Phase space reconstruction of semiconductor laser dynamics using reservoir computing

Alejandro Cunillera Pérez

Master’s Thesis

Master’s degree in Physics of Complex Systems at the UNIVERSITAT DE LES ILLES BALEARS

Date 08/01/2019

UIB Master’s Thesis Supervisor: Ingo Fischer

UIB Master’s Thesis Co-Supervisor: Miguel C. Soriano
Abstract

The geometry of the phase space of a dynamical system contains information about the dynamics of the system. The Takens embedding theorem shows that the full dynamical evolution of the system can be extracted from the structure of the phase space and it can be reconstructed just by measuring one of the variables of the dynamical system. This result has many applications such as recovering lost time series data, testing data encryption security in chaotic synchronization cryptography or data forecasting. This can also be used in control engineering to create a state observer. There are real-world systems that have some variables that can be measured easily but it might be unfeasible to measure the others. In this work we implement reservoir computing techniques to reconstruct and forecast the dynamics of a 3-dimensional dynamical system that describes the evolution of an optically injected class B semiconductor laser. This system has 3 variables, the amplitude, phase and carrier density but usually only the first one is measured. A reservoir computing state observer is utilized to infer the evolution of unmeasured variables provided that time series are available for measurements of one of the dynamical variables. Later on, an autonomous reservoir computing algorithm is used to predict the evolution of the system dynamics. In this same context, a hybrid method is also considered by introducing an approximate mathematical model into the autonomous algorithm in order to extend the prediction horizon.
## Contents

1 **Introduction** ........................................... 3

1.1 Objectives and outline of the master thesis .......................... 5

1.2 Observability in nonlinear dynamical systems. A class B semiconductor laser rate equations ..................................................... 6

1.3 Reservoir computing phase space reconstruction and prediction methods ........................................... 9

1.3.1 Cross-prediction method ................................... 9

1.3.2 Autonomous reservoir computing prediction .......................... 11

1.3.3 Hybrid prediction scheme ................................... 12

1.3.4 General considerations of the prediction methods ................. 14

1.4 Differential topology in a nutshell ................................... 15

2 **Methodology** .................................................. 19

2.1 Cross-prediction algorithm ................................... 19

2.2 Autonomous reservoir computing predictor algorithm ............... 22

2.3 Hybrid scheme predictor algorithm ................................ 23

3 **Results** ..................................................... 25

3.1 Cross-prediction results ................................... 25

3.2 Reservoir computing forecasting results ............................... 49

4 **Conclusion** .................................................... 55
1 Introduction

The need to process data stands as a common factor across many scientific disciplines. Data is collected from observations, experiments and measurements. Statistics and data science have been developed to analyze, understand and organize data. Experimental data is compared with that obtained from models and simulations in order to check the accuracy of our models. If they are accurate, they can be used for forecasting new data. Ultimately, in control theory, data is linked to the concept of observability, by which it is important to have access to measurements of some of the dynamical variables of the system in order to determine the state of the system.

However, there are times when the availability of data is limited. Experimental issues and data losses may lead to incomplete time series data. It might be difficult, unfeasible or impossible to make measurements of one or more of the dynamical variables of the system, so the time series for the measurements of these variables are not available. This can be a great problem, especially when forecasting or controlling the dynamics of a system. Rudolf E. Kalman solved the observability problem for linear systems [12], but this is still not the case for nonlinear systems.

From now on we are considering a system whose evolution is described by a dynamical system. The dynamical variables of a dynamical system are all the variables whose evolution is described by its equations. If the system is nonlinear and shows chaotic dynamics, an initial state in its phase space tends to evolve into an attractor, so it can be said that the phase space includes an attractor. The key point here is that the geometry of the attractor contains crucial information about the dynamics of the dynamical system, so we may obtain information about the dynamical variables and even predict them if we have a good understanding about the properties of the attractor. In this work we want to use the information contained in the geometry of the phase space of a system to reconstruct the dynamics of some of its dynamical variables. We also utilize this information to predict the dynamics of the system.

Phase space reconstruction is a technique used to infer the dynamics of a nonlinear chaotic dynamical system from measurements of the attractor contained in its phase space. This method takes advantage of the information contained in the topology of the attractor. Given time series for measurements of at least one of the dynamical variables of the system it is possible to construct an attractor that has the same topological properties of the one included in the phase space of the system. Then it may be possible to obtain time series for all the other dynamical variables. This technique can also be used for forecasting and recovering lost data from time series of dynamical systems [15].

This method consists on mapping the attractor onto a higher dimensional space where it can be unfolded. In mathematics, the unfolding procedure is called embedding. The
maximum amount of information of the geometry of an attractor can be obtained only when it is completely unfolded. For instance, there are many ways of defining the dimension of an attractor, but it can be measured only when it is embedded in a higher order space. Recovering the data is then just a matter of translating in a smart way the system’s trajectory in that space into the time series for the dynamical variables that we want to infer or predict.

A way to implement this technique in practice is by using the delay embedding procedure. The mathematics behind this approach are introduced in Section 1.4. It is based on some deep results and concepts in differential topology. One of these results is Takens embedding theorem [25], which shows that 2n + 1 time-delayed measurements of one dynamical variable of an n-dimensional dynamical system \{y(t), y(t − τ),..., y(t − 2nτ)\} are enough to unfold the attractor in a (2n + 1)-dimensional space, where \(τ\) is the sampling time of the time series. Packard et al. [20] show that the unfolding can also be done with the set \{y(t), \(\frac{dy(t)}{dt}\),..., \(\frac{d^2y(t)}{dt^2}\)\}. This set is equivalent to the previous one when the sampling time is small, as the derivatives can be approximated by linear combinations of the delayed terms of the time series. Aeyels [2] found this result independently, but from a control theory point of view.

In this work we use a machine learning technique called reservoir computing to implement a technique that is similar to the delay embedding, the reservoir computing phase space reconstruction. Reservoir computing was introduced independently by Jaeger under the name of echo state network [10], and by Maass et al., under the name of liquid-state machine [16]. First, the time series for the measured dynamical variables are injected into the reservoir, which is a random neural network whose nodes evolve nonlinearly. The input connections are implemented using usually fixed random weights. The input data makes the reservoir evolve and results in a nonlinear mapping of the input state onto a high-dimensional reservoir state. The output is obtained as a linear transformation of the state of the reservoir. At this point a sufficiently large data set of the dynamical variables that we want to infer is needed. The weights of this linear transformation are chosen by fitting the reservoir output to the time series of the target dynamical variables. This fitting phase is called training. Once the output weights are determined, if more input data is fed to the reservoir the output data might accurately reproduce the target dynamical variables. The key point here is that the size of the reservoir, which is given by the number of nodes in the network, has to be much larger than the dimension of the dynamical system. The reservoir has the role of being equivalent to the high dimensional space where the dynamics is unfolded. This technique is discussed in more detail in Section 1.3, and the methodology used is described in Section 2.

This phase space reconstruction implementation in reservoir computing was introduced by Jaeger and Haas [11], although in this work we follow the procedure described by Ott et al. [15]. In the latter work this technique is applied for inferring 2 of the 3 variables of the Rössler and the Lorenz dynamical systems, which are two paradigmatical examples of three-dimensional chaotic dynamical systems. In the same reference this approach is also
used to reconstruct the spatial dynamics produced by the Kuramoto-Sivashinsky equation, which shows spatiotemporal chaos. This method is named cross-prediction by Zimmermann and Parlitz [28], who use it for inferring the dynamics of two partial differential equation models that describe excitable media in cardiac tissue.

Ott et al. also proposed two variants of the reservoir computing scheme to predict the dynamics of dynamical systems. An autonomous reservoir computing predictor scheme is used in Ref. [21] to forecast the dynamics of the Kuramoto-Sivashinsky equation. This method requires introducing time series from all the dynamical variables of the dynamical system as input to the reservoir and the output is fitted to the values of the dynamical variables for the next time step. After the training phase, the output is injected back as the input introducing a feedback loop, so this scheme can be used for time series short-term prediction and long-term autonomous generation. A hybrid scheme was introduced in Ref. [22]. It combines the previous autonomous reservoir computing predictor with an approximate mathematical model for the dynamical system in a way that is similar to the predictor-corrector methods in numerical analysis. The mathematical model produces an initial guess of the variables at the next time step and the reservoir is trained to improve this prediction. It can be used even when there is a parameter mismatch between a parameter used in the model and its exact value. In [22] it is shown that, given a good guess of the underlying model, the hybrid scheme is able to predict the dynamics of the Lorenz dynamical system and the Kuramoto-Sivashinsky equation accurately for longer time than the autonomous reservoir computing predictor and with less computational cost.

The cross-prediction method can also be used to create a state observer, which could find applications in control engineering. This might be of practical use when it is easy to make measurements of some dynamical variables of a system but some are difficult to be measured and it is important to have approximate time series of the latter. It might also be applied for reconstructing missing data. This technique could also be used for testing data encryption security in chaotic laser synchronization based cryptography [4, 9]. In addition, the hybrid scheme setup could be used for improving weather forecasting. Many approximate models are used for doing long range predictions. These reservoir computing methods can even be implemented experimentally in a field-programmable gate array, FPGA [8], in a semiconductor laser optical network [5] and in a delay system with a single nonlinear node [3].

1.1 Objectives and outline of the master thesis

The main goal of this master thesis is to test whether the cross-prediction method can be used to reconstruct the dynamics of a three-dimensional nonlinear dynamical system, focusing on a model of an optically injected class B semiconductor laser. This model has five parame-
Phase space reconstruction of sc. laser dynamics using reservoir computing  A.Cunillera

ters that can be determined experimentally. It is an approximate model that, however, has proven to provide a good description for single mode semiconductor lasers [18, 27]. This dynamical system shows a rich variety of dynamics when varying two key parameters, including chaotic behaviour. The complex electric field that is produced by the laser can be described in terms of its real and imaginary components or of its amplitude and phase. The cross-prediction method is used to infer the dynamics of two of the three dynamical variables when measurements for the remaining one are unlimitedly available. The dynamics is inferred for each of the sets of coordinates that describe the complex electric field. The robustness of the prediction is also tested by adding noise to the measurements.

Another goal is to compare the performance of the autonomous reservoir computing and the hybrid schemes when used for forecasting the dynamics of the laser. The accuracy of the approximate mathematical model included in the hybrid scheme is tuned by introducing a mismatch in one of the parameters of the model.

The semiconductor laser rate equations are described in Section 1.2. The reservoir computing techniques used are briefly introduced in Section 1.3. A small discussion of the mathematical concepts behind the reservoir computing prediction methods is presented in Section 1.4. Section 2 covers the algorithms used for implementing the cross-prediction, the autonomous reservoir computing predictor and the hybrid scheme methods. Results are discussed in Section 3. Conclusions are presented in Section 4.

1.2 Observability in nonlinear dynamical systems. A class B semiconductor laser rate equations

We consider an optically injected class B semiconductor laser, the dynamics of which can be described by the following rate equations:

\begin{align*}
\dot{E} &= K + \left( \frac{1}{2}(1 + i\alpha)n - i\omega \right) E, \\
\dot{n} &= -2\Gamma n - (1 + 2Bn)(|E|^2 - 1),
\end{align*}

where \( E = \text{Re}(E) + i\text{Im}(E) \) is the complex electric field and \( n \) is the carrier density above threshold, which is equivalent to the population inversion that is needed to maintain the stimulated emission in gas lasers. The dot indicates the time derivative. \( B \) is the photon lifetime, \( \Gamma \) is the damping rate of the relaxation oscillation, \( K \) is the dimensionless injected field strength, \( \omega \) is the detuning between the injected field and the free-running laser frequencies and \( \alpha \) is the linewidth enhancement factor. There is an alternative description of the electric
field, $E = A \exp(-i\phi)$, being $A$ the amplitude of the field and $\phi$ its phase. Therefore, the intensity of the electric field is $I = A^2 = \text{Re}(E)^2 + \text{Im}(E)^2$.

This 3-dimensional dynamical system shows a rich variety of dynamics, including chaotic behaviour. In this work we fix $B$, $\Gamma$ and $\alpha$ and we aim to reconstruct and predict the laser dynamics when changing the injected field strength $K$ and the detuning $\omega$. These two parameters are of special experimental relevance since they are well accessible operational parameters. They can be adjusted via the injection laser that drives the semiconductor laser dynamics. Therefore, we explore the 2-dimensional parameter space for $K$ and $\omega$. This exploration shows that there are six topologically different regimes for the laser dynamics [27], that is to say, there are six different types of phase portraits that can be obtained for six different sets of values for $K$ and $\omega$ that cannot be mapped onto each other by a continuous and invertible map that preserves the direction of time [14].

The values for $K$ and $\omega$ used for obtaining results for the six qualitatively different dynamics are presented in Table 1. The other parameters are kept fixed as follows: $B = 0.015$, $\Gamma = 0.035$ and $\alpha = 2$. Figure 1 shows the experimental stability map of a semiconductor laser, which includes regions of stable, periodic and chaotic dynamics. In the chaotic regime the maximum Lyapunov exponent is around 0.053.

<table>
<thead>
<tr>
<th>Parameter set</th>
<th>$K$</th>
<th>$\omega$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stationary point dynamics</td>
<td>0.27</td>
<td>-0.28</td>
</tr>
<tr>
<td>Bounded-phase limit cycle dynamics</td>
<td>0.27</td>
<td>-0.2</td>
</tr>
<tr>
<td>Unbounded-phase limit cycle dynamics</td>
<td>0.1</td>
<td>-1.2</td>
</tr>
<tr>
<td>Quasiperiodic dynamics on an invariant torus</td>
<td>0.18</td>
<td>-1.2</td>
</tr>
<tr>
<td>Period 3 dynamics on a torus</td>
<td>0.2</td>
<td>-1.25</td>
</tr>
<tr>
<td>Chaotic laser dynamics</td>
<td>0.18</td>
<td>0.6</td>
</tr>
</tbody>
</table>

Table 1: Values of $K$ and $\omega$ for the 6 topologically different laser dynamics. $B = 0.015$, $\Gamma = 0.035$ and $\alpha = 2$ are kept fixed.
The observability of the dynamical variables of a semiconductor laser is a fundamental problem when working with those devices. Recall the amplitude and phase description of the electric field, \( E = A \exp(-i\phi) \). High-speed photodiodes and fast real-time scopes are needed to measure the dynamics of the laser intensity. The amplitude \( A \) can be obtained by taking the square root of the intensity. However, measuring the phase and the carrier density requires a more elaborate setup. A phase-space tomography technique can be used for measuring these three variables simultaneously [6, 13].

In many cases the unavailability of measurements for the phase and the carrier density is a technical limitation, but it is sometimes feasible to estimate their values. The cross-prediction technique may be a more accurate alternative to estimations when the phase and carrier density measuring setup can be implemented and used for a certain amount of time. This way, if enough data is gathered, the cross-prediction algorithm might reconstruct the phase space when feeding the reservoir with the easily-obtainable time series data for the amplitude.

In contrast, the autonomous reservoir computing and the hybrid scheme predictors can
be used to predict the dynamics of a semiconductor laser provided that there is enough data available for training the algorithms. The hybrid scheme method is particularly relevant for this system since the rate equations (1, 2) are an approximate model for the laser dynamics. For example, the linewidth enhancement factor $\alpha$ depends on the detuning and the carrier density, so it is sometimes difficult to determine this parameter accurately. Therefore, there might be a parameter mismatch between the exact value and the approximate one used in the model. This would cause a loss of accuracy in the model predictions. The hybrid scheme setup can be used to improve the accuracy of these predictions.

Now that the semiconductor laser model has been presented we introduce the reservoir computing techniques used to reconstruct and predict its dynamics in a general way.

\section{1.3 Reservoir computing phase space reconstruction and prediction methods}

\subsection{1.3.1 Cross-prediction method}

Let us consider a dynamical system $\dot{x}(t) = g(x(t))$ that describes the dynamics of a system, where $x$ are its dynamical variables and $t \geq 0$. Let $u(t) \in \mathbb{R}^M$ and $v(t) \in \mathbb{R}^P$ be two vectors that represent two different sets of measurable variables that depend on the dynamical state $x(t)$ at time $t$.

Let us assume that we have time series of measurements $v(t)$ over a sufficiently long but finite time period $0 \leq t \leq T$ but we know $u(t)$ over a longer time. The aim of the cross-prediction method is to reconstruct $v(t)$ accurately beyond time $T$, when we can only measure $u(t)$. In the following discussion we are considering time as a discrete variable with a constant rate, so $U = \{u(t) : t \geq 0\}$ and $V = \{v(t) : 0 \leq t \leq T\}$ are time series sampled at a constant time rate $\Delta t$.

We use a reservoir computing algorithm to implement the cross-prediction procedure. This approach was introduced by Jaeger and Haas [11], while the algorithm that we are using was described by Ott et al. [15] and it was called cross-prediction method by Parlitz and Zimmermann [28]. We construct a reservoir, which is a random neural network with $N \gg M$ nodes. The reservoir is a high order nonlinear dynamical system whose nodes evolve in time. It creates the high dimensional space in which we want to unfold the system dynamics. We define the state of the reservoir at time $t$, $r(t) \in \mathbb{R}^N$, which is a vector that contains the states of all the nodes.

The idea of the cross-prediction method can be described in a few words. The reservoir is fed with the time series of the variables that can be measured without time limit and it
produces raw output that still may not reproduce well the dynamics of the system. This can be solved by comparing the raw output with the known time series of the variables that we want to reconstruct. The comparison process is called training. If it is satisfactory, the output of the reservoir will be an approximation of the variables that we want to infer.

The input data \( u(t) \) is coupled to the reservoir through the input layer \( In[u(t)] \), where \( In : \mathbb{R}^M \rightarrow \mathbb{R}^N \) is a map, so that the vector \( u(t) \) is mapped onto the reservoir nodes. The reservoir evolves due to this coupling,

\[
\mathbf{r}(t) = R[In[u(t)], \mathbf{r}(t - \Delta t)],
\]

where \( R : \mathbb{R}^N \times \mathbb{R}^N \rightarrow \mathbb{R}^N \) is a nonlinear function called the reservoir nonlinearity. The reservoir produces a raw output vector \( Out[\mathbf{r}(t), \mathbf{p}] \), where \( Out : \mathbb{R}^N \times \mathbb{R}^S \rightarrow \mathbb{R}^P \) is a map and \( \mathbf{p} \in \mathbb{R}^S \) is a set of \( S \) parameters that can be adjusted, as we aim to approximate the reservoir output to \( \mathbf{v}(t) \). The time series for the state of the reservoir \( \mathbf{r}(t) \) has to be stored for \( 0 \leq t \leq T \).

We can distinguish two phases in the cross-prediction, the first one, for \( 0 \leq t \leq T \), is the training phase. In this phase we determine the parameters \( \mathbf{p}^* \) that minimize the error between the reservoir output \( Out[\mathbf{r}(t), \mathbf{p}^*] \) and the measurements \( \mathbf{v}(t) \), for \( 0 \leq t \leq T \), by using a linear regression. The second phase takes place for \( t > T \) and we reconstruct the unmeasured variables during it. The input data \( u(t) \) is mapped onto the reservoir for \( t > T \) and, if the training was successful, the output will be an approximation to the unmeasured variables, \( Out[\mathbf{r}(t), \mathbf{p}^*] = \hat{\mathbf{v}}(t) \approx \mathbf{v}(t) \), for \( t > T \). A scheme of the cross-prediction algorithm is presented in Figure 2.

![Figure 2: Scheme of the cross-prediction algorithm.](image-url)
cardiac tissue described by 2-dimensional partial differential equations [28]. In this work we are applying this technique to the rate equations of a semiconductor laser (Eqs. 1-2).

1.3.2 Autonomous reservoir computing prediction

Consider now the problem of forecasting dynamics. If we understand the mechanisms underlying a dynamical system we may make a mathematical model and use it to predict its dynamics. Let \( u \in \mathbb{R}^M \) be a set of \( M \) measurable variables that depend on the dynamics of the system. An approximate mathematical model that describes their evolution is, for instance,

\[
\text{Model}[u(t)] = \hat{u}(t + \Delta t) \approx u(t + \Delta t),
\]

where \( \text{Model} : \mathbb{R}^M \rightarrow \mathbb{R}^M \) is a function and \( \hat{u} \) is the prediction of the model. To forecast \( \hat{u}(t) \) for \( t > 0 \), an initial condition \( u(0) \) is needed.

In this context, reservoir computing can also be used for creating an artificial dynamical system that emulates the dynamics of another dynamical system. Here, we introduce how the autonomous reservoir computing method works.

The autonomous reservoir computing prediction method is similar to the cross-prediction algorithm. In this case we have time series for the full set of measurements only in the training interval \( 0 \leq t \leq T \). Let \( u(t) \in \mathbb{R}^M \) be measurements for all the dynamical variables at time \( t \). As stated before, we know \( u(t) \) for \( 0 \leq t \leq T \) and we want to predict the time series beyond time \( T \). The scheme of the setup of the autonomous predictor is shown in Figure 3.

![Figure 3: Scheme of the autonomous reservoir computing predictor.](image)

In the training phase we feed the reservoir with \( u(t) \) through the input map \( \text{In}[u(t)] \),
For every time step $0 \leq t < T$ the next state of the reservoir is computed

$$r(t + \Delta t) = R[In[u(t)], r(t)],$$  \hspace{1cm} (5)

where $R : \mathbb{R}^N \times \mathbb{R}^N \rightarrow \mathbb{R}^N$ is again a nonlinear function. We aim to predict the measurements for the next time step, $t + \Delta t$, so the output should be close to them, $Out[r(t + \Delta t), p^*] \approx u(t + \Delta t)$, for every $0 \leq t < T$. The parameters $p^*$ are computed as in the cross-prediction method, using a linear regression.

The prediction phase takes place for $t \geq T$. We use the output of the reservoir $\hat{u}(t + \Delta t) = Out[r(t + \Delta t), p^*]$ as the input for the next time step. This feedback loop produces the reservoir prediction $\hat{u}(t)$ for $t > T$. In this second phase the input map would be $In[\hat{u}(t)]$, and the reservoir evolution equation,

$$r(t + \Delta t) = R[In[\hat{u}(t)], r(t)],$$  \hspace{1cm} (6)

Ott et al. use the autonomous reservoir computing predictor for forecasting the dynamics of the Kuramoto-Sivashinsky equation [21, 22] and the Lorenz dynamical system [22].

### 1.3.3 Hybrid prediction scheme

The hybrid scheme predictor combines the autonomous reservoir computing approach with a mathematical model. This technique may improve the performance of these two algorithms under certain conditions. The main difference between the autonomous and the hybrid procedures is that in the latter method a mathematical model (4) is used to calculate an initial guess of the measurements $u(t)$ at the next time step, $Model[u(t)]$. This is mapped onto the reservoir together with the input data and to the output layer. Figure 4 shows the hybrid scheme setup.
The training phase takes place first, for $0 \leq t < T$. Data and the mathematical model’s initial guess are mapped onto the reservoir through the input map $In[Model[u(t)], u(t)]$, where $In : \mathbb{R}^M \times \mathbb{R}^M \rightarrow \mathbb{R}^N$. The reservoir evolves analogously as in the autonomous algorithm,

$$ r(t + \Delta t) = R[In[Model[u(t)], u(t)], r(t)], $$

where $R : \mathbb{R}^N \times \mathbb{R}^N \rightarrow \mathbb{R}^N$ is the reservoir nonlinearity. As stated before, the mathematical model prediction is also introduced in the output layer, $Out[Model[\hat{u}(t)], r(t + \Delta t), p^*] \approx \hat{u}(t + \Delta t)$. The parameters $p^*$ are determined in the same way as in the autonomous method.

In the prediction phase, we proceed exactly as in the autonomous reservoir computing method. After closing the feedback loop the reservoir evolves accordingly to the following equation

$$ r(t + \Delta t) = R[In[Model[\hat{u}(t)], \hat{u}(t)], r(t)], $$

where $\hat{u}(t + \Delta t) = Out[Model[\hat{u}(t)], r(t + \Delta t), p^*]$ is the prediction, for $t > T$, and $\hat{u}(T) = u(T)$.

If the mathematical model introduced in the hybrid scheme is sufficiently accurate, this technique might produce more precise predictions for longer time and with less computa-
tional effort than the autonomous algorithm. This means that the hybrid scheme may forecast the dynamics of a system using a smaller reservoir.

This technique is used in [22] to improve the performance of the autonomous reservoir computing predictor when forecasting the dynamics of the Kuramoto-Sivashinsky and the Lorenz equations.

1.3.4 General considerations of the prediction methods

A helpful feature of these reservoir computing techniques is the training reusability. Once the training is performed for a certain system, the output parameters can be stored. If we want to use these techniques for predicting its dynamics again, there is no need to train the reservoir again. To restart the prediction phase we only need to reinitialize the state of the reservoir. This can be done just by mapping a smaller amount of data onto the reservoir as in the training phase, but without doing the comparison between the output of the reservoir and the measurements. Since the training phase is the most computationally expensive phase, the training reusability leads to a significant saving in computational time.

A reservoir initialization is successful if after it the state of the reservoir does not depend on the initial state of the reservoir. Reservoir computing has fading memory, that is to say, the state of the reservoir at a certain time depends on a few previous states. This way, the reservoir can be initialized to an operational state regardless of the initial state. This fading memory feature takes a crucial role in phase space reconstruction, as will be discussed in the next section.

However, these forecasting approaches have some drawbacks that are particularly relevant when the dynamical system shows chaotic dynamics. Chaotic systems have a strong sensitivity to initial conditions, so the error between \( \hat{u}(T) \) and \( u(T) \) will grow exponentially. Consider two sufficiently close initial conditions, the time it takes for the trajectories that start from them to separate by a factor \( e \) is called the Lyapunov time. The inverse of the Lyapunov time is the Lyapunov exponent. These concepts are strongly related to the prediction horizon, which is the time from which on the prediction is no longer accurate. Thus there is dynamical divergence between the exact and the predicted trajectories.

The cross-prediction is not affected by this exponential divergence since at every time iteration the exact value of at least one of the dynamical variables is mapped into the reservoir, so the reservoir has a partial reference of the state of the system that prevents having this kind of divergence.

Moreover, the longer the training time series are, the more precise the unfolding of the dynamics will be. A perfect unfolding would need an infinite amount of data, yet it is im-
possible to gather or handle infinite amounts of data. The finiteness of the training data time series may lead to inaccuracies. The training efficiency depends also on phase space coverage. If the system dynamics moves into a scarcely visited region of the phase space during the training period, then the training may not be so effective for this kind of data. This may also introduce inaccuracies in the prediction.

Finally, one of the main goals of science is explaining the mechanisms underlying a phenomenon. However, being able to predict the dynamics of a system does not mean understanding how it works. These reservoir computing techniques may be useful tools for predicting and reconstructing data but they do not provide further understanding of the dynamics or the mechanisms behind them.

1.4 Differential topology in a nutshell

We need to get to the heart of the definition of a dynamical system in order to understand why do all these reservoir computing techniques work when trying to reconstruct the phase space of dynamical systems. Since the main idea is based on some results in differential topology, we provide a brief introduction to the topic [17, 19].

Let $U \subset \mathbb{R}^p$ and $V \subset \mathbb{R}^q$ be open sets, a mapping $f : U \to V$ is called smooth if all of its partial derivatives exist and are continuous. More generally, let $X \subset \mathbb{R}^p$ and $Y \subset \mathbb{R}^q$ be arbitrary sets, then $f : X \to Y$ is a smooth map if for each $x \in X$ there exist an open set $U \subset \mathbb{R}^p$ containing $x$ and a smooth map $F : U \to \mathbb{R}^q$ that coincides with $f$ throughout $U \cap X$. A smooth mapping $f : X \to Y$ is called a diffeomorphism if it is bijective and its inverse is also smooth. In that case, $X$ and $Y$ are diffeomorphic. A continuous mapping $f : X \to Y$ is called a homeomorphism if it is bijective and its inverse is also continuous.

A manifold is a generalization of the concept of curve and surface. More precisely, a subset $M \subset \mathbb{R}^p$ is called a smooth manifold of dimension $q$ if each $x \in M$ has a neighbourhood $U \subset M$ that is diffeomorphic to an open set $W$ of an Euclidean space $\mathbb{R}^q$. This way, a diffeomorphism $\varphi = (\varphi_1, \ldots, \varphi_p) : W \to U \subset M$ is a parametrization of $U \subset M$. Given that a set $S$ is compact if every open cover of $S$ has a finite subcover, a compact manifold is a manifold that is compact when considered as a subset of an Euclidean space.

Now, given two smooth manifolds $M$ and $N$, a smooth mapping $y : M \to N$ is called an embedding if it is injective, its derivative is everywhere injective and it is a homeomorphism onto its image. This is a key concept in our discussion, as will be shown below.

A relevant case is embedding a manifold in a sufficiently high dimensional Euclidean space. It is not only a matter of fitting a manifold into that space, but also of preserving its
structure. For instance, the Möbius strip is a 2-dimensional smooth manifold that cannot be embedded in the Euclidean plane $\mathbb{R}^2$, but in $\mathbb{R}^3$. This means that such an object would not exist in Flatland [1]. Mathematically it is the set $[0, 1] \times (-1, 1)$ but with the points $(0, t)$ and $(1, -t)$ identified. Identifying two points means making these points to be the same point. This can be thought of as gluing these points. This rectangle is represented in Figure 5a. We need an extra dimension to have enough room for the Möbius strip structure to fit into an Euclidean space. We can easily construct an $\mathbb{R}^3$-embedded Möbius strip by half twisting a rectangle shaped paper and gluing the ends together. Figure 5b shows a picture of a Möbius strip embedded in $\mathbb{R}^3$.

![Möbius strip flat representation](image1)

(a) Möbius strip flat representation. The points in the red line have to be identified in the direction of the arrow.

![Möbius strip picture as would be seen when embedded in $\mathbb{R}^3$.](image2)

(b) Möbius strip picture as would be seen when embedded in $\mathbb{R}^3$.

Figure 5: Möbius strip

There is a fundamental result in differential topology due to Hassler Whitney that addresses the question of manifold embedding in Euclidean spaces. The Whitney Embedding Theorem [26] states that any $m$-dimensional compact manifold can be embedded in $\mathbb{R}^{2m+1}$.

Now, since in this work we are considering systems the dynamics of which is described by a dynamical system, let us discuss its definition from a mathematical point of view.

A real-time dynamical system is a tuple $(T, M, \varphi)$, where the time domain $T \subset \mathbb{R}$ is an open interval, the phase space of the dynamical system is a manifold $M$ and $\varphi : U \subset T \times M \rightarrow M$, $(t, x) \mapsto \varphi(t, x)$ is a continuous function that describes the evolution of the system. This
map verifies $\Phi_0(x) = x$ and $\Phi(\Phi_t(x)) = \Phi_{t+s}(x) = \Phi_s(\Phi_t(x))$, for every $(x,t), (x,s) \in U$. The phase space manifold $M$ is the ensemble of all the possible states of the dynamical system.

Since $\Phi_0(x) = x$, the initial state of the system is a point $x$ in the phase space manifold $M$. This initial state $x$ describes a trajectory in the phase space that is characterized by the evolution map $\Phi_t(x)$ at time $t$.

The phase space manifold is an abstract object, although there is a way to translate a point in the manifold into observables. We already know that the phase space manifold is diffeomorphic to an open set of an Euclidean space. Let $y : M \rightarrow W \subset \mathbb{R}^m$ be a diffeomorphism, so $y(x) = (y_1(x), ..., y_m(x))$ is a system of coordinates of the phase space and it is the inverse map of a parametrization. Each of the $m$ coordinates of $\mathbb{R}^m$ can be identified with an observable, therefore computing $y_k(\Phi_t(x))$ is measuring the observable $k$ at time $t$. This way, $y(\Phi_0(x))$ are the $m$ measurements of the initial state or the initial conditions of the system.

For instance, let $M$ be the phase space manifold of an optically injected semiconductor laser like the one described in Section 1.2. The dynamical variables $y(\Phi_t(x)) = (\text{Re}(t), \text{Im}(t), n(t))$ and $\tilde{y}(\Phi_t(x)) = (A(t), \Phi_t(n(t)))$ are two different sets of measurements of the same point $x$ in the phase space since they originate from two different sets of coordinates, $y$ and $\tilde{y}$.

Now that we know that the map $y : M \rightarrow \mathbb{R}^m$ defines a set of $m$ observables and $t \mapsto y(\Phi_t(x))$ is the set of the measurements of the $m$ variables at time $t$, let us discuss how we can extract information about the dynamics using the Takens delay embedding theorem.

**Takens delay embedding theorem:** Let $M$ be a compact manifold of dimension $m$. For pairs $(\varphi_t, y_k)$, $\varphi_t : M \rightarrow M$ a diffeomorphism and $y_k : M \rightarrow \mathbb{R}$ a smooth function, the map $\Phi_{(\varphi_t, y_k)} : M \rightarrow \mathbb{R}^{2m+1}$, defined by

$$\Phi_{(\varphi_t, y_k)}(x) = \{y_k(x), y_k(\varphi_t(x)), ..., y_k(\varphi_2^m(x))\} = \{y_k(x), y_k(\varphi_t(x)), ..., y_k(\varphi_{2^m}(x))\}$$

is an embedding.

In other words, $2m + 1$ delayed measurements of one observable are enough to create an embedding of the phase space into a high dimensional Euclidean space where the dynamics is unfolded. If more vectors of consecutive delayed measurements are mapped through $\Phi_{(\varphi_t, y_k)}$, $\Phi_{(\varphi_t, y_k)}(x), \Phi_{(\varphi_t, y_k)}(\varphi_t(x)), \Phi_{(\varphi_t, y_k)}(\varphi_{2^r}(x)), ...$, then a geometrically identical but unfolded phase space is reconstructed in $\mathbb{R}^{2m+1}$. It has the same dynamical and topological properties of the original one and therefore the original dynamics are geometrically equivalent to those in the reconstructed phase space [23].

However, Takens embedding theorem does not provide access to the explicit information
of the dynamical variables that are reconstructed. Finding the mappings $y_s$, $1 \leq s \leq m$, $s \neq k$ that translate the points of the reconstructed phase space into the remaining physical observables is not trivial. Sometimes those mappings can be computed but the reconstructed observables might differ significantly from the true magnitudes.

Reservoir computing can be used to build an operational phase space reconstructor. The cross-prediction algorithm is conceptually similar to the delay embedding procedure, but it has some differences. The core of this method is the reservoir, which is a random network that evolves as a high order dynamical system. The delayed measurements of one observable are mapped onto the reservoir, whose dimension is high enough so that it takes the role of the embedding space. The phase space will be reconstructed in the reservoir when a sufficiently long time series of measurements is mapped onto it. Inferring the dynamics of the remaining observables is conceptually simple, it is just a matter of projecting the state of the reservoir onto the observables that we want to reconstruct. To do so, besides the unlimited time series of measurements of the input variable, we need sufficiently long time series of measurements of the observables to be reconstructed. The reservoir state projection is chosen so that it is a combination of the states of the nodes and the weights of that combination can be tuned so that the output approximates the latter time series, this is the training phase of the cross-prediction method. Afterwards, if the training was successful, the cross-prediction algorithm can be used for obtaining the measurements for those observables if more delayed measurements of the input observable are mapped into the reservoir.

There are a few differences between the Takens delay embedding and the reservoir computing cross-prediction. The latter can produce explicitly approximate values for the inferred measurements, while the delay embedding procedure can reconstruct the phase space, but not the time series of the unmeasured variables. It also needs sufficiently long time series of measurements for those variables for training the reservoir. The phase space embedding is conceptually different in the cross-prediction method since the reservoir has fading memory, that is to say, its state depends on a few previous states. Therefore, the phase space is never globally embedded in the reservoir, as it will lose information introduced with the previous delayed measurements. Nevertheless, the reservoir states can keep enough information about the local structure of the unfolded phase space and the dynamics of the evolution function $\varphi_t$ that defines the evolution of the dynamical system in the phase space manifold. This information may be enough for reproducing the system dynamics accurately.

In the next section we discuss the reservoir computing algorithms used for phase space reconstruction and prediction in detail.
2 Methodology

2.1 Cross-prediction algorithm

Let \( u \in \mathbb{R}^M \) and \( v \in \mathbb{R}^P \) be two different sets of measurements, as defined in Section 1.3. Again, let us assume that we have measurements for \( v(t) \) only in the training interval \( 0 \leq t \leq T \), and we have measurements for \( u(t) \) beyond the training time \( T \).

Let \( r(t) \in \mathbb{R}^N \) be the state of the \( N \) nodes of the reservoir at time \( t \). The reservoir is in this case an Erdős-Rényi weighted random neural network with average degree \( D \). This way, we compute the weighted adjacency matrix \( A \) that defines the links between nodes in the reservoir. The weights are first random and independently generated from a uniform distribution between \(-1\) and \(1\). Then, the spectral radius of the adjacency matrix is tuned to be \( \rho \). To do so, we compute the eigenvalues of \( A \) and we multiply this matrix by a convenient number so that the maximum of the eigenvalues, which is the spectral radius, becomes \( \rho \). This matrix \( A \) is computed at the very beginning of the algorithm and kept fixed.

The input layer \( In: \mathbb{R}^M \rightarrow \mathbb{R}^M \) is chosen to be linear. Therefore it can be represented by a matrix \( W_{in} \in \mathbb{R}^{N \times M}, In[u(t)] = W_{in}u(t) \). This matrix is selected such that the elements in vector \( u \) are mapped onto disjoint sets of nodes of the reservoir. This can be easily achieved by placing a single non-zero entry in each row of \( W_{in} \). In this work we have connected each element of \( u \) to \( N/M \) nodes. These connections are weighted links drawn random and independently from a uniform distribution between \(-\sigma\) and \(\sigma\).

As stated before, the reservoir evolution is described by a nonlinear function (3). There are many possible choices of that nonlinearity but in this work we use a hyperbolic tangent.

\[
\begin{align*}
r(t) = (1 - \delta)r(t - \Delta t) + \delta \tanh(Ar(t - \Delta t) + W_{in}u(t) + \xi1),
\end{align*}
\]

where, in spite of the abuse of notation, the hyperbolic tangent is evaluated for every element of its argument vector. \( 0 < \delta \leq 1 \) is the leakage rate, which controls the speed of the reservoir evolution in the sense that it controls the dependence of the reservoir state on its previous states. This dependence is weaker when \( \delta \) is small. The time step \( \Delta t \) is the sampling time of the time series. \( 1 \in \mathbb{R}^N \) is a vector of ones. \( \xi \) is the bias constant. It is introduced for breaking the symmetry in some dynamical systems. \( r(t) \) is an odd function if \( \xi = 0 \). Symmetries may lead to ambiguities between possible states of the system and, ultimately, to wrong inference of the dynamics.

For example, the Lorenz dynamical system (10) described below is invariant under the change of sign of the \( x \) and \( y \) variables.
$\dot{x} = a(x - y),$
$\dot{y} = x(b - z) - y,$
$\dot{z} = xy - cz. \tag{10}$

If we feed the reservoir with time series data from the $z$ variable of the Lorenz system, there is an ambiguity between 2 possible states of the system, one for the + sign in the $x$ and $y$ variables, another for the - sign. Since the hyperbolic tangent is an odd function, if the bias is zero it cannot resolve this ambiguity. The system would also be invariant under the change of sign of variables $x$, $y$ and the state of the reservoir $r$, so the inference could fail \cite{15}. This is why a non-zero bias term is sometimes needed, it can break this kind of symmetry.

The output map $Out : \mathbb{R}^N \times \mathbb{R}^S \rightarrow \mathbb{R}^P$ is assumed to be $\hat{v}(t) = Out[r(t), p] = W_{out}r(t) + c,$ where $W_{out} \in \mathbb{R}^{P \times N}$ is a matrix and $c \in \mathbb{R}^P$ is a vector. These two objects, $W_{out}$ and $c$ take the role of the parameters $p$ to be determined at the training phase.

Another way of breaking symmetries is by modifying the output map, $Out[r] = W_{out}r^* + c,$ where $r^*$ is chosen such that its elements $r^*_j$ are equal to the elements $r_j$ of $r$ when $j$ is odd and $r^*_j = r^2_j$ when $j$ is even. This symmetry breaking procedure is not used in this work.

In the cross-prediction method, the time series are pre-processed so that they have zero mean and unit variance for performance comparison purposes.

After generating the input mask $W_{in}$ and the reservoir adjacency matrix $A$, we initialize the state of the reservoir. First, we draw the raw initial state $r(-\tau)$ randomly from a uniform distribution between $-\sigma$ and $\sigma$. We call $\tau$ the transient time. The idea is to map the input $u$ onto the reservoir for making it evolve into the initial state $r(-\Delta t)$. We want this initial state to be independent of $r(-\tau)$, therefore we have to choose the transient time $\tau$ to be sufficiently large.

The reservoir initialization process also helps to avoid introducing transient dynamics of the system into the training data. The transient dynamics may also cover larger parts of the phase space, so avoiding introducing it into the training phase increases its effectiveness. Introducing data only from the dynamics that we want to reconstruct also boosts the performance of the training phase.

Now that we have the initial state of the reservoir $r(-\Delta t)$, we make it evolve up to time $T$ and we save the reservoir states time series for $0 \leq t \leq T$, $R = \{r(0), r(\Delta t), ..., r(T - \Delta t), r(T)\}$. Let $V = \{v(0), v(\Delta t), ..., v(T - \Delta t), v(T)\}$ be the time series of the time-limited measurements. Let $L = T/\Delta t$. To train the reservoir we perform a Tikhonov regularized linear regression. This way we determine $W_{out}$ and $c$ so that the output approximates the
measurements $V$. This method consists of minimizing Eq. (11) with respect to $W_{out}$ and $c$.

$$\sum_{k=0}^{L} \| W_{out} r(k\Delta t) + c - v(k\Delta t) \|^2 + \beta \text{Tr}(W_{out} W_{out}^T), \quad (11)$$

where $\|c\|^2 = c \cdot c^T$, $\text{Tr}(W)$ stands for the trace of matrix $W$ and $\beta$ is a positive but small number that is called the Ridge regression parameter. The term $\beta \text{Tr}(W_{out} W_{out}^T)$ is introduced to avoid overfitting $W_{out}$. Let $W_{out}^*$ and $c^*$ be the parameters that minimize Eq. (11). They can be described in terms of the time series $R$ and $V$.

Let $\bar{r}$ and $\bar{v}$ be the average of the time series $R$ and $V$, respectively

$$\bar{r} = \frac{1}{L+1} \sum_{k=0}^{L} r(k\Delta t), \quad (12)$$

$$\bar{v} = \frac{1}{L+1} \sum_{k=0}^{L} v(k\Delta t), \quad (13)$$

$\delta R$ and $\delta V$ stand for the difference between the elements of the time series $R$ and $V$ and their average.

$$\delta R = (r(0) - \bar{r}, r(\Delta t) - \bar{r}, ..., r(T - \Delta t) - \bar{r}, r(T) - \bar{r}), \quad (14)$$

$$\delta V = (v(0) - \bar{v}, v(\Delta t) - \bar{v}, ..., v(T - \Delta t) - \bar{v}, v(T) - \bar{v}), \quad (15)$$

and, finally, we compute the parameters that minimize Eq. (11).

$$W_{out}^* = \delta V \cdot \delta R^T \cdot (\delta R \cdot \delta R^T + \beta I)^{-1}, \quad (16)$$

$$c^* = -(W_{out}^* \bar{r} - \bar{v}), \quad (17)$$

where $I \in \mathbb{R}^{N \times N}$ is the identity matrix.

This way we can obtain an approximation to the unmeasured variables for $t > T$ by letting the reservoir evolve while mapping $u(t)$ onto its nodes and computing
\[ \hat{v}(t) = W_{\text{out}}^* r(t) + c^*. \] (18)

2.2 Autonomous reservoir computing predictor algorithm

The autonomous reservoir computing predictor implementation is analogous to the cross-prediction algorithm, although there are some modifications.

Let \( \mathbf{u} \in \mathbb{R}^M \) be the set of variables that can be measured for the dynamical system that we want to predict. Assume that we have measurements for those variables in the training and the initialization intervals, \(-\tau \leq t \leq T\). Let \( U = \{ \mathbf{u}(-\tau), \mathbf{u}(-\tau + \Delta t), \ldots, \mathbf{u}(T - \Delta t), \mathbf{u}(T) \} \) be the time series for these measurements. Beware that in this case the transient time should be chosen carefully. There is a trade-off between the reservoir initialization and the training process efficiency due to the finiteness of the measurements time series. This can be specially important when dealing with limited experimental data.

The input map and the reservoir are constructed in the same way as in the cross-prediction method. This time reservoir nonlinearity depends on \( r(t) \) instead of \( r(t - \Delta t) \) since at every time step \( t \) we are computing the state of the reservoir at the next time step, \( r(t + \Delta t) \). Therefore, we have

\[
\mathbf{r}(t + \Delta t) = (1 - \delta) \mathbf{r}(t) + \delta \tanh(A\mathbf{r}(t) + W_{\text{in}} \mathbf{u}(t) + \xi \mathbf{1}).
\] (19)

The reservoir is initialized as in the previous algorithm. The time series \( U \) is mapped onto the reservoir and the initial state \( \mathbf{r}(-\Delta t) \) is computed. Then, the training phase begins and the state of the reservoir \( \mathbf{r}(t) \) is stored for \( 0 \leq t \leq T \), let \( R = \{ \mathbf{r}(0), \mathbf{r}(\Delta t), \ldots, \mathbf{r}(T - \Delta t), \mathbf{r}(T) \} \) be the reservoir time series. Let \( V = \{ \mathbf{u}(0), \mathbf{u}(\Delta t), \ldots, \mathbf{u}(T - \Delta t), \mathbf{u}(T) \} \) be the time series of the training data.

The reservoir produces raw output,

\[
\hat{\mathbf{u}}(t + \Delta t) = W_{\text{out}} \mathbf{r}(t + \Delta t) + \mathbf{c}.
\] (20)

where \( W_{\text{out}} \in \mathbb{R}^{M \times N} \) and \( \mathbf{c} \in \mathbb{R}^M \).

This raw output time series has to be approximated to the training data \( V \). To do so, we have to compute the \( W_{\text{out}}^* \) and \( \mathbf{c}^* \) that minimize Eq. (11). The Tikhonov regularization is therefore performed for these time series following Eqs. (11-17).
Phase space reconstruction of sc. laser dynamics using reservoir computing  A.Cunillera

Finally, the prediction phase starts and the prediction $\hat{\mathbf{u}}(t)$ is used as an input of the reservoir. The reservoir evolution equation for this closed loop is,

$$r(t + \Delta t) = (1 - \delta)r(t) + \delta \tanh(Ar(t) + W_{in}\hat{\mathbf{u}}(t) + \xi 1).$$  \hspace{1cm} (21)

The variables $\mathbf{u}$ can be predicted by mapping the state of the reservoir at time $t \geq T$ through the output layer,

$$\hat{\mathbf{u}}(t + \Delta t) = W_{out}^*r(t + \Delta t) + \mathbf{c}^*.$$  \hspace{1cm} (22)

Beware that, unlike in the cross-prediction, the predictions can be highly sensitive to the final measurements $\mathbf{u}(T)$ and they could be accurate for a limited amount of iterations, specially when the system is chaotic.

### 2.3 Hybrid scheme predictor algorithm

The main difference between the hybrid scheme and the autonomous predictor is the implementation of an approximate mathematical model. The model takes the input data and produces an initial guess of the variables at the next time step. The input data is mapped onto some of the nodes of the reservoir, while the model’s initial guess is mapped onto the remaining nodes. The reservoir evolves and its state and the initial guess are combined to create the prediction.

Once again, let $\mathbf{u} \in \mathbb{R}^M$ be the set of variables measured in the training and the initialization intervals and $U = \{\mathbf{u}(-\tau), \mathbf{u}(-\tau + \Delta t), ..., \mathbf{u}(T - \Delta t), \mathbf{u}(T)\}$ the associated time series. The reservoir adjacency matrix is computed as in the two previous algorithms, while the input layer $W_{in} \in \mathbb{R}^{N \times 2M}$ is a matrix that has twice the number of columns as in the previous examples. Each node receives input from a single element of the initial guess input data bundle. This way, we can tune the matrix so that we map the model’s initial guess onto a fraction $f$ of nodes of the reservoir and the input data is mapped onto the remaining fraction of nodes.

The reservoir evolution equation is

$$r(t + \Delta t) = (1 - \delta)r(t) + \delta \tanh \left( Ar(t) + W_{in} \left( Model[\mathbf{u}(t)] \right) + \xi 1 \right),$$  \hspace{1cm} (23)
where $Model[u(t)]$ is the next time step prediction of the mathematical model. We consider here a model of the system with a small parameter mismatch to predict the evolution of the dynamics. This way we can test if the hybrid scheme method can be used to improve the accuracy of this approximate model.

The reservoir initialization process is exactly the same as in the autonomous reservoir computing prediction method. The reservoir states time series $R = \{r(0), r(\Delta t), ..., r(T - \Delta t), r(T)\}$ and the training data $V = \{u(0), u(\Delta t), ..., u(T - \Delta t), u(T)\}$ are stored.

Another difference between the prediction methods is that the mathematical model output also takes a role in the output layer.

$$\hat{u}(t + \Delta t) = W_{out} \left( \begin{array}{c} Model[u(t)] \\ r(t + \Delta t) \end{array} \right) + c,$$

where $W_{out} \in \mathbb{R}^{M \times (M+N)}$ and $c \in \mathbb{R}^M$. Both $W_{out}^*$ and $c^*$ are determined by using the Tikhonov linear regression. This terminates the training phase. Thus, the prediction phase starts and the predictor output $\hat{u}$ is introduced as the input of the algorithm. This way, the reservoir evolution and the output are computed according to the following equations

$$r(t + \Delta t) = (1 - \delta)r(t) + \delta \tanh \left( A r(t) + W_{in} \left( \begin{array}{c} Model[\hat{u}(t)] \\ \hat{u}(t) \end{array} \right) + \xi \mathbf{1} \right),$$

$$\hat{u}(t + \Delta t) = W_{out}^* \left( \begin{array}{c} Model[\hat{u}(t)] \\ r(t + \Delta t) \end{array} \right) + c^*.$$

The reservoir computing inference and prediction algorithms have been introduced. In the next section these methods are tested.
3 Results

In this section we present and discuss the results obtained when using the reservoir computing time series inference and prediction algorithms. The corresponding programs have been implemented in Python and are applied to a variety of dynamics produced by the optically injected semiconductor laser rate equations (1-2).

3.1 Cross-prediction results

First, let us discuss the results for the cross-prediction method. The reservoir input is one of the dynamical variables of the semiconductor laser and we use this algorithm to infer the time series of the other two dynamical variables, as explained in Section 2.

We need a way to test the accuracy of this method quantitatively. Let \( \{\hat{v}(t)\}_{t>T} \) be the time series of the inferred variables obtained by using the cross-prediction method and let \( \{v(t)\}_{t>T} \) be the exact value of those variables. We use the root mean square (RMS) error between the time series of the \( k \)-th dynamical variable (27) as a measure of the accuracy of the inference of that variable.

\[
\text{RMS}_k = \sqrt{\frac{\sum_t (v_k(t) - \hat{v}_k(t))^2}{\sum_t (v_k(t))^2}} \tag{27}
\]

Therefore, if RMS is 0, the inferred time series are identical to the exact ones, while if it is large it means that the inferred time series are not an accurate approximation to the exact dynamics.

The cross-prediction method requires an unlimited set of measurements of one dynamical variable as the reservoir input. Sufficiently long time series of measurements for the remaining variables are also needed for training the reservoir. To obtain the time series we integrate numerically the rate equations (1-2) by using a fourth order Runge-Kutta method with a time step \( \Delta t = 0.1 \). However, every time series is sampled every three integration time steps, \( \Delta t_s = 0.3 \), since it has been found that this sampling rate was the best in terms of accuracy of the inference.

Let us consider now the real and imaginary components of the electric field and the carrier density set of coordinates of the semiconductor laser rate equations. As described in Section 1.2, this dynamical system shows 6 different dynamical regimes, although for brevity we will sometimes present results for some of those regimes. The phase space of all those
Phase space reconstruction of sc. laser dynamics using reservoir computing A.Cunillera

dynamical regimes is presented in Figure 6. These are the stationary fixed point (Figure 6a), the bounded-phase limit cycle (Figure 6b), the unbounded-phase limit cycle (Figure 6c), the period-3 (Figure 6d), the quasiperiodic (Figure 6e) and the chaotic laser dynamics (Figure 6f).

In Figure 6a we can see a trajectory spiraling down towards a fixed point in the phase space. Figures 6b and 6c look very similar, although this would not be the case if we looked at the amplitude and phase of the electric field. With this set of coordinates Figure 6b would still show periodic dynamics, while Figure 6c would show a helix that drives any trajectory towards infinite phase. In Figure 6d we can see a trefoil knot. This period-3 dynamics is topologically different to the periodic dynamics in Figures 6b and 6c since the trefoil knot cannot be transformed by an orientation-preserving homeomorphism into a circle, unlike the limit cycles in Figures 6b and 6c. In Figure 6e the phase space looks like a torus. It is said that the dynamics is quasiperiodic since the dynamical variables oscillate with frequencies whose ratio is irrational, so an arbitrary long trajectory would span the torus. Last, we can see that there is an attractor in the phase space in Figure 6f for the chaotic laser regime.

To illustrate the performance of the cross-prediction method qualitatively, we plot the inferred time series obtained for some of the dynamical regimes in the real and imaginary components of the electric field and carrier density set of coordinates. We also use different variables as the reservoir input. The inferred time series are displayed as red dashed lines, while the exact time series are represented in blue. This way, if the inference is accurate enough, the red dashed lines will be superimposed upon the blue curves. Figures 7, 8 and 9 show the reconstructed dynamics corresponding to some of the nonchaotic dynamical regimes and Figure 10 to the chaotic laser.

To obtain the results presented for the nonchaotic regimes, we fed a small reservoir ($N = 400$ nodes) with the time series for Re($E$). We trained it for 900 time steps, since the sampling rate is $\Delta t_s = 0.3$, this corresponds to 270 units of time, and we performed the cross-prediction for 100 more units of time, which correspond to 333 iterations of the algorithm. The remaining reservoir parameters are presented in Table 2 and the dependence of the performance of the method on them will be discussed later. In this table, $T$ is the length of the training interval, $\tau$ is the initialization time of the reservoir and $T_{end}$ is the length of the inference phase that goes from $t = T$ to $t = T + T_{end}$. 
Phase space reconstruction of sc. laser dynamics using reservoir computing  

(a) Stationary fixed point  
(b) Bounded-phase limit cycle  
(c) Unbounded-phase limit cycle  
(d) Period-3 dynamics  
(e) Quasiperiodic dynamics on a torus  
(f) Chaotic laser dynamics  

Figure 6: $\text{Re}(E)$-$\text{Im}(E)$-$n$ phase space representations for the optically injected semiconductor laser.

27
Phase space reconstruction of sc. laser dynamics using reservoir computing  A.Cunillera

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>$N$</td>
<td>400</td>
</tr>
<tr>
<td>$D$</td>
<td>15</td>
</tr>
<tr>
<td>$\rho$</td>
<td>1.3</td>
</tr>
<tr>
<td>$\sigma$</td>
<td>1</td>
</tr>
<tr>
<td>$\delta$</td>
<td>0.4</td>
</tr>
<tr>
<td>$\xi$</td>
<td>1</td>
</tr>
<tr>
<td>$T$</td>
<td>270</td>
</tr>
<tr>
<td>$\tau$</td>
<td>300</td>
</tr>
<tr>
<td>$T_{end}$</td>
<td>100</td>
</tr>
<tr>
<td>$\beta$</td>
<td>$10^{-7}$</td>
</tr>
</tbody>
</table>

Table 2: Reservoir parameters for the cross-prediction method used for reconstructing non-chaotic dynamics for the real and imaginary components of the electric field and carrier density set of coordinates.

In Figure 7, we show the inferred time series of $\text{Im}(E)$ and $n$ when $\text{Re}(E)$ is used as the reservoir input for the bounded-phase limit cycle dynamical regime.

Figure 7: Cross-prediction of bounded-phase limit cycle dynamics. The input variable is the real part of the electric field $\text{Re}(E)$. The exact time series are represented by blue lines, while the inferred time series are represented by dashed red lines. The cross-prediction algorithm parameters used for obtaining this result are shown in Table 2.
In Figure 7 we can see that the red dashed lines are overimposed on the blue ones, meaning that the cross-prediction method is able to reconstruct the periodic dynamics accurately. We found that this dynamics can also be inferred with similar accuracy when using either $\text{Im}(\mathbf{E})$ or $n$ as the reservoir input variable. An accurate inference can be achieved when the reservoir is relatively small ($N = 400$) and the training phase is short.

Figure 8 shows the results of using the cross-prediction algorithm for reconstructing the variables $\text{Re}(\mathbf{E})$ and $n$ in the period-3 dynamical regime when using $\text{Im}(\mathbf{E})$ as the input variable.

![Cross-prediction results](image1)

Figure 8: Cross-prediction of period-3 dynamics. The input variable is the imaginary part of the electric field $\text{Im}(\mathbf{E})$. The exact time series are represented by blue lines, while the inferred time series are represented by dashed red lines. The cross-prediction algorithm parameters used for obtaining this result are shown in Table 2.

Again, we can see in Figure 8 that the period-3 dynamics can be reconstructed accurately when using a reservoir of size $N = 400$. 
The quasiperiodic dynamics are reconstructed in Figure 9, where the carrier density $n$ is used as the reservoir input variable.

![Cross-prediction of quasiperiodic dynamics](image)

Figure 9: Cross-prediction of quasiperiodic dynamics. The input variable is the carrier density $n$. The exact time series are represented by blue lines, while the inferred time series are represented by dashed red lines. The cross-prediction algorithm parameters used for obtaining this result are shown in Table 2.

As shown in Figure 9, the dynamics can be reproduced accurately when using the carrier density $n$ as the input variable even though this is not a periodic regime.

We have shown that the cross-prediction algorithm can be used to infer the dynamics of an optically injected semiconductor laser when using any of its 3 dynamical variables as the reservoir input. We now move on to check whether the cross-prediction method can be used to infer the chaotic dynamics. We use the real part of the electric field, $\text{Re}(E)$, as the reservoir input. The inference is not accurate when utilizing the parameter configuration used for the nonchaotic dynamical regimes, so we have to use a larger reservoir ($N = 3000$)
for achieving good accuracy when reconstructing more complex dynamics. We also changed
the other parameters to optimize the algorithm performance. These parameters are shown in
Table 3. The training time is also longer, as it is performed for 20000 iterations ($T = 6000$),
while the inference phase is as long as it was in the nonchaotic examples, 333 iterations. An example of cross-prediction of chaotic laser dynamics under this simulation conditions
is presented in Figure 10. A detailed evaluation of the influence of the reservoir parameters
will be presented at the end of this section.

Figure 10: Cross-prediction of chaotic laser dynamics. The input variable is the real part of
the electric field Re($E$). The exact time series are represented by blue lines, while the inferred
time series are represented by dashed red lines. The cross-prediction algorithm parameters
used for obtaining this result are shown in Table 3.

Figure 10 shows that the chaotic laser dynamics can be reconstructed accurately when
using the real part of the electric field as the input variable of the reservoir of the cross-
prediction method. Notice that the dynamics here is more complex than in the previous
examples.
In Figure 10 we can also see that in the chaotic regime there are large gradient peaks and oscillations that are faster than those in other regimes. This is remarkable for the real and the imaginary components of the electric field time series. It is found that reconstructing these large gradients and fast peaks is harder for the reservoir. This may happen since the slow oscillations prevail in the dynamics and the fast peaks are less frequent. The reservoir might be used to evolve according to the predominant slow dynamics and then it is harder for the reservoir to reconstruct this eventual dynamics. The scarcity of fast peaks in the training phase may also take a role in this.

Figure 11 shows the chaotic regime attractor in the phase space of the semiconductor laser when using the real component of the electric field as the cross-prediction input. Two parts can be distinguished when the time series is divided in small pieces and the RMS error is computed for them. The blue part corresponds to low RMS error areas while the red part stands for the high RMS error zones.
Figure 11: RMS error colorplot for the cross-prediction of chaotic laser dynamics. The low RMS error part of the attractor is represented in blue, while the high RMS error part is represented in red.

We can observe that some of the most external parts of the attractor are high RMS error zones. For instance, one of these zones corresponds to the highest values of the carrier density, so the largest peaks in the carrier density and the imaginary component of the field when the carrier density has large peaks seem to be difficult to reconstruct.

Now that we know that the cross-prediction method can reproduce the dynamics of the semiconductor laser in any regime and using any dynamical variable as the reservoir input, it is time to quantify the accuracy of the cross-prediction. In Table 4 we show results of the cross-prediction for each observable, $\text{Re}(E)$, $\text{Im}(E)$ and $n$, as the reservoir input in each dynamical regime. We computed the mean RMS error over 20 realizations with different initial conditions. The cross-prediction parameters used in the simulations are presented in Table 2 for the nonchaotic dynamical regimes and in Table 3 for the chaotic laser regime except for the time parameters. For comparison purposes we used the same length of the
training phase for all the dynamical regimes $T = 50000$ and $T_{end} = 1000$, although it was shown in Figures 7, 8 and 9 that a 100 times shorter training phase can be used for the nonchaotic dynamical regimes to achieve good accuracy in the inference.

<table>
<thead>
<tr>
<th>Regime</th>
<th>Input variable</th>
<th>Inferred variable 1</th>
<th>Inferred variable 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stationary dynamics</td>
<td>Re($E$)</td>
<td>Im($E$): $(4.96 \pm 0.94) \times 10^{-10}$</td>
<td>n: $(4.20 \pm 0.63) \times 10^{-9}$</td>
</tr>
<tr>
<td></td>
<td>Im($E$)</td>
<td>Re($E$): $(2.66 \pm 0.53) \times 10^{-9}$</td>
<td>n: $(3.08 \pm 0.52) \times 10^{-9}$</td>
</tr>
<tr>
<td></td>
<td>n</td>
<td>Re($E$): $(1.07 \pm 0.22) \times 10^{-8}$</td>
<td>Im($E$): $(2.73 \pm 0.64) \times 10^{-10}$</td>
</tr>
<tr>
<td>Bounded phase limit cycle dynamics</td>
<td>Re($E$)</td>
<td>Im($E$): $(7.92 \pm 0.85) \times 10^{-5}$</td>
<td>n: $(1.34 \pm 0.18) \times 10^{-4}$</td>
</tr>
<tr>
<td></td>
<td>Im($E$)</td>
<td>Re($E$): $(1.04 \pm 0.11) \times 10^{-4}$</td>
<td>n: $(8.19 \pm 0.56) \times 10^{-5}$</td>
</tr>
<tr>
<td></td>
<td>n</td>
<td>Re($E$): $(7.16 \pm 0.80) \times 10^{-5}$</td>
<td>Im($E$): $(1.47 \pm 0.23) \times 10^{-4}$</td>
</tr>
<tr>
<td>Unbounded phase limit cycle dynamics</td>
<td>Re($E$)</td>
<td>Im($E$): $(9.3 \pm 1.3) \times 10^{-5}$</td>
<td>n: $(9.4 \pm 1.1) \times 10^{-5}$</td>
</tr>
<tr>
<td></td>
<td>Im($E$)</td>
<td>Re($E$): $(7.9 \pm 1.2) \times 10^{-5}$</td>
<td>n: $(7.83 \pm 0.93) \times 10^{-5}$</td>
</tr>
<tr>
<td></td>
<td>n</td>
<td>Re($E$): $(1.04 \pm 0.11) \times 10^{-4}$</td>
<td>Im($E$): $(9.0 \pm 1.3) \times 10^{-5}$</td>
</tr>
<tr>
<td>Period-3 dynamics</td>
<td>Re($E$)</td>
<td>Im($E$): $(3.78 \pm 0.35) \times 10^{-5}$</td>
<td>n: $(4.44 \pm 0.29) \times 10^{-5}$</td>
</tr>
<tr>
<td></td>
<td>Im($E$)</td>
<td>Re($E$): $(2.153 \pm 0.091) \times 10^{-5}$</td>
<td>n: $(2.47 \pm 0.14) \times 10^{-5}$</td>
</tr>
<tr>
<td></td>
<td>n</td>
<td>Re($E$): $(2.74 \pm 0.20) \times 10^{-5}$</td>
<td>Im($E$): $(2.40 \pm 0.14) \times 10^{-5}$</td>
</tr>
<tr>
<td>Quasiperiodic dynamics</td>
<td>Re($E$)</td>
<td>Im($E$): $(7.02 \pm 0.17) \times 10^{-5}$</td>
<td>n: $(6.88 \pm 0.20) \times 10^{-5}$</td>
</tr>
<tr>
<td></td>
<td>Im($E$)</td>
<td>Re($E$): $(8.44 \pm 0.32) \times 10^{-5}$</td>
<td>n: $(8.25 \pm 0.23) \times 10^{-5}$</td>
</tr>
<tr>
<td></td>
<td>n</td>
<td>Re($E$): $(7.50 \pm 0.27) \times 10^{-5}$</td>
<td>Im($E$): $(9.88 \pm 0.42) \times 10^{-5}$</td>
</tr>
<tr>
<td>Chaotic laser dynamics</td>
<td>Re($E$)</td>
<td>Im($E$): $(2.8 \pm 1.1) \times 10^{-2}$</td>
<td>n: $(3.4 \pm 1.2) \times 10^{-2}$</td>
</tr>
<tr>
<td></td>
<td>Im($E$)</td>
<td>Re($E$): $(5.2 \pm 3.3) \times 10^{-2}$</td>
<td>n: $(1.49 \pm 0.54) \times 10^{-2}$</td>
</tr>
<tr>
<td></td>
<td>n</td>
<td>Re($E$): $(1.24 \pm 0.35) \times 10^{-2}$</td>
<td>Im($E$): $(3.7 \pm 1.5) \times 10^{-2}$</td>
</tr>
</tbody>
</table>

Table 4: Mean RMS error of the cross-prediction inference over 20 realizations for the real and imaginary components of the electric field and carrier density set of coordinates. The cross-prediction parameters used for obtaining the results for the nonchaotic dynamical regimes are presented in Table 2 and those for the chaotic laser regime in Table 3 except for the length of the training and inference phases, which are $T = 50000$ and $T_{end} = 1000$ respectively.

We state that a variable has been inferred accurately when the RMS error is lower than 0.5 per 1000 units of time since if the RMS error is larger than that threshold, inaccuracies can be appreciated when comparing the exact and the inferred time series. This way, we may conclude that the dynamics can be reconstructed accurately in any dynamical regime when using any variable as the reservoir input. We can also observe that the accuracy of the inference of the dynamical variables does not depend significantly on which the input variable is.

In the nonchaotic regimes, if we fix the cross-prediction algorithm parameters, we can see that the accuracy of the inference decreases as the complexity of the dynamics increases.
We used a small reservoir for reconstructing nonchaotic dynamics but this configuration does not produce accurate results when applied to the reconstruction of chaotic dynamics. It is observed that, generally, using a larger reservoir produces more accurate results, provided that the training phase is sufficiently large. However, the computation time is also proportional to the size of the reservoir.

Similarly, we now test the cross-prediction for an alternative ansatz of the electric field. The dynamics can also be inferred when using one dynamical variable as the reservoir input in the amplitude and phase of the electric field and carrier density set of variables. In this case, the amplitude of the electric field is a more accessible quantity from the experimental point of view. The cross-prediction parameters used for reconstructing the dynamics are shown in Table 5.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>$N_{\text{nonchaotic}}$</td>
<td>400</td>
</tr>
<tr>
<td>$N_{\text{chaotic}}$</td>
<td>1200</td>
</tr>
<tr>
<td>$\rho$</td>
<td>1.4</td>
</tr>
<tr>
<td>$\delta$</td>
<td>0.5</td>
</tr>
<tr>
<td>$T$</td>
<td>6000</td>
</tr>
<tr>
<td>$T_{\text{end}}$</td>
<td>300</td>
</tr>
<tr>
<td>$\sigma$</td>
<td>1.4</td>
</tr>
<tr>
<td>$\xi$</td>
<td>1</td>
</tr>
<tr>
<td>$\tau$</td>
<td>400</td>
</tr>
<tr>
<td>$\beta$</td>
<td>$10^{-6}$</td>
</tr>
</tbody>
</table>

Table 5: Reservoir parameters for the cross-prediction method. They are used for reconstructing the chaotic and nonchaotic laser dynamics for the amplitude, phase and carrier density set of coordinates. Notice that we use a larger reservoir with $N_{\text{chaotic}}$ nodes for reconstructing chaotic dynamics.

The dynamics can be reconstructed for the stationary fixed point and the bounded-phase limit cycle dynamics but the cross-prediction method fails when trying to reconstruct the phase for the remaining dynamical regimes. Figure 12 shows how the reconstruction of the phase fails for the quasiperiodic regime when the amplitude is used as the reservoir input.

The main difference between the two mentioned regimes and the others is that in the former ones the phase is bounded. When it is unbounded the dynamics moves away from the local region of the phase space manifold for which the reservoir has been trained. The reservoir output map is computed so that the reservoir states in a certain region of the reservoir phase space are projected to reproduce the behaviour of the variables to be inferred, but if the unbounded dynamics drives the reservoir states away from that region, the output map might produce inaccurate or wrong values of the target variables and therefore the cross-prediction might not work. Taking the $2\pi$-modulus on the phase does not solve this since this operation is introduced independently of the nonlinear behaviour of the other dynamical variables and therefore the cross-prediction method is unable to extract this information from these
variables.

![Figure 12: Failure in the cross-prediction of quasiperiodic dynamics. The input variable is the amplitude $A$. The exact time series are represented by blue lines, while the inferred time series are represented by dashed red lines. The phase is not inferred correctly. The cross-prediction algorithm parameters used for obtaining this result are shown in Table 5.](image)

A way to overcome this is using a finite-time phase instead of the usual phase. The finite-time phase at time $t$ is defined as the difference between the phase at that time and the phase at the previous sampling time, $\phi_d(t) = \phi(t) - \phi(t - \Delta t)$. In Figure 13 we can observe that implementing this finite-time phase allows us to reconstruct the quasiperiodic dynamics accurately. In this case we use the amplitude as the input variable. The parameters used for the reconstruction are shown in Table 5.
Figure 13: Cross-prediction of quasiperiodic dynamics. The input variable is the amplitude $A$. The exact time series are represented by blue lines, while the inferred time series are represented by dashed red lines. In this case the phase is inferred accurately. The cross-prediction algorithm parameters used for obtaining this result are shown in Table 5.

We can also reconstruct chaotic dynamics in this amplitude, finite-time phase and carrier density set of coordinates, but to be accurate enough we have to use a larger reservoir with 1200 nodes. In Figures 14, 15 and 16 we exemplify that chaotic dynamics can be reconstructed accurately when using the amplitude, the finite-time phase and the carrier density as the input variable, respectively. We used 20000 samples in the training phase and 1000 in the prediction phase. Recall that the cross-prediction algorithm parameters used for obtaining these figures are presented in Table 5.
Figure 14: Cross-prediction of chaotic laser dynamics. The input variable is the amplitude $A$. The exact time series are represented by blue lines, while the inferred time series are represented by dashed red lines. The cross-prediction algorithm parameters used for obtaining this result are shown in Table 5.
Figure 15: Cross-prediction of chaotic laser dynamics. The input variable is the finite-time phase. The exact time series are represented by blue lines, while the inferred time series are represented by dashed red lines. The cross-prediction algorithm parameters used for obtaining this result are shown in Table 5.
Figure 16: Cross-prediction of chaotic laser dynamics. The input variable is the carrier density $n$. The exact time series are represented by blue lines, while the inferred time series are represented by dashed red lines. The cross-prediction algorithm parameters used for obtaining this result are shown in Table 5.

As stated before, we can observe that the chaotic laser dynamics are reproduced accurately when using the amplitude as the input variable in Figure 14, when using the finite-time phase in Figure 15 and when using the carrier density, as can be seen in Figure 16.

We quantify now the accuracy of the cross-prediction when applied to the optically injected semiconductor laser (Eqs. 1-2) when using the amplitude, finite-time phase and carrier density set of coordinates. In Table 6 we present the mean RMS error obtained when using the cross-prediction algorithm over 20 realizations. The parameters of this algorithm are those presented in Table 5 except for the time parameters. In this case the training phase lasts for $T = 50000$ and the inference phase for $T_{end} = 1000$. 

40
<table>
<thead>
<tr>
<th>Regime</th>
<th>Input variable</th>
<th>Inferred variable 1</th>
<th>Inferred variable 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stationary dynamics</td>
<td>$A$</td>
<td>$\phi_d: (5.59 \pm 0.82) \times 10^{-9}$</td>
<td>$n: (9.1 \pm 1.6) \times 10^{-9}$</td>
</tr>
<tr>
<td></td>
<td>$\phi_d$</td>
<td>$A: (1.04 \pm 0.17) \times 10^{-8}$</td>
<td>$n: (5.49 \pm 0.83) \times 10^{-9}$</td>
</tr>
<tr>
<td></td>
<td>$n$</td>
<td>$A: (4.13 \pm 0.61) \times 10^{-9}$</td>
<td>$\phi_d: (6.70 \pm 0.93) \times 10^{-9}$</td>
</tr>
<tr>
<td>Bounded phase limit cycle dynamics</td>
<td>$A$</td>
<td>$\phi_d: (2.42 \pm 0.36) \times 10^{-5}$</td>
<td>$n: (1.68 \pm 0.23) \times 10^{-5}$</td>
</tr>
<tr>
<td></td>
<td>$\phi_d$</td>
<td>$A: (1.34 \pm 0.17) \times 10^{-5}$</td>
<td>$n: (1.86 \pm 0.29) \times 10^{-5}$</td>
</tr>
<tr>
<td></td>
<td>$n$</td>
<td>$A: (2.47 \pm 0.31) \times 10^{-5}$</td>
<td>$\phi_d: (1.72 \pm 0.21) \times 10^{-5}$</td>
</tr>
<tr>
<td>Unbounded phase limit cycle dynamics</td>
<td>$A$</td>
<td>$\phi_d: (1.53 \pm 0.18) \times 10^{-5}$</td>
<td>$n: (1.55 \pm 0.21) \times 10^{-5}$</td>
</tr>
<tr>
<td></td>
<td>$\phi_d$</td>
<td>$A: (1.83 \pm 0.23) \times 10^{-5}$</td>
<td>$n: (1.53 \pm 0.16) \times 10^{-5}$</td>
</tr>
<tr>
<td></td>
<td>$n$</td>
<td>$A: (1.88 \pm 0.22) \times 10^{-5}$</td>
<td>$\phi_d: (1.64 \pm 0.20) \times 10^{-5}$</td>
</tr>
<tr>
<td>Period-3 dynamics</td>
<td>$A$</td>
<td>$\phi_d: (5.21 \pm 0.39) \times 10^{-5}$</td>
<td>$n: (2.07 \pm 0.22) \times 10^{-5}$</td>
</tr>
<tr>
<td></td>
<td>$\phi_d$</td>
<td>$A: (2.81 \pm 0.23) \times 10^{-5}$</td>
<td>$n: (1.60 \pm 0.11) \times 10^{-5}$</td>
</tr>
<tr>
<td></td>
<td>$n$</td>
<td>$A: (1.79 \pm 0.17) \times 10^{-5}$</td>
<td>$\phi_d: (4.87 \pm 0.54) \times 10^{-5}$</td>
</tr>
<tr>
<td>Quasiperiodic dynamics</td>
<td>$A$</td>
<td>$\phi_d: (4.66 \pm 0.22) \times 10^{-4}$</td>
<td>$n: (1.91 \pm 0.092) \times 10^{-4}$</td>
</tr>
<tr>
<td></td>
<td>$\phi_d$</td>
<td>$A: (3.86 \pm 0.20) \times 10^{-4}$</td>
<td>$n: (1.18 \pm 0.061) \times 10^{-4}$</td>
</tr>
<tr>
<td></td>
<td>$n$</td>
<td>$A: (1.756 \pm 0.090) \times 10^{-4}$</td>
<td>$\phi_d: (8.24 \pm 0.33) \times 10^{-4}$</td>
</tr>
<tr>
<td>Chaotic laser dynamics</td>
<td>$A$</td>
<td>$\phi_d: (6.9 \pm 1.5) \times 10^{-2}$</td>
<td>$n: (5.7 \pm 2.9) \times 10^{-2}$</td>
</tr>
<tr>
<td></td>
<td>$\phi_d$</td>
<td>$A: (3.18 \pm 0.65) \times 10^{-2}$</td>
<td>$n: (2.53 \pm 0.74) \times 10^{-3}$</td>
</tr>
<tr>
<td></td>
<td>$n$</td>
<td>$A: (4.9 \pm 3.3) \times 10^{-2}$</td>
<td>$\phi_d: (1.55 \pm 0.33) \times 10^{-1}$</td>
</tr>
</tbody>
</table>

Table 6: Mean RMS error of the cross-prediction inference over 20 realizations for the amplitude and finite-time phase of the electric field and carrier density set of coordinates. The cross-prediction parameters used are presented in Table 5 except for the length of the training and inference phases, which are $T = 50000$ and $T_{end} = 1000$ respectively.

The results shown in Table 6 are similar to those in Table 4 for the real and imaginary description of the electric field. It is found that the dynamics can be reconstructed accurately in any dynamical regime regardless of which the input variable is. When fixing the cross-prediction algorithm parameters it can be observed that the RMS error is larger when the dynamics is more complex. The chaotic laser regime shows the most complex dynamics and the RMS error is larger for this regime. Again, the accuracy of the inference does not depend significantly on the input variable, although we had to use the finite-time phase instead of the usual phase.

We consider now the semiconductor laser in the chaotic regime and we describe its dynamics using the amplitude, finite-time phase and carrier density set of coordinates. We proceed to show the dependence of the accuracy of the cross-prediction method on the algorithm parameters. To do so, we use the parameters as stated in Table 5 except for the length of the training and inference phases. In this case the training phase length is $T = 50000$ and
Phase space reconstruction of sc. laser dynamics using reservoir computing A.Cunillera

the inference phase length is $T_{end} = 1000$. We fix all the reservoir parameters but one and we reconstruct the dynamics for different values of the free parameter and we take the mean over 20 realizations with different initial conditions for the laser. For the following results we use the amplitude as the input variable, so we represent the mean RMS error for the reconstructed finite-time phase on the figures on the left and the carrier density on the figures on the right.

In Figure 17 we show the dependence of the accuracy of the cross-prediction on the size of the reservoir.

![Figure 17](image.png)

(a) RMS error of the reconstruction of the finite-time phase in the chaotic laser regime for different sizes of the reservoir.

(b) RMS error of the reconstruction of the carrier density in the chaotic laser regime for different sizes of the reservoir.

Figure 17: Dependence of the cross-prediction accuracy on the size of the reservoir for the chaotic laser regime.

We can see that a larger reservoir usually produces more accurate results, however the computation time also increases. A reservoir with 1000 nodes can be used to obtain fast and accurate results. In Figure 17b we observe that the reconstruction of the dynamics of the carrier density is not accurate when using a small reservoir. The parameter configuration shown in Table 5 is optimized for obtaining the best results for the finite-time phase, since when using the amplitude as the input variable it is more difficult to obtain accurate results for the finite-time phase reconstruction than for the carrier density. We observed that a small reservoir already obtains an accurate inference of the carrier density time series when using a different parameter configuration, but in that case the finite-time phase reconstruction is not accurate.

In Figure 18 we show the dependence of the accuracy of the cross-prediction on the
average degree of the reservoir, $D$.

![Graph](image)

(a) RMS error of the reconstruction of the finite-time phase in the chaotic laser regime for different values of the average degree of the reservoir, $D$.

(b) RMS error of the reconstruction of the carrier density in the chaotic laser regime for different values of the average degree of the reservoir, $D$.

Figure 18: Dependence of the cross-prediction accuracy on the density of the reservoir for the chaotic laser regime.

We can see that the accuracy of the cross-prediction does not depend significantly on the density of the reservoir when it is dense enough, although dense reservoirs work better than the sparse ones.

In Figure 19 we represent the dependence of the RMS error on the spectral radius of the adjacency matrix of the reservoir, $\rho$. 

43
Phase space reconstruction of sc. laser dynamics using reservoir computing  A.Cunillera

Figure 19: Dependence of the cross-prediction accuracy on the spectral radius of the adjacency matrix of the reservoir.

In the figure above we can see that the RMS error is minimum for \( \rho \) around 1 for the finite-time phase reconstruction, while for the carrier density it is minimum around 1.3. In this case, a small spectral radius \( \rho \) does not produce good results for the carrier density reconstruction.

We now consider in Figure 20 the dependence of the accuracy of the cross-prediction algorithm on the strength of the input layer connections, which is determined by the parameter \( \sigma \).
Phase space reconstruction of sc. laser dynamics using reservoir computing  A.Cunillera

Figure 20: Dependence of the cross-prediction accuracy on the parameter $\sigma$ for the chaotic laser regime.

In this case, the smaller $\sigma$ is, the more accurate the inferences are, although $\sigma$ cannot be arbitrarily small since $\sigma = 0$ implies that there is no input to the reservoir. We keep a value of $\sigma$ of order 1 to avoid numerical issues.

In Figure 21 we show the dependence of the accuracy of the cross-prediction on the parameter $\delta \in (0, 1]$, which was described in Eq. (9).
In the figures above we observe that the most accurate inferences of the finite-time phase and the carrier density may require different parameter configurations of the algorithm. The RMS error of the finite-time phase is minimum around $\delta = 0.4$, while the minimum for the carrier density is around $\delta = 0.2$. We used $\delta = 0.4$ since it was good enough for reconstructing both variables accurately. When $\delta$ is close to 1 the carrier density is not reconstructed accurately. Notice also that in this work we use the cross-prediction algorithm to reconstruct these two variables at the same time, but one could use two different reservoir to reconstruct these variables separately.

Last, we show the dependence of the RMS error on the bias parameter $\xi$ in the following figure.
In Figure 22 we observe that the cross-prediction is more accurate when using a bias around 1.2 and that the carrier density reconstruction is not accurate when the bias parameter is large.

To test the robustness of the cross-prediction method we introduce now gaussian white noise in the measurements. We do not introduce it in the rate equations but in the measurement procedure. We define the amplitude of the time series of a variable as the difference between the global maximum and the global minimum of the time series of that variable. To create the noisy measurements, we add gaussian white noise of zero mean and standard deviation a certain percentage of the amplitude of the measurements of that observable to its time series.

In Figure 23 the RMS error of the inferred variables is represented as an error bar plot. The input variable is the amplitude of the electric field. The RMS error on Figure 23a corresponds to the reconstructed finite-time phase, while the RMS error on Figure 23b corresponds to the carrier density. The cross-prediction algorithm parameters are those in Table 5 for the chaotic laser regime, except for the time parameters. In this case the training phase length is $T = 50000$ and the inference phase length is $T_{end} = 1000$. We take the mean over 20 realizations, with different initial conditions in each realization. We add gaussian white noise to the time series from 0.01% to 1% of the amplitude of the time series.
Phase space reconstruction of sc. laser dynamics using reservoir computing

A.Cunillera

Figure 23: Dependence of the cross-prediction accuracy on the strength of the noise for the chaotic laser regime.

In the figure above we may observe that the RMS error increases approximately linearly with the strength of the noise, although this might not be the case if the noise was stronger. We may conclude then that the dynamics can still be reconstructed when there is small noise in the measurements, with values of the RMS error below 0.1 for noise strengths below 1%.

We have shown that the cross prediction method can be used to reconstruct accurately the dynamics of an optically injected semiconductor laser described by Eqs. (1-2) in any dynamical regime. To do so, we only need time series for one of the dynamical variables of the laser and we can use both the real and imaginary components and the amplitude and finite-time phase descriptions of the electric field. We have also computed the dependence of the method on the parameters of the algorithm for the amplitude and finite-time phase description of the chaotic laser. We have discussed the robustness of the method when adding noise to the time series.
3.2 Reservoir computing forecasting results

Last, we present the results obtained for the autonomous and the hybrid scheme predictors. We introduce a parameter mismatch in the $\alpha$ parameter in the mathematical model inside the hybrid algorithm. We compare the performance of these two machine learning methods and the model (Eqs. 1-2) with the same parameter mismatch as in the hybrid predictor.

From now on, we will consider the laser model in the chaotic regime and we will describe the electric field using its real and imaginary components.

The time beyond which a prediction is no longer accurate is called the prediction horizon. In [22], it is defined as the time in which the normalized error

$$\text{Err}(t) = \frac{\|u(t) - \hat{u}(t)\|}{\sqrt{\langle \|u(t)\|^2 \rangle}}$$

first exceeds a certain threshold $h \in (0, 1)$. In this work, $h = 0.4$.

Figure 24 shows an example of autonomous reservoir computing time series prediction. The parameters of the algorithm are presented in Table 7, but in the training phase length is $T = 3000$ and the prediction phase length is $T_{\text{end}} = 100$. Recall that $f$ is the fraction of the reservoir nodes that receive input from the mathematical model in the hybrid algorithm.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>$N$</td>
<td>600</td>
</tr>
<tr>
<td>$D$</td>
<td>10</td>
</tr>
<tr>
<td>$\rho$</td>
<td>1</td>
</tr>
<tr>
<td>$\sigma$</td>
<td>1</td>
</tr>
<tr>
<td>$\delta$</td>
<td>1</td>
</tr>
<tr>
<td>$\xi$</td>
<td>1</td>
</tr>
<tr>
<td>$T$</td>
<td>100000</td>
</tr>
<tr>
<td>$T_{\text{end}}$</td>
<td>30000</td>
</tr>
<tr>
<td>$\beta$</td>
<td>$10^{-4}$</td>
</tr>
<tr>
<td>$f$</td>
<td>0.5</td>
</tr>
</tbody>
</table>

Table 7: Reservoir parameters for the autonomous reservoir computing and the hybrid predictors. They are used for forecasting the chaotic laser dynamics for the real and imaginary components of the electric field and the carrier density.

In these forecasting algorithms the time series sampling time step is the same as the integration one, $\Delta t = 0.1$, as opposed to the cross-prediction algorithm. Since in the hybrid predictor we integrate the rate equations (1-2) with a parameter mismatch in the $\alpha$ parameter, if we used in the hybrid algorithm an integration time step different from the sampling time
and the integration time step that we used when obtaining the time series, we would introduce errors that come from the numerical integration.

In Figure 24 the exact time series are represented by blue lines, while the predicted time series are represented by dashed red lines. The error $\text{Err}(t)$ is plotted in black, and the vertical dashed line shows the prediction horizon, in this case it is around 64 (30.64-3000). Since the Lyapunov exponent is around 0.053, so the Lyapunov time of this dynamical system is around 19 units of time and therefore the forecast is accurate for 3.4 Lyapunov times. Notice that beyond that time the predicted time series differ significantly from the exact ones.

![Figure 24](image.png)

Figure 24: Autonomous reservoir computing prediction of chaotic laser dynamics. The exact time series are represented by blue lines, while the predicted time series are represented by dashed red lines. The error $\text{Err}(t)$ is plotted in black, and the vertical dashed line shows the prediction horizon.

Now we will compare the performance of the reservoir computing prediction algorithms and an approximate mathematical model. To do so, we introduce a mismatch in the $\alpha$ pa-
rameter of the model and the hybrid scheme, as stated before. First, we compute the time series of the chaotic laser for $\alpha = 2$, then we use these three methods to predict the dynamics of the laser and finally we compare the predictions with the original time series. We compute the prediction horizon with these three methods for 20 realizations with different initial conditions in each realization and for each value of the mismatched parameter. The algorithm parameters are presented in Table 7. We use three different values of the mismatched $\alpha$: a small mismatch $\alpha = 2.01$, a medium one $\alpha = 2.1$ and a large one $\alpha = 3$.

In the following figure we show a comparison of the prediction horizons of these methods as function of the size of the reservoir when a small mismatch, $\alpha = 2.01$, is introduced. We represent in red the prediction horizon computed by integrating the rate equations with that parameter mismatch, in blue the autonomous reservoir computing predictor results and in black the hybrid predictor scores.

Figure 25: Comparison of the prediction accuracy of an approximate mathematical model, an autonomous reservoir computing predictor and a hybrid method when a parameter mismatch $\alpha = 2.01$ is introduced in these algorithms (the real value is $\alpha = 2$) for different sizes of the reservoir of the algorithms. We represent in red, blue and black the prediction horizon for the mathematical model, the autonomous predictor and the hybrid scheme, respectively.
In the figure above we can observe that the hybrid predictor doubles the prediction horizon of the mathematical model for most of the reservoir sizes considered and it outperforms the other methods when the reservoir is small. The autonomous predictor can produce predictions as accurate as the hybrid scheme when the reservoir is large, but when it is small the autonomous algorithm predictions are worse than those from the mathematical model. The hybrid algorithm prediction horizon is between 4.6 and 6.1 Lyapunov times, the autonomous reservoir computing algorithm’s is between 0.4 and 5.5 Lyapunov times and the approximate model prediction horizon is 3.1 Lyapunov times.

In Figure 26 the mismatch introduced is a 5% of the value of the original parameter. Again, we present the prediction horizon obtained from the three predictors.

Figure 26: Comparison of the prediction accuracy of an approximate mathematical model, an autonomous reservoir computing predictor and a hybrid method when a parameter mismatch $\alpha = 2.1$ is introduced in these algorithms (the real value is $\alpha = 2$) for different sizes of the reservoir of the algorithms. We represent in red, blue and black the prediction horizon for the mathematical model, the autonomous predictor and the hybrid scheme, respectively.
Phase space reconstruction of sc. laser dynamics using reservoir computing A.Cunillera

If we compare Figures 25 and 26, the prediction horizon of the approximate model has decreased from 3.1 to 1.3 Lyapunov times, while the mismatch has increased from $\alpha = 2.01$ to 2.1. The autonomous reservoir computing predictor does not depend on the mismatch. The hybrid predictor shows a similar behaviour in both figures, it produces better predictions when the reservoir is small while when the reservoir is large they are as good as the ones produced by the autonomous predictor.

In Figure 27 we introduce a large mismatch, $\alpha = 3$. The simulation configuration is the same as in the previous two figures.

![Figure 27: Comparison of the prediction accuracy of an approximate mathematical model, an autonomous reservoir computing predictor and a hybrid method when a parameter mismatch $\alpha = 3$ is introduced in these algorithms (the real value is $\alpha = 2$) for different sizes of the reservoir of the algorithms. We represent in red, blue and black the prediction horizon for the mathematical model, the autonomous predictor and the hybrid scheme, respectively.](image)

This time, the approximate model forecast is inaccurate, it is only valid for 0.1 Lyapunov
times, and both reservoir computing methods produce better predictions even when a small
reservoir is used in the case of the autonomous reservoir scheme. Again, the hybrid method
produces better forecasts than the autonomous one when the reservoir has few nodes but
the accuracy of the predictions are similar when the reservoir is large. Notice that even
though the internal model of the hybrid scheme does not produce good forecasts by itself,
this hybrid reservoir computing method still manages to produce predictions as accurate as
when the internal model had a small mismatch.

Since the computation time grows with the size of the reservoir, it is shown that the hy-
brid predictor can be used with a small reservoir to produce fast and accurate predictions.
This method outperforms the autonomous reservoir computing algorithm for small networks,
but when the reservoir is large, the accuracy of both methods are similar. It is found that the
hybrid scheme accuracy does not depend significantly on the mismatch on its internal math-
ematical model. When the mismatch is large the mathematical model produces inaccurate
predictions but the hybrid predictor still produces good predictions.
4 Conclusion

In this work we have used a reservoir computing algorithm for inferring the time series of unmeasured dynamical variables of a dynamical system that describes the behaviour of an optically injected class B semiconductor laser. Two reservoir computing methods have been used for predicting the laser dynamics, one of them uses a mathematical model of the system. We have also discussed why these techniques work from a mathematical point of view.

We found that the cross-prediction algorithm can be used to reconstruct accurately the dynamics of two of the three variables of the semiconductor laser when the third one is used as the algorithm input. All the dynamical regimes that can be shown by the semiconductor laser dynamical system (Eqs. 1-2) can be reconstructed with this method. Moreover, the laser dynamics can be described with two different sets of coordinates, one is the real and imaginary components of the electric field and the carrier density and the other is the amplitude and phase of the electric field and the carrier density. This algorithm can be used for both sets of coordinates and for any input variable, although the phase must be replaced by a finite-time phase when the semiconductor laser shows unbounded phase dynamics. We have also discussed the dependency of the method on 6 different algorithm parameters. Furthermore, we have demonstrated the robustness of the method under the addition of small gaussian white noise on the time series.

We have also shown that the dynamics can be predicted by using two different reservoir computing algorithm: an autonomous and a hybrid predictor. The latter one combines the autonomous reservoir computing algorithm with an approximate mathematical model, which in this case is the rate equations (1-2) with a mismatch in the linewidth enhancement factor $\alpha$. We have compared the performance of these two methods with the approximate mathematical model. To do so, we have computed the prediction horizon, which is the time beyond which the prediction is no longer accurate, for reservoirs of different sizes and different values of the mismatched parameter. We have demonstrated that the hybrid algorithm outperforms the other two methods and it can extend the accuracy of its forecasts beyond 6 Lyapunov times. Moreover, this algorithm can produce long and accurate predictions when using a small reservoir, so it can produce accurate results faster than the autonomous reservoir computing predictor, whose accuracy is only similar to the hybrid algorithm’s when the reservoir is large. The hybrid predictor is found to be robust under changes on the parameter mismatch on its internal mathematical model.

As we mentioned before, these methods can be of interest in data science, control engineering and experimental physics. An open question is whether these techniques can be used for testing data encryption security in chaotic laser synchronization based cryptography [4, 9]. When synchronizing two chaotic lasers, complex dynamics emerges and one laser can be used to send a message to the other laser. It is shown that if a modulation is introduced
in the transmitter dynamics, the receiver strongly suppresses that perturbation. This way, a message can be encrypted in the transmitter laser dynamics. To recover the message, the receiver has to measure and subtract the transmitter and the receiver laser outputs. This method provides some level of security, but might have a backdoor if there was a way of training a receiver by machine learning methods. If the time series of the transmitter laser’s amplitude could be measured and a reservoir was trained for reproducing the receiver laser dynamics, the message encryption could be jeopardized.
Phase space reconstruction of sc. laser dynamics using reservoir computing A.Cunillera

References


Phase space reconstruction of sc. laser dynamics using reservoir computing

A. Cunillera


Phase space reconstruction of sc. laser dynamics using reservoir computing

A.Cunillera


