systems' ability to perform computations. It considers the interaction between inputs and outputs of systems, with "computational capability" encompassing a system's ability to process and predict the dynamics of the universe.

Thus, computational capability can be defined as the ability of a device to process and analyze data. Note that the study of [Dueñas-Díez and Pérez-Mercader, 2019] is only focusing on the computational capability of a single chemical oscillator, showing that all the accepted words by the chemical Turing machine have the same free-energy difference between the oxidized and reduced states of the catalyst after the processing the input. However, so far no one has studied the computational capability of two chemical oscillators, which could be critical to understand if there is any advantage of connecting such entities or whether a single chemical pot would suffice.

1.3 Two problems with an adjacent solution

Two parallel problems to the paradigm of unconventional computation, whose solution is adjacent to the construction of a computer based on a network of programmable chemical oscillators, will be introduced superficially below. As we will see in the next two subsections, the first problem is related to the experimental aspects of chemical computation, while the second is more concerned with the theoretical implications of computation using networks of finite automata. In this way, we will see that understanding chemical computation goes beyond being a branch of unconventional computing, being as well a way to understand the origin and evolution of information handling by living beings.

1.3.1 Biocomputing

If we broadly define computation as a procedure by which input information is transformed according to pre-defined rules and turned into output data, then we can find computation not only in electronic devices, but also in living entities [De Castro, 2006]. Since the early 1980s we have been studying the computational abilities of biological matter [Bennett, 1982]. However, it was not until 1994 that the possibility of implementing computation using molecular (i.e., genetic) hardware was demonstrated [Adleman, 1994].

Although there have been multiple developments in the world of biocomputing, it seems that we are still quite far from obtaining an adequate formalism that allows us to decipher how life computes [Goñi-Moreno, 2024]. Moreover, to date most advances in biocomputation use almost exclusively genetic material [Amos and Goñi-Moreno, 2018]. As has been well explained in [Goñi-Moreno and Nikel, 2019], the use of metabolism as a means of computation holds strong promise for biocomputing.

The idea is very simple. Real cells contain within them an intricate computational network that mixes genetic (digital) and metabolic (analog) aspects. Understanding the information flows between the two components is fundamental to achieving what some have called "cellular supremacy in biocomputing" [Grozinger et al., 2019]. But before understanding the intricate interactions between genetic and metabolic networks that evolution has shaped over time, it is first necessary to understand the computational aspects of metabolism. After all, almost all biochemical interactions

in the cell handle information in an analog format [Walker and Davies, 2013].

But what is metabolism if not more than a set of chemical reactions with a kind of periodicity? In 1951 Boris Belusov was trying to find the non-organic analogue of the Krebs cycle, which led him to the discovery of an oscillatory reaction [Belousov, 1959] that later was named the Belousov-Zhabotinsky reaction, or BZ reaction. The second name of this reaction corresponds to Anatol Zhabotinsky, who explained the mechanism behind Belusov's reaction in detail [Zhabotinsky, 1964]. Thus, the BZ reaction proves to be a suitable artifact when studying the computational abilities of primordial metabolism in primitive life forms.

As mentioned in the previous section, Belousov-Zhabotinsky chemical oscillators have been shown to be capable of universal computation in the linear-bounded limit. Thus, by studying the computational capacity of two chemical oscillators with different coupling levels, we also gain a deeper understanding of the metabolic facet of computation in the cell, not only at the individual but also at the collective level. So far the methodologies proposed to study metabolic biocomputing have focused on carbon-based life forms [Goñi-Moreno and Nikel, 2019]. However, to truly understand the nature of metabolic computation, it is necessary to understand how biochemistry-free life-like forms were able to manipulate information.

1.3.2 Algoritmic Networks

Algorithmic Networks is a recently developed proposal [Abrahao et al., 2019], which seeks to investigate how topological properties of a network can trigger emergent behavior capable of irreducibly increasing the computational power of the entire network with respect to the network components. Roughly speaking, an algorithmic network is a population of automata whose members can share information with each other according to a given topology. In this way, each node of the network is an automaton and each edge of the network is a communication channel.

Algorithmic networks can be seen as the generalization of Boolean networks [Kauffman, 1969], where each node is a Boolean automaton with an associate set of rules (a truth table) for every possible input. However, algorithmic networks go beyond Boolean. In [Abrahão et al., 2020] it is shown that a slight modification in the communication protocol is sufficient to allow an algorithmic network (with random topology) to synergistically solve problems in a higher computational class than the computational class of its individual nodes. This is quite relevant, because as was pointed out in [Abrahão and Zenil, 2022], we can use algorithmic networks as a model of emergence, in which the whole can solve problems of a higher computational class than its parts.

Despite the above, it is important to note that the canonical model of algorithmic networks considers only Turing machines as nodes. At the same time, although the authors have shown theoretical results in both fixed and adaptive networks, the

topologies used are still random. Thus, in order to test the theoretical toolbox of algorithmic networks in experiments (both *in vivo* and *in silico*), further research is needed to establish how a resource-bounded version of algorithmic networks with non-random topologies performs mathematically.

As mentioned in Section 1.2, the findings of [Dueñas-Díez and Pérez-Mercader, 2019] show that BZ oscillators are equivalent to a resource-bounded version of Turing machines. From this we glimpse an opportunity: studying the computational capacity of two or more chemical automata (coupled or uncoupled) brings with it the chance to deepen our understanding of how resource-bounded algorithmic networks work. Moreover (and as we will see later), the methodology proposed in this thesis also opens the doors to explore non-random topologies and intermediate coupling states, where two automata can be weakly coupled. This in a certain way generalizes the original proposal [Abrahao et al., 2019] and only by making use of our computational outputs can we try to infer theoretical results with the level of abstraction proposed by [Abrahão et al., 2020].

Chapter 2

Methods

2.1 Reservoir Computing

As emphasized in the Introduction, it is necessary to find a computational paradigm that is compatible with neuromorphic architectures. In particular, we seek a computational framework that allows us to study the interaction of two or more chemical automata (represented here as BZ oscillators). From its inception, reservoir computing (RC) has shown to be an excellent machine learning algorithm for processing information generated by dynamical systems using observed time-series data [Jaeger, 2001, Maass et al., 2002].

An RC system consists of a reservoir, which maps inputs into a high-dimensional space;

it also contains a readout, which performs pattern analysis from the high-dimensional states in the reservoir [Tanaka et al., 2019]. In RC only the readout is trained with a simple method, while the rest of its architecture remains completely fixed. Thus, the major advantage of RC compared to other recurrent neural networks is fast learning, resulting in the use of minimal computing resources [Lukoševičius and Jaeger, 2009].

Another advantage of RC is that the reservoir is very malleable, which opens the possibility of using a variety of physical systems, substrates, and devices to implement RC: from ecological networks [Ushio et al., 2023] to brain organoids [Cai et al., 2023]. Due to the enormous diversity of possible architectures to implement, this feature makes RC a leading archetype for neuromorphic software development. Figure 2.1 shows the common workflow used in reservoir computing.

Nevertheless, as we previously pointed out, to successfully program a neuromorphic configuration, we must begin by understanding how the reservoir components are capable of processing, storing and transmitting information. This is where the Belousov-Zhabotinsky chemical oscillators become relevant, since being equivalent to a Linear Bounded Automaton, it is possible to characterize their ability to handle information and perform computations. Before focusing on how to perform such a task—and how at the same time this is compatible with the adjacent solution problems described in Section 1.3—let us first start by understanding the logic behind RC.

To implement a standard RC, consider a dynamical system whose n-dimensional state $\mathbf{x}$ obeys a set of n autonomous differential equation of the form

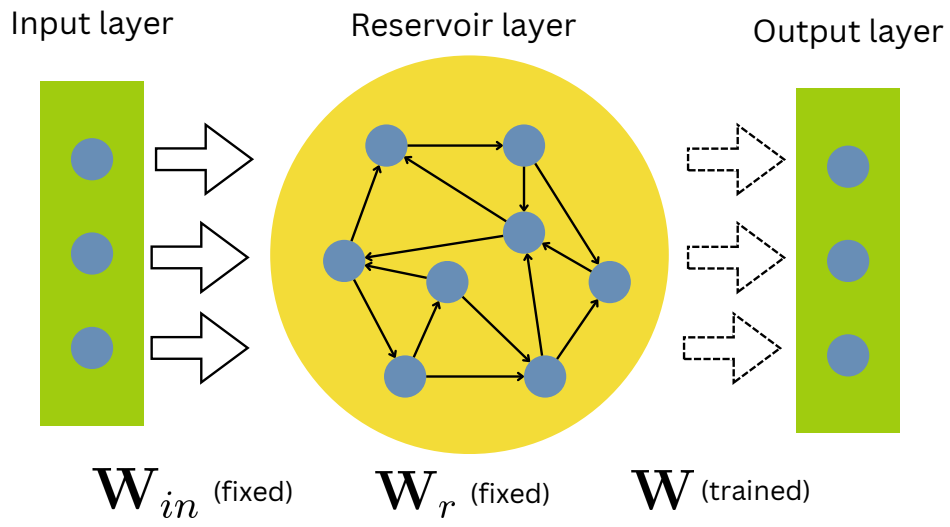


Figure 2.1: Typical RC architecture. The reservoir layer consists of randomly connected neurons. The connections between the input and reservoir layer $\mathbf{W}_{in}$ and connections within the reservoir layer $\mathbf{W}_r$ are fixed (solid arrows), whereas the output weights $\mathbf{W}$ are trained (dashed arrows). Adapted from [Sakemi et al., 2020].

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}). \tag{2.1.1}$$

In general, the goal of RC is to approximate the equation above in discrete time by the following map:

$$\mathbf{x}_{t+1} = \mathbf{F}(\mathbf{x}_t). \tag{2.1.2}$$

In standard RC, one views the state of the real system as a linear readout from an

auxiliary reservoir system, whose state is an N_r-dimensional vector $\mathbf{r}_t$. Specifically,

$$\mathbf{x}_t = \mathbf{W} \cdot \mathbf{r}_t, \tag{2.1.3}$$

where $\mathbf{W}$ is an $n \times N_r$ matrix of trainable output weights. The dynamics of the reservoir system is usually described by

$$\mathbf{r}_{t+1} = f\left(\mathbf{W}_{in} \cdot \mathbf{u}_t + \mathbf{W}_r \cdot \mathbf{r}_t\right). \tag{2.1.4}$$

Here $\mathbf{W}_{in}$ is the $N_r \times n$ input matrix and $\mathbf{W}_r$ is the $N_r \times N_r$ reservoir matrix. The input $\mathbf{u}_t$ is an n-dimensional vector that represents either a state of the real system ($\mathbf{u}_t = \mathbf{x}_t$) during training, or the model's own output ($\mathbf{u}_t = \mathbf{W} \cdot \mathbf{r}_t$) during prediction. Usually, the activation f is taken as a non-linear element-wise function. As noted above, in general only the output matrix $\mathbf{W}$ is trained, with $\mathbf{W}_r$ and $\mathbf{W}_{in}$ generated randomly from appropriate ensembles. However, following [Ushio et al., 2023], in order to use a system of ordinary differential equations as a reservoir we should modified Eq. (2.1.4) by

$$\mathbf{r}_{t+1} = f\left(\mathbf{W}_{in} \cdot \mathbf{u}_t + g(\mathbf{r}_t)\right), \tag{2.1.5}$$

where now the vector $\mathbf{r}_t$ contains the variables associated to the dynamical system

and g represents the right hand-side of the model (represented as a set of ordinary differential equations) that we are going to use. In order that Eq. (2.1.5) could be truly interpreted as the dynamical system, we can let f to be the identity function [Ushio et al., 2023].

To train an RC model from a given initial condition $\mathbf{x}_0$, we first integrate the real dynamics (2.1.1) to obtain N_{train} additional states $\{\mathbf{x}_t\}_{t=1,...,N_{\text{train}}}$. We then iterate the reservoir dynamics (2.1.5) for N_{train} times from $\mathbf{r}_0$, using the training data as inputs ($\mathbf{u}_t = \mathbf{x}_t$). This produces a corresponding sequence of reservoir states, $\{\mathbf{r}_t\}_{t=1,...,N_{\text{train}}}$. Finally, we solve the following equation for the output weights $\mathbf{W}$—that render Eq. (2.1.3) the best fit to the training data—using Ridge regression with Tikhonov regularization,

$$\mathbf{W} = \mathbf{X}\mathbf{R}^T(\mathbf{R}\mathbf{R}^T + \lambda\mathbb{I})^{-1}. \tag{2.1.6}$$

Here, $\mathbf{X}$ ($\mathbf{R}$) is a matrix whose columns are the vectors $\mathbf{x}_t$ ($\mathbf{r}_t$) for $t = 1, ..., N_{\text{train}}$, $\mathbb{I}$ is the identity matrix, and $\lambda > 0$ is regularization coefficient *"that prevents ill conditioning of the weights, which can be symptomatic of overfitting the data"* [Zhang and Cornelius, 2023]. Note that when $\lambda = 0$, Eq. (2.1.6) reduces to an ordinary least squares (OLS) regression. Last but not least, once we have found the weights, we simply replace the input with the model's own output at the previous iteration ($\mathbf{u}_t = \mathbf{W} \cdot \mathbf{r}_t$) and make the prediction. When measuring the accuracy of time series forecasting there are several evaluation metrics available [Hyndman, 2018]. Here I am

focusing to Error Measures, which focus on measuring the accuracy and magnitude of errors in the forecasted values when compared to the actual values. Particularly, root mean square error (RMSE) it measures the average difference between the predicted and actual values, taking into account the squared differences to emphasize larger errors. Most of the time the preferred error measure in Reservoir Computing is the RMSE between the actual time series and our prediction. Why? because it penalizes large errors due to squaring and it is easily interpretable thanks to expressing the number in the magnitude of the variables.

2.2 An experimentally valid model for chemical oscillators

To implement reservoir computing (RC) using chemical oscillators as reservoir, there are at least three possible paths to be explored. If we stick to the origins of the RC, the most intuitive thing to do is to construct a network capable of representing the dynamics of the Belousov-Zhabotinsky (BZ) reaction and use it as reservoir. As has been well demonstrated in [Müller et al., 2022], there are multiple ways to represent chemical networks using the paradigm of graphs and networks.

However, although it is possible to create a complex network that dynamically mimics the BZ reaction, the task of coupling at least two of these networks to capture the dynamics of two coupled chemical oscillators is not trivial at all. Basically, depending

on the model used (a Boolean network, a bipartite graph, a hypergraph), we must perform the concatenation of the two graphs in a different fashion, which makes the way we define the coupling between two oscillators arbitrary.

On the other hand, there is the possibility of using time series produced by a real BZ reaction as a reservoir [Ushio et al., 2023]. For this it is necessary to implement a technique called *time-delay embedding* [Takens, 2006]. With the above we now only need the time series produced by two coupled oscillators (or failing that, the individual time series of each oscillator) to implement the coupling. The problem with using this approach is that to make accurate predictions we must define a suitable embedding dimension for each case [Deyle and Sugihara, 2011], which is not trivial either.

Lastly, as it was shown in [Ushio et al., 2023], it is possible to perform RC using a dynamical system if such a system can be represented by a system of differential equations. Our question now is whether there is a system of differential equations suitable for such a task. In 2022 the same Harvard group that laid the foundations for non-hybrid chemical computation published a paper showing that it is possible to experimentally observe the spontaneous synchronization of two electrochemically coupled BZ oscillators [Liu et al., 2022].

In order to test their experimental results, the authors extended the Oregonator BZ model to take into account the drifting natural frequencies in batch condition and electrochemical coupling, arriving at a model that is able to reproduce their experi-

mental results, requires only five differential equations to represent each oscillator and whose few dimensionless parameters are fine tuned with *in vivo* experiments [Liu et al., 2022]. Since it is very straightforward to define the coupling of two or more oscillators using this model, we will use it to perform RC.

In order to represent a single oscillator, the equations that we will use are the following:

$$\varepsilon \frac{dx}{d\tau} = qay - xy + ax - x^2, \tag{2.2.1}$$

$$\gamma \frac{dy}{d\tau} = -qay - xy + fbz, \tag{2.2.2}$$

$$\frac{dz}{d\tau} = ax - bz, \tag{2.2.3}$$

$$\alpha \frac{da}{d\tau} = -qay - ax + x^2, \tag{2.2.4}$$

$$\beta \frac{db}{d\tau} = -bz. \tag{2.2.5}$$

In Eqs. (2.2.1)-(2.2.5) $\varepsilon = 0.0667$, $\gamma = 8.89 \times 10^{-5}$, $q = 8 \times 10^{-4}$, $\alpha = 270$, $\beta = 430$ and $f = 0.65$. For two uncoupled oscillators, an additional set of five equations was added:

$$\varepsilon \frac{dx'}{d\tau} = qa'y' - x'y' + a'x' - x'^2, \tag{2.2.6}$$

$$\gamma \frac{dy'}{d\tau} = -qa'y' - x'y' + fb'z', \tag{2.2.7}$$

$$\frac{dz'}{d\tau} = a'x' - b'z', \tag{2.2.8}$$

$$\alpha \frac{da'}{d\tau} = -qa'y' - a'x' + x'^2, \tag{2.2.9}$$

$$\beta' \frac{db'}{d\tau} = -b'z'. \tag{2.2.10}$$

In Eqs. (2.2.6)-(2.2.10) the value of the parameters are almost the same, excepting $\beta' = 412.8$. If the oscillators are coupled, we need to modify Eqs. (2.2.3) and (2.2.8), substituting them by

$$\frac{dz}{d\tau} = ax - bz + k \ln \frac{(1-z)z'}{z(1-z')}, \tag{2.2.11}$$

$$\frac{dz'}{d\tau} = a'x' - b'z' - k \ln \frac{(1-z)z'}{z(1-z')}. \tag{2.2.12}$$

In Eqs. (2.2.11) and (2.2.12) k is the coupling strength, which quadratic approximation used in [Liu et al., 2022] is given by

$$k = \kappa[H]^2, \tag{2.2.13}$$

where $\kappa = 1.1 \times 10^{-5}$ and $[H]$ is the hydrogen ion concentration of the solution. At high acid concentration, which experimentally reflects strong coupling between the oscillators, $[H] = 3$. Otherwise, $[H] = 1.5$, which experimentally is associated with low coupling between the oscillators [Liu et al., 2022].

With all these equations, we will have a total of four cases to study computational efficiency under the reservoir computing paradigm. The first one is having a single oscillator. In that case, it is enough to use Eqs. (2.2.1)-(2.2.5). For two uncoupled oscillators we can use Eqs. (2.2.1)-(2.2.10). When the oscillators are coupled we use Eqs. (2.2.11) and (2.2.12) instead of Eqs.(2.2.3) and (2.2.8). In this latter situation we will consider the case where we have low acid concentration ($[H] = 1.5$) and where we have high acid concentration ($[H] = 3$).

Another point that I think it is also important to emphasize is that in the work of [Liu et al., 2022], the effects of different initial conditions on the synchronization of BZ oscillators are studied. Essentially, the authors propose three types of initial conditions, each associated with the initial synchronization of two oscillators. The first one is the In-phase initial condition, which is

$$(x_0, y_0, z_0, a_0, b_0) = (0, 3.287, 0.006225, 0.4, 0.4)\,. \tag{2.2.14}$$

This initial condition refers to the fact that the two oscillators are at the same point in their cycle. On the other hand, the Out-of-phase initial condition is

$$(x_0, y_0, z_0, a_0, b_0) = (0, 24.26, 0.04515, 0.4, 0.404)\,. \tag{2.2.15}$$

As can be guessed, in this case the oscillators have the same frequency but different phases. Finally, we have the Anti-phase initial condition:

$$(x_0, y_0, z_0, a_0, b_0) = (0.1411, 0, 0.06109, 0.4, 0.404)\,. \tag{2.2.16}$$

In this case the two oscillators have phases or cycles that are in opposite directions. In general, the results of [Liu et al., 2022] show that the initial conditions are of little relevance to obtain spontaneous synchronization. In order to study a possible relationship between such spontaneous synchronization and computational capability in coupled oscillators, for my experiments I will also make use of the initial conditions (2.2.14), (2.2.15), and (2.2.16).

In the case of having a single oscillator, I'm going to use Eqs. (2.2.14)-(2.2.16) individually. Given that the equations describing two oscillators differ in parameter β', for the cases where we have two oscillators (uncoupled or coupled) I will test all the possible combinations of initial conditions. This is in order to make an exhaustive search and do not omit anything interesting.

In general, a larger reservoir leads to better prediction efficiency, as it allows for a greater capacity to store and process temporal information [Zhong et al., 2021]. Thus,

I expect that the two coupled oscillators is always more efficient than two uncoupled oscillators, and I also expect this latter to be more efficient than a single oscillator. Additionally, I will contrast the above results with a classical Ridge regression, since this is a conventional method which serves as a landmark for contrasting the results of reservoir computing [Ushio et al., 2023].

2.3 Why time series forecasting?

Having laid the groundwork for reservoir computation and found a suitable mathematical representation that allows us to capture the experimental characteristics of the substrate we are interested in using as a reservoir, the time has come to choose a proper task that will allow us to study the computational capability of the chemical oscillators. To make such a selection, the first part of this section is devoted to analyzing the experimental set used in [Dueñas-Díez and Pérez-Mercader, 2019], which will allow us to glimpse the reasons why time series forecasting is suitable as a metric of computational efficiency in our context. As we will see, this choice is also aligned to the solution of the problems introduced in Section 1.3. I will then introduce the methodology I used to obtain the results presented in this thesis.

In their experiments, [Dueñas-Díez and Pérez-Mercader, 2019] start with a chemical reactor previously prepared with the necessary ingredients to carry out the chemical reaction. Thus, the input to be executed is represented by a combination of *aliquots*, which are constant quantities (carefully chosen) of certain chemical species. The

input is added aliquot after aliquot to the one-pot reactor at constant time intervals. Finally, the output of the computation is in the form of a chemical response. In the case of the BZ reaction, in principle this is the frequency and amplitude changes in the oscillations were observed.

Subsequently, after some corrections in the accept/reject interpretation and optimizing the quantities for some aliquots, the authors succeeded in recharacterizing the acceptance and rejection of aliquot sequences based on energetic considerations. In other words, the free-energy difference between the oxidized and reduced states of the catalyst after the processing the word is the same for all words in the language accepted by the chemical TM [Dueñas-Díez and Pérez-Mercader, 2019]. This is quite relevant because, since it is a constant value, under a translation in such free-energy difference we convert the acceptance/rejection problem of a sequence of aliquots to a minimization problem.

Therefore, based on the fact that even the signals that encode binary values in electronic logic circuits are in reality stored as continuous-valued voltages [Grozinger et al., 2019], it is not at all unreasonable to assume that time series forecasting of one or more chemical oscillators captures the acceptance/rejection intuition shown by [Dueñas-Díez and Pérez-Mercader, 2019]. Why? Because in both cases we want to minimize something. The interpretation of the latter is as follows. The complexity of the time series is associated with the complexity of the language used as input. Therefore, the more efficient one or more chemical automata are at time series forecasting, the more efficient the chemical Turing machine is at processing complex

computable languages.

Let us now briefly discuss how the latter fact is also compatible with solving the problems of studying metabolic biocomputing and extending our knowledge in algorithmic networks, both introduced in Section 1.3. As was mentioned in [Grozinger et al., 2019]: *"Although genetic circuits may appear to behave digitally, it is only the collective behaviour of a large number of inherently analogue components that give rise to this property".*

From that perspective, by studying the time series forecasting of one or more chemical oscillators, we are glimpsing the computational capabilities of primitive metabolic systems (biochemistry-free). This of course can give us information about how the metabolism computes and what is the impact of collectivity on it. In this way, we are advancing towards the development of a formal framework within which to argue that a cell computes, according to any understood model of computation.

On the other hand, so far only theoretical advances have been made in the paradigm of algorithmic networks. Although such progress points to very interesting directions such as the computation beyond the Turing barrier from emergent behavior in distributed systems [Abrahão et al., 2020], the theoretical model that has been used as a node in such networks is nothing more and nothing less than a Turing machine, an abstraction that, since it requires an infinite amount of physical resources, cannot be materialized.

Studying the computational capacity of coupled automata belonging to a class lower

than Turing machines can give rise to the application of the theoretical results in the framework of algorithmic networks. In fact, what is proposed here aims at the same time to expand such a mathematical tool, since the topologies proposed here are not random and we implement different types of coupling that go beyond the dichotomous. These two properties turn out to be relevant in biological systems [López-Díaz et al., 2023].

2.3.1 Methodology

We will now describe how I generated the different time series that were used to test our reservoir. Taking into account that the time series under our context can be seen as the complexity of the language used as input in the chemical Turing machine, the ideal would be to look for a representation of it that allows us to analyze different degrees of difficulty when forecasting. I therefore decided to use the chaotic time series generated by the Z-variable of the Lorenz system, a rather famous dynamical system given by

$$\frac{dX}{dt} = -pX + pY, \tag{2.3.1}$$

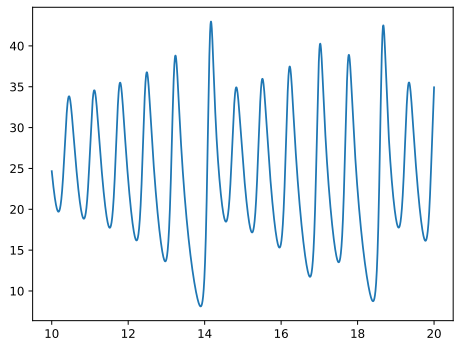
$$\frac{dY}{dt} = -XZ + rX - Y, \tag{2.3.2}$$

$$\frac{dZ}{dt} = XY - bZ, \tag{2.3.3}$$

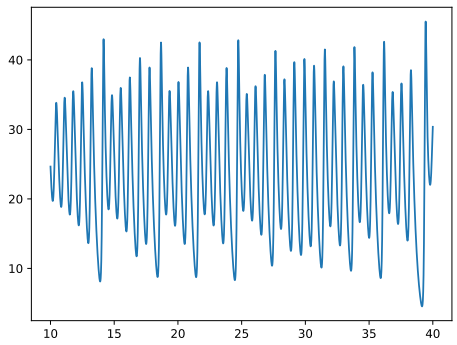
where $(p, r, b) = (10, 28, 8/3)$. Following [Ushio et al., 2023], the initial values of X, Y, and Z were set to one. Then variable Z was integrated over a temporal interval $[0, t]$, such that the time series to train the reservoir is the one given by the interval $[10, t]$. Here t defines what I will be calling *oscillatory level*. The reason for integrating over $[10, t]$, and not over $[0, t]$, is because the time series is in an initialization period during $[0, 10]$. Therefore, during $[10, t]$ we avoid training the time series over that warm-up phase. By doing $t \in \{20, 30, ..., 90, 100\}$ we get a total of nine time series, each associated with the integration interval $[10, t]$. Figure 2.2 shows four of the nine different time series used, obtained by varying the oscillatory level t.

A very important observation must be made here. When doing the integration, for each oscillatory level the number of points to be considered was manipulated in such a way that all time series have the same number of points regardless of the oscillatory level. The reason behind this is that if we let the time series have different resolutions (given by the number of points), due to the nature of the algorithm used to train the reservoir, the time series with high resolution (with higher number of points) will have a higher prediction quality. Thus, by normalizing the number of points in each time series we solve that problem.

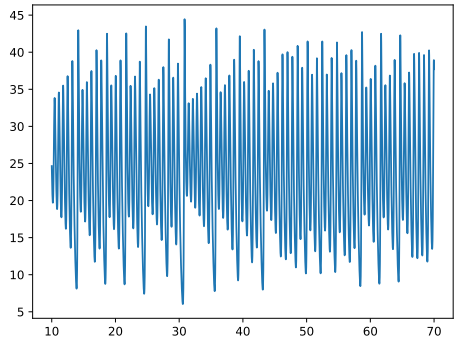
Subsequently, for each of the four reservoirs (single oscillator, two uncoupled oscillators, two weakly coupled oscillators and two strongly coupled oscillators) each of the nine time series was processed leaving fixed the model parameters (outlined in Section 2.2), the test fraction (its value being 0.2, as usual), and the value of the regularization coefficient (λ). Each time series was standardized by subtracting the minimum of the



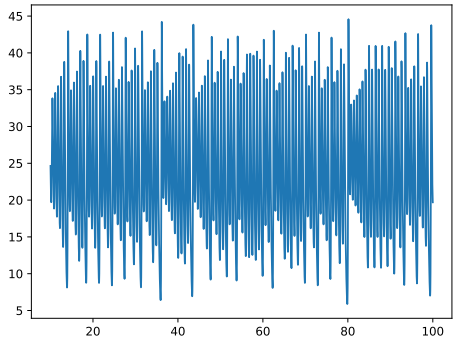
(a) $t = 20$



(b) $t = 40$



(c) $t = 70$



(d) $t = 100$

Figure 2.2: Four different time series associated with different oscillatory levels (denoted by t). In all plots the x axis is associated with time, while the y axis represents the value of the Z variable in the system (2.3.1)-(2.3.3).

time series from all values and dividing all resulting values by the difference of the maximum and minimum of the time series, as suggested in [Ushio et al., 2023].

Depending on the case (single oscillator or two oscillators) different families of initial conditions were used (see Section 2.2). Returning to the architecture shown in Figure 2.1, we are using only one input node and one output node since both input

and output are scalar values (represented by the value acquired by the time series). Another important point is that for the stochastic case the same set of equations described in section 2.2 were used, with the difference that an additive form of noise will be added to each of the equations. The addition of noise will be further explained below. The above is inspired by [Muñuzuri and Pérez-Mercader, 2022], where additive-noise terms modulate the rates of change in concentrations for each of the participating chemical species.

Chapter 3

Results

In the following I will present the results obtained for both the deterministic and stochastic cases. In order not to clutter this section with plots, I will focus on showing the key results associated with both cases, leaving the rest of the plots in an appendix at the end of this thesis. After each plot I will give a brief analysis explaining the interpretation of the results. Subsequently, in the Discussion section, I will focus on exploring the implications of the results described here.

3.1 Terminology

Before showing the results, it is necessary to introduce some terminology that will help us to delve into the meaning of the results.

Definition 3.1.1. Given a reservoir composed of chemical oscillators, the efficiency of the reservoir is given by the average RMSE when forecasting a time series across different initial conditions. Thus, the efficiency is given by the average RMSE.

Definition 3.1.2. Given a reservoir composed of chemical oscillators, we say that such a system is invariant when all possible combinations of initial conditions produce the same efficiency. If a system is not invariant, we say that it is non-invariant. Numerically, a system is invariant if the standard deviation of the efficiency is less than 10^{-6}.

Definition 3.1.3. Given a reservoir described by a system of differential equations, we define the complexity of the reservoir by the number of equations and nonlinear terms contained in the mathematical description of the reservoir. If we label the model by A, the complexity of the reservoir will be denoted by C_A.

Definition 3.1.4. Let A and B be two differential equation models describing two reservoirs, forecasting the same time series, such that $C_A < C_B$. If the efficiency of A is less than or equal to the efficiency of B, we say that B is reducible to A. If the two reservoirs turn out to have the same efficiency we say that they are equivalent. Numerically, two reservoirs are equivalent when the absolute difference between their efficiencies is less than 10^{-6}.

The aforementioned terms will be of importance when introducing what we will later call invariance and reducibility diagrams. Before proceeding, it is important to note that in both Definition 2.1.3 and Definition 2.1.4 we are making use of an arbitrary numerical threshold of 10^{-6}, which may have an impact when interpreting

such invariance and reducibility diagrams. The value of this threshold was assigned with respect to my observations on the results obtained for a reservoir with two chemical automata. As we will see below, for the cases of two uncoupled oscillators, two weakly coupled oscillators and two strongly coupled oscillators, the value of the efficiency is very similar. However, the difference between these values is remarkable up to the order of magnitude 10^{-6}. That is why we are making use of this threshold value.

3.2 Deterministic Case

Given one of the nine time series we generated using the methodology exposed in Section 2.3.1 we will test each of the four reservoir cases (single, uncoupled, coupled low and coupled high) when forecasting such time series. Remember that each reservoir is a system of differential equations. Thus, we need to set up an initial condition. In [Liu et al., 2022] the authors propose different initial conditions associated to the initial synchronization states of the oscillators. For each reservoir, I obtained the RMSE for each of those initial conditions. Then I averaged the RMSE for different initial conditions to obtain the average RMSE (referred in this work as *efficiency*) associated to the reservoir when forecasting one of the nine time series. Fig. 3.1 shows the average RMSE for the different chemical reservoirs used. Observe that we are comparing the chemical oscillator reservoirs with a simple Ridge regression. This is common in the Reservoir Computing literature in order to contrast results obtained

with and without reservoir.

Taking into account the terminology introduced in Section 2.1, to correctly interpret this plot we must remember that having a low average RMSE (with respect to multiple initial conditions) means that the reservoir is more efficient. At the same time, remember that as we increase the oscillatory level, the reservoir is forecasting a more complex time series. In this context complex means that the time series is more difficult to forecast given the high density of oscillations. An interesting observation follows from the above. From the oscillatory level $t = 50$ the chemical reservoirs are more efficient than a simple Ridge regression and we can observe a clear hierarchy of efficiency between the reservoir composed of a single chemical automaton and the ones composed of two chemical automata. This same behavior is rescued by using different values $\lambda > 0$, as shown in Figures 5.4-5.10 in Appendix I.

As is observed in Fig. 3.1, the values of the cases U, CL and CH are very closed to each other. In fact, they are the same until the sixth decimal place. In order to explore if there is any small preference towards each of these cases, let's analyze the invariance and reducibility diagrams for a regularization coefficient of $\lambda = 0.5$ (same parameter configuration as in Fig. 3.1). These diagrams will give us information about a couple of intrinsic and comparative properties between the reservoirs. On the one hand, in Fig. 3.2(a) we observe that there is a certain time threshold ($t = 50$) from which the behavior of the system is independent of the initial conditions assigned to it. On the other hand, Fig. 3.2(b) tells us that from that same time threshold there is a deterministic behavior in the efficiency hierarchy associated to different models.

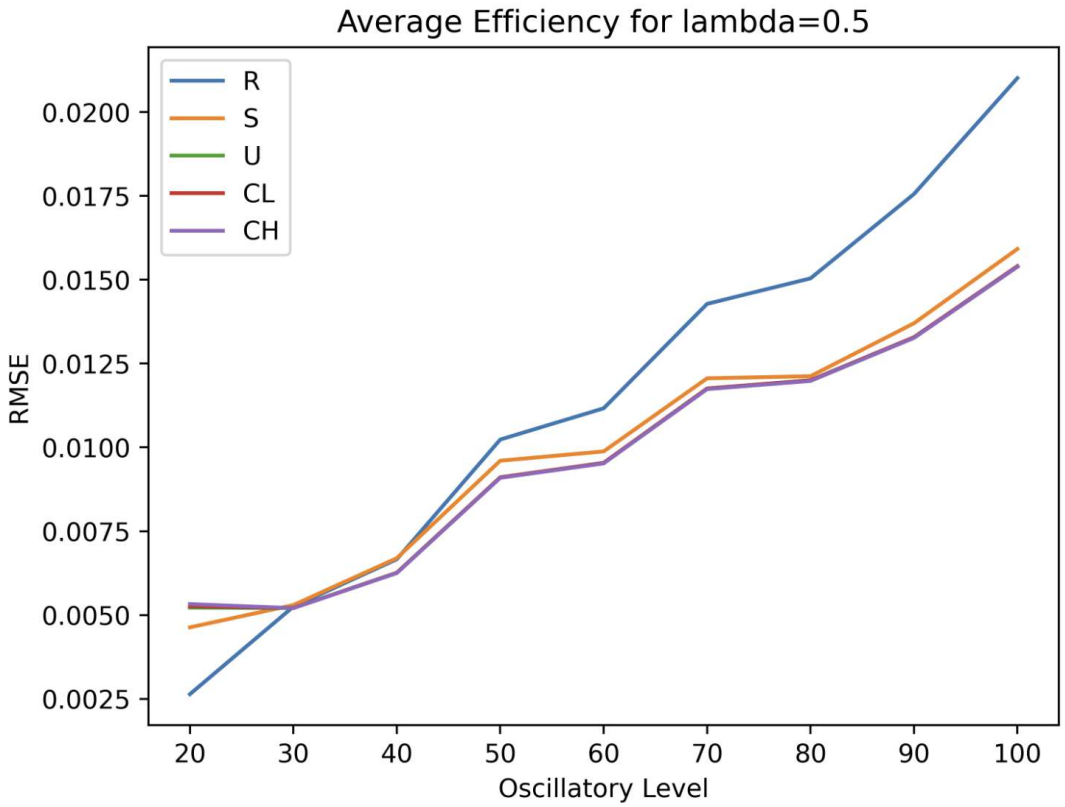
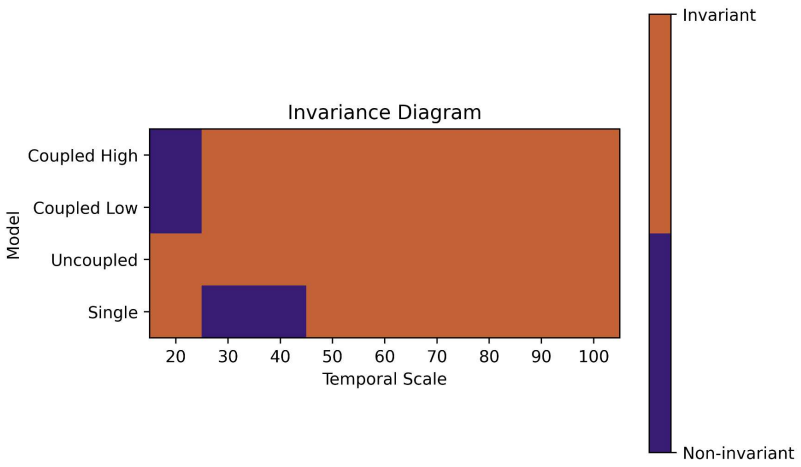
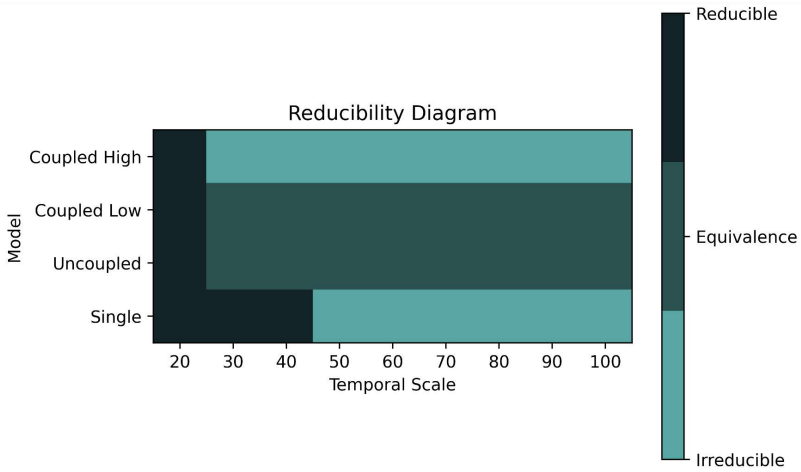


Figure 3.1: Average RMSE obtained for the different reservoirs using $\lambda = 0.5$. R stands for Ridge Regression, S for Single Oscillator, U for Uncoupled Oscillators, CL for Coupled Low Oscillators and CH for Coupled High Oscillators.

Let me explain the latter in more detail. First of all, let us recall that the oscillatory levels (here called temporal scale for better interpretation) are associated with the degree of "complexity" of the time series, which means the degree of difficulty when forecasting such time series. As the Figure 2.2 illustrates, by increasing the temporal scale the time series becomes more intricate, so we would expect that the reservoir (no matter how many oscillators it contains and how they are coupled) will have a slightly harder time making such a prediction.



(a)



(b)

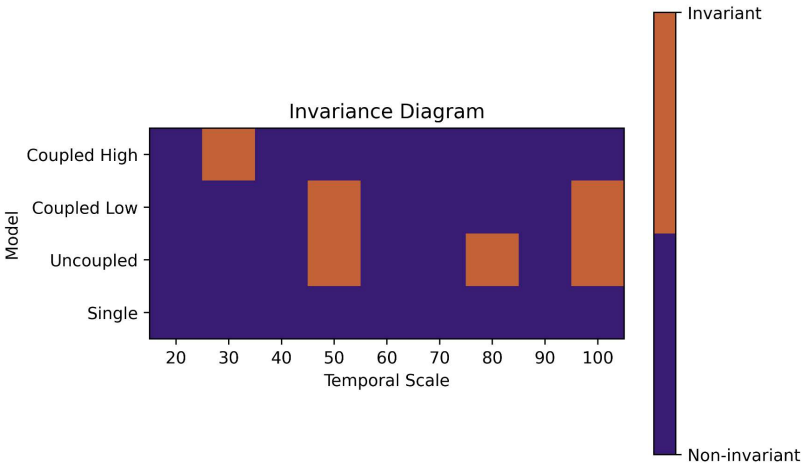
Figure 3.2: Diagrams obtained for $\lambda = 0.5$. In both the x axis represents the different oscillatory levels described in Sections 2.3.1.

What the Figure 3.2(b) shows us is that from $t = 50$ the reservoir with a single chemical automaton always has a higher efficiency than a Ridge regression (without

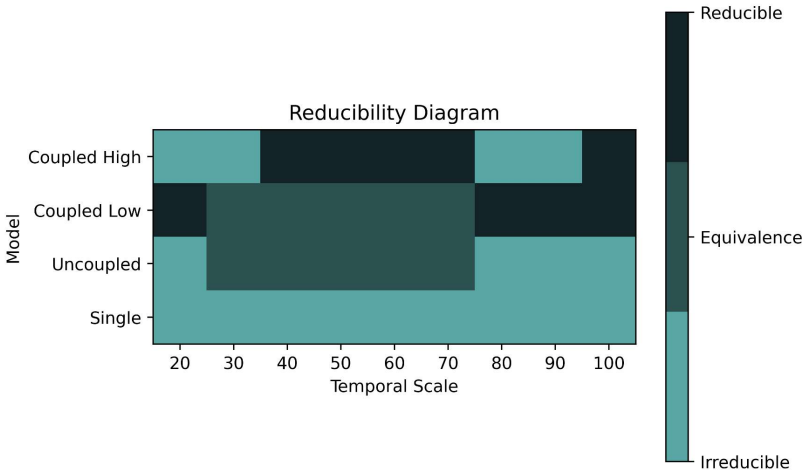
reservoir). At the same time, there is an equivalence between a reservoir with two uncoupled and weakly coupled automata, both being more efficient than a single chemical automaton. Finally, the system with two strongly coupled automata is much more efficient than the cases described above.

At this point it is very interesting to note that both thresholds (for both the invariance and reducibility diagrams) coincide. In fact, this coincident and hierarchical behavior is consistent for other values $\lambda > 0$ (under an increase in threshold as we increase λ). In general such a change in the value of λ does not critically affect reservoir efficiency, as illustrated in the Figures 5.1-5.3 shown in Appendix I of this thesis. However, there is a difference when studying the degenerate case $\lambda = 0$, which implies that Eq. (2.1.6) acquires the OLS form. This information coincides with what is described in Fig. 3.1, but the latter did not give us a glimpse of the intrinsic hierarchy between the reservoirs composed of two oscillators.

As illustrated in Figs. 3.3, when $\lambda = 0$ the behavior of the different reservoirs is completely arbitrary with respect to the variation of the temporal scale. This implies that it is not possible to identify a threshold beyond which the system behaves in a deterministic manner. A straightforward question that follows from the above is why this happens. As noted in Section 2.1, the value of λ is chosen to be strictly positive to prevent overfitting the data. By setting $\lambda = 0$ we are leaving out any regularization that avoids that. Therefore, it is normal to observe anomalous behavior in such a degenerate case.



(a)



(b)

Figure 3.3: Diagrams obtained for $\lambda = 0$.

To wrap up this section and move on to study the stochastic case, let us make some general remarks which will be extended later in the discussion. For the deterministic

case we observe that, regardless of the value $\lambda > 0$, there exists an oscillatory level such that when crossing it

1. the value of the efficiency becomes fully invariant to the initial conditions used,
2. there is a hierarchy of efficiency in which reservoirs with two oscillators are more efficient than the reservoir with one oscillator, which in turn is more efficient than a simple Ridge regression.

From the above it can be concluded that, for significant temporal scales (where the time series becomes more "complex" as we mentioned above) it is necessary to have at least two chemical automata conforming the reservoir in order to have a higher forecasting accuracy. Moreover, the diagram 3.2(b) shows that after this threshold temporal scale there is an equivalence between the reservoirs of two coupled automata and two weakly coupled automata, both being less efficient than the reservoir composed of two strongly coupled automata.

3.3 Stochastic Case

As has been pointed out in [Grozinger et al., 2019], noise is inherent to the computational dynamics of living beings. Although this could be interpreted as a disadvantage, stochasticity is part of one of the evolutionary mechanisms that life has used since its origin. It is clear that if the noise level is high the system will enter a state of

catastrophe and fragility. However, an adequate amount of noise may be sufficient and necessary for evolutionary purposes [Roy and Majumdar, 2022]. That is why, following the methodology proposed by [Muñuzuri and Pérez-Mercader, 2022], I have taken on the task of studying the computational efficiency of different reservoirs under a fixed amount of noise of different colors.

In total I have studied the effects of five types of noise: white, pink, brown, blue and violet. This selection was based on ecological observations showing the importance of each of these color noises in different environments [Vasseur and Yodzis, 2004]. Let $\mathbf{u}$ be the vector containing the variables of a system of n ordinary differential equations and let $\bar{\alpha}$ be a vector of k parameters of the system of ordinary differential equations. We add noise in the following way.

$$\nabla_t \mathbf{u} = \mathbf{F}(\mathbf{u}; \bar{\alpha}) + \boldsymbol{\xi}(t) \tag{3.3.1}$$

where ∇_t represent the temporal gradient applied to the variables of the system of ordinary differential equations, $\mathbf{F}$ is a function representing the right hand side of the system of ordinary differential equations and $\boldsymbol{\xi}(t)$ is a temporally updated vector with n samples of Gaussian $(1/\varphi)^\eta$ noise, where φ is the frequency and the parameter η determines the colored noise we are using. Since we are using Gaussian noise, the function $\boldsymbol{\xi}(t)$ shown in Eq. (3.3.1) is such that

$$\langle \boldsymbol{\xi}(t) \rangle = 0$$

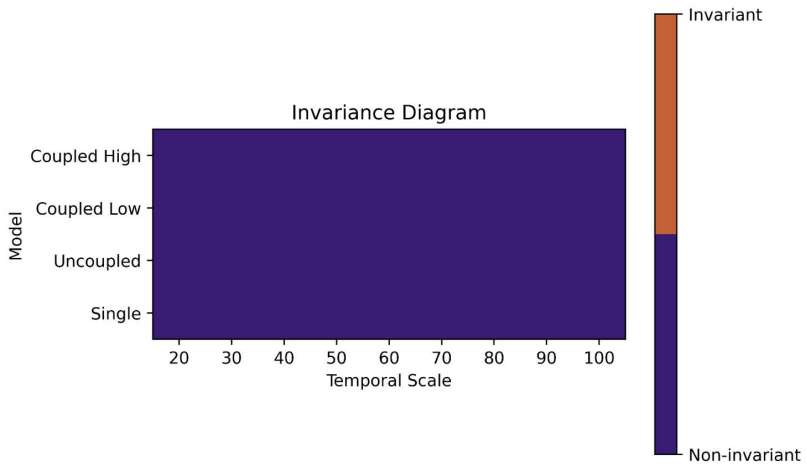
$$\langle \boldsymbol{\xi}(t) \boldsymbol{\xi}(t') \rangle = D \; \delta(t - t')$$

where the noise intensity D was set as in [Muñuzuri and Pérez-Mercader, 2022], and the probability density function is such that

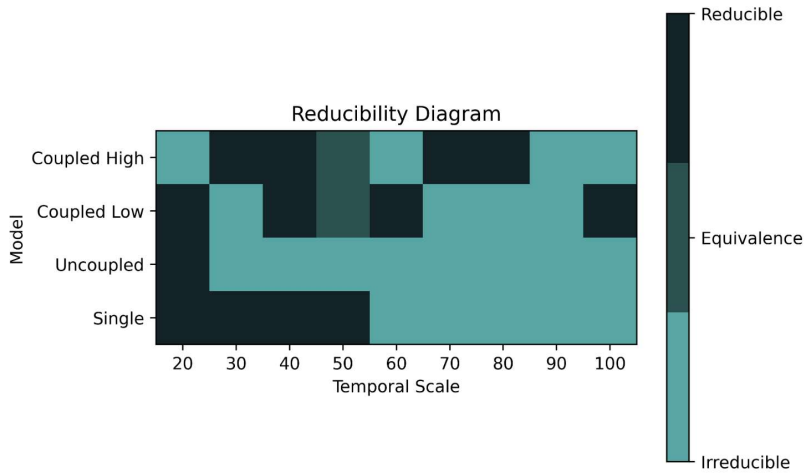
$$p_{\boldsymbol{\xi}} \; \propto \; \exp\left(-\frac{1}{2D} \int \boldsymbol{\xi}^2(t) \; dt \right) \tag{3.3.2}$$

In this way, by setting $\eta = 0$ we obtain white noise, $\eta = 1$ pink noise, $\eta = 2$ brown noise, $\eta = -1$ blue noise and $\eta = -2$ violet noise. This is equivalent to the methodology used by [Muñuzuri and Pérez-Mercader, 2022] when studying the effects of noise on primordial life forms. The idea is the same as in the previous section. I will take white noise as a canonical example, primarily because of its ecological importance in terrestrial environments [Vasseur and Yodzis, 2004], and also because in general the behavior is the same for other types of noise. In particular, I will again select $\lambda = 0.5$. This will allow us to contrast the results obtained for the deterministic case (presented in the previous section) and the stochastic case. Notwithstanding the above, all other results for different types of noise and different values of $\lambda > 0$ will be presented in the appendix of this thesis.

Figure 3.4 shows the invariance and reducibility behavior for the stochastic case. As



(a)



(b)

Figure 3.4: Diagrams obtained for $\lambda = 0.5$ using additive white noise. In both the x axis represents the different oscillatory levels described in Sections 2.3.1.

we can see by comparing with Fig. 3.2, there are important differences between this and the deterministic case. In Fig. 3.4(a) we observe that absolutely for all temporal

scales (and for all models used as reservoir) the system is always non-invariant, which implies that there is a dependence on the initial conditions we select. Regardless of $\lambda > 0$, this behavior is the same as that obtained for the other types of noise, as shown in Figures 5.14-5.20 Appendix II of this thesis. On the other hand, in Fig. 3.4(b) we observe a much more disordered behavior in terms of reducibility. While in the deterministic case there was a regular (flag-like) behavior, in the stochastic case this is not true. Although we sometimes observe irreducible and deterministic behavior for the Single and Uncoupled cases, this rule is not general for any type of noise or for any value of $\lambda > 0$. This is shown in detail in Figures of Appendix II-VI.

However, by looking at Fig. 3.5 we obtain a much more illuminating result than that obtained with the diagrams 3.4. We note that, despite the additive white noise implementation, there is a threshold oscillatory level ($t = 70$) such that we get back the previously observed efficiency hierarchy. Thus, we obtain that after $t = 70$ the reservoirs with two chemical automata are more efficient than the reservoir with a single automaton, and in turn the latter is more efficient than a Ridge regression. As shown in Figures 5.18-5.20, in this case by increasing $\lambda > 0$ there an increasing behavior with the threshold oscillatory level (as in the deterministic case). Another relevant question is whether the value of the efficiency varies too much as we vary $\lambda > 0$. As supported by Appendixes II-VI the variations of the average RMSE are negligible regardless of the noise used.

To conclude this section let us highlight the differences between the stochastic case and the deterministic case. Regardless of the type of noise used, there is always a

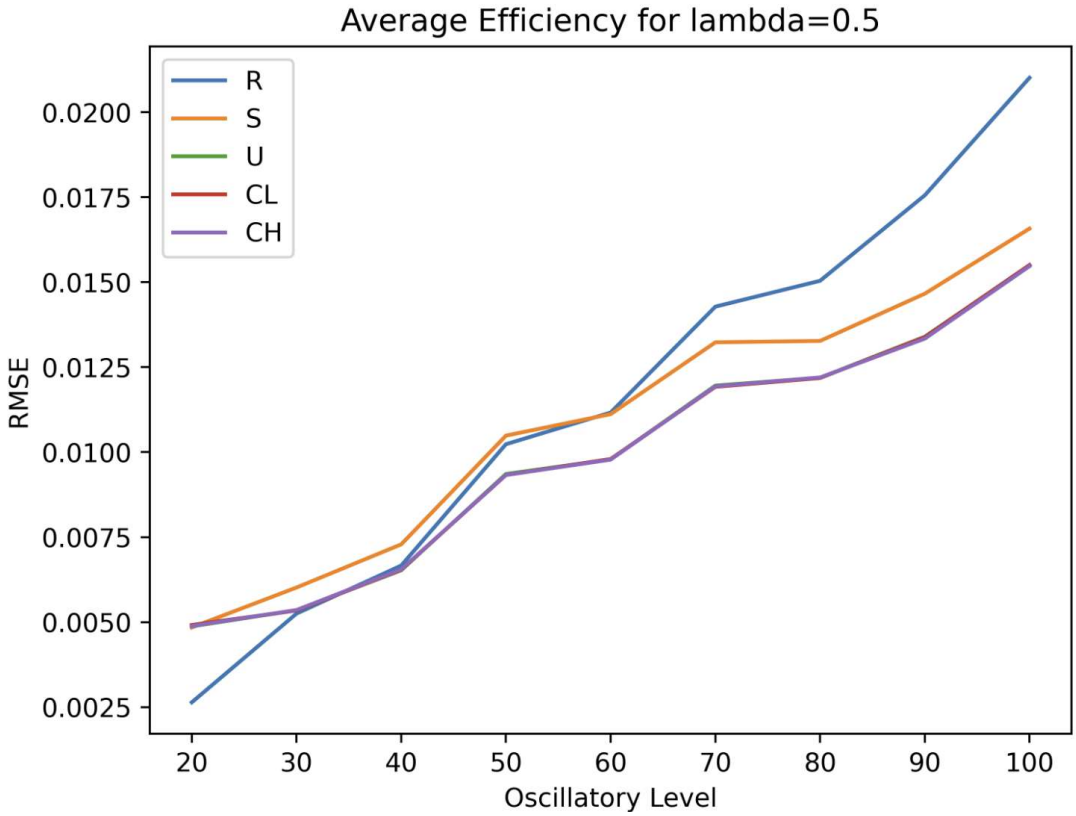


Figure 3.5: Average RMSE obtained for the different reservoirs using $\lambda = 0.5$ and white noise. R stands for Ridge Regression, S for Single Oscillator, U for Uncoupled Oscillators, CL for Coupled Low Oscillators and CH for Coupled High Oscillators.

threshold oscillatory level such that exceeding it respects the efficiency hierarchy previously observed in the deterministic case. In fact, despite a translation in the oscillatory level, by increasing $\lambda > 0$ we observe that the threshold oscillatory level also increases, as in the deterministic case. In contempt of the above, there is now always a dependence on the initial conditions, as shown by the invariance diagrams. Likewise, in the stochastic case it is not always possible to distinguish which reservoir (consisting of more than one automaton) is the most efficient, as shown by the reducibility diagrams. Note that the stochastic case and the degenerate deterministic

case ($\lambda = 0$) behave similarly, as shown in Fig. 3.3. In the next section we will discuss what are the implications of these results, highlighting again not only the construction of a non-hybrid chemical computer, but also in the development of metabolic biocomputing and the extension of algorithmic networks paradigm.

Chapter 4

Discussion

4.1 Chemical Circuits

A Turing machine can accept any language that is considered *computable*, meaning that no matter how grammatically complex a language is, as long as there is a defined set of rules to identify strings belonging to that language, a Turing machine can be designed to recognize it. However, the grammatical complexity of a language can significantly impact how efficient or resource-intensive the Turing machine needs to be to accept it, with more grammatical complex languages potentially requiring more complex Turing machine designs to process them effectively [Hopcroft, 2001]. Let's recall again the time series interpretation. Since the problem of accepting/rejecting sequences of aliquots in a chemical Turing machine is equivalent to a minimization

problem (see Section 1.3), it is possible to argue that a time series with more dense oscillations (which in our terminology is associated with a high oscillatory level) means that we are using a much more intricate language as input.

As a consequence of the above, our results show that independently of the presence of noise, there is always a complexity threshold in the density oscillations of a time series (determined in our methodology by the oscillatory level) such that when this point is exceeded there is a clear hierarchy of efficiency between the reservoir composed of a single chemical automaton and the ones composed of two chemical oscillators (regardless of how they are coupled), the latter family being more efficient than the former. Interpreting the above, this means that in order to identify grammatical complex computable languages efficiently, it is necessary to use a reservoir composed of at least two chemical automata. In the deterministic case we observe that there is a clear preference to the case where the automata are strongly coupled, while in the stochastic case there is a dependence on the type of additive noise used.

Moreover, for the deterministic case we can infer a set of rules in circuits made up of chemical automata (referred here as chemical circuits) that could well be seen as an analogue to Kirchhoff's laws for electrical circuits. These are:

1. For sufficiently complex languages, two strongly coupled chemical automata are irreducible in terms of efficiency.

2. For sufficiently complex languages, two weakly coupled chemical automata are equivalent to two uncoupled chemical automata in terms of efficiency.

In this way, depending on the complexity of the grammatical language we want to use as input, we can optimize the construction of the chemical computer and at the same time maximize its efficiency in processing that language. Of course, the above prewritten laws are only for systems of two automata. By [Liu et al., 2022] we know that the coupling strength between the oscillators was dependent on the acid concentration. At low acid concentrations, the coupling was insufficient for consistent synchronization, while higher acid concentrations significantly enhanced coupling, enabling stable synchronization. With the above, let us prove the following theorem.

Theorem 4.1.1. *A chain of $n \geqslant 2$ strongly coupled chemical oscillators will spontaneously synchronize.*

The proof is by induction. By [Liu et al., 2022] we know that the base case is true. Now, suppose the statement is valid for a chain of n strongly coupled chemical oscillators. For the jth chemical oscillator, we can write its amplitude as $A_j(t)$ and its phase as $\phi_j(t)$, where $j = 1, \ldots, n$. In this way, the states of the jth chemical oscillator are represented by $z_j(t) = A_j(t)e^{i\phi_j(t)}$. If the oscillators are strongly coupled, then for $j, k \in \{1, \ldots, n\}$

1. $\dot{\phi}_j(t) = \omega$,
2. $\phi_j(t) - \phi_k(t) = c$ where c is a constant,
3. $A_j(t)$ approaches a steady value or a well-defined collective average.

Thus, the collective behavior of the system can be described by a single effective oscillator:

$$Z_{\text{eff}}(t) = \frac{1}{n}\sum_{j=1}^{n} z_j \bar{A} e^{i\Phi(t)}, \tag{4.1.1}$$

where $\bar{A} = \frac{1}{n}\sum_{j=1}^{n} A_j(t)$ is the effective amplitude and $\Phi(t) = \frac{1}{n}\sum_{j=1}^{n} \phi_j(t)$ is the effective phase. After the n oscillators have reached fully synchronization, we can represent its behavior by Eq. (4.1.1), as if they were a single oscillator [Pikovsky et al., 2001]. Then, let us strongly couple one more oscillator to the averaged oscillator. By [Liu et al., 2022] we know that both chemical oscillators will synchronize. Thus the statement is valid for a chain of $n+1$ strongly coupled chemical oscillators. This proves the theorem. As a corollary, we have the following statement.

Corollary 4.1.2. *A chain of $n \geqslant 2$ strongly coupled chemical oscillators are more efficient at recognizing sufficiently complex computable languages than a chain of $n-1$ strongly coupled chemical oscillators.*

The proof is again by induction. The present thesis was devoted to show the base case. We know that two strongly coupled chemical oscillators are more efficient at recognizing sufficiently complex computational languages than one chemical oscillator. Let us assume the statement holds for a chain of n strongly coupled chemical oscillators. By theorem 4.1.1 we know that a chain of n strongly coupled chemical oscillators will synchronize. Let us represent the behavior of the chain of n strongly coupled chemical oscillators as a single oscillator by using Eq. (4.1.1). If we strongly couple a

new chemical oscillator to the averaged oscillator, from the present thesis we know that the array composed by the averaged oscillator and the new oscillator will be more efficient at recognizing sufficiently complex computational languages than the averaged oscillator per se. Thus the statement is valid for a chain of $n + 1$ strongly coupled chemical oscillators. This proves the corollary.

There are many interesting things about the above results. First of all, let us note that the theorem 4.1.1 is quite independent of topology, while the corollary 4.1.2 is only valid for chains (linear arrays) of strongly coupled chemical oscillators. It remains to explore alternative topologies that facilitate the transfer and handling of information. Despite the restriction, we can conjecture some ways to exploit this linear topology in order to achieve universal computation or solve NP-hard problems. Some of these proposals are described below.

4.1.3 What type of problems can we solve with a chemical computer?

So imagine that we have finally built a chemical computer composed of n chemical automata connected with a topology that allows to optimize its operation for a sufficiently complex language. What's next? What are the types of problems for which such a chemical computer will be more efficient than a conventional computer? As was well pointed out by [Grozinger et al., 2019], comparing unconventional computing architectures with traditional computers to solve tasks where the latter have been the

starting point (or inspiration) does not make much sense. Thus, chemical supremacy will be determined by the set of problems that chemical computers can practically solve while conventional microprocessor-based computers cannot.

Although Turing's model [Turing, 1936] provides a framework for answering fundamental questions about computation, *"as soon as one leaves the comfort provided by the abundance of mathematical machinery used to describe digital computation, the world seems to be packed with paradoxes"* [Horsman et al., 2017]. The nature of computation is not of purely theoretical interest. Mathematical models of computation and their properties inform the engineering of their physical manifestations [Grozinger et al., 2019]. There are multiple models of computation that are theoretically possible but have so far failed to be implemented. Since the chemical substrate is quite different from silicon, chemical computers offer us the opportunity to implement models of computation that have not yet been explored and could bring us advantages in terms of both computational recurrence and parallelism [Moore and Mertens, 2011].

Assuming a restricted topology as in the corollary 4.1.2, in the first instance it is possible to construct logic gates. Based on the fact that if we alter the chemistry of one of the chemical reactors it will modify its oscillations and by spontaneous synchronization this will spread along the chain, we can follow the train of thought proposed by [Adamatzky, 2004] and build logic gates based on the collision of wavefronts produced by the spontaneous synchronization of chemical oscillators along the chain. This approach has been shown to be capable of building combinatorial logic circuits [Costello and Adamatzky, 2005] and adaptive computation [Toth et al.,

2009].

If we define the power of a model of computation in terms of the range of computations that are possible to describe using that model, it is clear that combinatorial logic circuits are extremely weak, since the output of a circuit is purely a function of its current inputs (i.e., there is no memory). As was experimentally demonstrated by [Dueñas-Díez and Pérez-Mercader, 2019], chemical computation offers us the implementation of stateful computations, such that the output depends not only on the current input, but also on the current state, which encodes information about previous inputs that are no longer present.

Let us restrict again by the topology of 4.1.2. We know that the accepted words by a single chemical oscillator have the same free-energy difference between the oxidized and reduced states of the catalyst after the processing some input. There are many examples on how to use a single energetic variable to perform computation. If the single energy variable encodes the state of the system (e.g., binary states represented by high/low energy levels) we can represent signals, as in analog computation. If changes in the energetic variable drive transitions in the computational state space, then maybe we can perform some kind of thermodynamic computation. Here it is important to remember that both combinatorial logic circuits and Turing machines are deterministic models of computation. However, deterministic computation can be generalized to include stochastic and non-deterministic computation.

Nowadays probabilistic algorithms have become the cornerstone of computer science,

as they are able to efficiently approximate solutions to difficult optimization problems for which a deterministic solution is intractable [Motwani, 1995]. In general, stochastic (or probabilistic, or randomised) computation refers to algorithms that can make random choices during their execution. The results shown in the present thesis respect to the addition of stochastic noise in chemical automata point to the fact that chemical circuits could be viewed as stochastic processors that provide the substrate for implementing probabilistic algorithms [Alaghi and Hayes, 2015]. By generalizing the idea of a two-reactor chemical circuit to n reactors we can also glimpse the implementation of distributed computing, where concurrent parts of an algorithm may be executed in parallel, potentially speeding up computations significantly [Grozinger et al., 2019].

To conclude this subsection, let us reflect a little on the possibility of solving problems belonging to different complexity classes. By complexity class we mean a set of problems with related computational complexity, typically defined by a model of computation (like a Turing machine), the type of problem (decision, optimization, etc.), and a resource bound (like polynomial time or logarithmic space) [Johnson, 1990].

We have known for a decade that it is possible to reformulate a considerable number of NP-complete and NP-hard problems using Ising-type systems [Lucas, 2014]. This led to the development of Ising machines, computing devices capable of solving complex problems by mimicking how atoms in magnets and spin glasses arrange themselves to reach a low energy state [Mohseni et al., 2022]. While it is theoretically possible to

reformulate NP-hard problems in a one-dimensional Ising system (as the one obtained by the topological constraint of corollary 4.1.2), the expressiveness of such a system is limited by its locality. Practical implementations often require auxiliary spins, modified interactions, or approximations to overcome these limitations [Lucas, 2014].

Usually Ising machines are built using quantum devices [Kalinin and Berloff, 2022]. However, the bistable, reactive and computational nature of the BZ reaction may well be a candidate for building scalable Ising machines that will eventually allow us to achieve computational supremacy over the classical von Neumann architecture. Although this may seem like elaborate speculation, the theoretical results shown in [Abrahão et al., 2020] suggest that it is possible to go beyond the Turing computational barrier using distributed systems, such as the one that could be constructed by means of chemical circuits.

4.2 Metabolic Biocomputing

From a metabolic biocomputing perspective, our results indicate that primordial metabolic systems may have exploited multimetabolism as an evolutionary strategy to survive in a changing environment. To understand the last in detail, let us imagine that each chemical oscillator represents a primordial metabolic structure in a given protobiosphere. Such an assumption is not at all far-fetched, for as mentioned above, the BZ reaction was developed with the aim of finding a biochemistry-free alternative version of the Krebs cycle. As was recently well pointed out in [Jaeger et al., 2024],

one of the dialectical processes that defined life from its earliest forms was anticipation. This implies that all organisms, at all evolutionary scales, are anticipatory agents.

Let us now focus our attention on the time series. In this context, we can identify it as a signal coming from the environment, which contains relevant information that allows the system to survive. Thus, when systems composed of chemical oscillators forecast such a time series, this can be interpreted as a kind of inferential process that allows the system to adapt and maintain itself in a homeostatic state. Therefore, our results show that for complex environmental signals coupled metabolic architectures are able to adapt faster than isolated metabolic architectures. This is consistent with the results shown in [Cervera et al., 2019], which point to the use of bioelectrical synchronization as a potential memory mechanism.

It is important to note that the method proposed in the present thesis conforms a methodology that could well be exploited for the study of metabolic biocomputing. Given the flexibility of reservoir computation in terms of substrate and the large number of models that exist in the literature to represent chemical reaction networks [Müller et al., 2022], it is entirely possible to use much more complex chemical reactions to explore its computational capabilities. An excellent example of the above is [Baltussen et al., 2024], where it has recently been studied a chemical reservoir computer based on the formose reaction. The authors showed how this complex, self-organizing chemical reaction network can perform several nonlinear classification tasks in parallel, predict the dynamics of other complex systems and achieve time-series forecasting. Again, this leaves the door open for the characterization of

metabolic biocomputing.

4.3 Resource-bounded Algorithmic Networks

As mentioned in Subsection 1.3.2, the theoretical results concerning the paradigm of algorithmic networks focus on a case that is impossible to materialize. The results of the present thesis show that, even in the resource-bounded case and with external noise, the computational capacity of two strongly connected automata outperforms that of a single automaton. Moreover, in the deterministic case (used canonically in algorithmic networks) it is possible to detect an equivalence between two decoupled oscillators and two weakly coupled oscillators. This is fully consistent with the usual methodology in algorithmic networks [Abrahao et al., 2019], where adaptive behavior respects a dichotomy in the connection of the components.

However, this is not true for the stochastic case, and as far as I know, no algorithmic networks with external noise have been studied. Thus, the effect of different degrees of coupling in algorithmic networks without resource limitation remains to be explored. In this way we have shown that the theory of algorithmic networks is also applicable in the resource-bounded case and at the same time we have glimpsed a kind of extension of this paradigm, leaving open questions that are fundamental for the development of the same. The generalization of our methodology to n chemical automata will allow further extrapolation of the theoretical results obtained so far [Abrahão and Zenil, 2022], aiming at the use of chemical circuits as a minimal model of structural

emergence capable of being built in a laboratory.

Chapter 5

Conclusions

In [Dueñas-Díez and Pérez-Mercader, 2019] the computational capabilities of a single chemical oscillator were studied, concluding that there is an equivalence between the BZ reaction and a linear bounded automaton. The present thesis represents the first study regarding the computational capabilities of two coupled chemical oscillators. Given that all words accepted by the chemical Turing machine have the same value of the free-energy difference between the oxidized and reduced states of the catalyst after the processing of input [Dueñas-Díez and Pérez-Mercader, 2019], we can transform the problem of accepting/rejecting an input into a minimization problem. By associating the ability to process a computable language with the difficulty of forecasting a time series, reservoir computing paradigm is perfect for studying the computational capabilities of the different chemical oscillator configurations tested in [Liu et al., 2022].

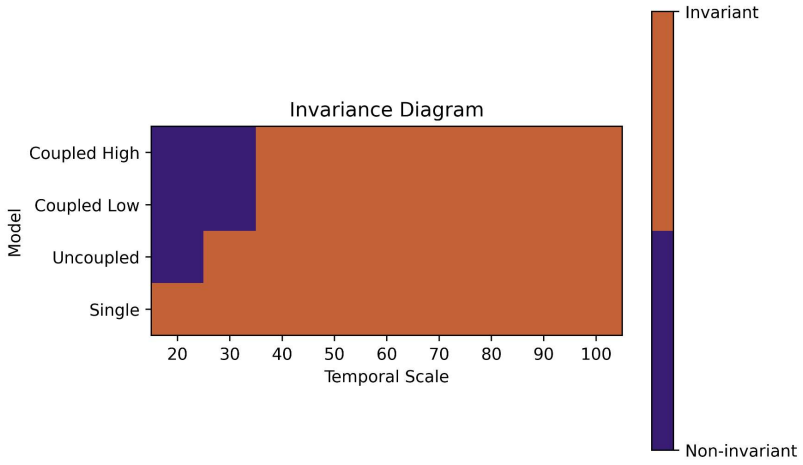
The results of this thesis specifically show that two chemical oscillators will be always better than one single chemical oscillator when forecasting complex time series, which means that in order to efficiently process sufficiently complex computable languages it is necessary to use two chemical automata. From the above, together with the experimental results shown in [Liu et al., 2022], it follows that a chain of $n \geqslant 2$ strongly coupled chemical automata is more efficient (in recognizing sufficiently complex computable languages) than a chain of $n-1$ strongly coupled chemical automata. This fact is proved by means of corollary 4.1.2. I then proposed different ways in which information could be manipulated within such unconventional architectures, considering both the topological constraint imposed by corollary 4.1.2 as well as more arbitrary topologies. In addition, the implications of the present thesis on problems such as the search for cell supremacy in biocomputing and the exploration of algorithmic networks in the resource-limited case were also inspected.

The computational capability of chemical automata arrays with a more arbitrary architecture remains to be studied. As such the model used to characterize the chemical oscillators is simple to extend for $n > 2$ oscillators. However, since it is based on experimental results [Liu et al., 2022], the extension of the parameters must be done with caution. Now that the feasibility of constructing networks of automata with arbitrary topology has been realized, the ability to manipulate information in such architectures remains to be examined, as well as the generalization of the results found here with respect to the efficient processing of sufficiently complex computational languages. This will help us to understand the advantages and limitations of building chemical circuits, giving us a better understanding of the operation, computational

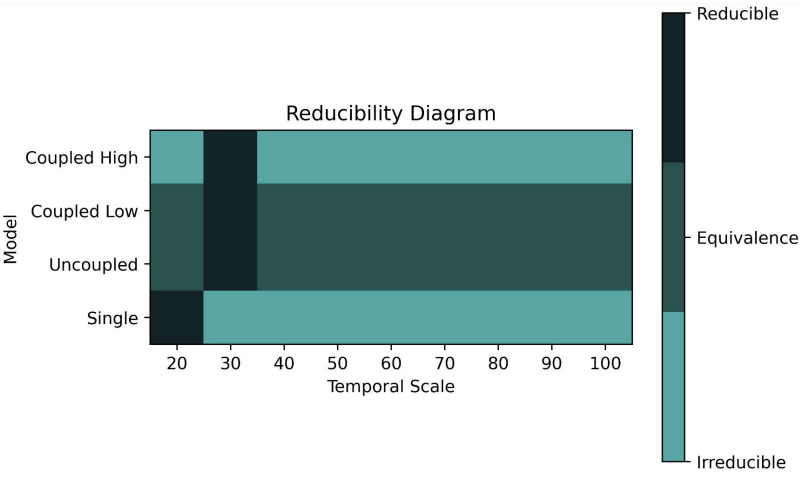
capacity and possible architectures that a chemical computer can have.

Additionally, it is clear that our methodology revolves around reservoir computation (RC). Despite that RC requires very small training data sets, uses linear optimization, and thus requires low training cost, it has been shown that has some disadvantages as any other machine learning approach. Fortunately, there are recent research lines to rectify such nuances. For instance, to use less metaparatemers we could use nonlinear vector autoregression, which has been shown to be equivalent to RC [Gauthier et al., 2021]. Similarly, RC could be very dependent of the warm-up time when training, but recently it has been proposed a next-generation reservoir computing to overcome this difficulty [Zhang and Cornelius, 2023]. By preserving the RC architecture we rescue certain advantages, because being so versatile we can test different chemical reactions, using different time series or even testing other computational tasks, as shown in [Baltussen et al., 2024].

Appendix: Deterministic Case

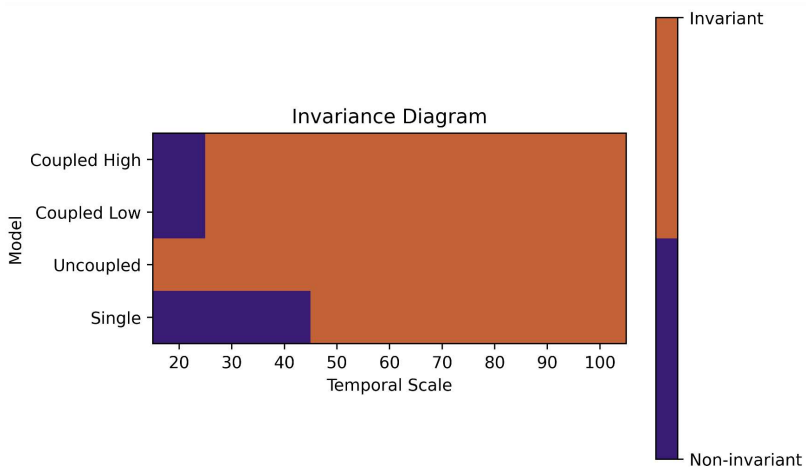


(a)

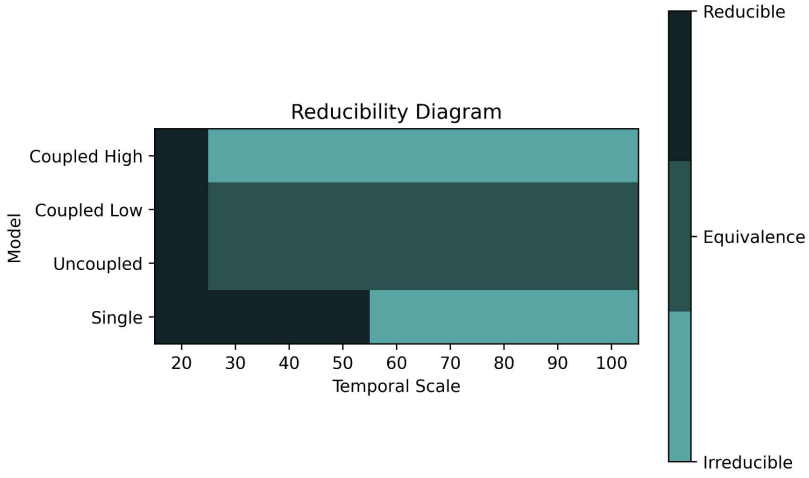


(b)

Figure 5.1: Diagrams obtained for $\lambda = 0.05$.

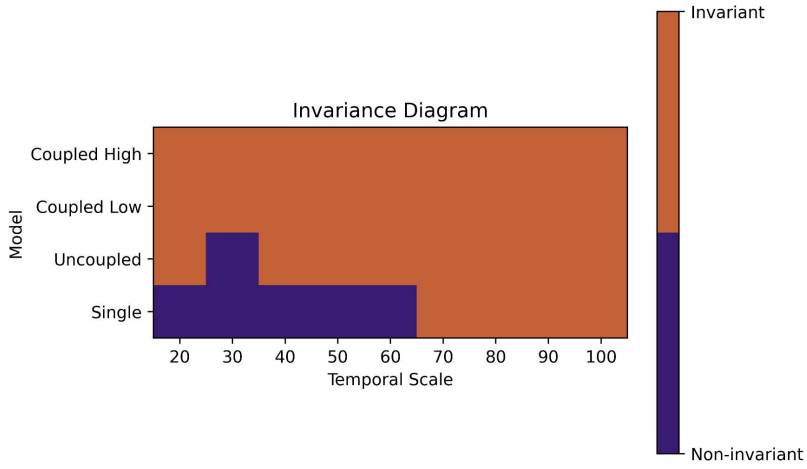


(a)

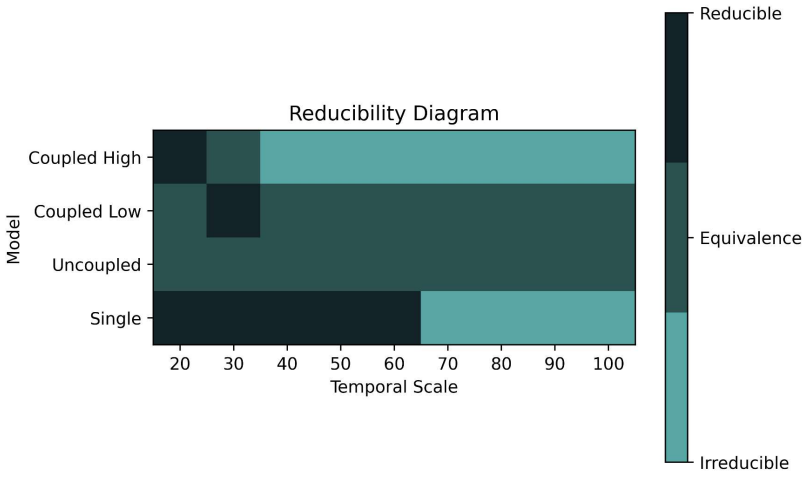


(b)

Figure 5.2: Diagrams obtained for $\lambda = 1$.



(a)



(b)

Figure 5.3: Diagrams obtained for $\lambda = 1.5$.

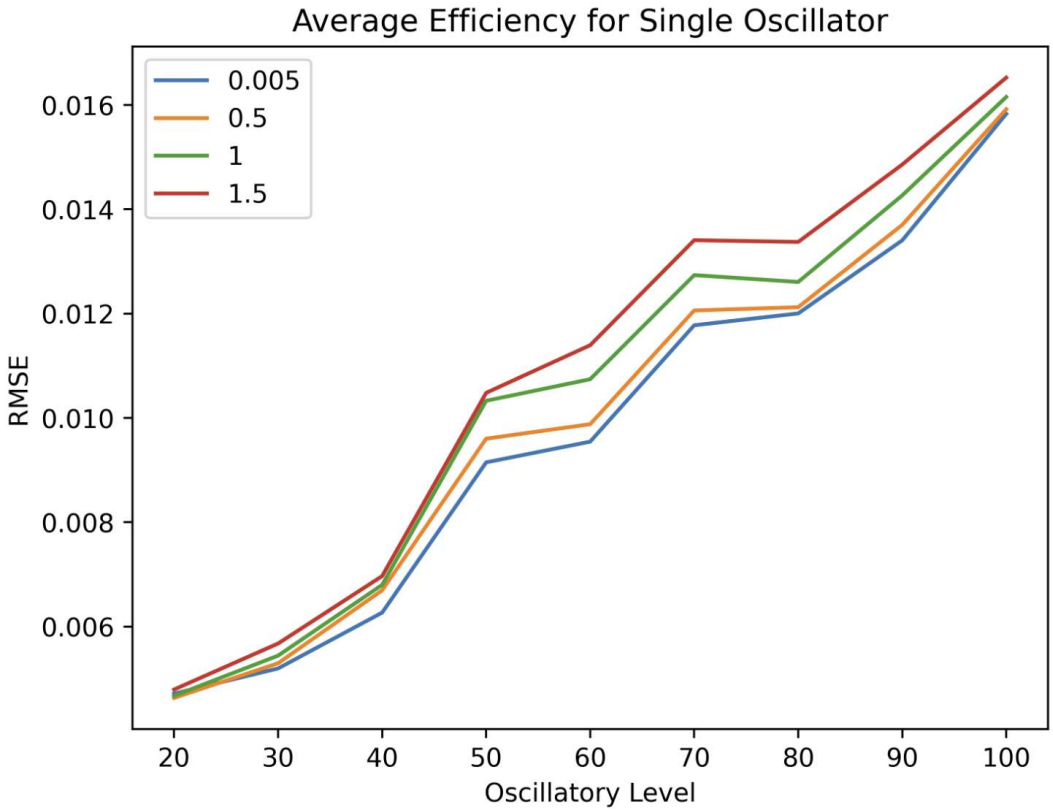


Figure 5.4: Average RMSE obtained for the single automaton reservoir using different values $\lambda > 0$.

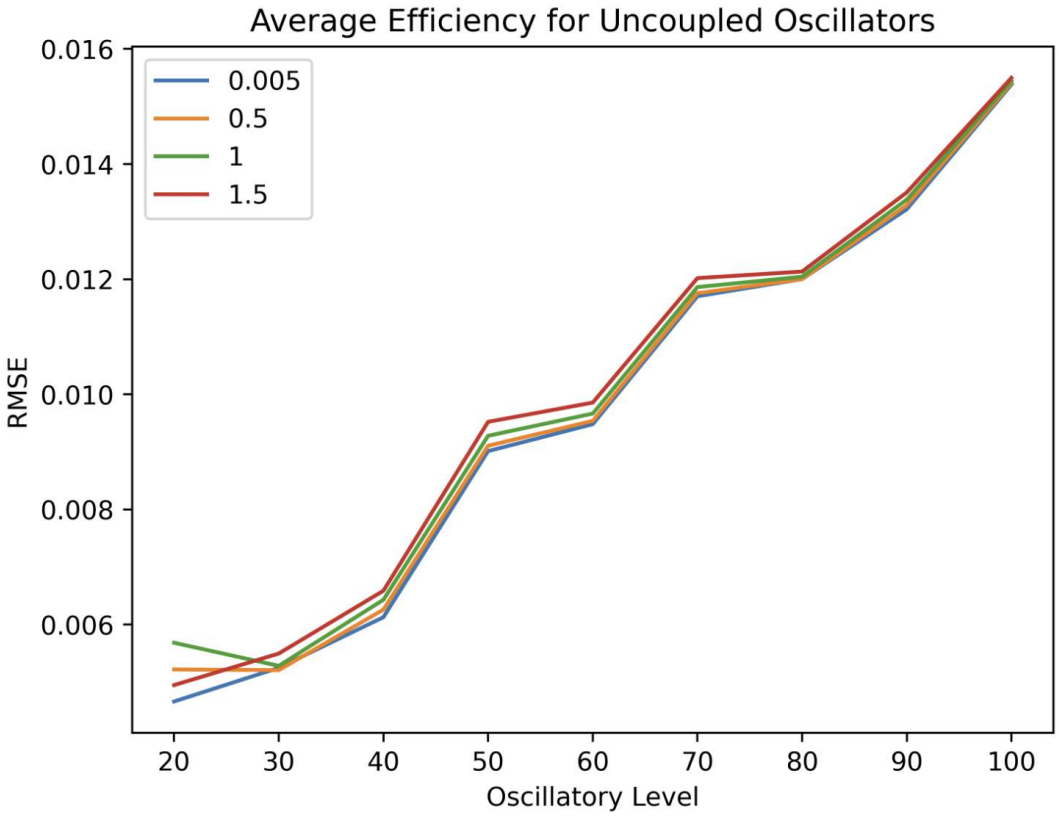


Figure 5.5: Average RMSE obtained for the two uncoupled automata reservoir using different values $\lambda > 0$.

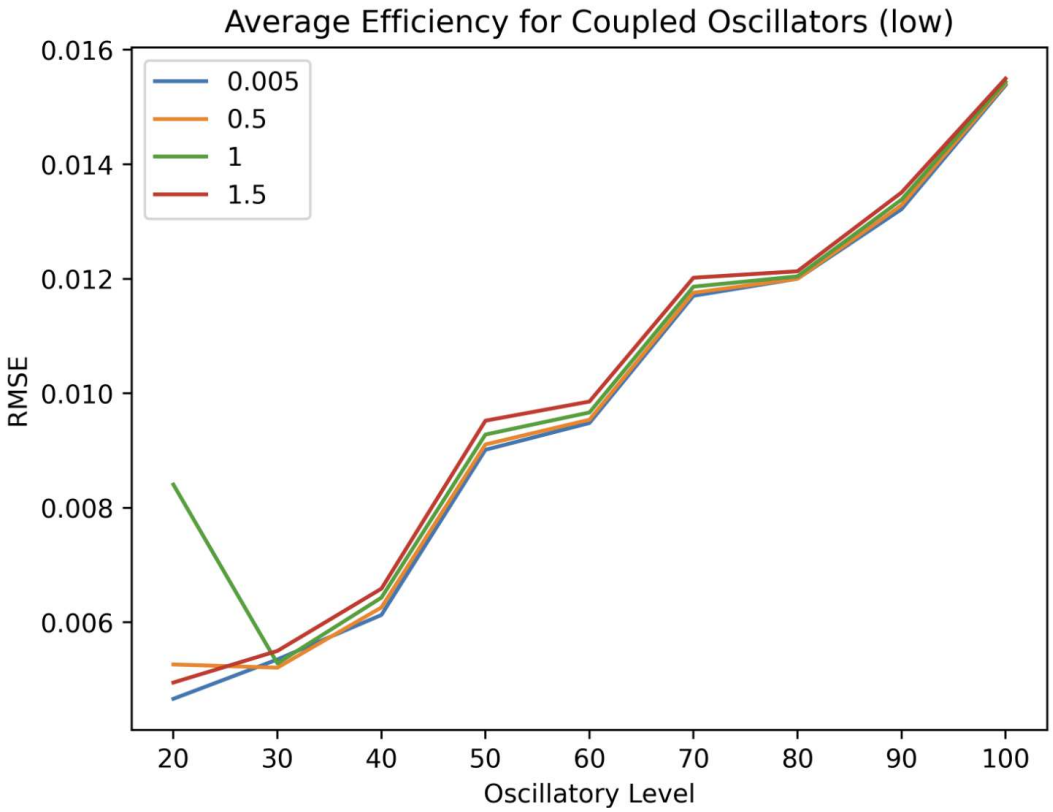


Figure 5.6: Average RMSE obtained for the two coupled low automata reservoir using different values $\lambda > 0$.

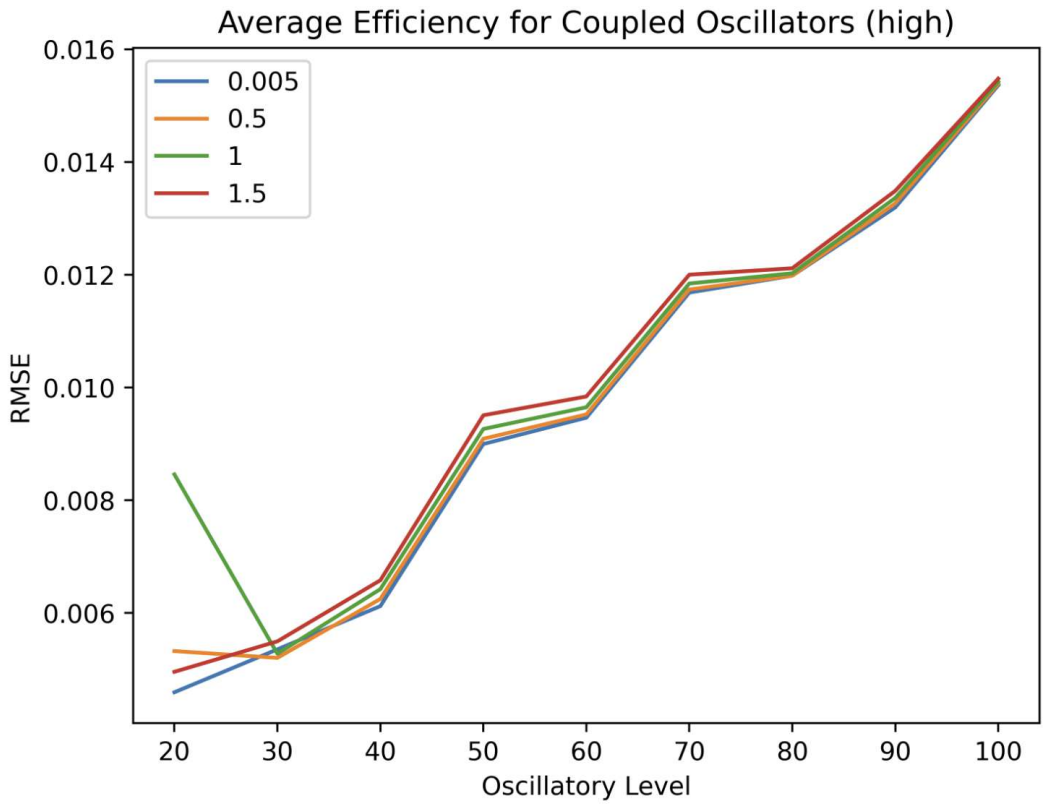


Figure 5.7: Average RMSE obtained for the two coupled high automata reservoir using different values $\lambda > 0$.

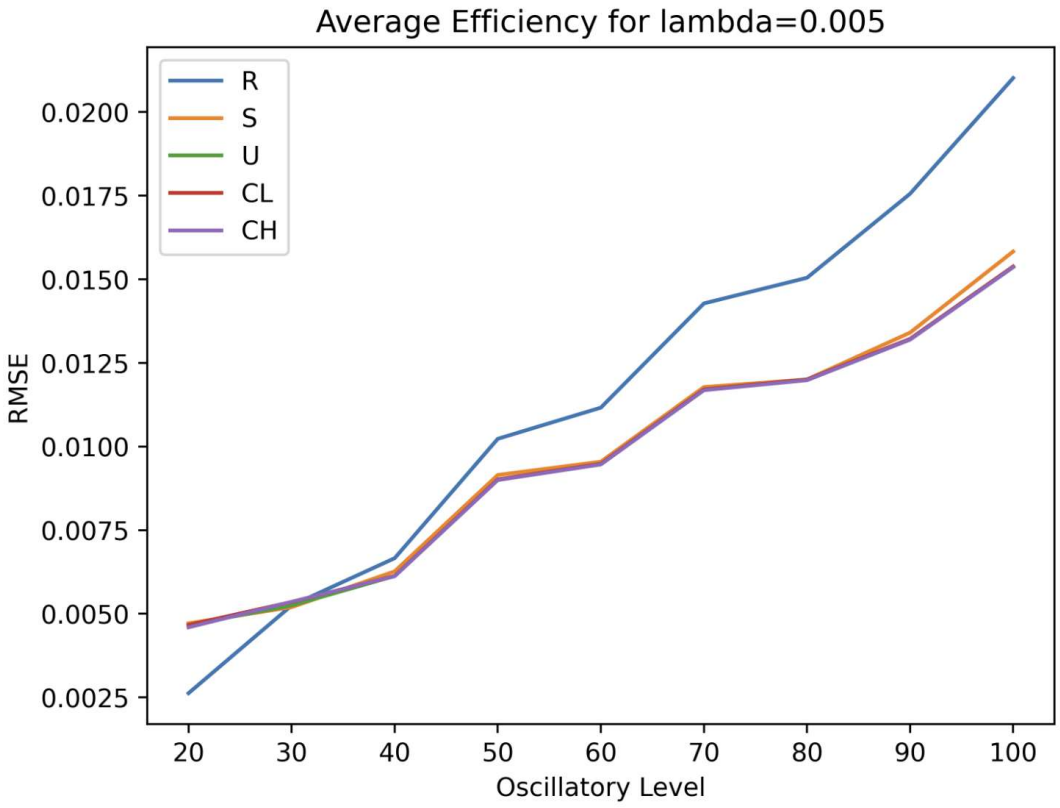


Figure 5.8: Average RMSE obtained for the different reservoirs using $\lambda = 0.05$. R stands for Ridge Regression, S for Single Oscillator, U for Uncoupled Oscillators, CL for Coupled Low Oscillators and CH for Coupled High Oscillators.

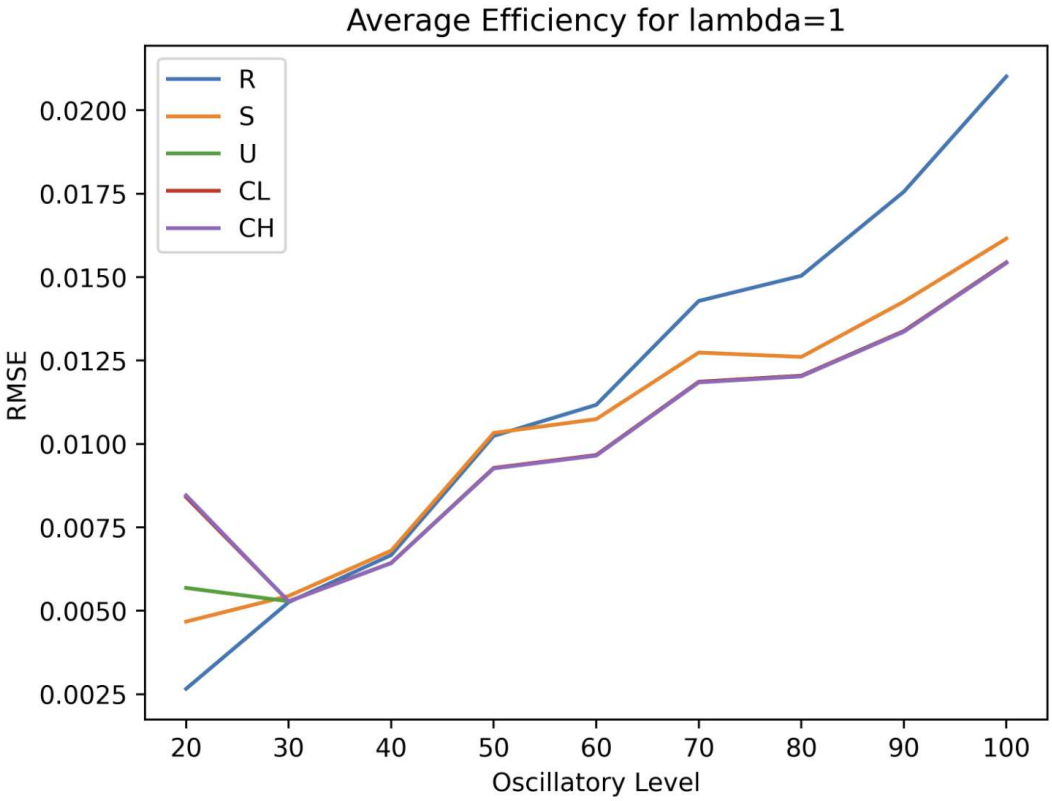


Figure 5.9: Average RMSE obtained for the different reservoirs using $\lambda = 1$. R stands for Ridge Regression, S for Single Oscillator, U for Uncoupled Oscillators, CL for Coupled Low Oscillators and CH for Coupled High Oscillators.

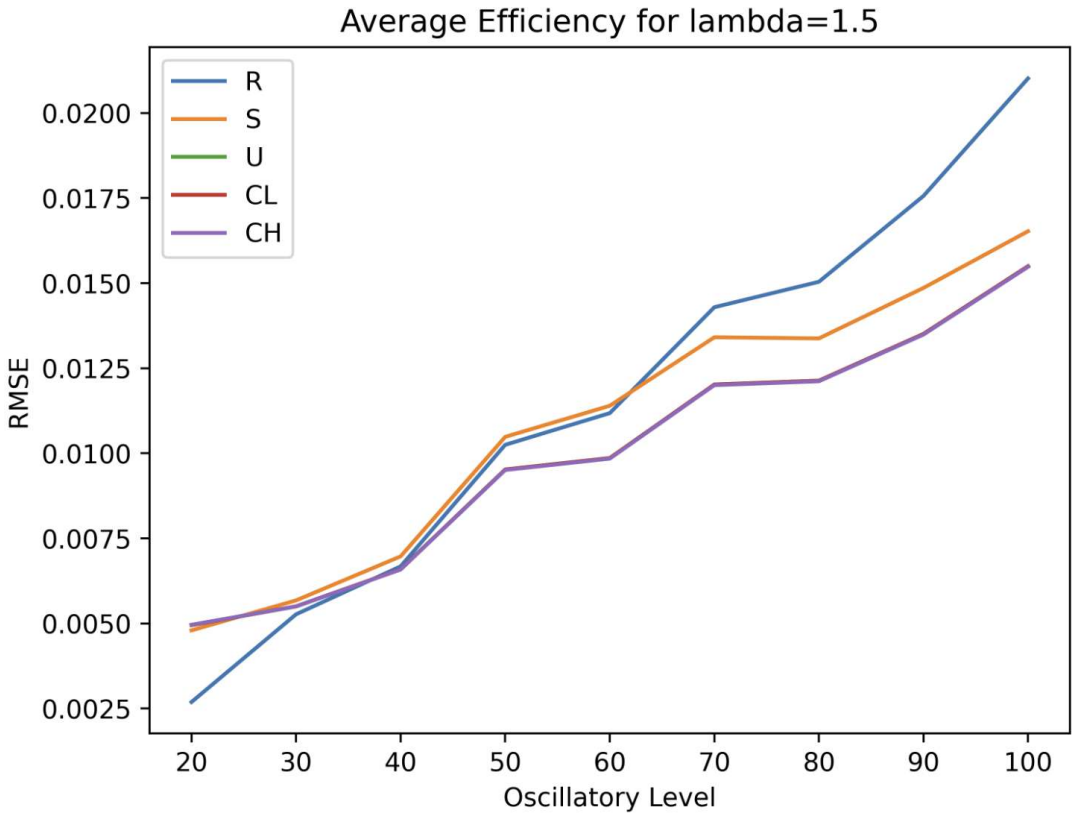
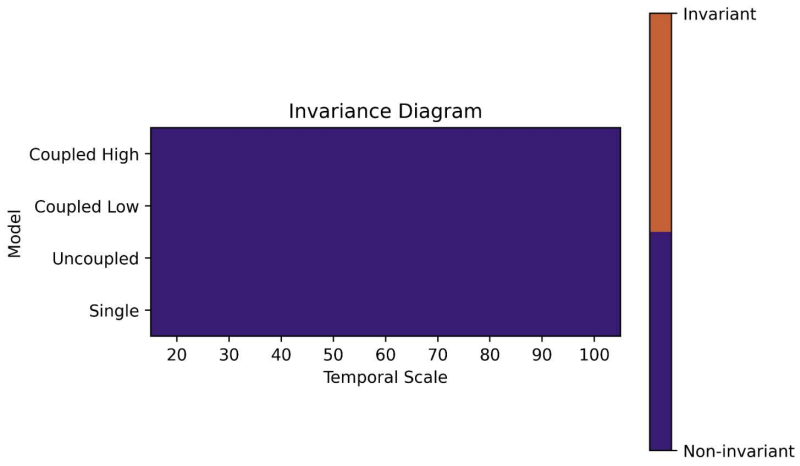


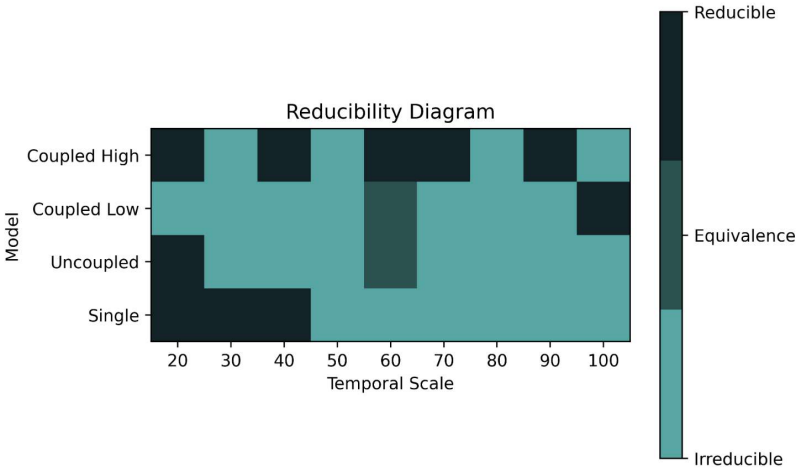
Figure 5.10: Average RMSE obtained for the different reservoirs using $\lambda = 1.5$. R stands for Ridge Regression, S for Single Oscillator, U for Uncoupled Oscillators, CL for Coupled Low Oscillators and CH for Coupled High Oscillators.

Appendix:

Stochastic Case | White Noise

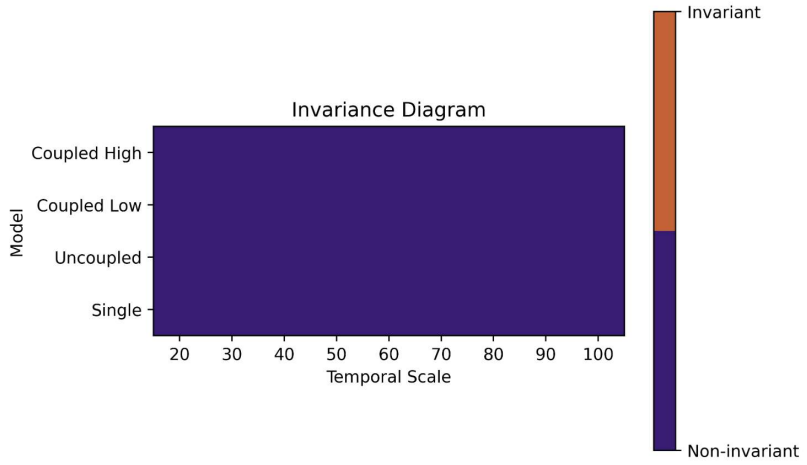


(a)

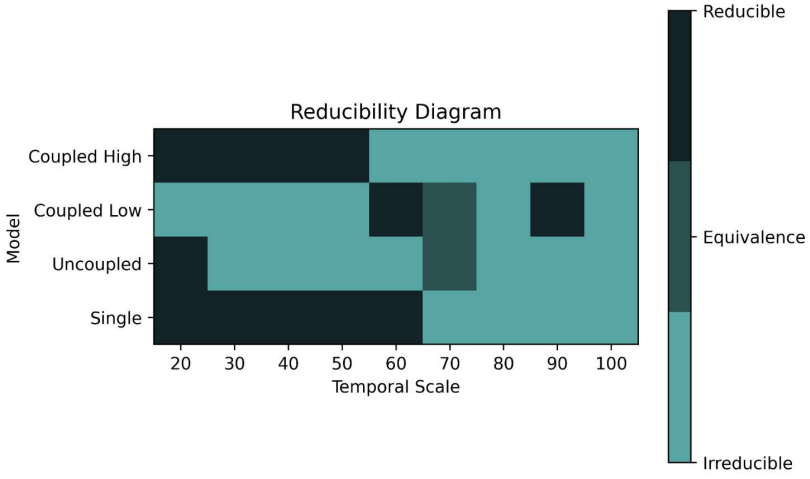


(b)

Figure 5.11: Diagrams obtained for $\lambda = 0.05$ using additive white noise. In both the x axis represents the different oscillatory levels described in Sections 2.3.1.

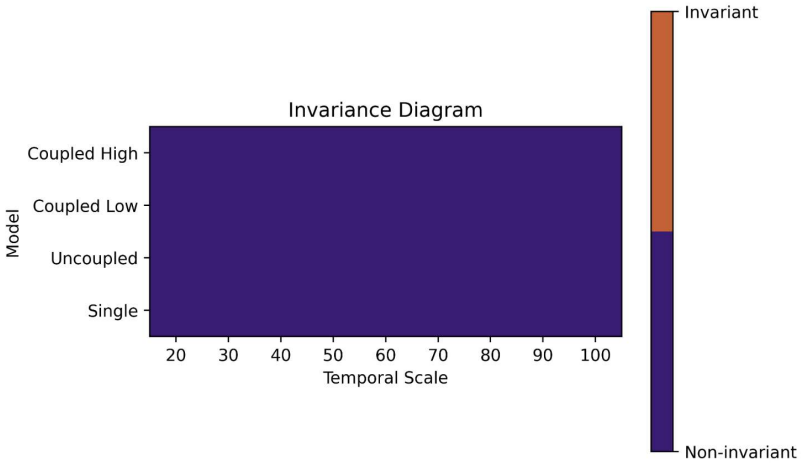


(a)

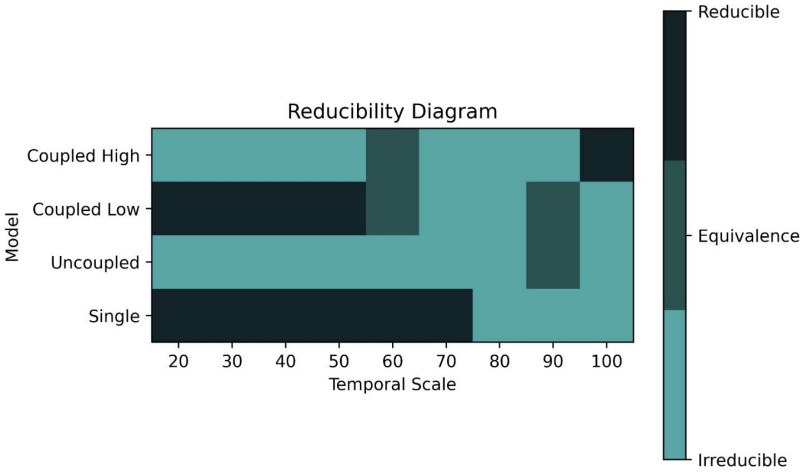


(b)

Figure 5.12: Diagrams obtained for $\lambda = 1$ using additive white noise. In both the x axis represents the different oscillatory levels described in Sections 2.3.1.



(a)



(b)

Figure 5.13: Diagrams obtained for $\lambda = 1.5$ using additive white noise. In both the x axis represents the different oscillatory levels described in Sections 2.3.1.

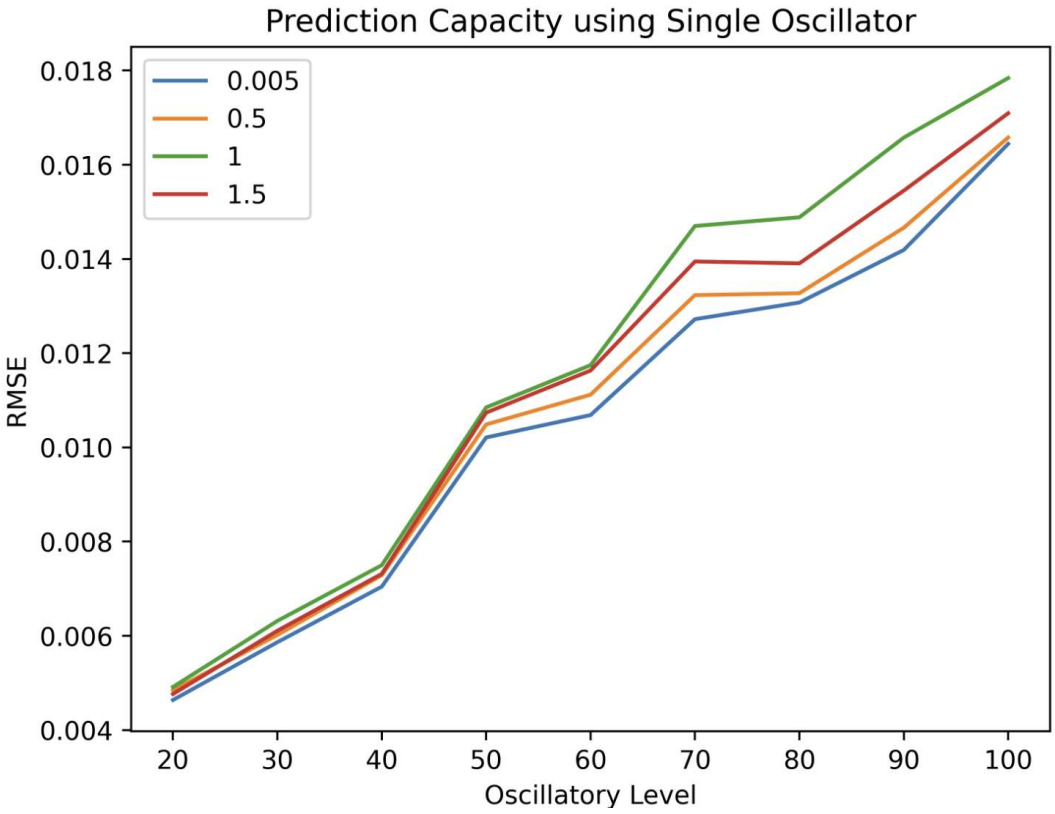


Figure 5.14: Average RMSE obtained for the single automaton reservoir using different values $\lambda > 0$ and additive white noise.

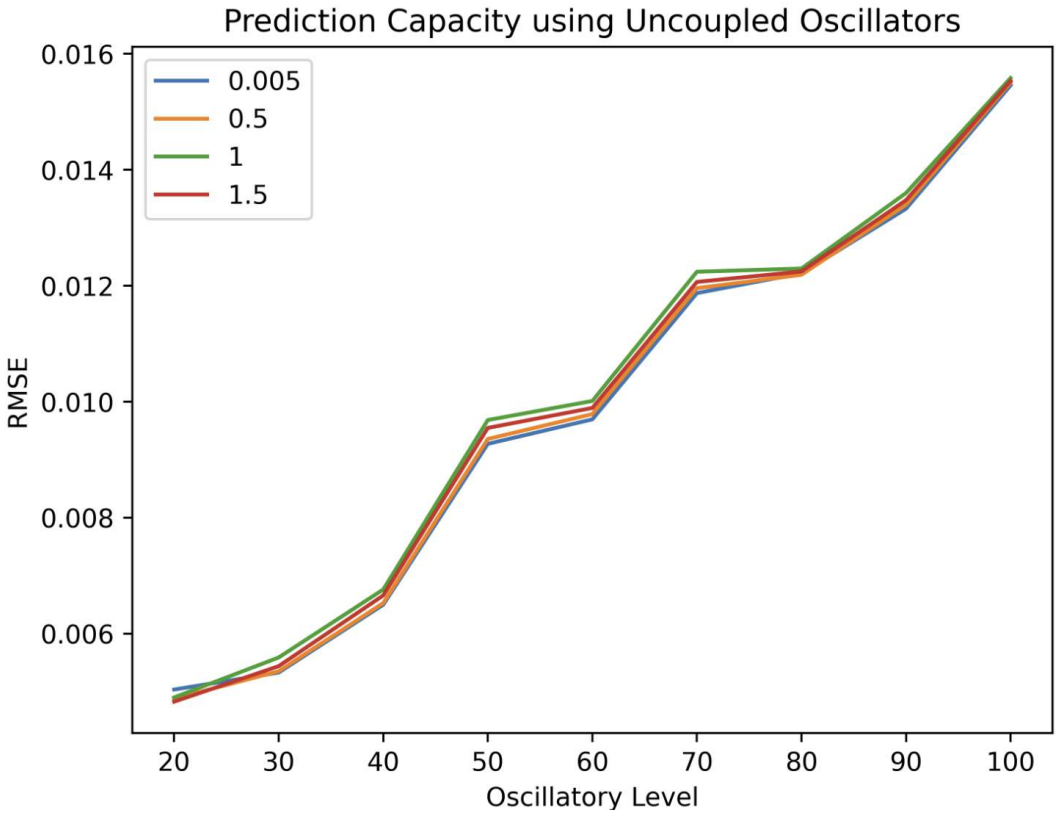


Figure 5.15: Average RMSE obtained for the uncoupled automata reservoir using different values $\lambda > 0$ and additive white noise.

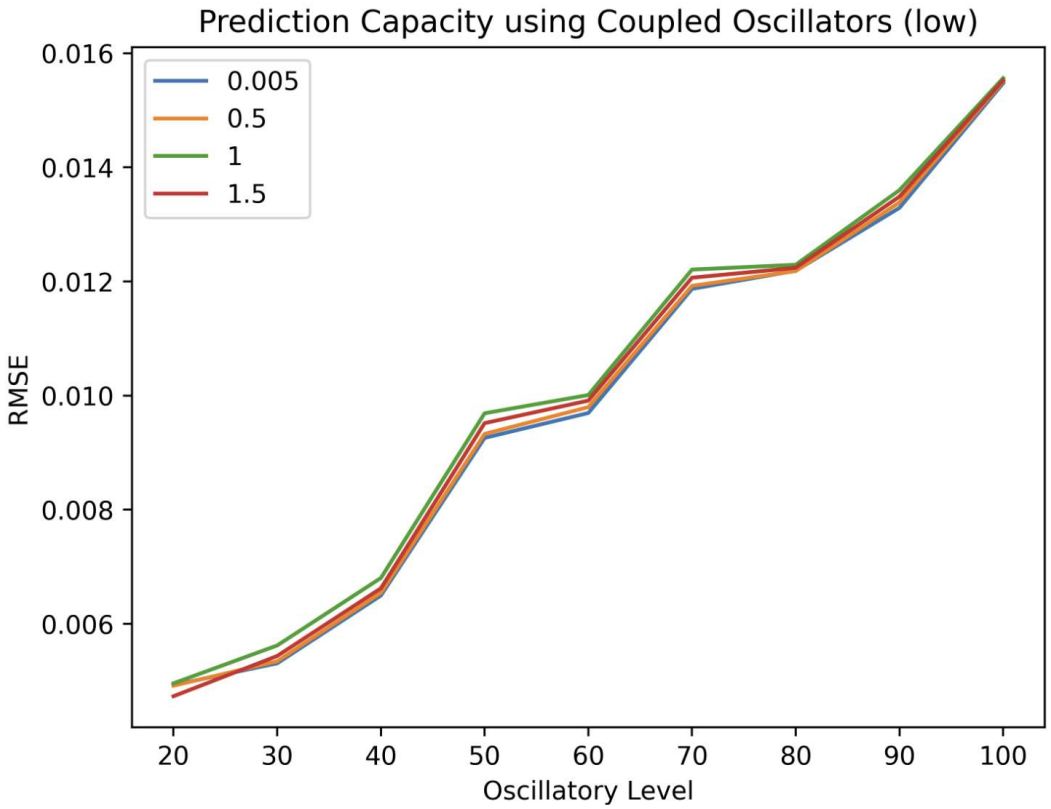


Figure 5.16: Average RMSE obtained for the coupled low automata reservoir using different values $\lambda > 0$ and additive white noise.

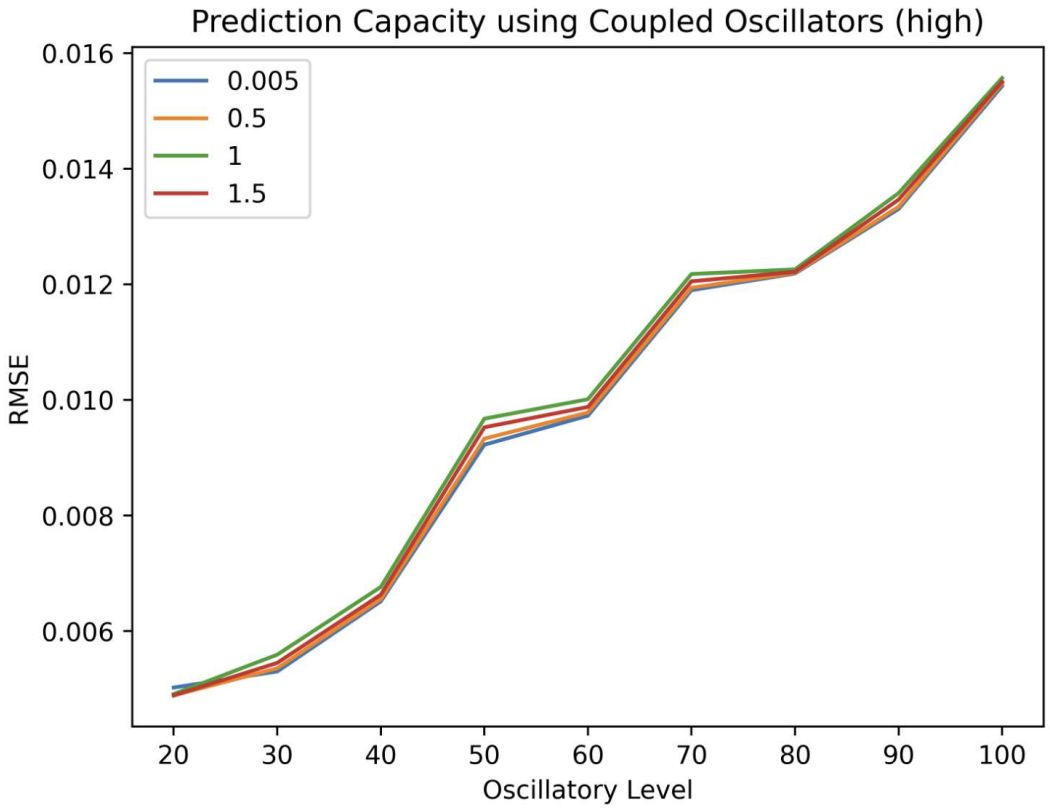


Figure 5.17: Average RMSE obtained for the coupled high automata reservoir using different values $\lambda > 0$ and additive white noise.

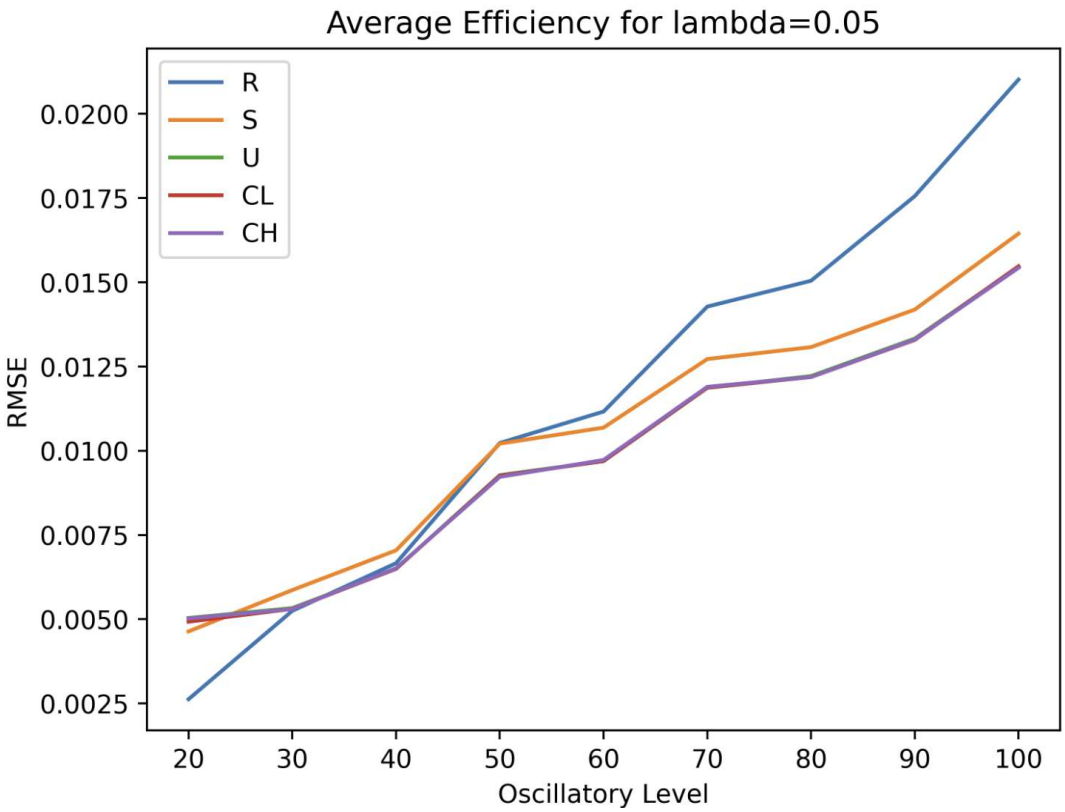


Figure 5.18: Average RMSE obtained for the different reservoirs using $\lambda = 0.05$. R stands for Ridge Regression, S for Single Oscillator, U for Uncoupled Oscillators, CL for Coupled Low Oscillators and CH for Coupled High Oscillators.

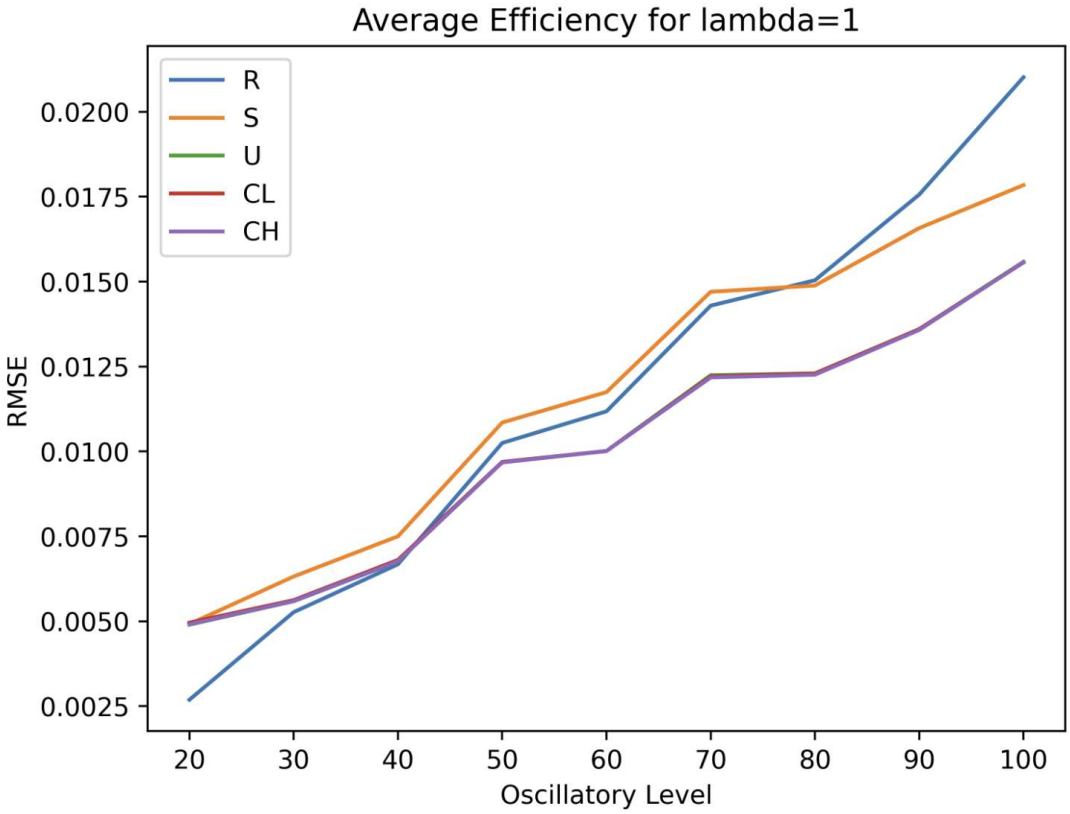


Figure 5.19: Average RMSE obtained for the different reservoirs using $\lambda = 1$. R stands for Ridge Regression, S for Single Oscillator, U for Uncoupled Oscillators, CL for Coupled Low Oscillators and CH for Coupled High Oscillators.

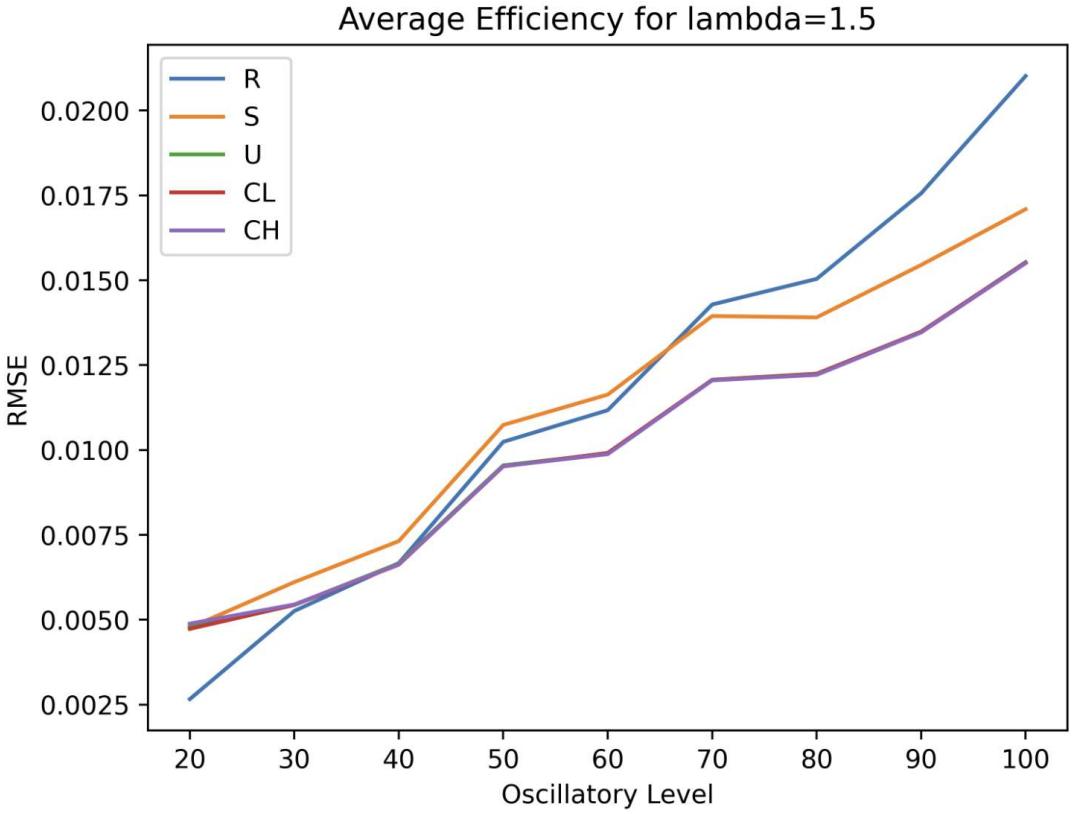
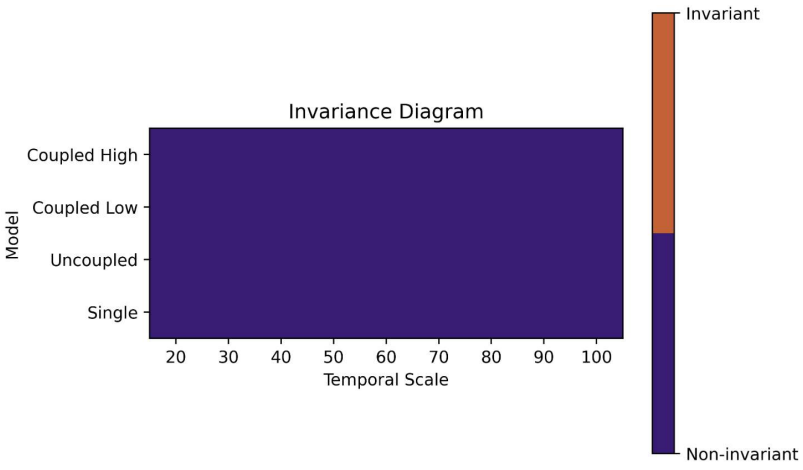


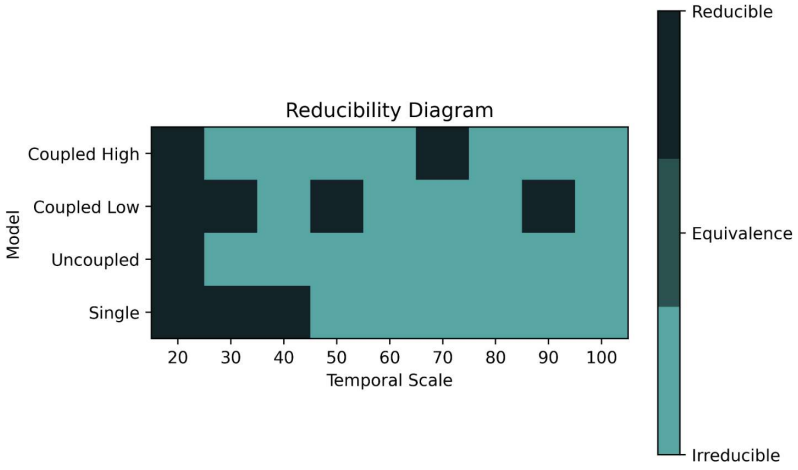
Figure 5.20: Average RMSE obtained for the different reservoirs using $\lambda = 1.5$. R stands for Ridge Regression, S for Single Oscillator, U for Uncoupled Oscillators, CL for Coupled Low Oscillators and CH for Coupled High Oscillators.

Appendix:

Stochastic Case | Pink Noise

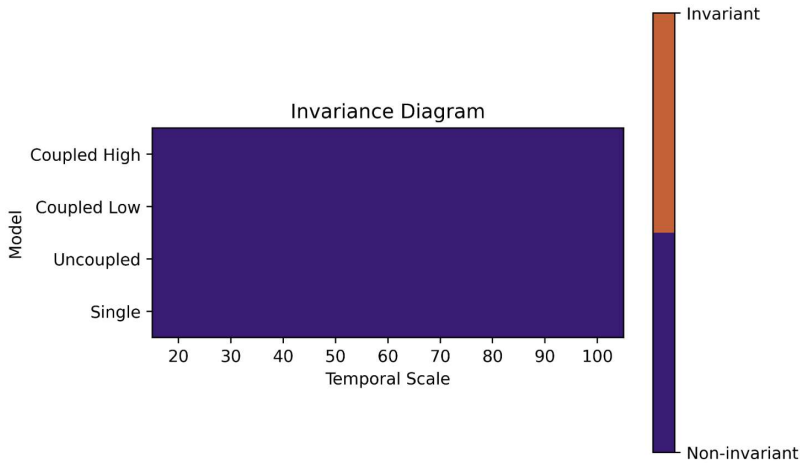


(a)

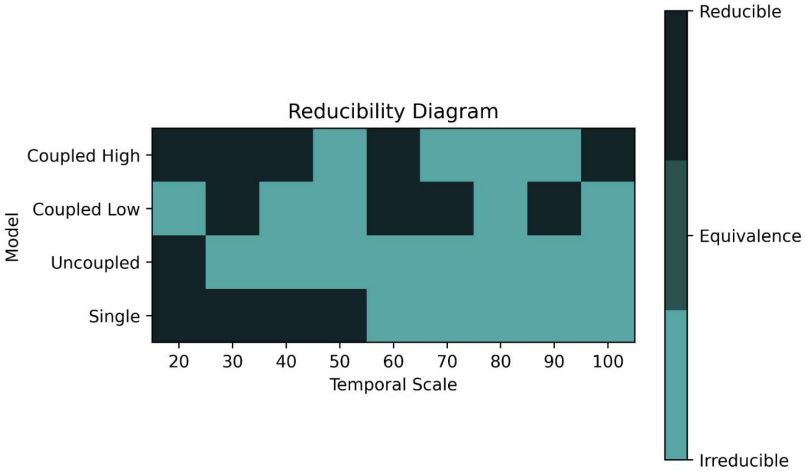


(b)

Figure 5.21: Diagrams obtained for $\lambda = 0.05$ using additive pink noise. In both the x axis represents the different oscillatory levels described in Sections 2.3.1.

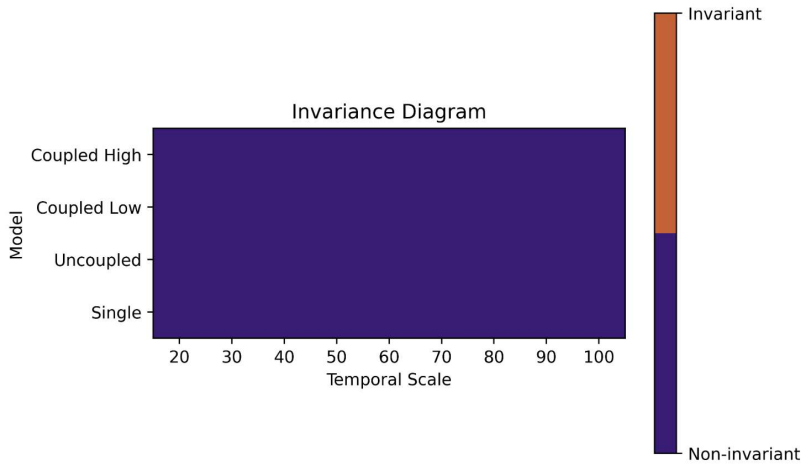


(a)

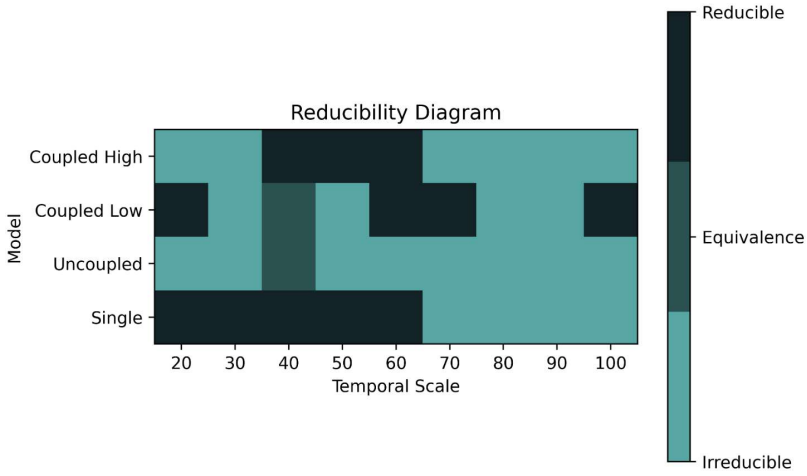


(b)

Figure 5.22: Diagrams obtained for $\lambda = 0.5$ using additive pink noise. In both the x axis represents the different oscillatory levels described in Sections 2.3.1.

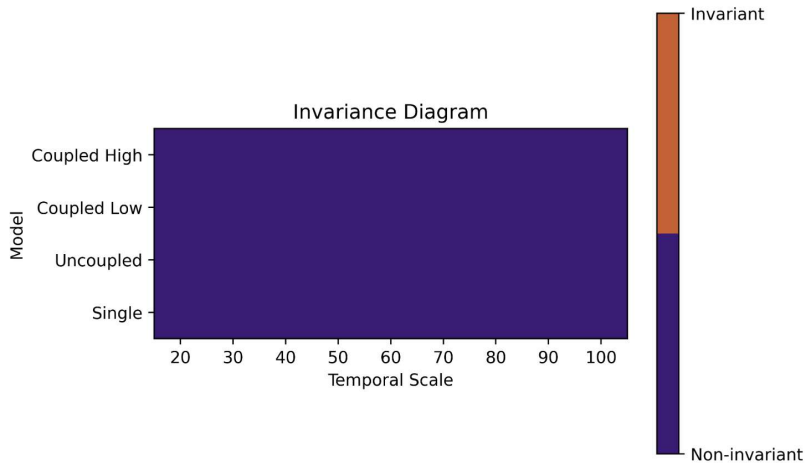


(a)

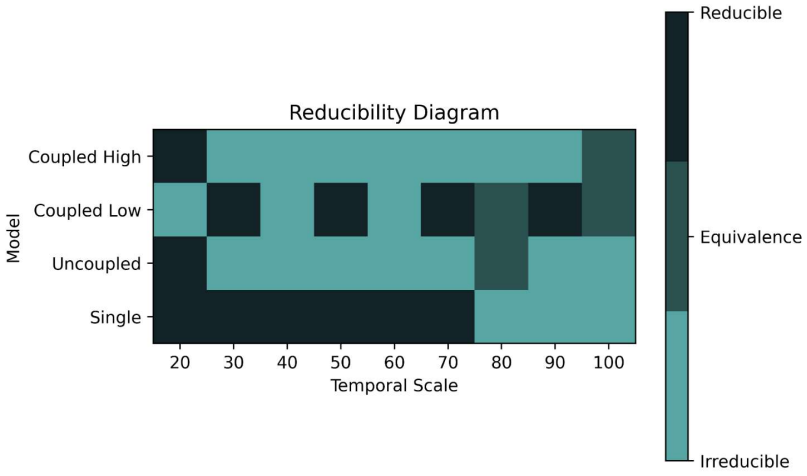


(b)

Figure 5.23: Diagrams obtained for $\lambda = 1$ using additive pink noise. In both the x axis represents the different oscillatory levels described in Sections 2.3.1.



(a)



(b)

Figure 5.24: Diagrams obtained for $\lambda = 1.5$ using additive pink noise. In both the x axis represents the different oscillatory levels described in Sections 2.3.1.

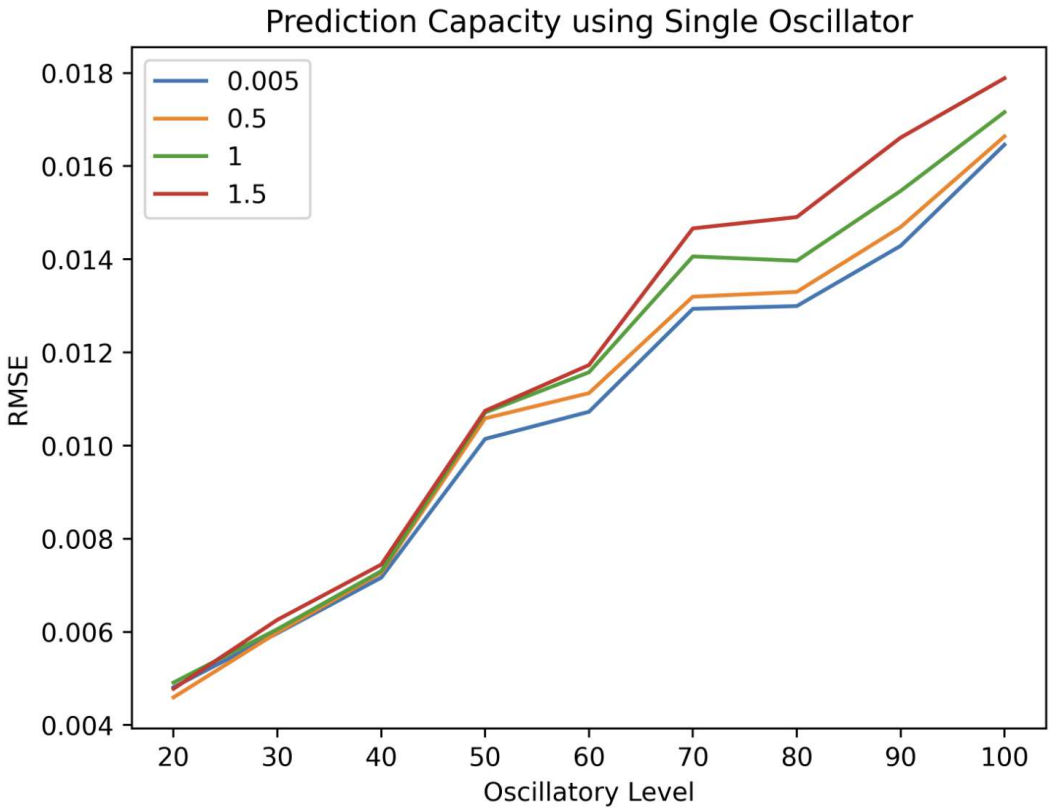


Figure 5.25: Average RMSE obtained for the single automaton reservoir using different values $\lambda > 0$ and additive pink noise.

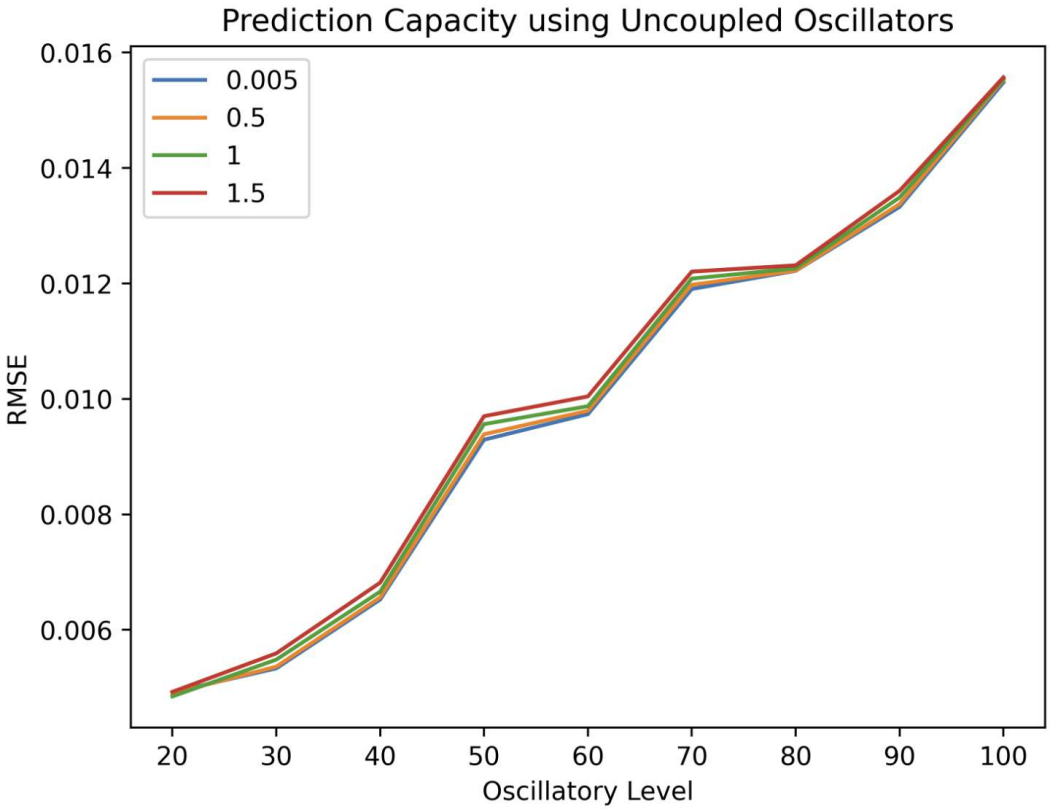


Figure 5.26: Average RMSE obtained for the uncoupled automata reservoir using different values $\lambda > 0$ and additive pink noise.

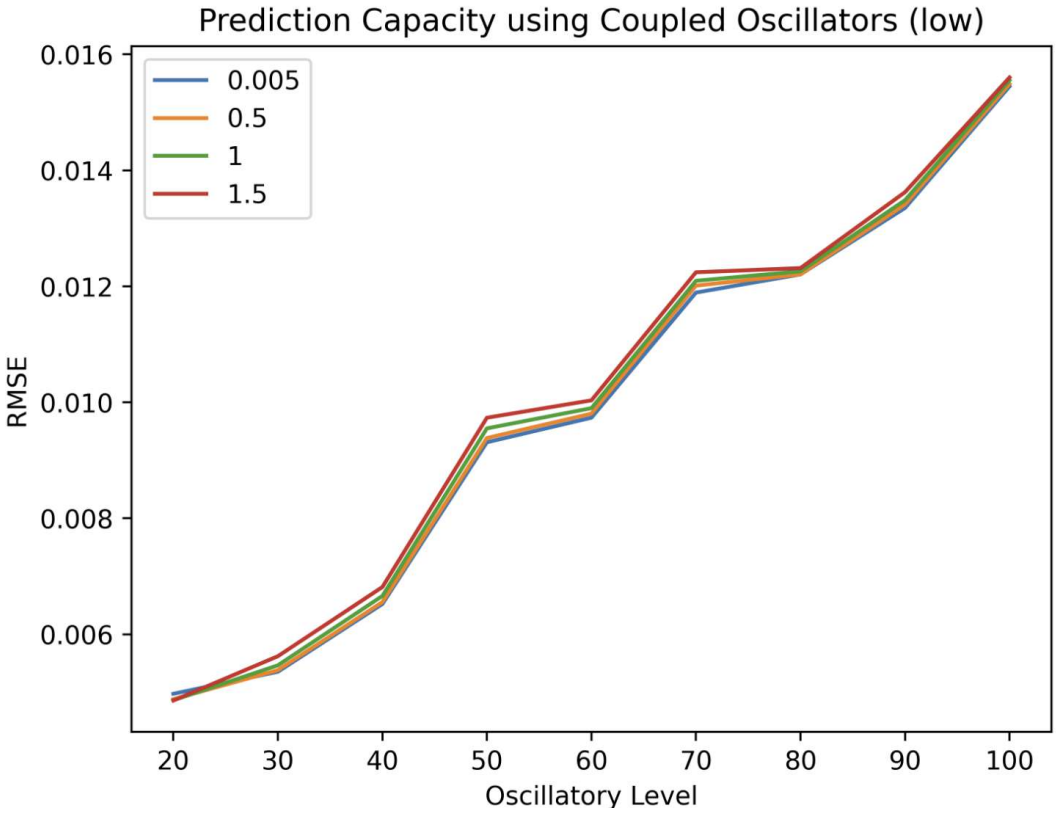


Figure 5.27: Average RMSE obtained for the coupled low automata reservoir using different values $\lambda > 0$ and additive pink noise.

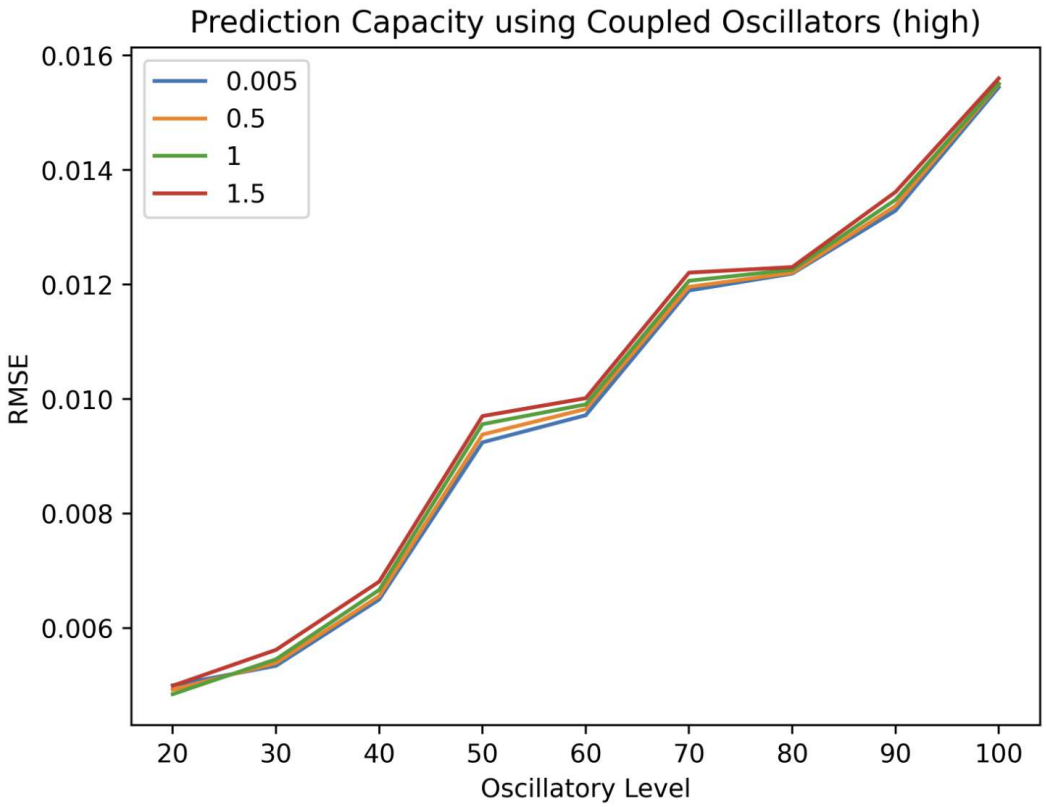


Figure 5.28: Average RMSE obtained for the coupled high automata reservoir using different values $\lambda > 0$ and additive pink noise.

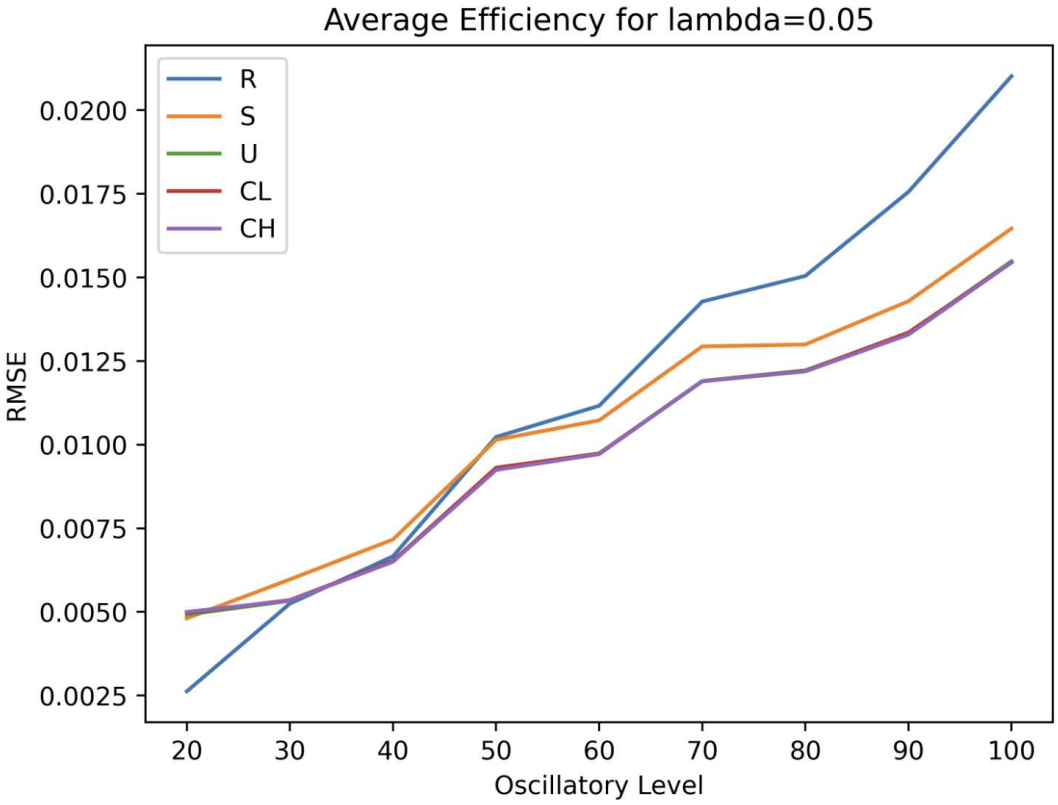


Figure 5.29: Average RMSE obtained for the different reservoirs using $\lambda = 0.05$. R stands for Ridge Regression, S for Single Oscillator, U for Uncoupled Oscillators, CL for Coupled Low Oscillators and CH for Coupled High Oscillators.

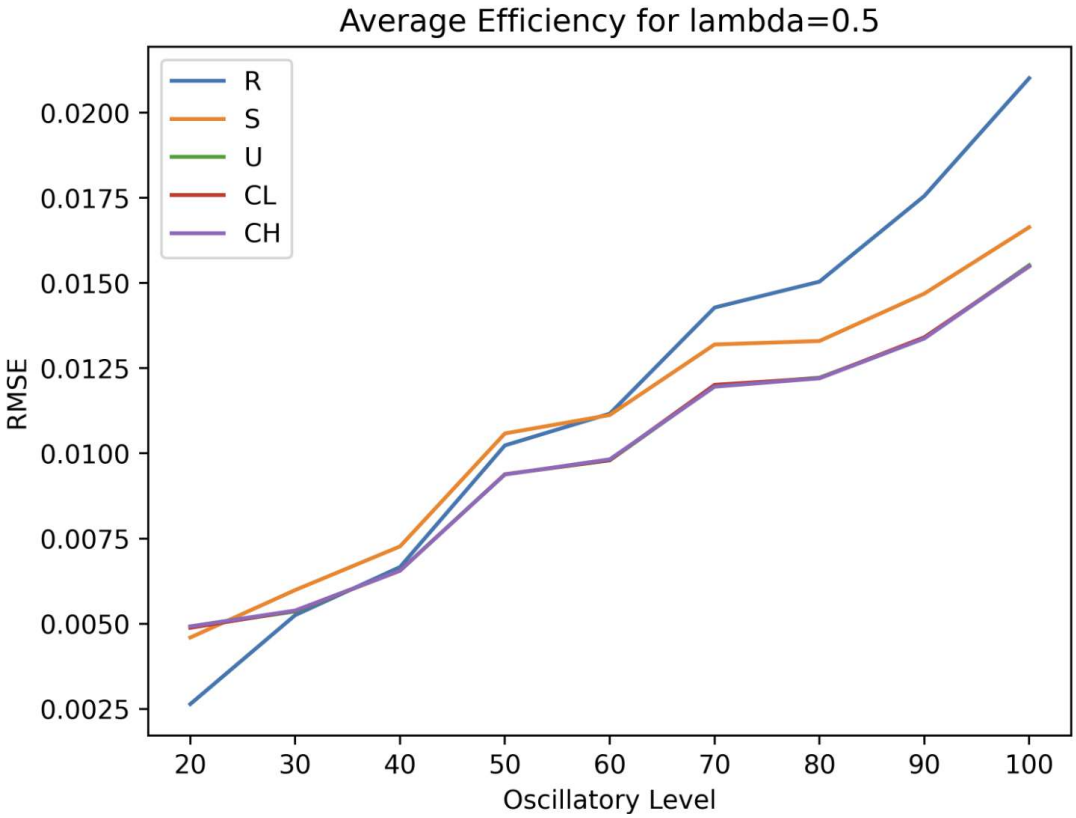


Figure 5.30: Average RMSE obtained for the different reservoirs using $\lambda = 0.5$. R stands for Ridge Regression, S for Single Oscillator, U for Uncoupled Oscillators, CL for Coupled Low Oscillators and CH for Coupled High Oscillators.

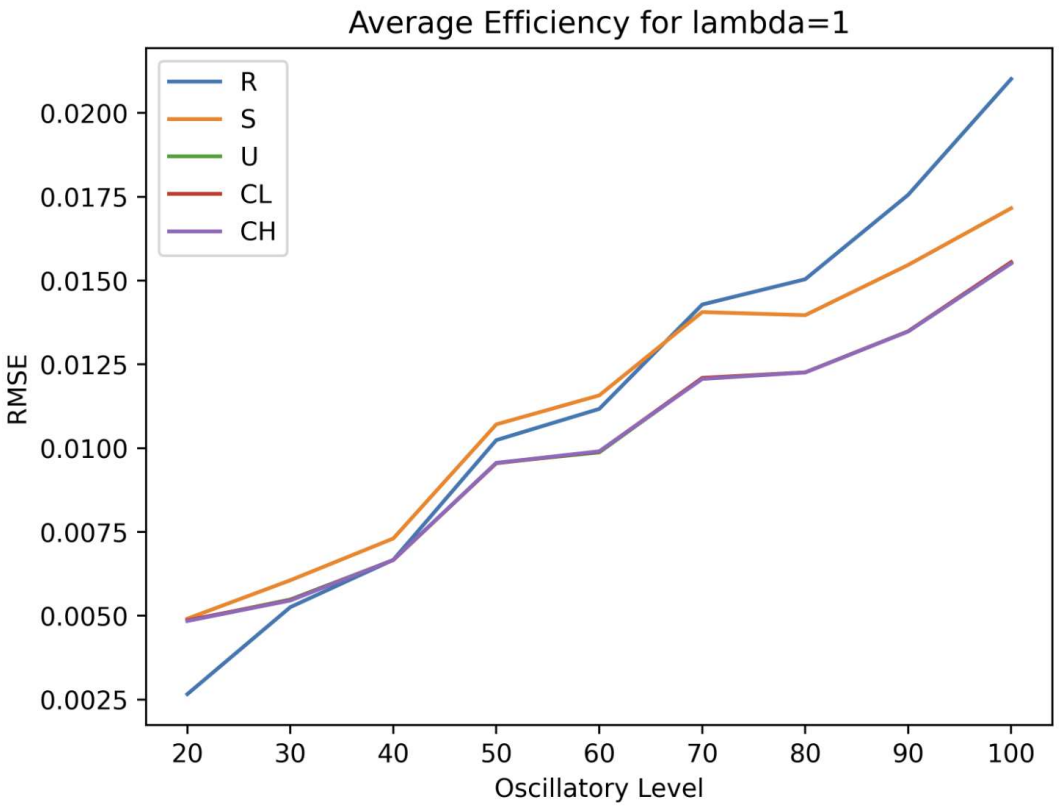


Figure 5.31: Average RMSE obtained for the different reservoirs using $\lambda = 1$. R stands for Ridge Regression, S for Single Oscillator, U for Uncoupled Oscillators, CL for Coupled Low Oscillators and CH for Coupled High Oscillators.

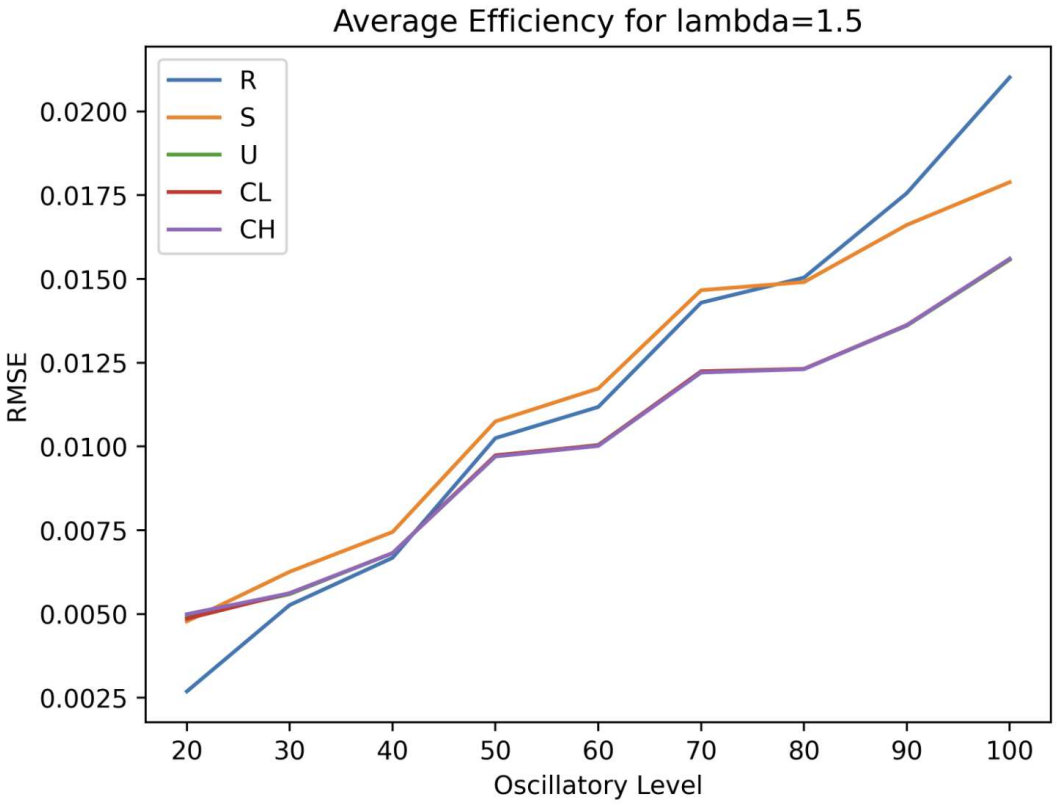
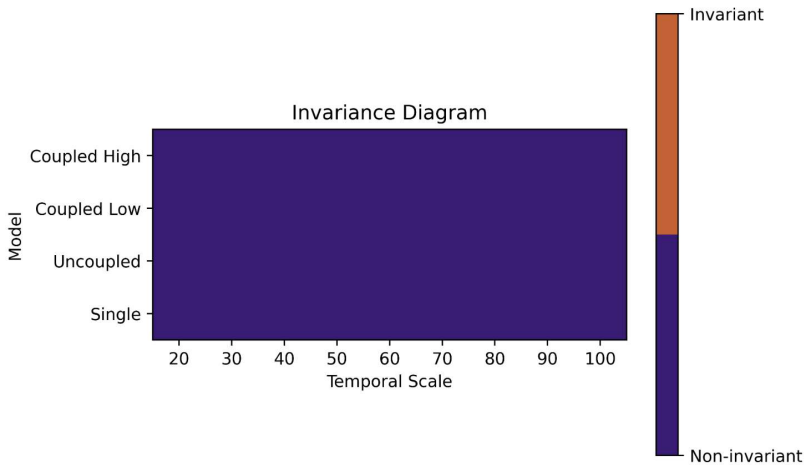


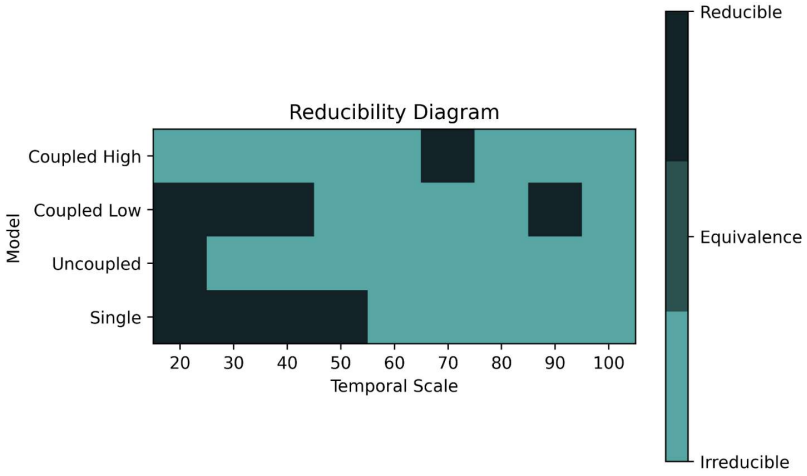
Figure 5.32: Average RMSE obtained for the different reservoirs using $\lambda = 1.5$. R stands for Ridge Regression, S for Single Oscillator, U for Uncoupled Oscillators, CL for Coupled Low Oscillators and CH for Coupled High Oscillators.

Appendix:

Stochastic Case | Brown Noise

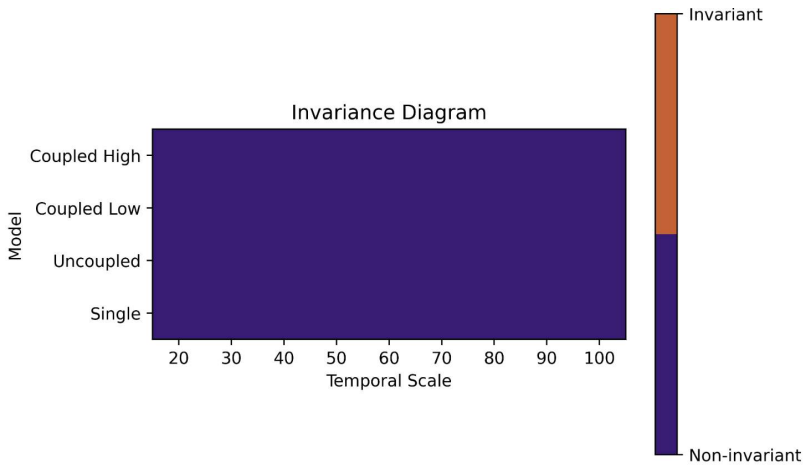


(a)

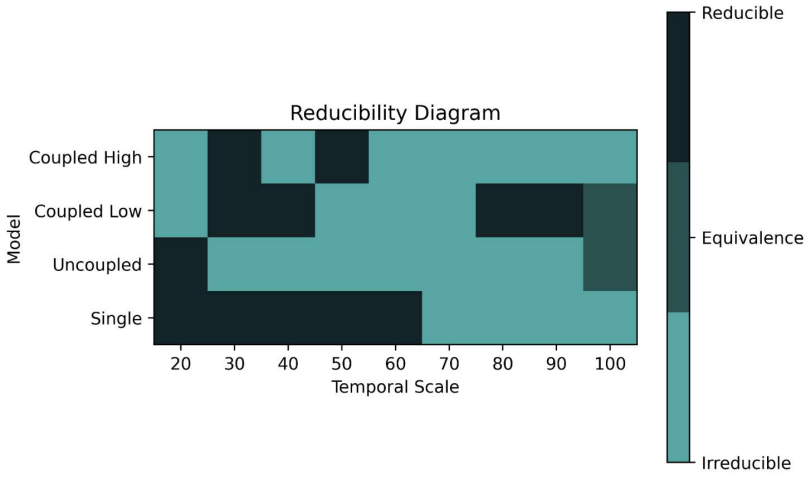


(b)

Figure 5.33: Diagrams obtained for $\lambda = 0.05$ using additive brown noise. In both the x axis represents the different oscillatory levels described in Sections 2.3.1.

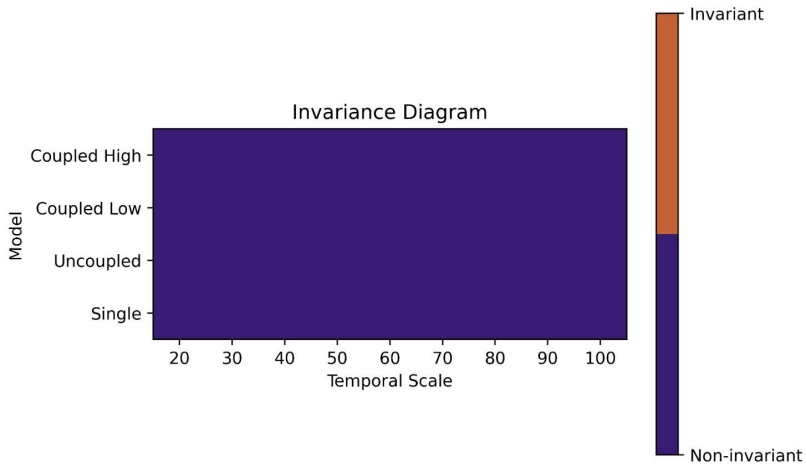


(a)

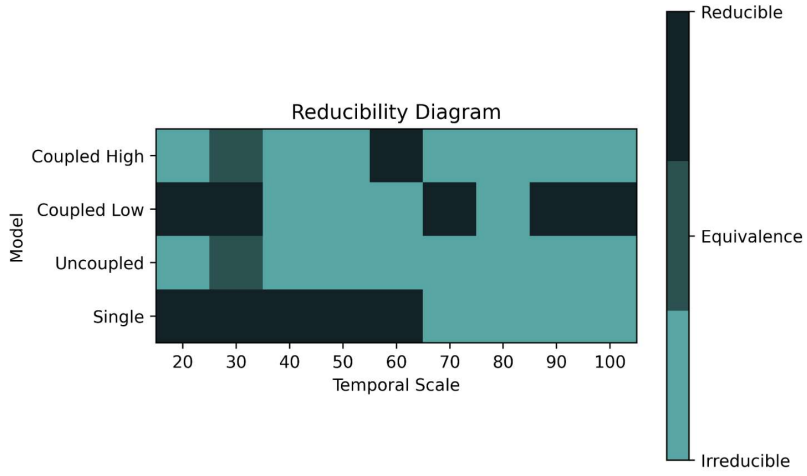


(b)

Figure 5.34: Diagrams obtained for $\lambda = 0.5$ using additive brown noise. In both the x axis represents the different oscillatory levels described in Sections 2.3.1.

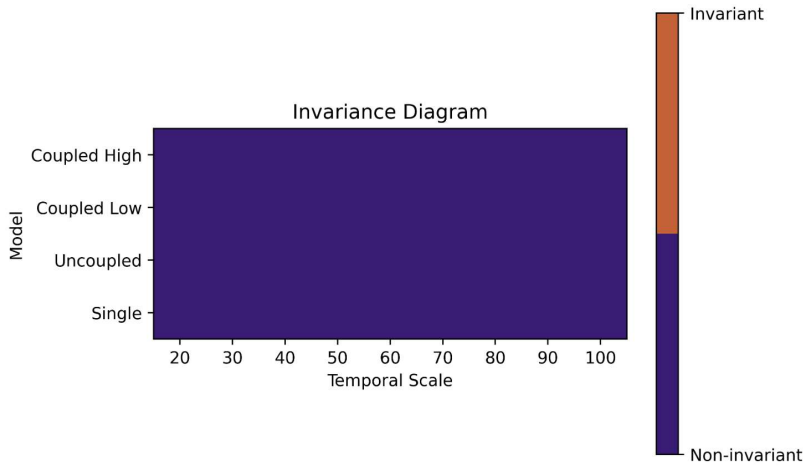


(a)

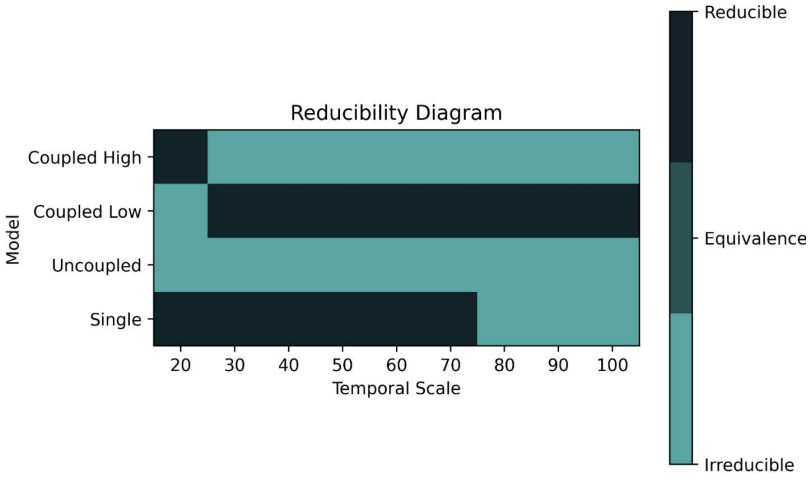


(b)

Figure 5.35: Diagrams obtained for $\lambda = 1$ using additive brown noise. In both the x axis represents the different oscillatory levels described in Sections 2.3.1.



(a)



(b)

Figure 5.36: Diagrams obtained for $\lambda = 1.5$ using additive brown noise. In both the x axis represents the different oscillatory levels described in Sections 2.3.1.

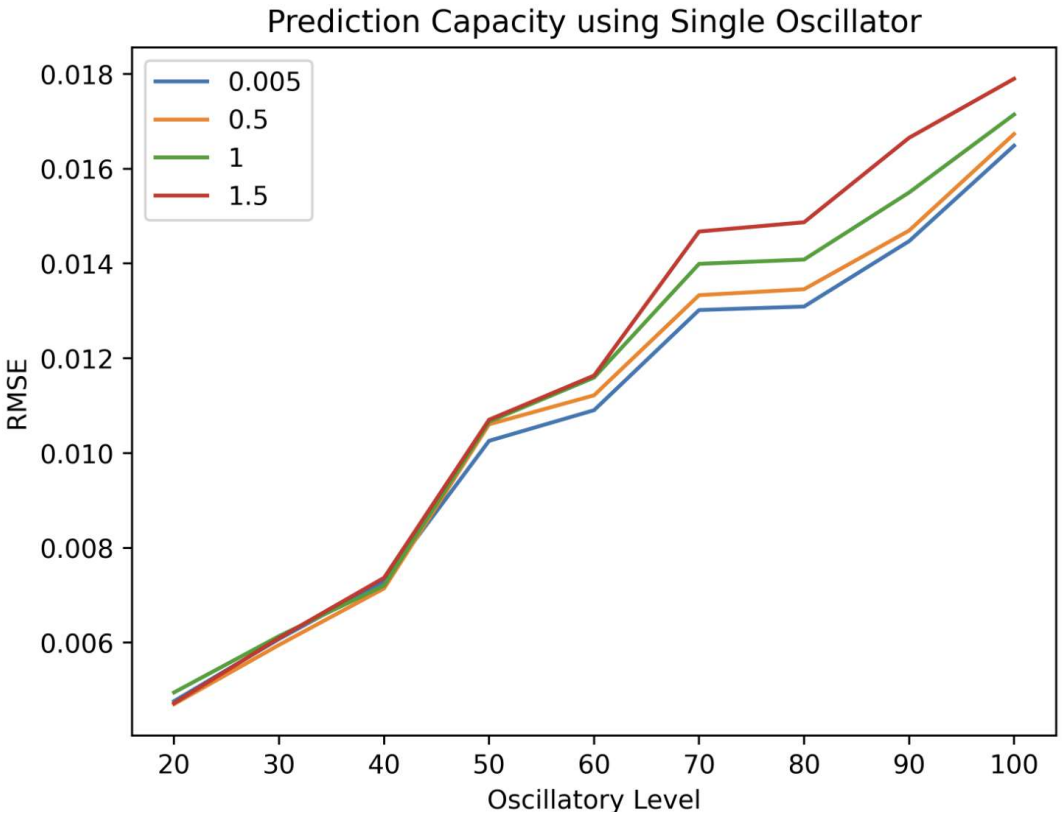


Figure 5.37: Average RMSE obtained for the single automaton reservoir using different values $\lambda > 0$ and additive brown noise.

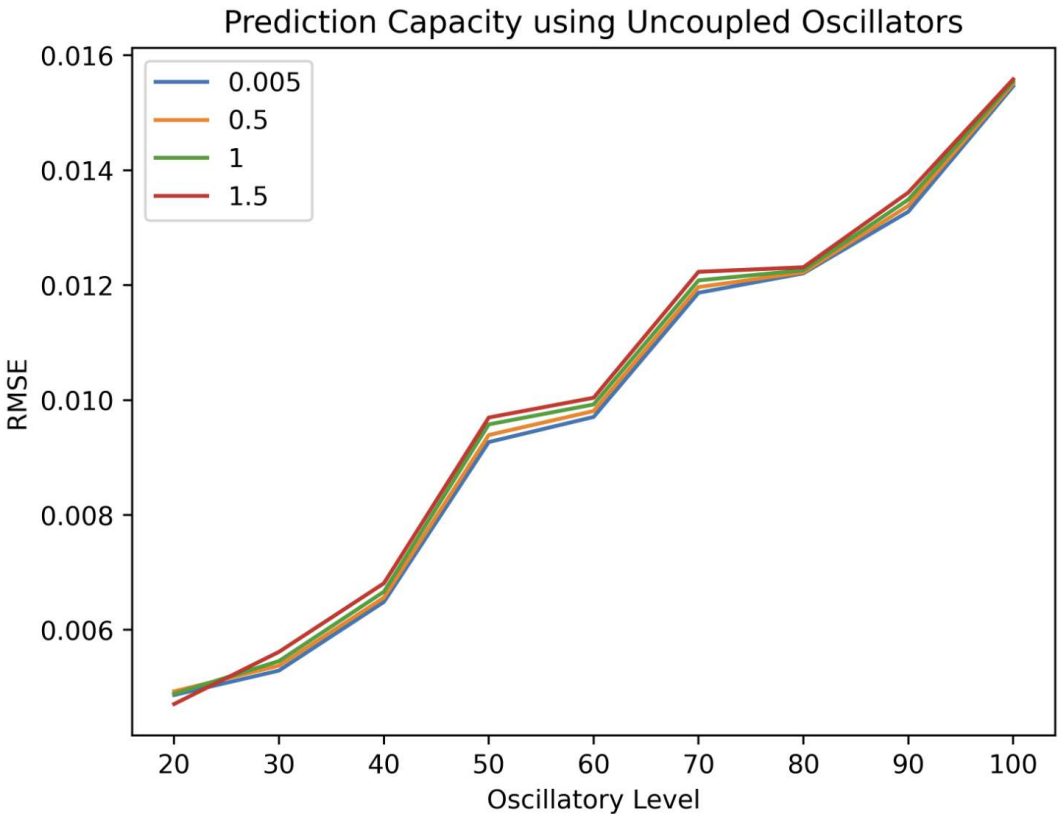


Figure 5.38: Average RMSE obtained for the uncoupled automata reservoir using different values $\lambda > 0$ and additive brown noise.

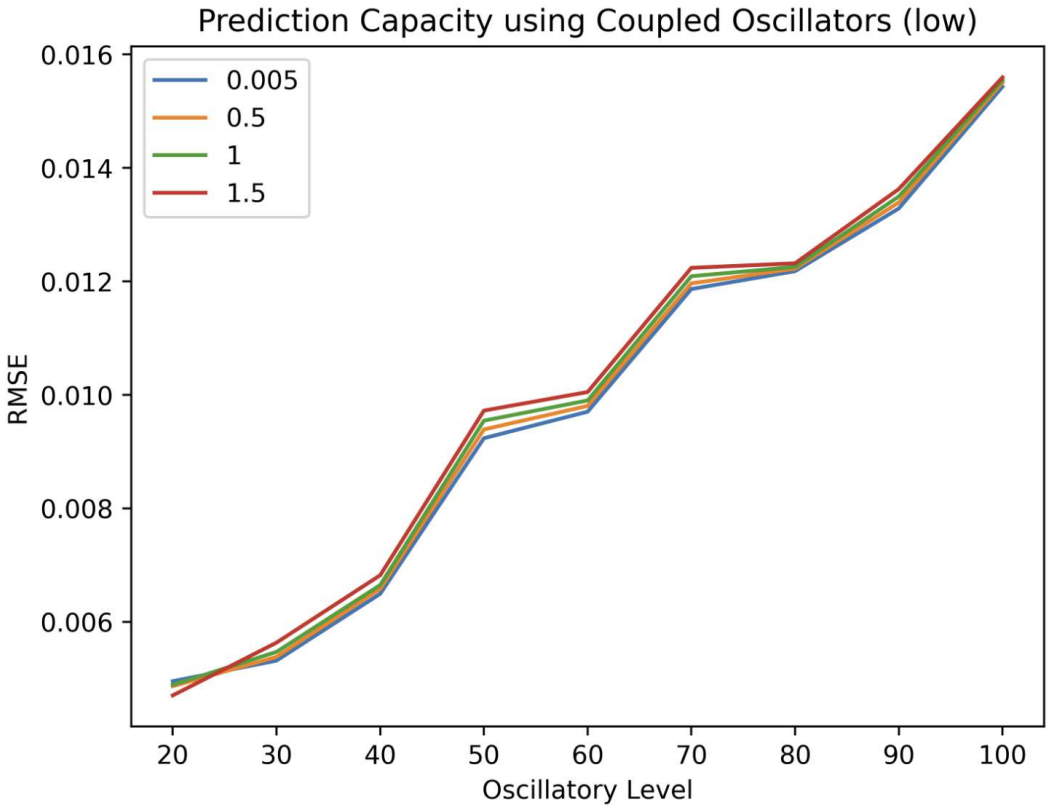


Figure 5.39: Average RMSE obtained for the coupled low automata reservoir using different values $\lambda > 0$ and additive brown noise.

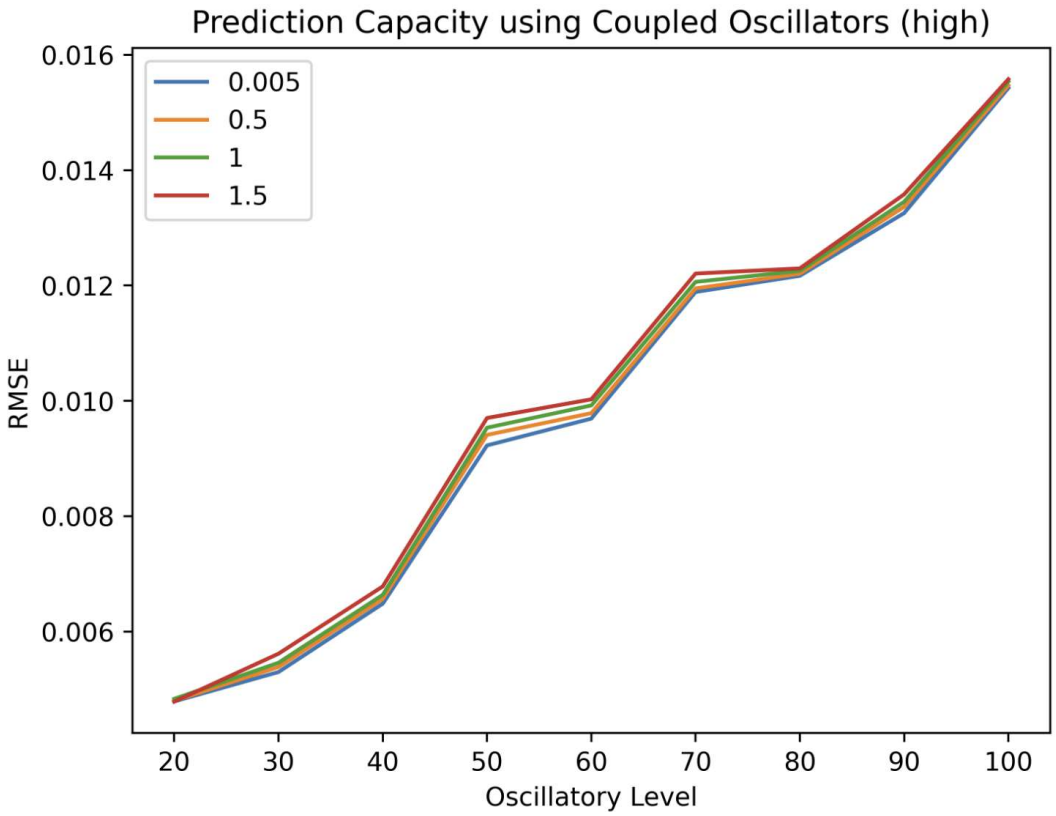


Figure 5.40: Average RMSE obtained for the coupled high automata reservoir using different values $\lambda > 0$ and additive brown noise.

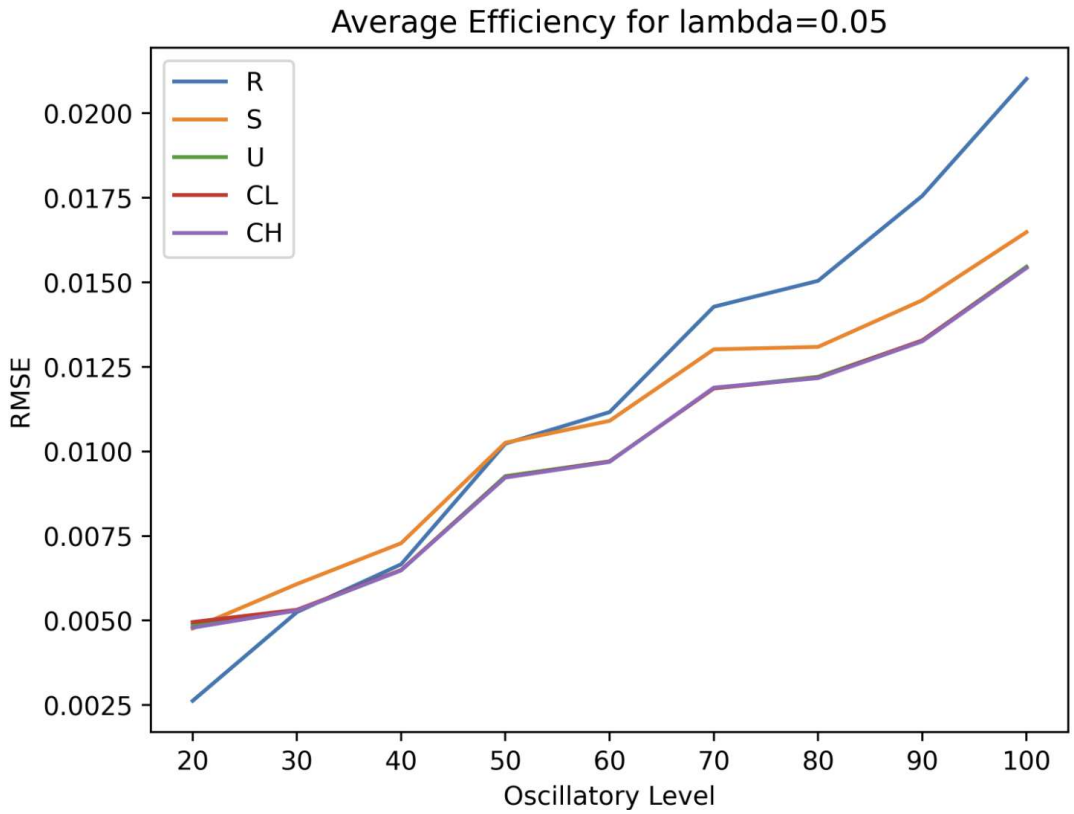


Figure 5.41: Average RMSE obtained for the different reservoirs using $\lambda = 0.05$. R stands for Ridge Regression, S for Single Oscillator, U for Uncoupled Oscillators, CL for Coupled Low Oscillators and CH for Coupled High Oscillators.

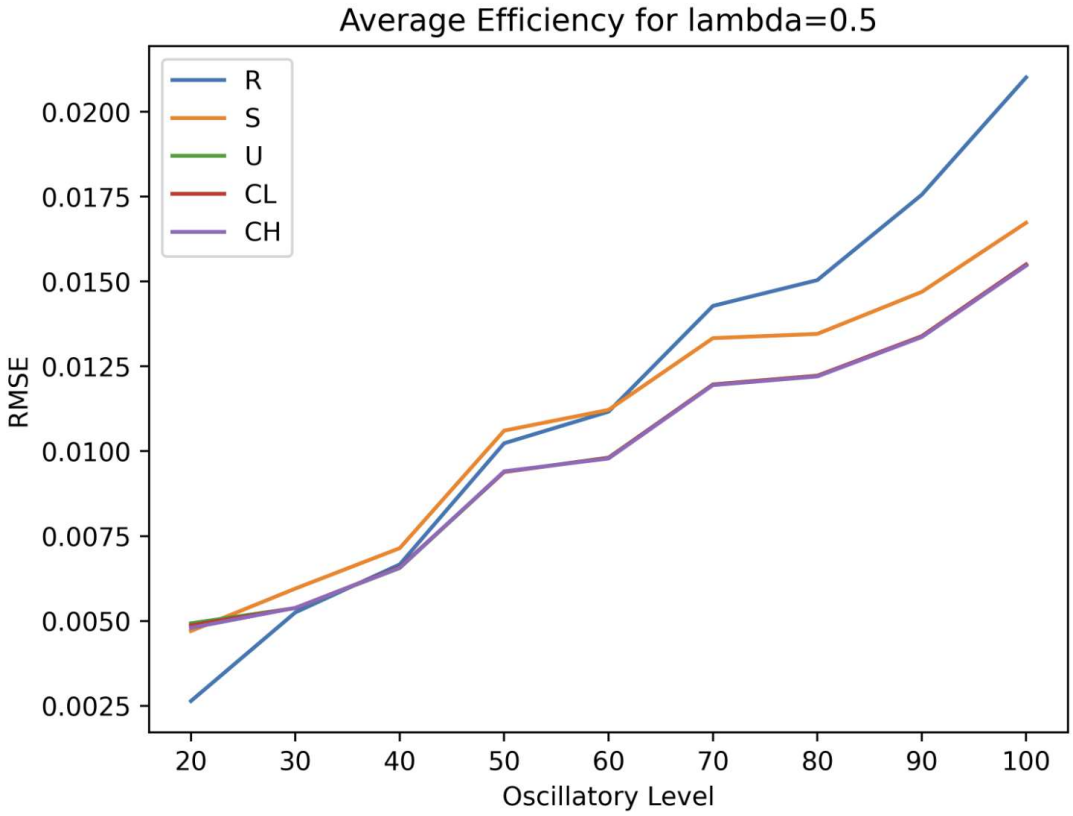


Figure 5.42: Average RMSE obtained for the different reservoirs using $\lambda = 0.5$. R stands for Ridge Regression, S for Single Oscillator, U for Uncoupled Oscillators, CL for Coupled Low Oscillators and CH for Coupled High Oscillators.

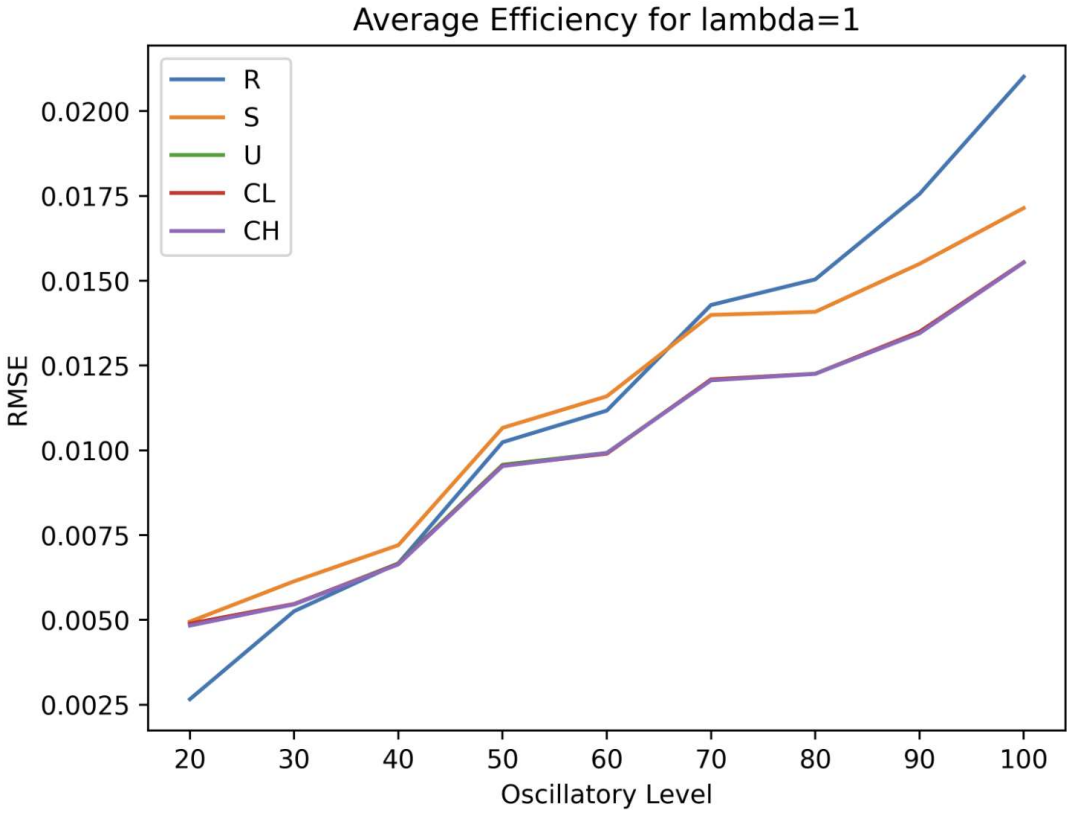


Figure 5.43: Average RMSE obtained for the different reservoirs using $\lambda = 1$. R stands for Ridge Regression, S for Single Oscillator, U for Uncoupled Oscillators, CL for Coupled Low Oscillators and CH for Coupled High Oscillators.

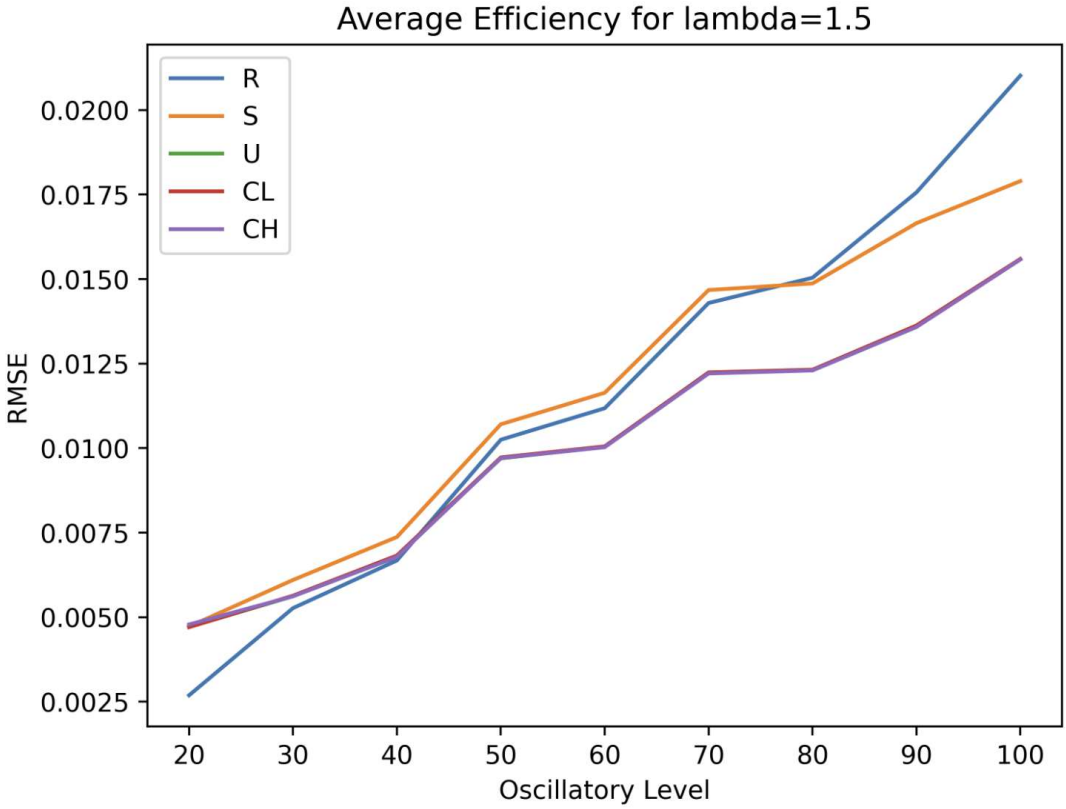
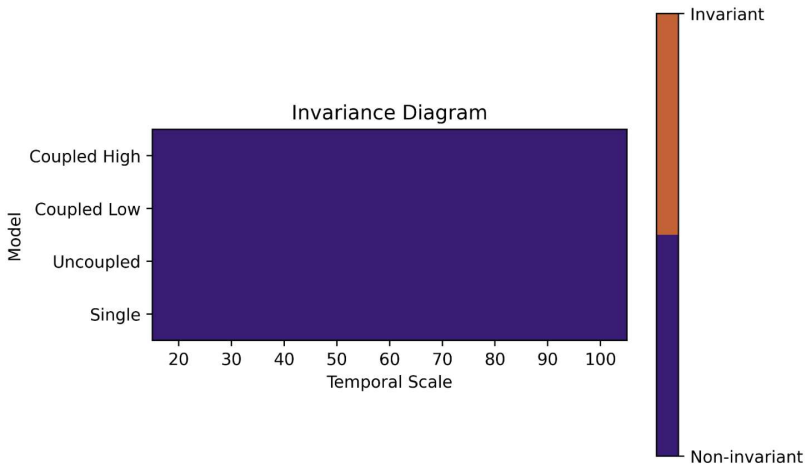


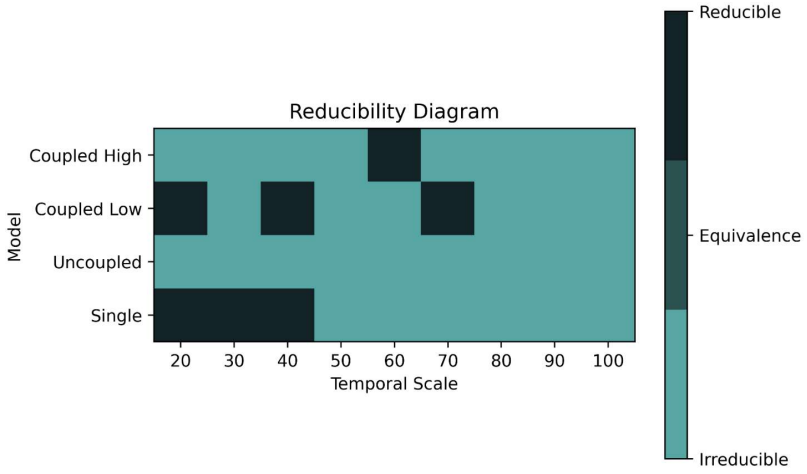
Figure 5.44: Average RMSE obtained for the different reservoirs using $\lambda = 1.5$. R stands for Ridge Regression, S for Single Oscillator, U for Uncoupled Oscillators, CL for Coupled Low Oscillators and CH for Coupled High Oscillators.

Appendix:

Stochastic Case | Blue Noise

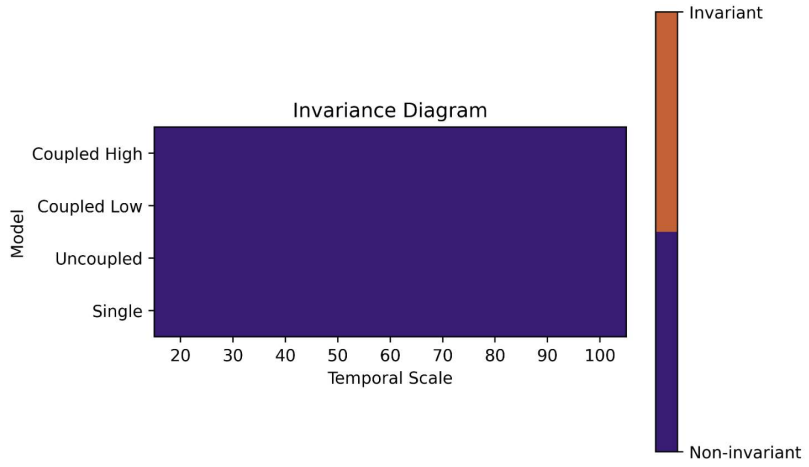


(a)

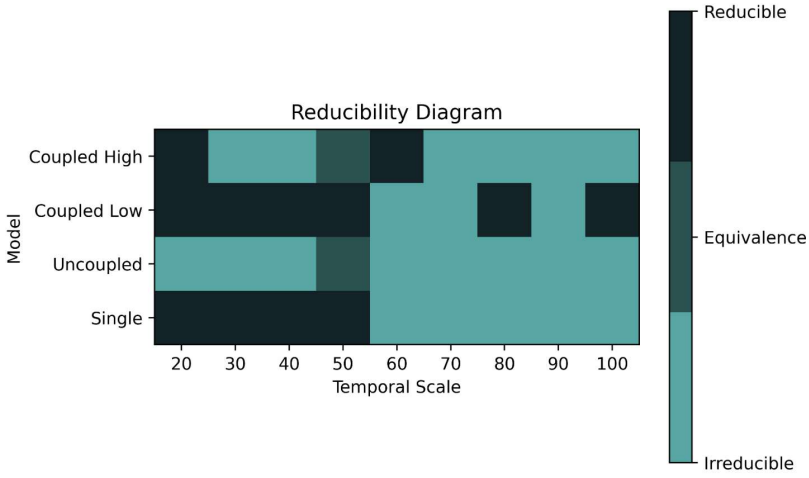


(b)

Figure 5.45: Diagrams obtained for $\lambda = 0.05$ using additive blue noise. In both the x axis represents the different oscillatory levels described in Sections 2.3.1.

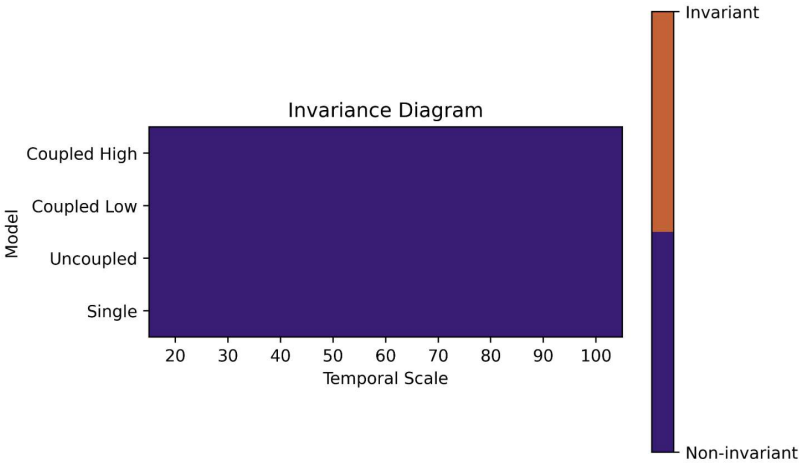


(a)

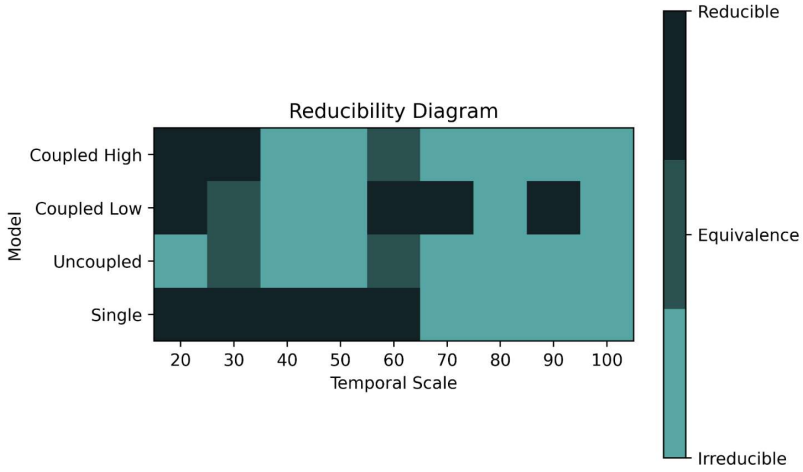


(b)

Figure 5.46: Diagrams obtained for $\lambda = 0.5$ using additive blue noise. In both the x axis represents the different oscillatory levels described in Sections 2.3.1.

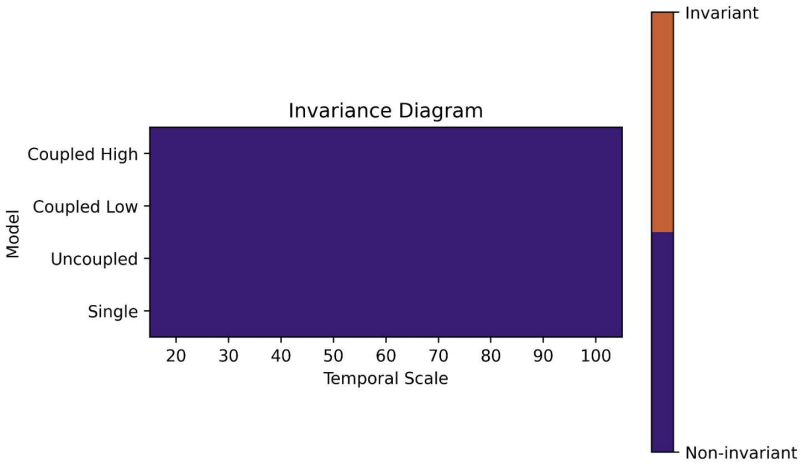


(a)

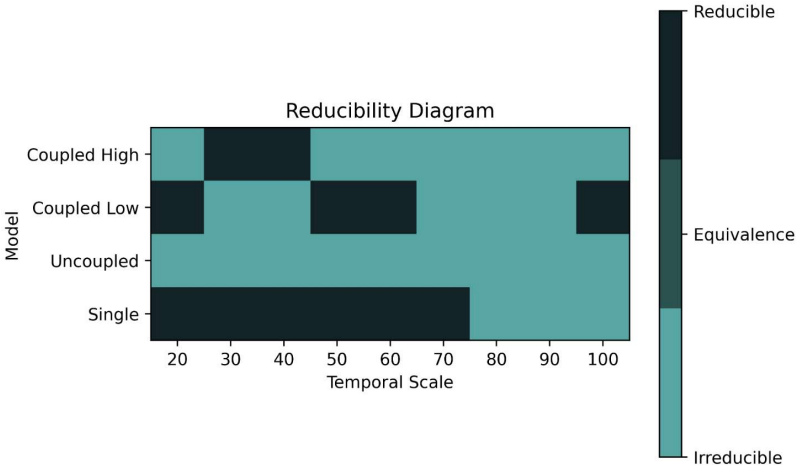


(b)

Figure 5.47: Diagrams obtained for $\lambda = 1$ using additive blue noise. In both the x axis represents the different oscillatory levels described in Sections 2.3.1.



(a)



(b)

Figure 5.48: Diagrams obtained for $\lambda = 1.5$ using additive blue noise. In both the x axis represents the different oscillatory levels described in Sections 2.3.1.

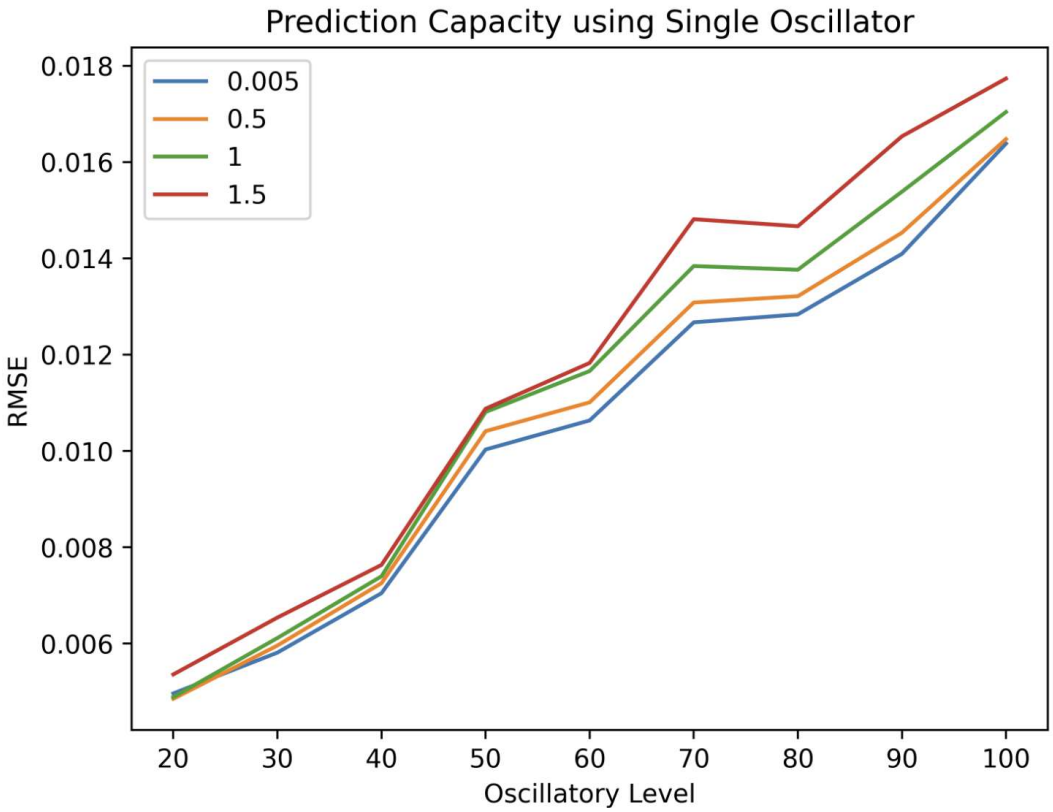


Figure 5.49: Average RMSE obtained for the single automaton reservoir using different values $\lambda > 0$ and additive blue noise.

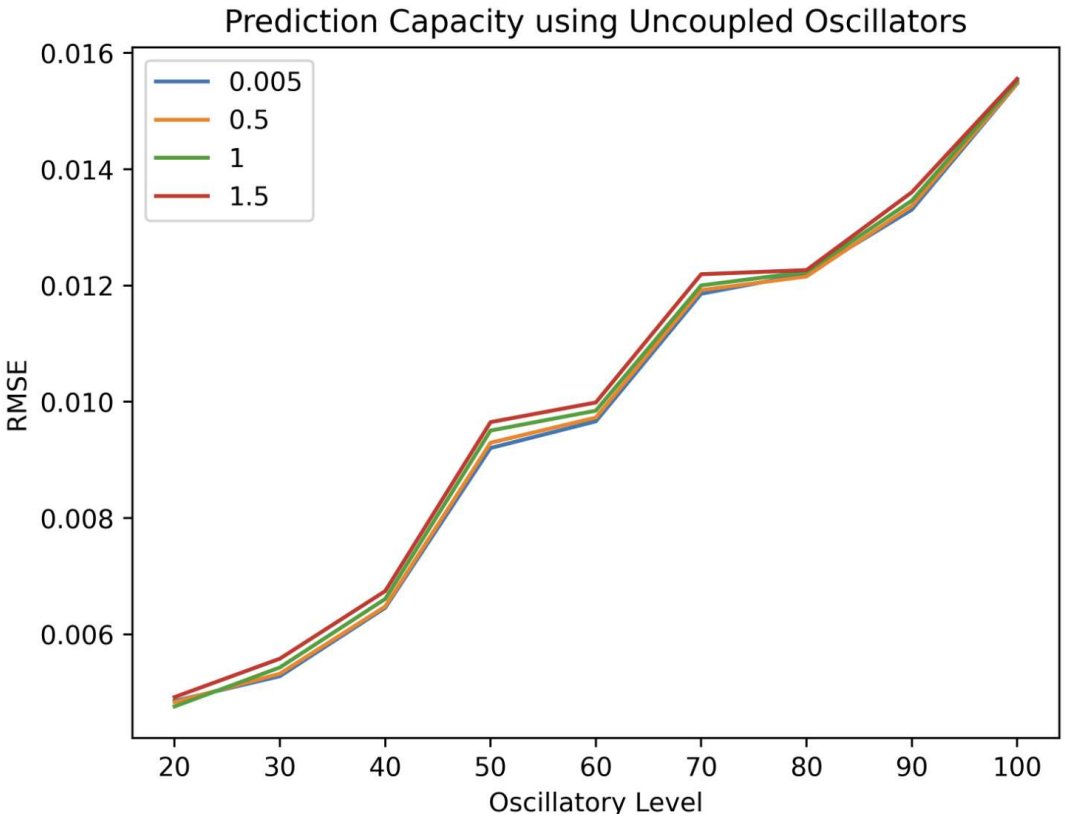


Figure 5.50: Average RMSE obtained for the uncoupled automata reservoir using different values $\lambda > 0$ and additive blue noise.

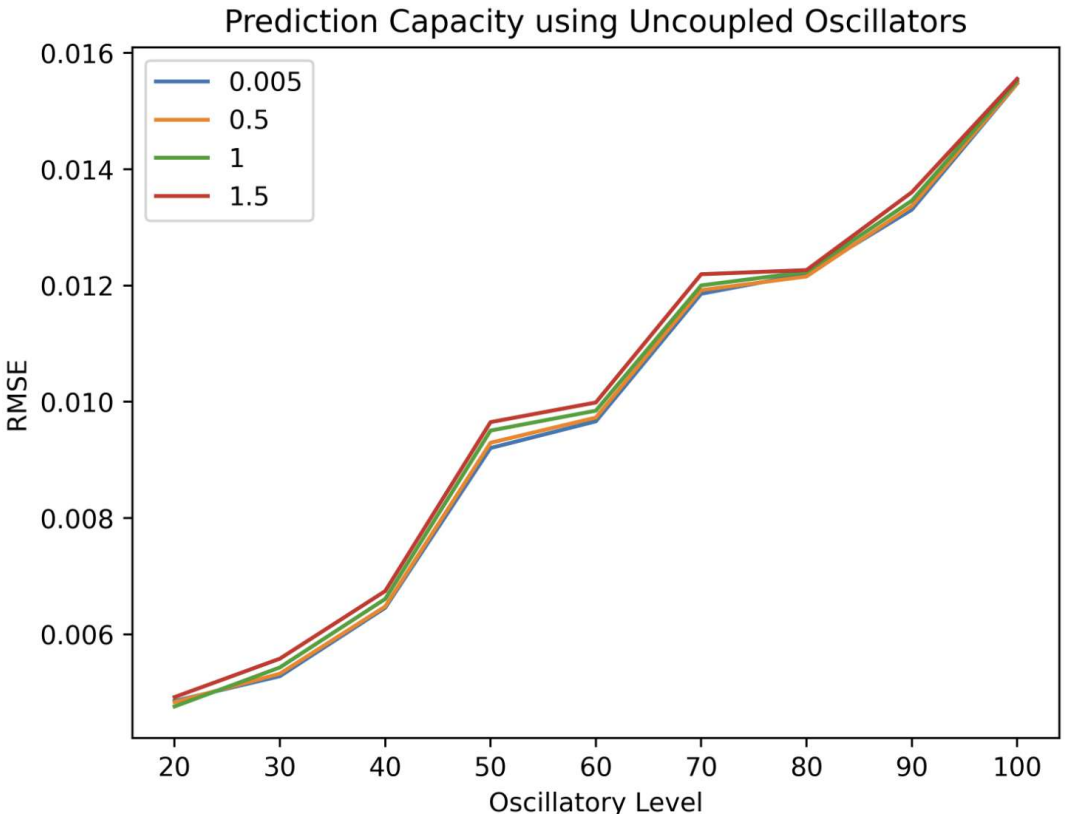


Figure 5.51: Average RMSE obtained for the uncoupled automata reservoir using different values $\lambda > 0$ and additive blue noise.

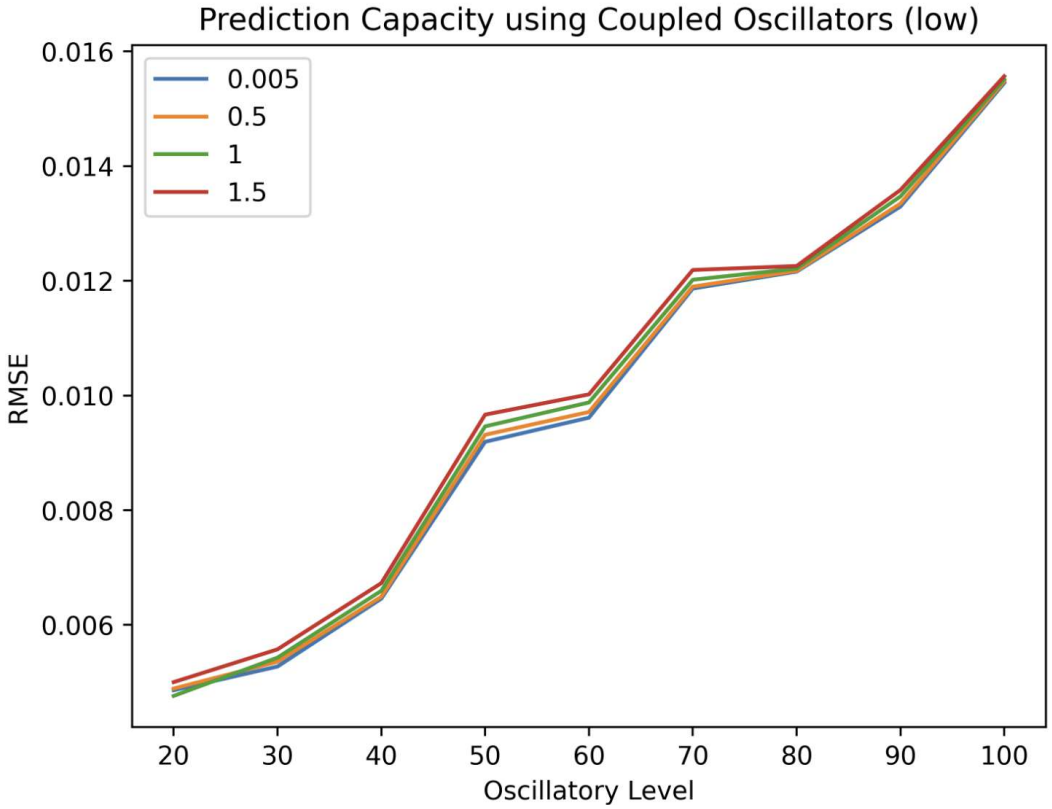


Figure 5.52: Average RMSE obtained for the coupled low automata reservoir using different values $\lambda > 0$ and additive blue noise.

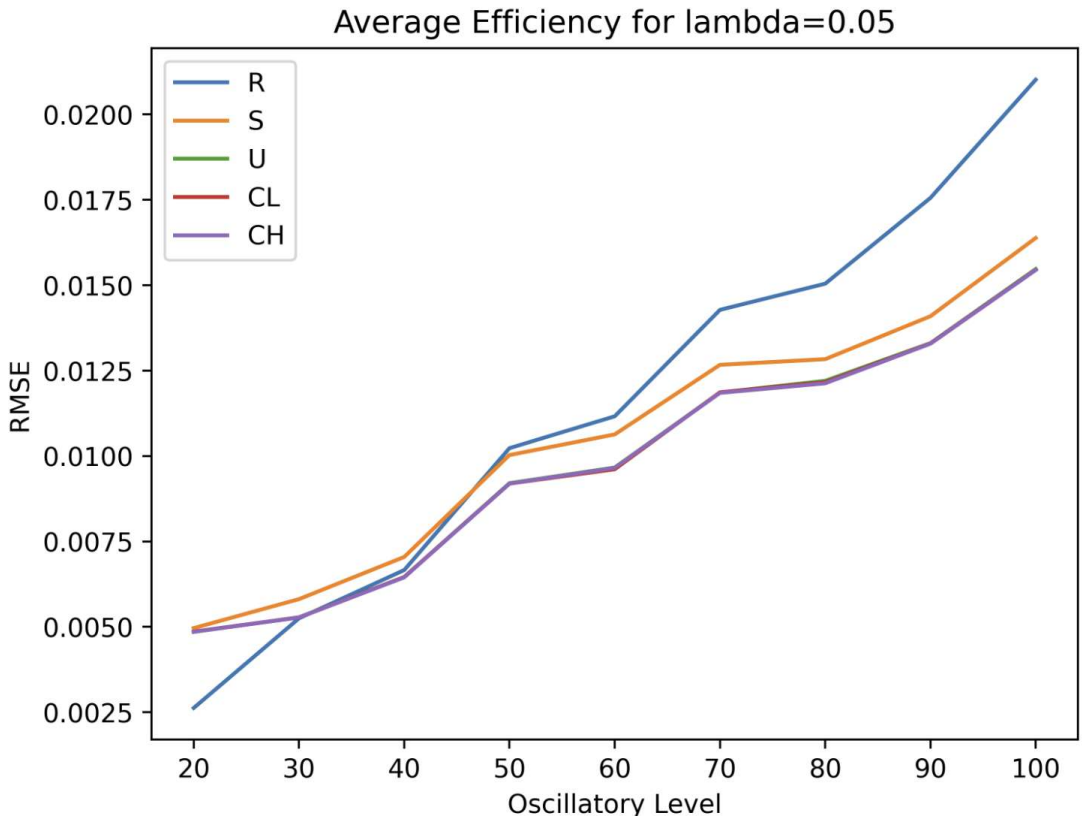


Figure 5.53: Average RMSE obtained for the different reservoirs using $\lambda = 0.05$. R stands for Ridge Regression, S for Single Oscillator, U for Uncoupled Oscillators, CL for Coupled Low Oscillators and CH for Coupled High Oscillators.

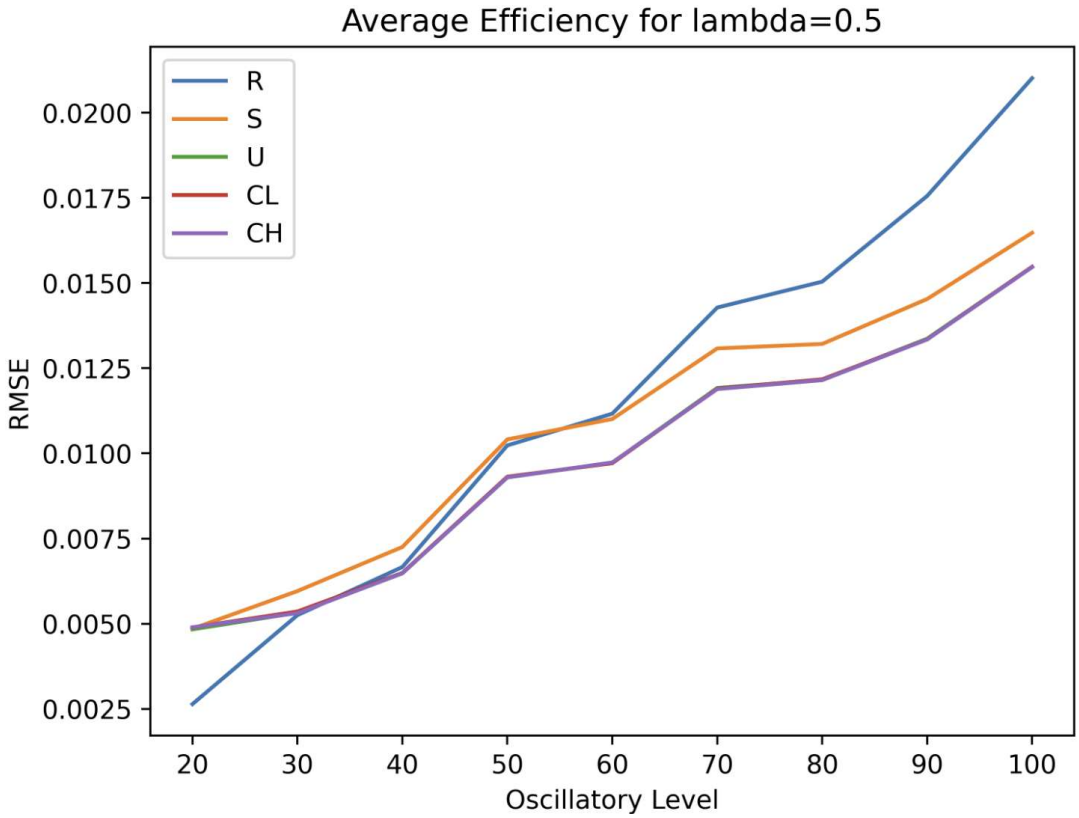


Figure 5.54: Average RMSE obtained for the different reservoirs using $\lambda = 0.5$. R stands for Ridge Regression, S for Single Oscillator, U for Uncoupled Oscillators, CL for Coupled Low Oscillators and CH for Coupled High Oscillators.

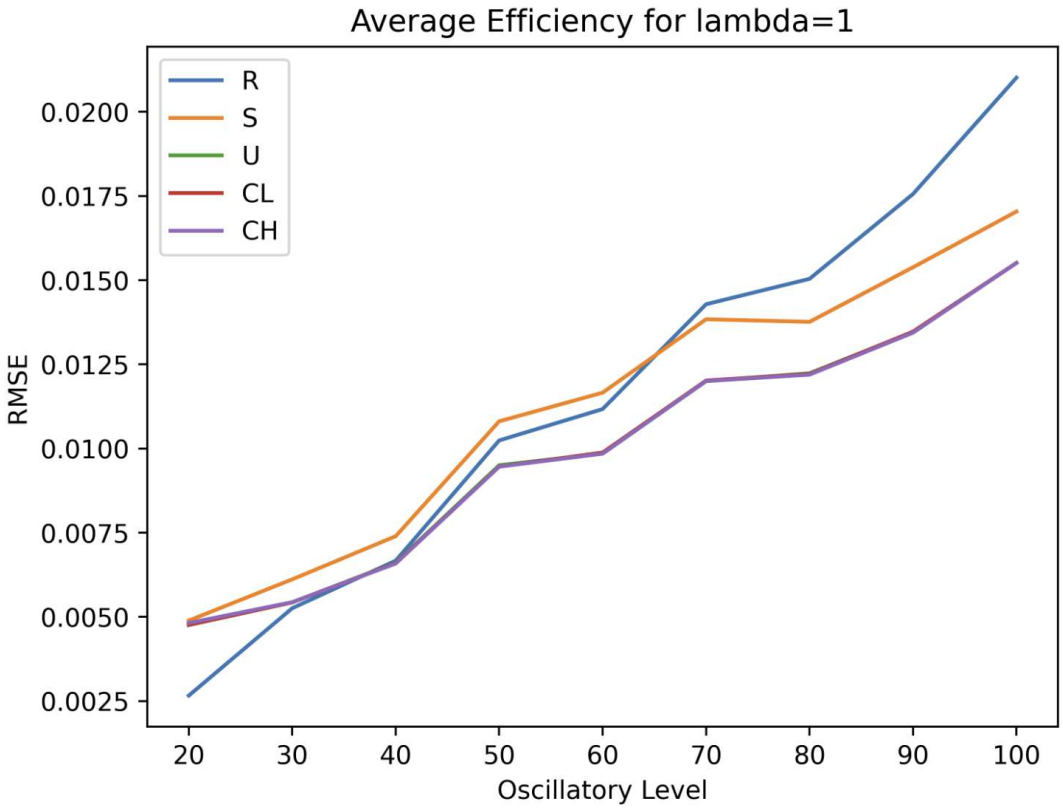


Figure 5.55: Average RMSE obtained for the different reservoirs using $\lambda = 1$. R stands for Ridge Regression, S for Single Oscillator, U for Uncoupled Oscillators, CL for Coupled Low Oscillators and CH for Coupled High Oscillators.

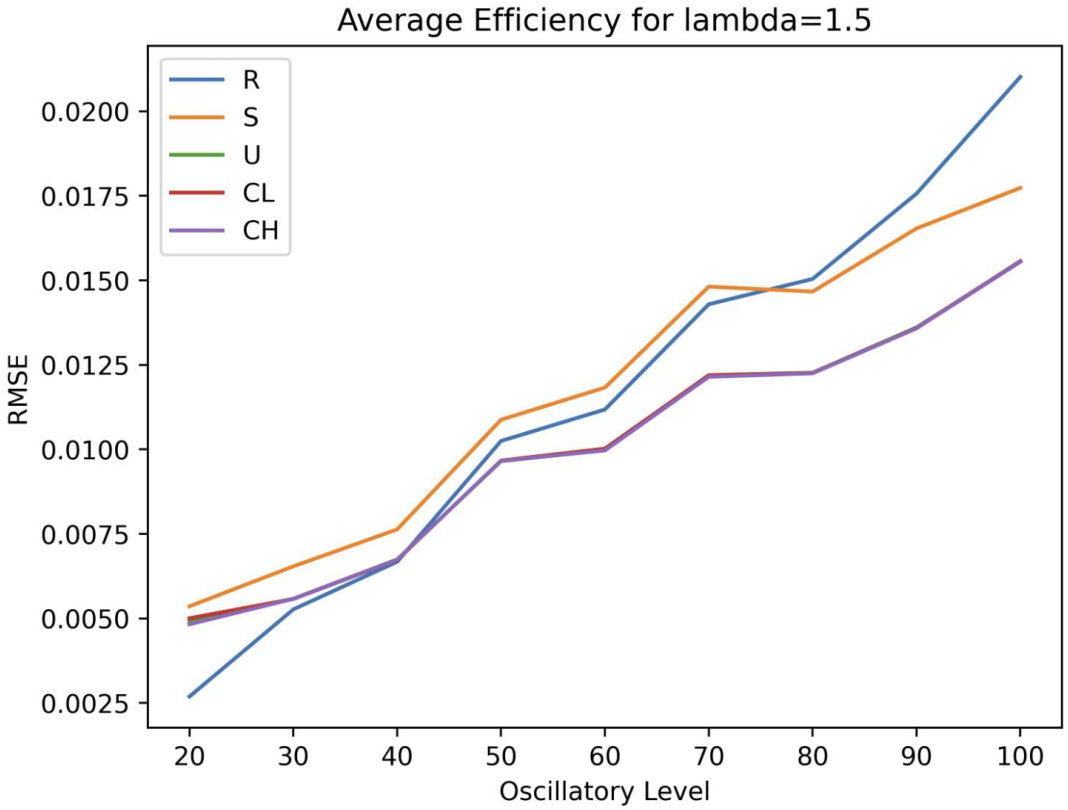
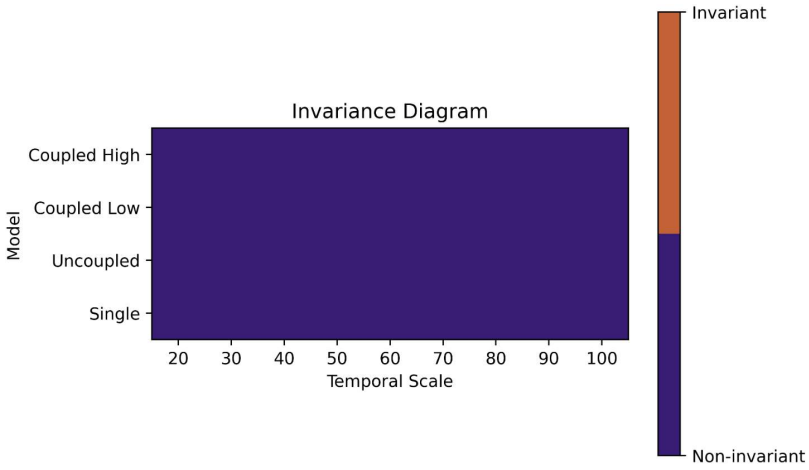


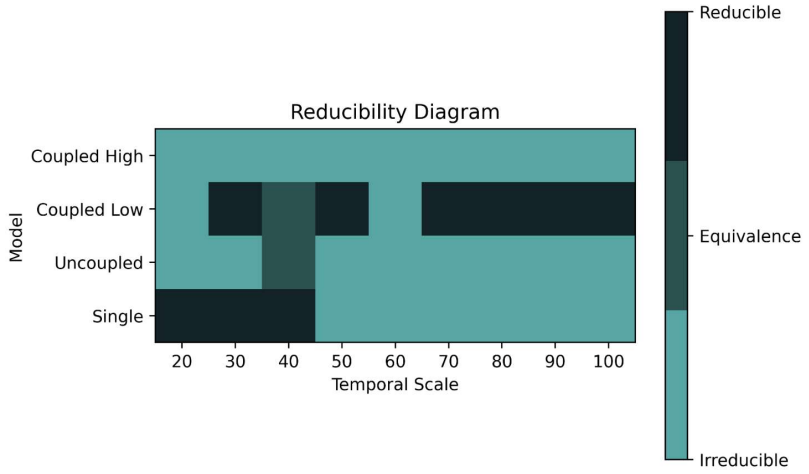
Figure 5.56: Average RMSE obtained for the different reservoirs using $\lambda = 1.5$. R stands for Ridge Regression, S for Single Oscillator, U for Uncoupled Oscillators, CL for Coupled Low Oscillators and CH for Coupled High Oscillators.

Appendix:

Stochastic Case | Violet Noise

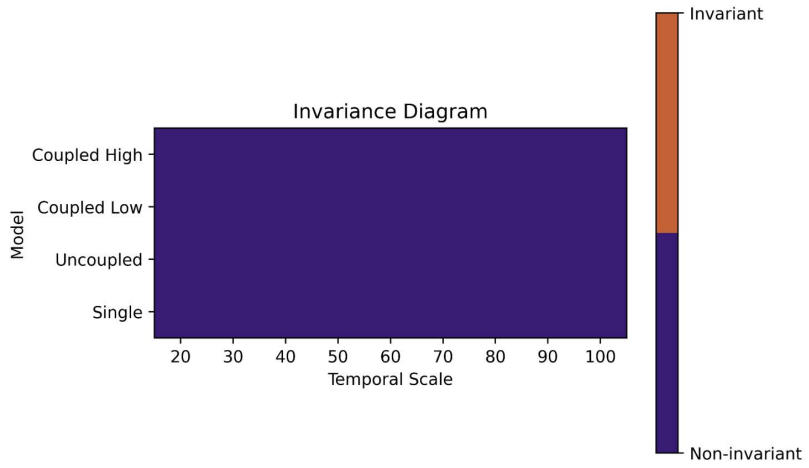


(a)

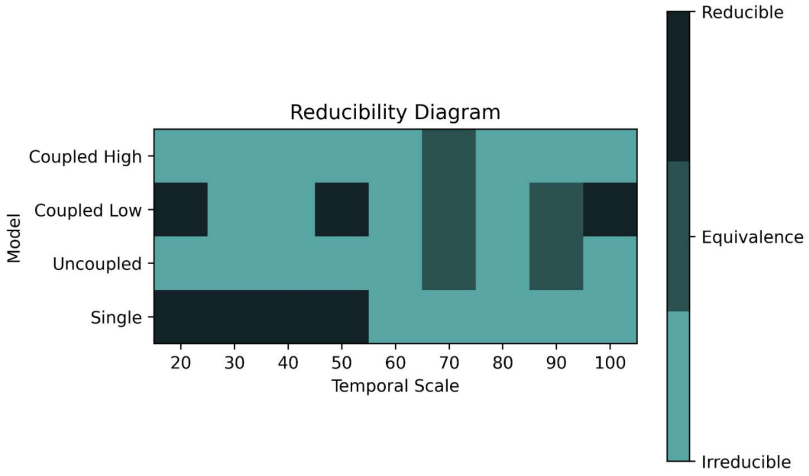


(b)

Figure 5.57: Diagrams obtained for $\lambda = 0.05$ using additive violet noise. In both the x axis represents the different oscillatory levels described in Sections 2.3.1.

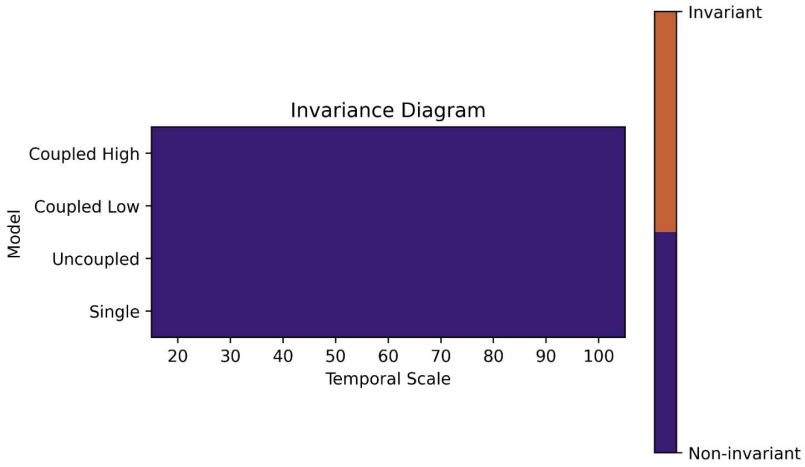


(a)

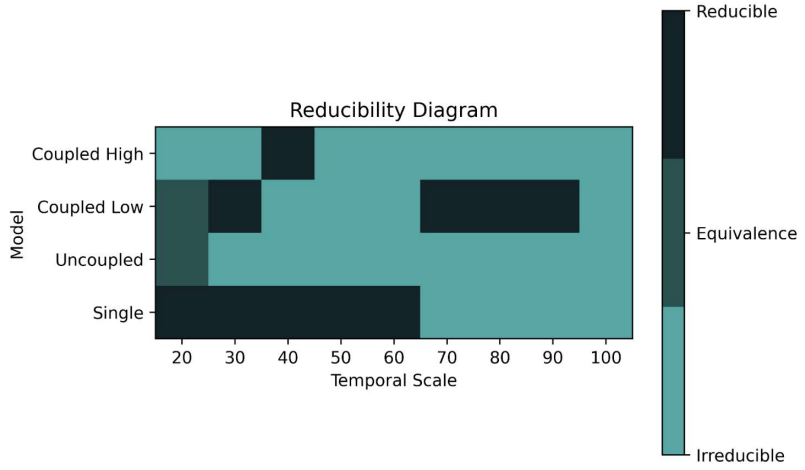


(b)

Figure 5.58: Diagrams obtained for $\lambda = 0.5$ using additive violet noise. In both the x axis represents the different oscillatory levels described in Sections 2.3.1.

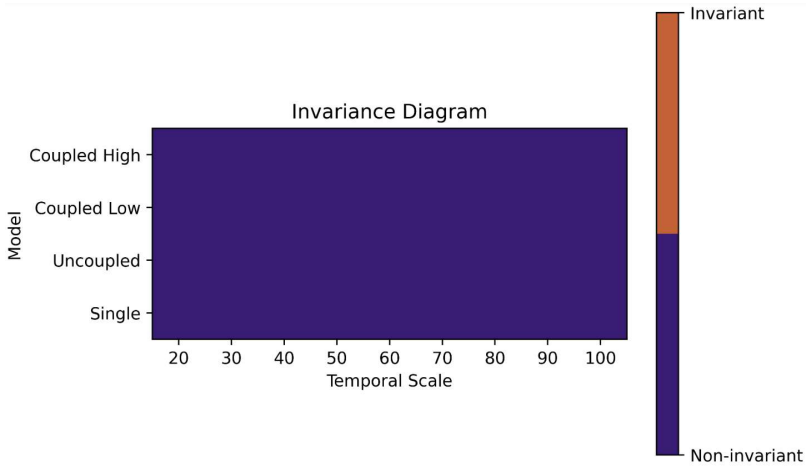


(a)

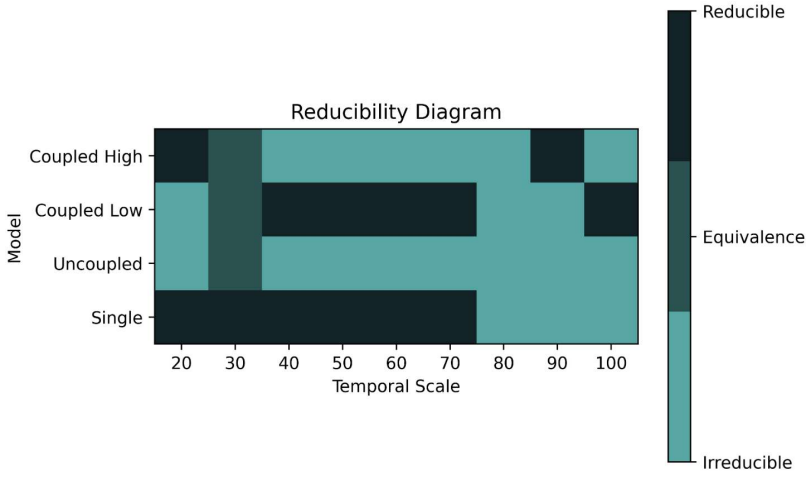


(b)

Figure 5.59: Diagrams obtained for $\lambda = 1$ using additive violet noise. In both the x axis represents the different oscillatory levels described in Sections 2.3.1.



(a)



(b)

Figure 5.60: Diagrams obtained for $\lambda = 1.5$ using additive violet noise. In both the x axis represents the different oscillatory levels described in Sections 2.3.1.

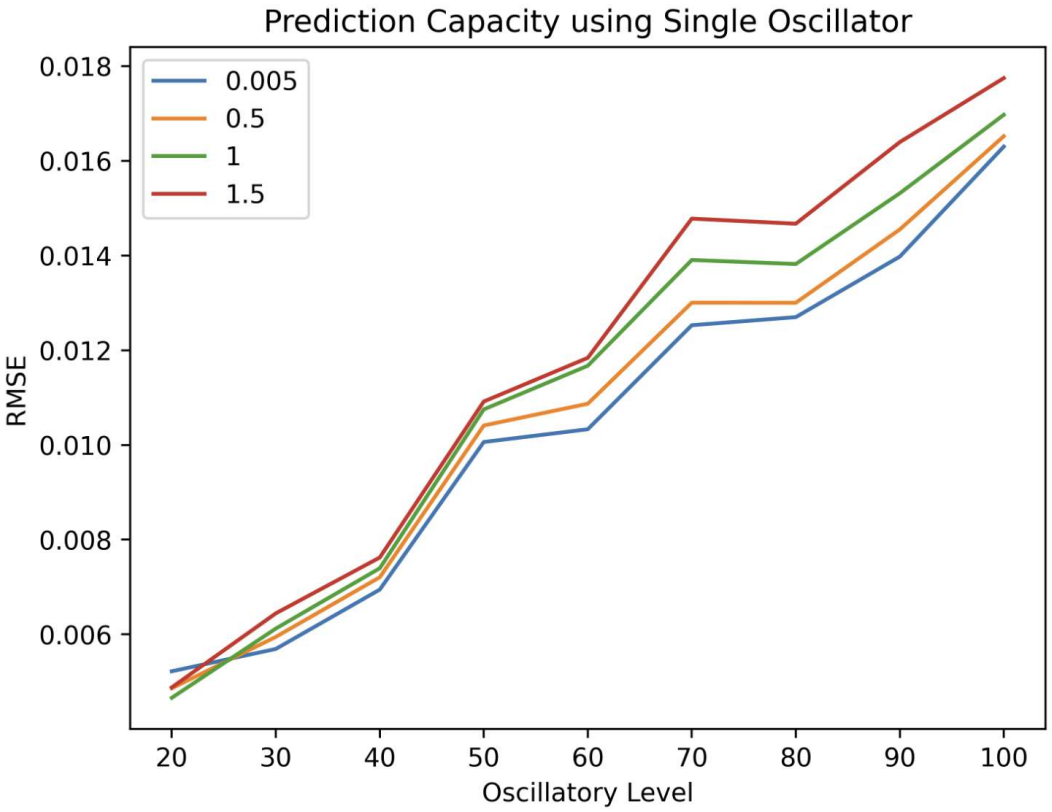


Figure 5.61: Average RMSE obtained for the single automaton reservoir using different values $\lambda > 0$ and additive violet noise.

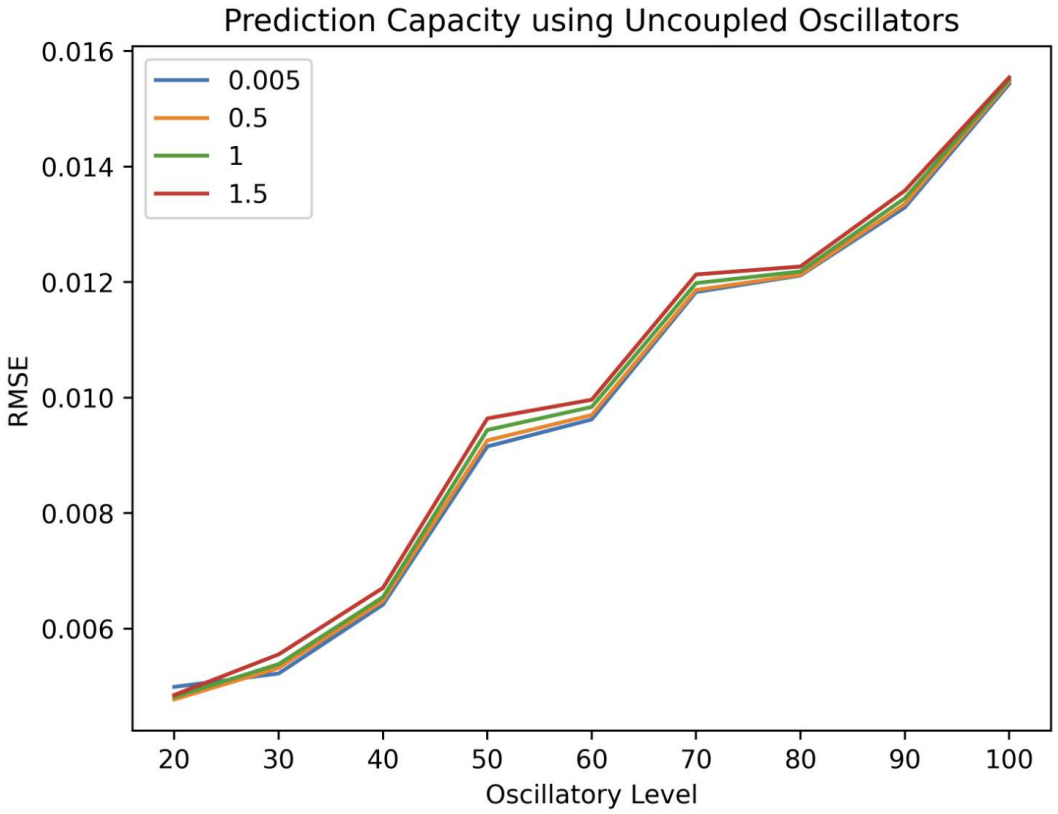


Figure 5.62: Average RMSE obtained for the uncoupled automata reservoir using different values $\lambda > 0$ and additive violet noise.

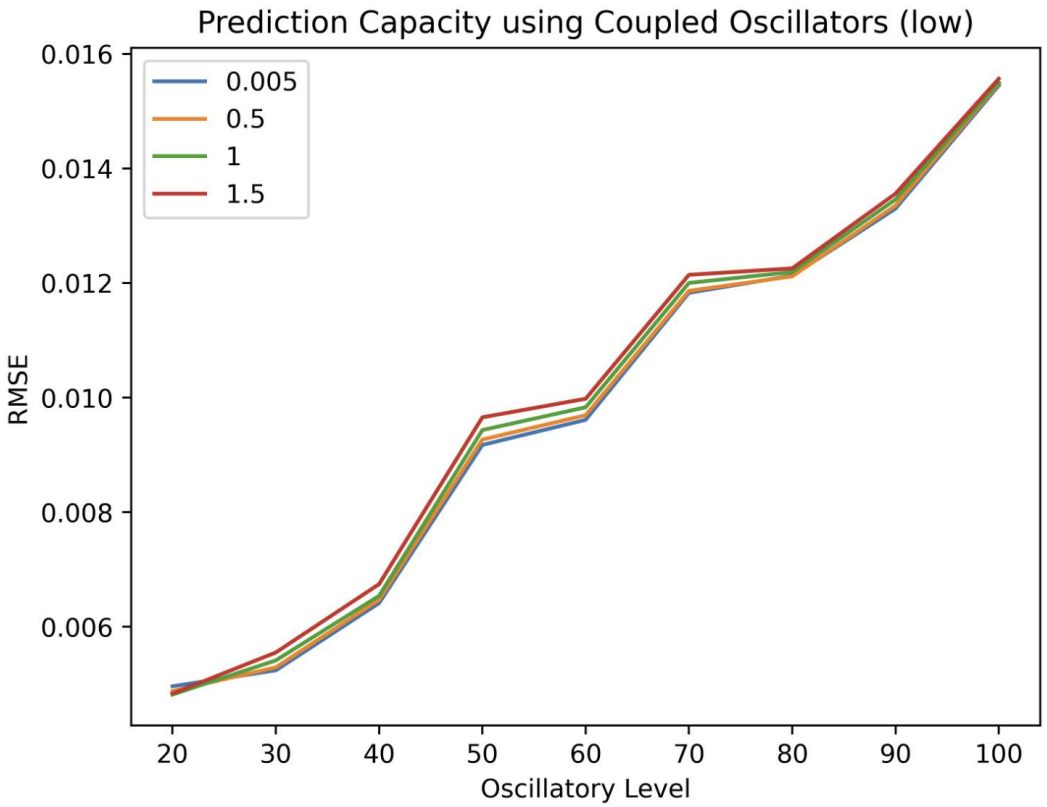


Figure 5.63: Average RMSE obtained for the coupled low automata reservoir using different values $\lambda > 0$ and additive violet noise.

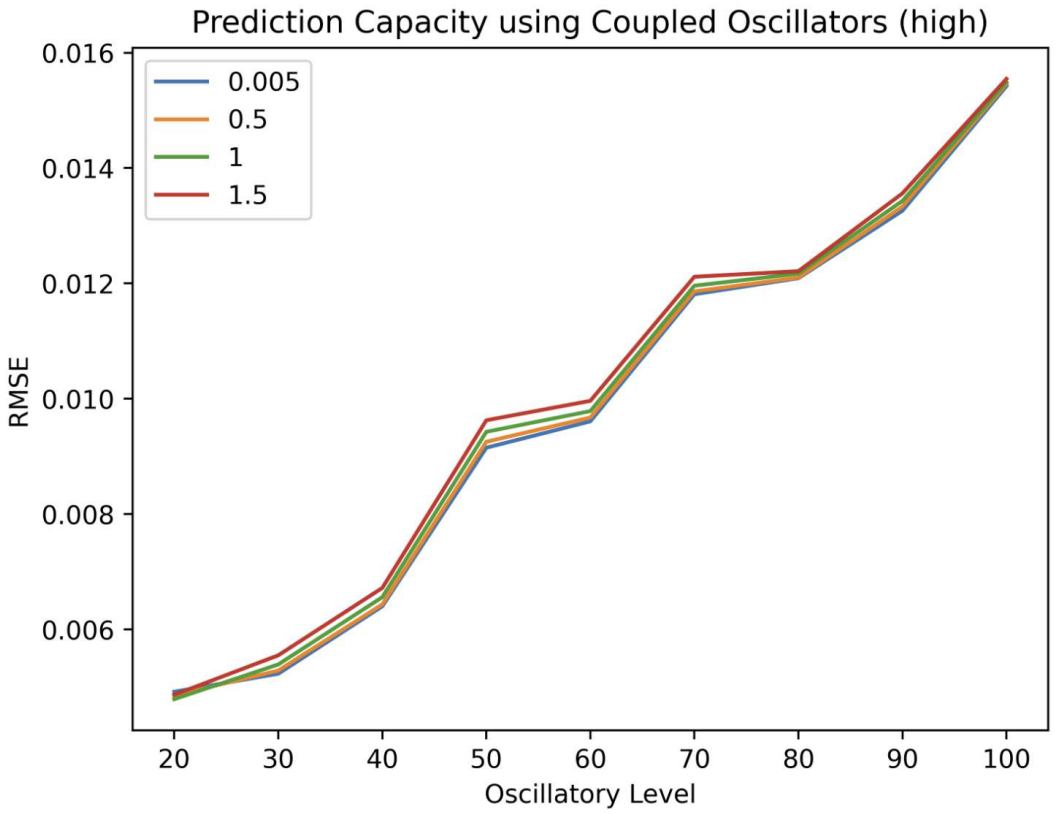


Figure 5.64: Average RMSE obtained for the coupled high automata reservoir using different values $\lambda > 0$ and additive violet noise.

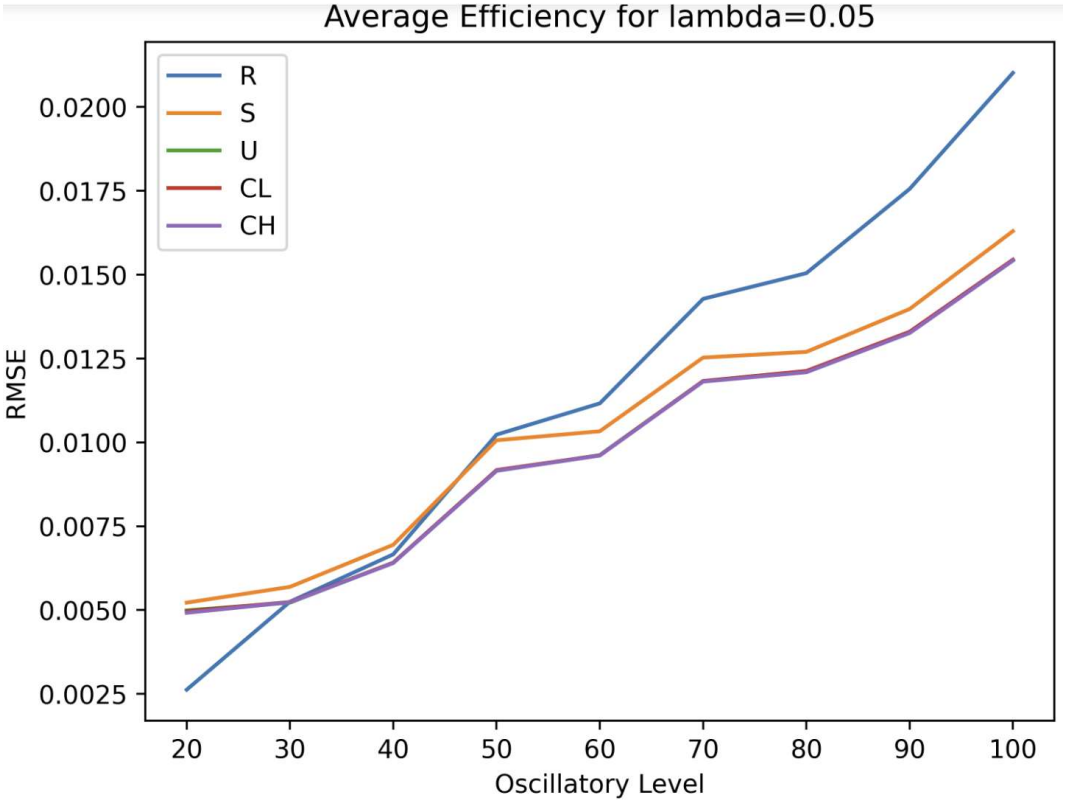


Figure 5.65: Average RMSE obtained for the different reservoirs using $\lambda = 0.05$. R stands for Ridge Regression, S for Single Oscillator, U for Uncoupled Oscillators, CL for Coupled Low Oscillators and CH for Coupled High Oscillators.

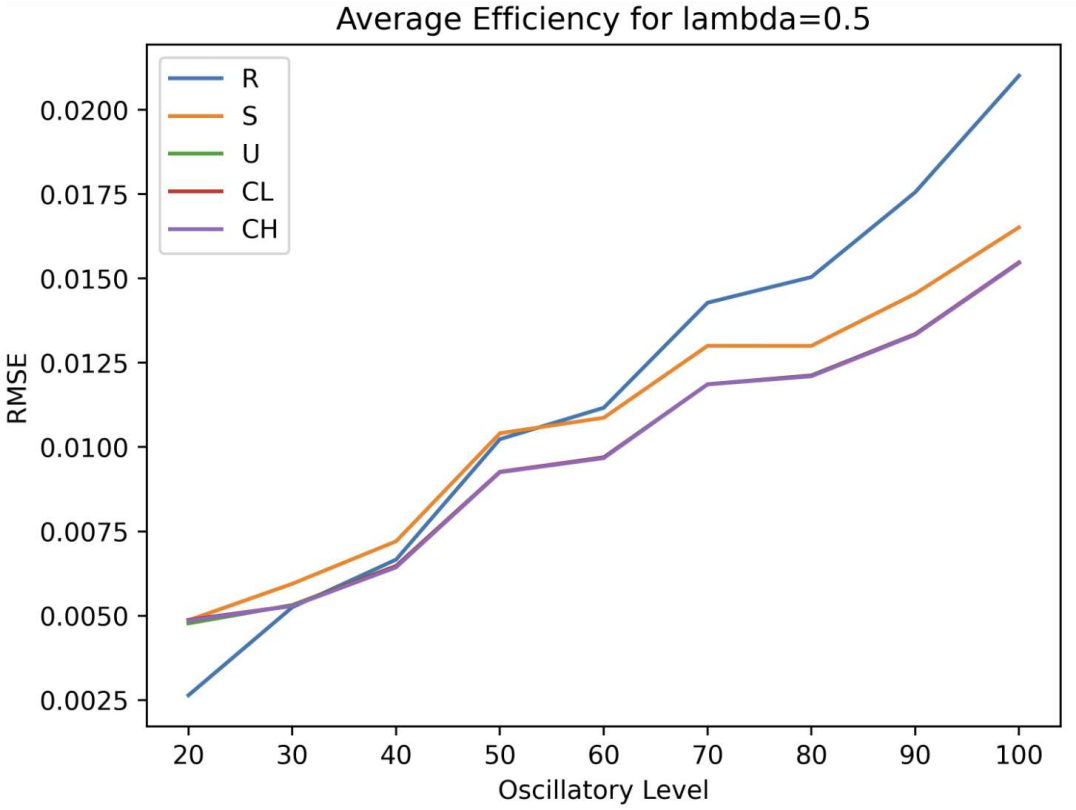


Figure 5.66: Average RMSE obtained for the different reservoirs using $\lambda = 0.5$. R stands for Ridge Regression, S for Single Oscillator, U for Uncoupled Oscillators, CL for Coupled Low Oscillators and CH for Coupled High Oscillators.

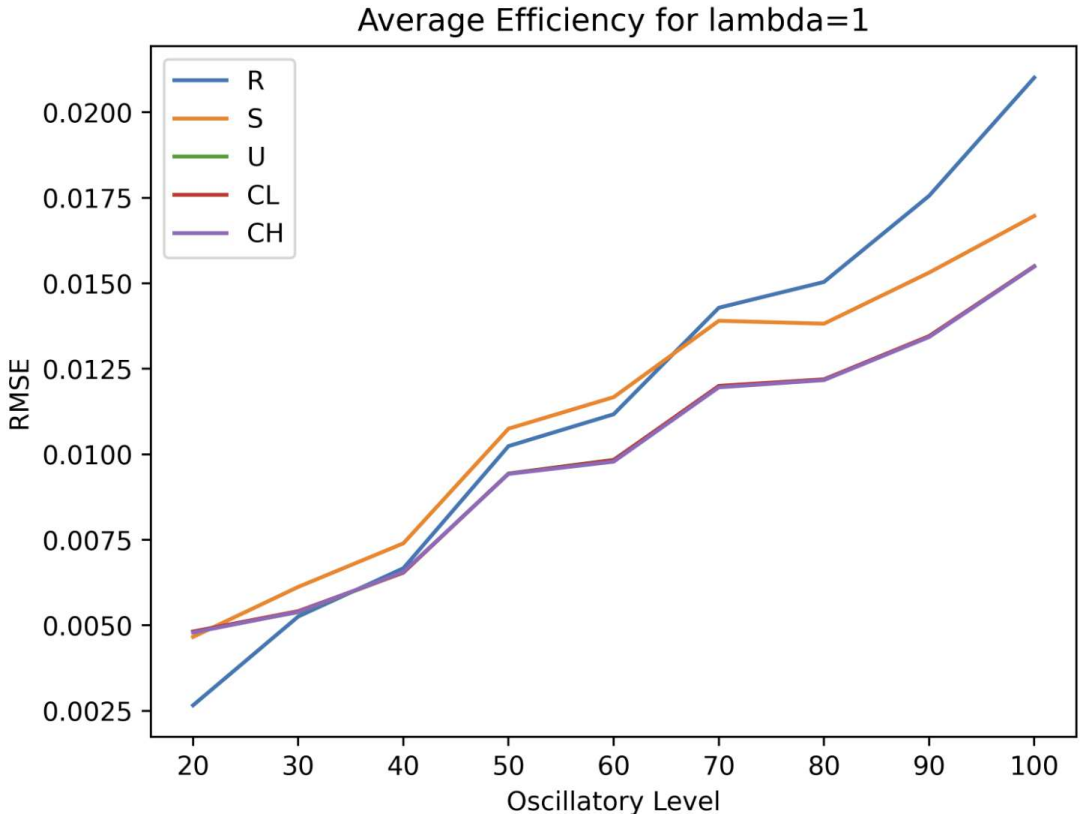


Figure 5.67: Average RMSE obtained for the different reservoirs using $\lambda = 1$. R stands for Ridge Regression, S for Single Oscillator, U for Uncoupled Oscillators, CL for Coupled Low Oscillators and CH for Coupled High Oscillators.

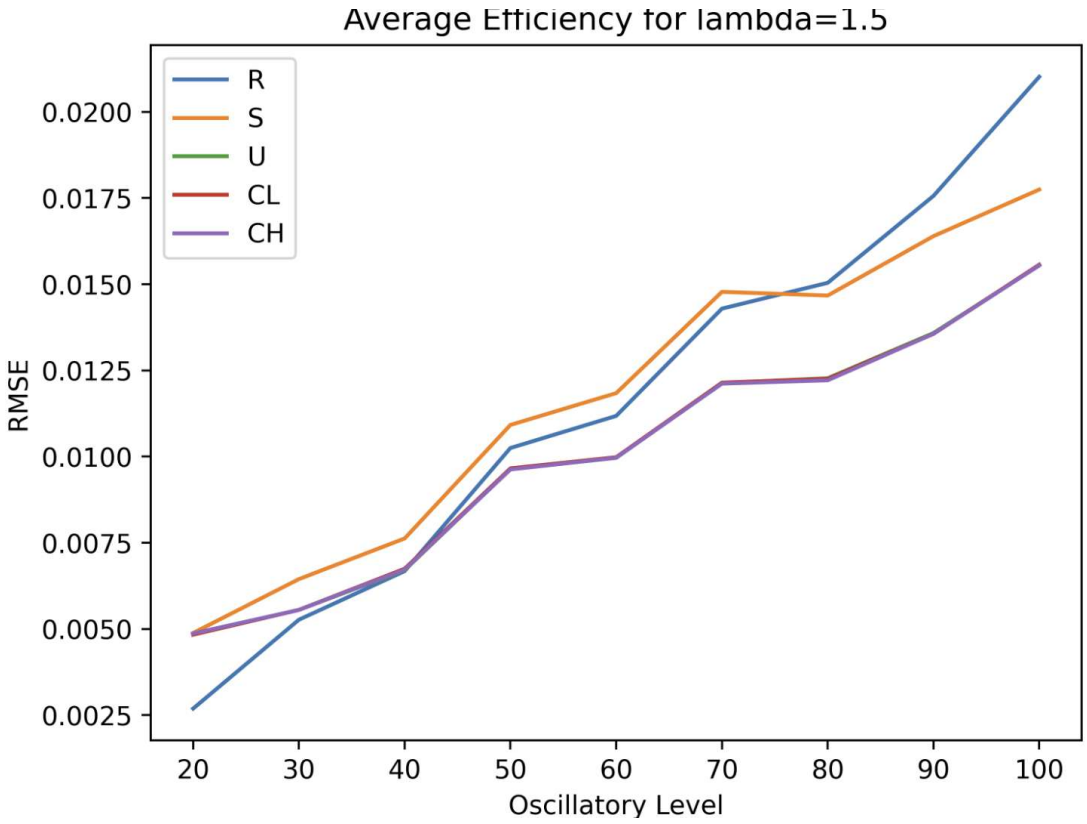


Figure 5.68: Average RMSE obtained for the different reservoirs using $\lambda = 1.5$. R stands for Ridge Regression, S for Single Oscillator, U for Uncoupled Oscillators, CL for Coupled Low Oscillators and CH for Coupled High Oscillators.

Bibliography

[Abrahão et al., 2020] Abrahão, F. S., D'Ottaviano, Í. M., Wehmuth, K., Doria, F. A., and Ziviani, A. (2020). Learning the undecidable from networked systems. In *UNRAVELLING COMPLEXITY: The Life and Work of Gregory Chaitin*, pages 131–179. World Scientific.

[Abrahao et al., 2019] Abrahao, F. S., Wehmuth, K., and Ziviani, A. (2019). Algorithmic networks: Central time to trigger expected emergent open-endedness. *Theoretical Computer Science*, 785:83–116.

[Abrahão and Zenil, 2022] Abrahão, F. S. and Zenil, H. (2022). Emergence and algorithmic information dynamics of systems and observers. *Philosophical Transactions of the Royal Society A*, 380(2227):20200429.

[Adamatzky, 2004] Adamatzky, A. (2004). Collision-based computing in belousov–zhabotinsky medium. *Chaos, Solitons & Fractals*, 21(5):1259–1264.

[Adamatzky and Costello, 2002] Adamatzky, A. and Costello, B. D. L. (2002). Experimental logical gates in a reaction-diffusion medium: The xor gate and beyond. *Physical Review E*, 66(4):046112.

[Adleman, 1994] Adleman, L. M. (1994). Molecular computation of solutions to combinatorial problems. *science*, 266(5187):1021–1024.

[Alaghi and Hayes, 2015] Alaghi, A. and Hayes, J. P. (2015). On the functions realized by stochastic computing circuits. In *Proceedings of the 25th edition on Great Lakes Symposium on VLSI*, pages 331–336.

[Amos and Goñi-Moreno, 2018] Amos, M. and Goñi-Moreno, A. (2018). Cellular computing and synthetic biology. *Computational matter*, pages 93–110.

[Baltussen et al., 2024] Baltussen, M. G., de Jong, T. J., Duez, Q., Robinson, W. E., and Huck, W. T. (2024). Chemical reservoir computation in a self-organizing reaction network. *Nature*, 631(8021):549–555.

[Belousov, 1959] Belousov, B. (1959). Периодически действующая реакция и ее механизм. *Сборник рефератов по радиационной медицине*, 147(145):145–147.

[Bennett, 1982] Bennett, C. H. (1982). The thermodynamics of computation—a review. *International Journal of Theoretical Physics*, 21:905–940.

[Cai et al., 2023] Cai, H., Ao, Z., Tian, C., Wu, Z., Liu, H., Tchieu, J., Gu, M., Mackie, K., and Guo, F. (2023). Brain organoid reservoir computing for artificial intelligence. *Nature Electronics*, 6(12):1032–1039.

[Cervera et al., 2019] Cervera, J., Manzanares, J. A., Mafe, S., and Levin, M. (2019). Synchronization of bioelectric oscillations in networks of nonexcitable cells: from single-cell to multicellular states. *The Journal of Physical Chemistry B*, 123(18):3924–3934.

[Costello and Adamatzky, 2005] Costello, B. D. L. and Adamatzky, A. (2005). Experimental implementation of collision-based gates in belousov–zhabotinsky medium. *Chaos, Solitons & Fractals*, 25(3):535–544.

[Crowley, 2000] Crowley, T. J. (2000). Causes of climate change over the past 1000 years. *Science*, 289(5477):270–277.

[De Castro, 2006] De Castro, L. N. (2006). *Fundamentals of natural computing: basic concepts, algorithms, and applications*. Chapman and Hall/CRC.

[Deyle and Sugihara, 2011] Deyle, E. R. and Sugihara, G. (2011). Generalized theorems for nonlinear state space reconstruction. *Plos one*, 6(3):e18295.

[Dueñas-Díez and Pérez-Mercader, 2019] Dueñas-Díez, M. and Pérez-Mercader, J. (2019). How chemistry computes: Language recognition by non-biochemical chemical automata. from finite automata to turing machines. *Iscience*, 19:514–526.

[Gauthier et al., 2021] Gauthier, D. J., Bollt, E., Griffith, A., and Barbosa, W. A. (2021). Next generation reservoir computing. *Nature communications*, 12(1):1–8.

[Goñi-Moreno, 2024] Goñi-Moreno, Á. (2024). Biocomputation: Moving beyond turing with living cellular computers. *Communications of the ACM*, 67(6):70–77.

[Goñi-Moreno and Nikel, 2019] Goñi-Moreno, A. and Nikel, P. I. (2019). High-performance biocomputing in synthetic biology–integrated transcriptional and metabolic circuits. *Frontiers in bioengineering and biotechnology*, 7:40.

[Grozinger et al., 2019] Grozinger, L., Amos, M., Gorochowski, T. E., Carbonell, P., Oyarzún, D. A., Stoof, R., Fellermann, H., Zuliani, P., Tas, H., and Goñi-Moreno, A. (2019). Pathways to cellular supremacy in biocomputing. *Nature communications*, 10(1):5250.

[Hopcroft, 2001] Hopcroft, J. (2001). Introduction to automata theory, languages, and computation.

[Horsman et al., 2017] Horsman, D., Kendon, V., Stepney, S., and Young, J. P. W. (2017). Abstraction and representation in living organisms: when does a biological system compute? *Representation and reality in humans, other living organisms and intelligent machines*, pages 91–116.

[Hyndman, 2018] Hyndman, R. (2018). *Forecasting: principles and practice*. OTexts.

[Jaeger, 2001] Jaeger, H. (2001). The "echo state" approach to analysing and training recurrent neural networks-with an erratum note. *Bonn, Germany: German National Research Center for Information Technology GMD Technical Report*, 148(34):13.

[Jaeger et al., 2024] Jaeger, J., Riedl, A., Djedovic, A., Vervaeke, J., and Walsh, D. (2024). Naturalizing relevance realization: Why agency and cognition are fundamentally not computational. *Frontiers in psychology*, 15:1362658.

[Johnson, 1990] Johnson, D. S. (1990). A catalog of complexity classes. In *Algorithms and complexity*, pages 67–161. Elsevier.

[Kaack et al., 2022] Kaack, L. H., Donti, P. L., Strubell, E., Kamiya, G., Creutzig, F., and Rolnick, D. (2022). Aligning artificial intelligence with climate change mitigation. *Nature Climate Change*, 12(6):518–527.

[Kalinin and Berloff, 2022] Kalinin, K. P. and Berloff, N. G. (2022). Computational complexity continuum within ising formulation of np problems. *Communications Physics*, 5(1):20.

[Katsikis et al., 2015] Katsikis, G., Cybulski, J. S., and Prakash, M. (2015). Synchronous universal droplet logic and control. *Nature Physics*, 11(7):588–596.

[Kauffman, 1969] Kauffman, S. A. (1969). Metabolic stability and epigenesis in randomly constructed genetic nets. *Journal of theoretical biology*, 22(3):437–467.

[Kim et al., 2016] Kim, D., Kung, J., Chai, S., Yalamanchili, S., and Mukhopadhyay, S. (2016). Neurocube: A programmable digital neuromorphic architecture with high-density 3d memory. *ACM SIGARCH Computer Architecture News*, 44(3):380–392.

[Kuhnert et al., 1989] Kuhnert, L., Agladze, K., and Krinsky, V. (1989). Image processing using light-sensitive chemical waves. *Nature*, 337(6204):244–247.

[Leffer, 2023] Leffer, L. (2023). The ai boom could use a shocking amount of electricity. *Sci. Am.*

[Liu et al., 2022] Liu, Y., Pérez-Mercader, J., and Kiss, I. Z. (2022). Synchronization of belousov–zhabotinsky oscillators with electrochemical coupling in a spontaneous process. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 32(9).

[López-Díaz et al., 2023] López-Díaz, A. J., Sánchez-Puig, F., and Gershenson, C. (2023). Temporal, structural, and functional heterogeneities extend criticality and antifragility in random boolean networks. *Entropy*, 25(2):254.

[Lu et al., 2024] Lu, W. D., Teuscher, C., Sarles, S. A., Yang, Y., Todri-Sanial, A., and Zhu, X.-B. (2024). A perfect storm and a new dawn for unconventional computing technologies. *npj Unconventional Computing*, 1(1):9.

[Lucas, 2014] Lucas, A. (2014). Ising formulations of many np problems. *Frontiers in physics*, 2:5.

[Lukoševičius and Jaeger, 2009] Lukoševičius, M. and Jaeger, H. (2009). Reservoir computing approaches to recurrent neural network training. *Computer science review*, 3(3):127–149.

[Maass et al., 2002] Maass, W., Natschläger, T., and Markram, H. (2002). Real-time computing without stable states: A new framework for neural computation based on perturbations. *Neural computation*, 14(11):2531–2560.

[Mohseni et al., 2022] Mohseni, N., McMahon, P. L., and Byrnes, T. (2022). Ising machines as hardware solvers of combinatorial optimization problems. *Nature Reviews Physics*, 4(6):363–379.

[Moore and Mertens, 2011] Moore, C. and Mertens, S. (2011). *The nature of computation*. Oxford University Press.

[Motwani, 1995] Motwani, R. (1995). *Randomized Algor ithms*. Cambridge University Press.

[Müller et al., 2022] Müller, S., Flamm, C., and Stadler, P. F. (2022). What makes a reaction network "chemical"? *Journal of cheminformatics*, 14(1):63.

[Muñuzuri and Pérez-Mercader, 2022] Muñuzuri, A. P. and Pérez-Mercader, J. (2022). Unified representation of life's basic properties by a 3-species stochastic cubic autocatalytic reaction-diffusion system of equations. *Physics of Life Reviews*, 41:64–83.

[Parrilla-Gutierrez et al., 2020] Parrilla-Gutierrez, J. M., Sharma, A., Tsuda, S., Cooper, G. J., Aragon-Camarasa, G., Donkers, K., and Cronin, L. (2020). A programmable chemical computer with memory and pattern recognition. *Nature communications*, 11(1):1442.

[Pikovsky et al., 2001] Pikovsky, A., Rosenblum, M., Kurths, J., and Synchronization, A. (2001). A universal concept in nonlinear sciences. *Self*, 2:3.

[Roy and Majumdar, 2022] Roy, S. and Majumdar, S. (2022). *Noise and Randomness in Living System*. Springer.

[Sakemi et al., 2020] Sakemi, Y., Morino, K., Leleu, T., and Aihara, K. (2020). Model-size reduction for reservoir computing by concatenating internal states through time. *Scientific reports*, 10(1):21794.

[Schuman et al., 2017] Schuman, C. D., Potok, T. E., Patton, R. M., Birdwell, J. D., Dean, M. E., Rose, G. S., and Plank, J. S. (2017). A survey of neuromorphic computing and neural networks in hardware. *arXiv preprint arXiv:1705.06963*.

[Suzuno et al., 2014] Suzuno, K., Ueyama, D., Branicki, M., Tóth, R., Braun, A., and Lagzi, I. (2014). Maze solving using fatty acid chemistry. *Langmuir*, 30(31):9251–9255.

[Takens, 2006] Takens, F. (2006). Detecting strange attractors in turbulence. In *Dynamical Systems and Turbulence, Warwick 1980: proceedings of a symposium held at the University of Warwick 1979/80*, pages 366–381. Springer.

[Tanaka et al., 2019] Tanaka, G., Yamane, T., Héroux, J. B., Nakane, R., Kanazawa, N., Takeda, S., Numata, H., Nakano, D., and Hirose, A. (2019). Recent advances in physical reservoir computing: A review. *Neural Networks*, 115:100–123.

[Toth et al., 2009] Toth, R., Stone, C., de Lacy Costello, B., Adamatzky, A., and Bull, L. (2009). Simple collision-based chemical logic gates with adaptive computing. *International Journal of Nanotechnology and Molecular Computation (IJNMC)*, 1(3):1–16.

[Turing, 1936] Turing, A. (1936). On computable numbers, with an application to the entscheidungs problem. *Proceedings of the London Mathematical Society Series/2 (42)*, pages 230–42.

[Ushio et al., 2023] Ushio, M., Watanabe, K., Fukuda, Y., Tokudome, Y., and Nakajima, K. (2023). Computational capability of ecological dynamics. *Royal Society Open Science*, 10(4):221614.

[Vasseur and Yodzis, 2004] Vasseur, D. A. and Yodzis, P. (2004). The color of environmental noise. *Ecology*, 85(4):1146–1152.

[Walker and Davies, 2013] Walker, S. I. and Davies, P. C. (2013). The algorithmic origins of life. *Journal of the Royal Society Interface*, 10(79):20120869.

[Wolpert, 2001] Wolpert, D. H. (2001). Computational capabilities of physical systems. *Physical Review E*, 65(1):016128.

[Xu et al., 2023] Xu, M., Chen, X., Guo, Y., Wang, Y., Qiu, D., Du, X., Cui, Y., Wang, X., and Xiong, J. (2023). Reconfigurable neuromorphic computing: Materials, devices, and integration. *Advanced Materials*, 35(51):2301063.

[Zhabotinsky, 1964] Zhabotinsky, A. M. (1964). Periodical oxidation of malonic acid in solution (a study of the belousov reaction kinetics). *Biofizika*, 9:306–311.

[Zhang and Cornelius, 2023] Zhang, Y. and Cornelius, S. P. (2023). Catch-22s of reservoir computing. *Physical Review Research*, 5(3):033213.

[Zhong et al., 2021] Zhong, Y., Tang, J., Li, X., Gao, B., Qian, H., and Wu, H. (2021). Dynamic memristor-based reservoir computing for high-efficiency temporal signal processing. *Nature communications*, 12(1):408.

[Ziegler, 2020] Ziegler, M. (2020). Novel hardware and concepts for unconventional computing. *Scientific reports*, 10(1):11843.