Università degli Studi di Padova

University of Padua
Department of Physics and Astronomy "G. Galilei"
Master Thesis in Physics of Data

Modeling brain dynamics through recurrent neural networks: a linearization approach

Degree candidate

Sandra Elsa Sanjai

ID 2113951

Supervisor

Dr. Michele Allegra

University of Padua

Academic Year 2025/2026

Contents

In computational neuroscience, recurrent Neural Networks (RNNs) have recently emerged as a flexible tool to model the dynamics of biological neural circuits. Often, different types of RNNs can adequately model the data, but build on different premises and lead to different descriptions of dynamics. In this thesis, we will review an RNN approach commonly used to model neural dynamics: a reservoir computing approach, where a high-dimensional, linear RNN is trained via linear regression Using both simulated data and real data from functional magnetic resonance imaging, the approach will be systematically analyzed, testing its accuracy in reproducing several types of dynamical systems.

1 Introduction

The study of complex dynamical systems is a central pillar of contemporary scientific research, with the human brain standing as one of the most challenging and intricate systems to model. In recent decades, the field has seen a rapid evolution of data-driven methodologies, particularly machine learning techniques, which have proven highly effective at modeling complex datasets and delivering accurate temporal predictions. Within this landscape, Reservoir Computing (RC) has emerged as a particularly powerful framework [1, 2, 3, 4]. By leveraging high-dimensional recurrent neural networks (RNNs) with fixed internal couplings, RC can transform input time series into rich representations, capturing a system's dynamics with remarkable computational efficiency. In particular, RC can be leveraged to construct "digital twins" - autonomous in silico replicas that are statistically equivalent to a system under investigation [5]. Digital twins serve as a paradigm for analyzing and simulating dynamical systems, offering a platform for in silico experiments. In neuroscience, digital twins of the brain may pave the way for personalized medicine and the design of targeted intervention strategies, such as neurorehabilitation.

The pursuit of predictive accuracy in machine learning has often come at the expense of interpretability. While inferred models may be accurate, they frequently function as "black boxes", obscuring the underlying physical principles of the observed system. To bridge this gap, Koopman theory provides a valuable framework by mapping nonlinear systems into higher-dimensional linear dynamics [6]. By inferring a finite-dimensional linear representation, researchers can regain straightforward interpretability, gaining insights into the specific time scales and dynamical modes of complex systems.

This thesis reviewed and tested a novel route to constructing digital twins using noisy linear recurrent neural networks (lRNNs) within the reservoir computing framework, recently developed in Ref. [7]. We first review the underlying mathematical theory, which relates the spectral properties of the trained networks - specifically their Laplace-transform poles — with the spectral properties of the observed system. In particular, we discuss how the lRNN model yield a high dimensional, linear representation of dynamics that can be directly related with the corresponding Koopman operator.

A significant portion of this work then focuses on the application of this framework to infraslow brain signals. Infra-slow oscillations are defined as brain activity fluctuations in the range $0.01 \leq f \leq 0.1$ Hz. Infraslow fluctuations (IFSs) have been observed through many measurement techniques, including functional magnetic resonance imaging (fMRI), electro- and magnetoencephalography (EEG/MEG), and single unit recordings [8]. In particular, the BOLD signal of fMRI is dominated by infraslow (< 0.1 Hz) fluctuations. ISFs are correlated across brain regions and are commonly used to infer functional connections in the brain [9]. ISFs are hypothesized to play a relevant role in modulating neural excitability [10], organizing large-scale cortical activity [11], and orchestrating brain function when coupled with faster oscillations.

While fMRI signals were already investigated in the reference article, here we aim to perform a replication study using a different fMRI data set, potentially highlighting some limitations of the approach. We show that stochastic lRNNs can effectively recover the correct dissipative structure of brain dynamics even when working with heavily filtered datasets and limited sample size conditions, where established linear modeling techniques often produce spurious spectral estimates. Moreover, we show that brain activity can be decomposed into a relatively low-dimensional latent state space that provides a compact description of dynamics in terms of a few latent modes oscillating at frequencies $\leq 0.1Hz$, matching real physiological timescales.

Importantly, this work also explores the boundaries of the linear framework by testing it against strongly nonlinear chaotic systems, such as the Lorenz and Rossler systems. We find that the model struggles to replicate these dynamics, as chaotic systems are characterized by infinite-dimensional, continuous-spectrum Koopman operators that cannot be easily approximated by a finite-dimensional linear system. In these cases, the predicted trajectories often fail to capture the complex geometric structure of the attractors, instead collapsing into simplified

spiral-like orbits. This highlights that while the lRNN framework is highly effective for systems that are Koopman-linearizable, such as resting-state brain activity, it represents a oversimplified "best attempt" when faced with genuinely chaotic dynamics.

Overall, this thesis work aims to review a robust, data-driven framework for brain modeling that captures the essential stochastic and dissipative nature of neural activity. .

2 Methods

2.1 Dynamical modeling with Reservoir Computing

In this section, we will review in detail the approach in [7], which leverages the reservoir computing framework to learn a linear dynamical system underlying observed time series. This approach will be applied to simulated time series and real time series from neuroimaing in Chapter 3.
Upon reviewing the basics of RC, we will describe how to use a linearized RC for time series modeling, and finally present the details of the implementation used in Chapter 3.

2.1.a Reservoir computing

Reservoir computing (RC) emerged in the early 2000's as a powerful and efficient machine learning paradigm specifically designed for processing temporal or sequential data [1, 2, 3]. It is based on a type of recurrent neural network in which the core component, known as the reservoir, consists of a large number of interconnected nodes with fixed weights. When input data is fed into the system, it propagates through this reservoir, generating a complex set of dynamic responses that capture the temporal structure and dependencies within the data. These internal states create a rich, high-dimensional representation of the input without requiring the network itself to be trained. Instead of adjusting all the network parameters, as is done in traditional recurrent neural networks, reservoir computing only trains the output layer, which maps the reservoir's internal states to the desired output. This significantly reduces computational cost, simplifies training, and avoids issues such as vanishing or exploding gradients.

One of the key advantages of reservoir computing is its ability to model nonlinear and time-dependent processes with relatively low effort and high efficiency. Because the reservoir remains fixed, it can be reused across different tasks, and training typically involves straightforward linear regression techniques. This makes RC particularly suitable for real-time applications and systems with limited computational resources. It has been successfully applied in various domains, including speech and handwriting recognition, financial forecasting, robotics, and the analysis of biological signals such as EEG data [4].

2.1.b RC: basic formulation

A Recurrent Neural Network (RNN) is an artificial neural network architecture specialized in processing sequential data, characterized by cyclic connections that allow the maintenance of an internal state (hidden state). While traditional deep learning networks assume that input-output pairs are independent, in recurrent neural networks the output depends on the sequence of previous inputs. In fact, RNNs have a form of "internal memory" that allows them to use information from previous steps in the sequence to influence the current output. Given an input sequence $\boldsymbol{\omega}(t) \in \mathbb{R}^M$, a RNN updates its hidden state $\mathbf{r}(t) \in \mathbb{R}^N$ according to the equation:

$$\mathbf{r}(t) = \Phi(W\mathbf{r}(t-1)) + W^{in}\boldsymbol{\omega}(t) + \mathbf{b}),$$

where $W \in \mathbb{R}^{N\times M}$ and $W^{in} \in \mathbb{R}^{N\times M}$ are learnable weight matrices, $\mathbf{b}$ is a bias vector, and Φ is a nonlinear activation function. The output $\mathbf{y}(t) \in \mathbb{R}^K$ is computed as:

$$\mathbf{y}(t) = W^{out}\mathbf{r}(t)$$

where $W^{out} \in \mathbb{R}^{K\times N}$. The weights can be trained to yield a target output $\mathbf{y}^*(t)$, i.e.,

$$\mathbf{y}(t) \to \mathbf{y}^*(t)$$

The reservoir computing approach uses a recurrent neural network (RNN) with fixed internal connections, called the *reservoir*. In this context, the "echo state" property emerges spontaneously from the collective dynamics of the RNN. Assuming that standard compactness conditions are satisfied (input and state spaces are closed and bounded) and that the network has no output feedback connections (outputs are not fed back into the reservoir), the network possesses echo states if the network state $\mathbf{r}(t)$ is uniquely determined by any left-infinite input sequence $\boldsymbol{\omega}(t-1), \boldsymbol{\omega}(t-2), \ldots$. In other words, the initial state does not

matter: after processing a sufficiently long input sequence, the internal state depends only on past inputs and not on the initial condition. This is fundamental because it means the network "forgets" initial conditions and responds consistently to inputs.
In a continuous-time formulation, each unit j of the network ($j = 1, \ldots, N$) has an activity state $r_j(t)$ evolving over time according to:

$$\tau \frac{dr_j(t)}{dt} = \Phi(h_j) - r_j,$$

where τ is the decay constant and $\Phi(h_j)$ is the activation function, identical for all units. The synaptic input h_j is the weighted sum:

$$h_j(t) = \sum_{k=1}^{N} W_{jk} r_k(t) + \sum_{k=1}^{M} W_{jk}^{in} \omega_k(t)$$

Due to the high dimensionality of the RNN state space, a simple linear transformation represented by W^{out} is often sufficient to accurately map the RNN state into a time series $\boldsymbol{\omega}(t)$ that is functionally related to the input.

Given $R \in \mathbb{R}^{N \times T}$ (where the k-th row represents the time series of the k-th unit over a training period of T steps) and $\Omega \in \mathbb{R}^{K \times T}$ (whose rows contain the K observables $\omega_k(t)$ to be reproduced), the linear mapping is computed using ridge regression. The latter is a variant of linear regression that introduces a quadratic penalty (L2 norm) on the coefficients to reduce their variance. The model optimizes the following cost function:

$$J(W) = ||Y - XW||_2^2 + \beta ||W||_2^2$$

$$= \text{squared error} + \beta \cdot \text{coefficient penalty}$$

where Y is the matrix of observations, X is the matrix of predictors, W is the coefficient matrix, and the hyperparameter $\beta \ll 1$ controls the strength of regularization. The closed-form solution is:

$$W^* = (X^T X + \beta I)^{-1} X^T Y,$$

where I is the identity matrix. Regularization prevents non-invertibility of $X^T X$ when variables are correlated.

2.1.c Linearized RC for dynamical system learning

A regular non-linear RNN obeys the following dynamics:

$$\tau \frac{d\mathbf{r}(t)}{dt} = \Phi(\mathbf{h}(t)) - \mathbf{r}(t)$$

where $\Phi(\mathbf{h}(t))$ is a non-linear activation function. In the case of weak recurrent coupling or small input, we can approximate $\Phi(\mathbf{h}(t)) \approx \mathbf{h}(t)$, simplifying the dynamics as:

$$\tau \frac{d\mathbf{r}(t)}{dt} = \mathbf{h}(t) - \mathbf{r}(t)$$

This simplification is valid if the recurrent weights are small or if the input driving the reservoir is weak. For a network driven by fMRI signals, as in Chapter 3, the assumption is valid, as these signals are slow, smooth, and low-amplitude with low frequency components.

Accounting for both recurrent and external input, the linearized RNN equation is:

$$\tau \frac{d\mathbf{r}(t)}{dt} = \mathbf{r}(t) + W^{in}\boldsymbol{\omega}(t) - \mathbf{r}(t)$$

$$\tau \frac{d\mathbf{r}(t)}{dt} = (W - I)\mathbf{r}(t) + W^{in}\boldsymbol{\omega}(t)$$

Dividing by τ, we obtain

$$\frac{d\mathbf{r}(t)}{dt} = \frac{W - I}{\tau}\mathbf{r}(t) + \frac{1}{\tau}W^{in}\boldsymbol{\omega}(t).$$

$$\frac{d\mathbf{r}(t)}{dt} - \frac{W - I}{\tau}\mathbf{r}(t) = \frac{1}{\tau}W^{in}\boldsymbol{\omega}(t)$$

We can multiply the whole equation by the integrating factor $e^{-\frac{W-I}{\tau}t}$:

$$e^{-\frac{W-I}{\tau}t}\frac{d\mathbf{r}(t)}{dt} - e^{-\frac{W-I}{\tau}t}\frac{W - I}{\tau}\mathbf{r}(t) = e^{-\frac{W-I}{\tau}t}\frac{1}{\tau}W^{in}\boldsymbol{\omega}(t)$$

Now, noticing that the LHS is the derivative of the product $e^{-\frac{W-I}{\tau}t}\mathbf{r}(t)$, we obtain

$$\frac{d}{dt}\left(e^{-\frac{W-I}{\tau}t}\mathbf{r}(t)\right) = e^{-\frac{W-I}{\tau}t}\frac{1}{\tau}W^{in}\boldsymbol{\omega}(t)$$

Integrating both sides, we get

$$\int_{t_0}^{t} \frac{d}{ds}\big(e^{-\frac{W-I}{\tau}s}\mathbf{r}(s)\big)ds = \int_{t_0}^{t} e^{-\frac{W-I}{\tau}s}\frac{1}{\tau}W^{in}\boldsymbol{\omega}(s)ds$$

$$\Rightarrow \quad \mathbf{r}(t) = e^{\frac{W-I}{\tau}(t-t_0)}\mathbf{r}(t_0) + \int_{t_0}^{t} e^{\frac{W-I}{\tau}(t-s)}\frac{1}{\tau}W^{in}\boldsymbol{\omega}(s)ds$$

If the system is stable, and we assume $t_0 \to -\infty$ (which corresponds to assuming that the system has evolved for a sufficiently long time), then only the second term survives and

$$\mathbf{r}(t) = \int_{-\infty}^{t} e^{\frac{W-I}{\tau}(t-s)}\frac{1}{\tau}W^{in}\boldsymbol{\omega}(s)ds \tag{2.1}$$

From (2.1), we see that $r(t)$ depends on all past inputs $\boldsymbol{\omega}(t)$. Taking the i-th element $r_i(t)$, one can see that the equation is a temporal filter:

$$r_i(t) = \int_{-\infty}^{t} \left[e^{(W-I)(t-s)/\tau}W^{in}\right]_i \boldsymbol{\omega}(s)ds$$

with weights given by the row matrix $\left[e^{(W-I)(t-s)/\tau}W^{in}\right]_i$, where recent inputs correspond to larger weights. Thus, the reservoir implements a delayed filter and it contains a rich memory of past inputs.

Notoriously, Takens [12] showed that if a dynamical system has dimension d, then one can reconstruct a state space with the same dynamics using time delayed observations

$$\mathbf{z}(t) = \begin{bmatrix} \boldsymbol{\omega}(t) \\ \boldsymbol{\omega}(t-\tau) \\ \boldsymbol{\omega}(t-2\tau) \\ \vdots \end{bmatrix}$$

as long as $m \geq 2d+1$, a method called delay embedding. Takens' theorem and Eq. (2.1) suggest that the reservoir will embed enough information to reconstruct the dynamics of the driving input.

In agreement with this hypothesis, one can try to reconstruct the input from the reservoir state, as

$$\boldsymbol{\omega}(t) \approx W^{out}\mathbf{r}(t)$$

Assuming this is possible, one obtains an autonomous, continuous-time autoregressive (AR) model for $\omega(t)$. An AR model is a time series model where the current value of a signal depends linearly on its past values. In discrete time, the

definition is:

$$x_t = \sum_{k=1}^{p} a_k x_{t-k} + \epsilon_t$$

where the x_t is the value of the signal at time t, a_k are the coefficients, ϵ_t is the noise and p stands for the order of the AR model. For a continuous time signal $\omega(t)$, an AR model involves integration over a continuous range of delays:

$$\boldsymbol{\omega}(t) = \int_0^\infty G(s)\boldsymbol{\omega}(t-s)ds$$

Substituting $\omega(t) \approx W^{out} r(t)$ in Eq. (2.1), we get

$$\boldsymbol{\omega}(t) = \int_{-\infty}^{t} W^{out} e^{(W-I)(t-s)/\tau} W^{in} \boldsymbol{\omega}(s) ds$$

$$\Rightarrow \qquad \boldsymbol{\omega}(t) = \int_0^\infty G(s)\boldsymbol{\omega}(t-s)ds$$

where $G(s) = W^{out} e^{(W-I)s/\tau} W^{in}$. Thus, the reservoir computer behaves like an autoregressive model with Green's function given by $G(s)$.

The model allows for a simple description in terms of independent modes. Indeed, one can diagonalize the synaptic matrix as $W = Q\Lambda Q^{-1}$, where $\Lambda = \text{diag}(\lambda_1, \ldots, \lambda_N)$ is the diagonal matrix containing the eigenvalues and $Q = [\mathbf{q}_1 \ldots \mathbf{q}_N]$ is the matrix whose columns are the eigenvectors. The spectral decomposition of W reads

$$W = \sum_{n=1}^{N} \lambda_n \mathbf{q}_n \mathbf{q'}_n^T$$

where $\mathbf{q'}_n$ are the columns of Q^{-1}, i.e., $Q^{-1} = [\mathbf{q'}_1 \ldots \mathbf{q'}_N]$. Therefore,

$$G(s) = W^{out} Q e^{(\Lambda - I)s/\tau} Q^{-1} W^{in}$$

$$\Rightarrow \qquad G(s) = \sum_{n=1}^{N} J_n e^{(\lambda_n - 1)s/\tau}$$

where $J_n = W^{out} \mathbf{q}_n \mathbf{q'}_n^T W^{in}$. Each exponential corresponds to one dynamical mode of the system. The spectral radius $\rho = \max Re(\lambda_n)$ must obey $\rho < 1$ to ensure that all modes decay. Since $G(s)$ has N exponential modes, the reservoir behaves like an AR model with N modes. Generally, though, most of the J_n are close to zero, hence only a few modes dominate the dynamics.

The system's state is reconstructed using $\boldsymbol{\omega}(t) = W^{out}\mathbf{r}(t)$. Each row of W^{out} is

a linear decoder that reconstructs one observable from the reservoir. Only this readout matrix is trained. To this aim, one can use ridge regression The solution to this regression problem is

$$W^{\text{out}} = \Omega R^T (RR^T + \beta I)^{-1}.$$

where $R \in \mathbb{R}^{N \times T}$ and $\Omega \in \mathbb{R}^{D \times T}$ are defined as

$$R = \begin{bmatrix} r(t_1) & r(t_2) & \dots \end{bmatrix}$$

$$\Omega = \begin{bmatrix} \omega(t_1) & \omega(t_2) & \dots \end{bmatrix}$$

After using all the training data and fitting $\omega(t) = W^{out} r(t)$, one can use the trained W^{out} to obtained an autonomous model of the dynamics of the input $\boldsymbol{\omega}(t)$. To do so, it is sufficient to drive the dynamics of the reservoir by substituting the original input term with the predicted output:

$$W^{in}\boldsymbol{\omega}(t) \to W^{in} W^{out} \mathbf{r}(t)$$

Then the reservoir dynamics become:

$$\tau \frac{d\mathbf{r}(t)}{dt} = W\mathbf{r}(t) + W^{in} W^{out} \mathbf{r}(t) - \mathbf{r}(t)$$

$$\tau \frac{d\mathbf{r}(t)}{dt} = \tilde{W}\mathbf{r}(t) - \mathbf{r}(t) \tag{2.2}$$

where $\tilde{W} = W + W^{in} W^{out}$ is the learned synaptic matrix.

$$\frac{d\mathbf{r}(t)}{dt} = \frac{\tilde{W} - I}{\tau}\mathbf{r}$$

This is the 'prediction mode' of the reservoir. The reservoir evolves without external input, using instead its own generated output as the input for the next step.

Spectral analysis and Koopman modes

At time t, the reservoir has a state:

$$\mathbf{r}(t) = \begin{bmatrix} \mathbf{r}_1(t) \\ \mathbf{r}_2(t) \\ \vdots \\ \mathbf{r}_N(t) \end{bmatrix}$$

After training, the reservoir evolves as:

$$\frac{d\mathbf{r}(t)}{dt} = \frac{\tilde{W} - I}{\tau}\mathbf{r}(t)$$

With a derivation analogous to that done for the synaptic matrix, we can diagonalize the trained $\tilde{W}$ to obtain $\tilde{W} = Q\tilde{\Lambda}Q^{-1}$ where Q are the eigenvectors and $\tilde{\Lambda}$ the eigenvalues.
The eigenvectors represent pure dynamical modes. The reservoir state is always a linear combination of many eigenvectors:

$$\mathbf{r} = v_1\mathbf{q}_1 + v_2\mathbf{q}_2 + \cdots + v_N\mathbf{q}_N$$

where v_k are the coefficients, which determine the weight of each dynamical mode. In matrix form,

$$\mathbf{r} = Q\mathbf{v} \qquad \Rightarrow \qquad \mathbf{v} = Q^{-1}\mathbf{r}$$

The advantage of this representation is that v_k is the amplitude of the dynamical mode k. Because the eigenvectors diagonalize the system, when the system state $\mathbf{r}$ is exactly the eigenvector $\mathbf{q}_k$ then replacing $\mathbf{r} = Q\mathbf{v}$ one obtains:

$$\frac{d(Q\mathbf{v})}{dt} = \frac{\tilde{W} - I}{\tau}Q\mathbf{v} \qquad \frac{d\mathbf{v}}{dt} = \frac{\tilde{\Lambda} - I}{\tau}\mathbf{v}$$

Thus, the coefficients evolve independently and are decoupled:

$$\frac{dv_k}{dt} = \frac{\tilde{\lambda}_k - 1}{\tau}v_k$$

Solving this simple ODE, we get:

$$v_k(t) = v_k(0)e^{(\tilde{\lambda}_k - 1)t/\tau} \tag{2.3}$$

Using the eigenvectors, we can compute an index of "mode relevance" from $\Xi = W^{out}Q$. Indeed, since the ouput is:

$$\boldsymbol{\omega}(t) = W^{out}\mathbf{r}(t)$$

and

$$\mathbf{r}(t) = Q\mathbf{v}(t)$$

$$\boldsymbol{\omega}(t) = W^{out}Q\mathbf{v}(t)$$

we get

$$\omega_j(t) = \sum_k \Xi_{jk} v_k(0) e^{(\tilde{\lambda}_k - 1)t/\tau} \tag{2.4}$$

The absolute value of the coefficient $|\Xi_{jk}|$ determines the relevance of mode k for time the input node j. As already mentioned, not all N modes are generally relevant. Using the relevance matrix Ξ, we can generally identify a subset of k modes that dominate the dynamics.

In the reservoir modes we derived in Eq. 2.1.c, let us define $\sigma_k := \frac{\tilde{\lambda}_k - 1}{\tau}$.

$$v_k(t) = v_k(0) e^{\sigma_k t}$$

and in Eq. 2.1.c:

$$\omega_j(t) = \sum_{k=1}^{N} \Xi_{jk} v_k(0) e^{\sigma_k t}$$

We can show that the σ_k are related to the poles of the Laplace transform of the input signals. Indeed, by using the Laplace transform definition:

$$\mathcal{L} f(t)(s) = \int_0^\infty f(t) e^{-st} dt$$

we obtain

$$\mathcal{L}\{\Xi_{jk} v_k(0) e^{\sigma_k t}\}(s) = \int_0^\infty \Xi_{jk} v_k(0) e^{\sigma_k t} e^{-st} dt$$

Combining the exponentials and integrating,

$$\mathcal{L}\{\Xi_{jk} v_k(0) e^{\sigma_k t}\}(s) = \frac{\Xi_{jk} v_k(0)}{s - \sigma_k}$$

We can find the Laplace transform of the whole signal exploiting the linearity of the Laplace transform:

$$\mathcal{L}\{\omega_j(t)\}(s) = \sum_{k=1}^{N} \mathcal{L}\{\Xi_{jk} v_k(0) e^{\sigma_k t}\}(s) = \sum_{k=1}^{N} \frac{\Xi_{jk} v_k(0)}{s - \sigma_k}$$

where $\sigma_k = \frac{\tilde{\lambda}_k - 1}{\tau}$. So the poles are at

$$s = \sigma_k = \frac{\tilde{\lambda}_k - 1}{\tau}$$

If we partition the modes into relevant and irrelevant blocks

$$\mathbf{v}(t) = \begin{bmatrix} \mathbf{v}_{\text{rel}}(t)\ \mathbf{v}_{\text{irr}}(t) \end{bmatrix}, \qquad \Psi = \begin{bmatrix} \Psi_{\text{rel}} & \Psi_{\text{irr}} \end{bmatrix}.]$$

then

$$\boldsymbol{\omega}(t) = \Psi_{\text{rel}}\mathbf{v}_{\text{rel}}(t) + \Psi_{\text{irr}}\mathbf{v}_{\text{irr}}(t)$$

if the irrelevant mode have tiny weights then

$$\boldsymbol{\omega}(t) \approx \Psi_{\text{rel}}\mathbf{v}_{\text{rel}}(t)$$

Now, the observable effectively depends only on a subset of modes.

The linear modes identified with the RC approach can be related to the 'Koopman modes' of Koopman analysis. For a general dynamical system, dynamics is non-linear,

$$\mathbf{z}_{t+1} = F(\mathbf{z}_t)$$

In this thesis, we can consider $\mathbf{z}_t$ to be the latent variable denoting the state of the system, which we do not directly observe, and $F(\mathbf{z}_t)$ is unknown. Let $g(\mathbf{z}_t)$ be the observables, that are (generally higher-dimensional) functions of the latent variables. Koopman theory states that one can find a linear operator K, termed Koopman operator, acting on observables as

$$(Kg)(\mathbf{z}_t) = g(F(\mathbf{z}_t)) = g(\mathbf{z}_{t+1})$$

Even if $F(\mathbf{z}_t)$ is non-linear, the Koopman operator is linear and hence we can get a linear evolution in the space of observables. In this framework, dynamical systems can be systematically studied with linear techniques and, in particular, are amenable to spectral analysis [13].
In principle, the Koopman operator lives in the infinite dimensional functional space of observables, [14] [15] and it cannot be written as a finite matrix. However, we can introduce a finite-dimensional approximation where we choose a finite set of basis functions [13], projecting the true Koopman operator into a finite subspace. Approaches like these are described in [16] [14].

The reservoir state $\mathbf{r}_t$ evolves linearly as

$$\mathbf{r}_{t+1} = W\mathbf{r}_t + W_{in}\boldsymbol{\omega}_t$$

Since $\boldsymbol{\omega}(t) \approx W^{out}\mathbf{r}(t)$, the reservoir state effectively realizes a high-dimensional projection of the dynamics. Thus, the reservoir space acts as the high-dimensional

space where the Koopman operator, or an approximation of it, can act linearly. In other words, the reservoir implicitly constructs a high-dimensional observable space that can approximate Koopman-invariant subspaces. [13].

2.2 Implementation of the RC model

2.2.a Reservoir

Reservoir Structure

The reservoir is implemented as a ring reservoir where each neuron is connected exactly to one neighboring neuron:

$$W = \begin{bmatrix} 0 & 0 & \dots & 1 \\ 1 & 0 & \dots & 0 \\ 0 & 1 & \dots & 0 \\ \vdots & & & \end{bmatrix}$$

So $(Wr)_j = r_{j-1}$. This structure was shown to be very efficient. The ring matrix ensures a peculiar eigenvalue spectrum. Let us consider the eigenvalue equation

$$Wq = \lambda q$$

Since $(Wq)_j = q_{j-1}$, the eigenvalue equation becomes

$$q_{j-1} = \lambda q_j$$

This equation can be solved recurrently, obtaining

$$q_j = \lambda^{-j} q_0$$

Due to the circular boundary condition, $q_N = q_0$ and applying the above equation we get $\lambda^N = 1$. The solutions are the N-th roots of unity:

$$\lambda_k = e^{2\pi i k/N}$$

The corresponding eigenvectors are known analytically and are pure Fourier modes, which makes the system very interpretable and close to Koopman-style evolution. In the reservoir computer we scale the ring as $W \to \rho W$, where $\rho < 1$ to ensure the spectral radius is less than one, so then the eigenvalues become:

$$\lambda_k = \rho e^{2\pi i k/N}$$

Network parameters

The network parameters used are summarized in Table 2.1, and detailed in the following paragraphs.

Parameter	Value	Description
N	500	Number of reservoir units
ρ	0.5	Spectral radius of recurrent matrix W
τ	1.0 (fMRI), 0.1 (toy system)	Reservoir time constant
g^{in}	10^{-4} (fMRI), $g^{in} = \frac{1}{\sqrt{M}}$ (toy system)	Standard deviation of input weight matrix W^{in}
σ_{noise}	10^{-6}	Standard deviation of thermal noise (stochastic case)
β	10^{-9}	Ridge regression regularization parameter
Δt	$\frac{samplinginterval}{100} = 0.01$	Integration time step
Transient duration	20 s	Washout period before training
Training duration	271 s	Time window used to train readout weights
Prediction horizon	29 s	Time interval used for forecasting
Reservoir topology	Ring	Deterministic cyclic connectivity structure
Activation function $\Phi(h)$	$\Phi(h) = h$	Linear reservoir units

Table 2.1: Hyperparameters used for the linear reservoir computing model following the setup described in Di Antonio et al. (2024).

Internal weight matrix

The reservoir size is 500 units, which is chosen as it is sufficiently large reservoir. For the toy model, which has dimension 15, the reservoir's dimension is much larger so it can safely represent the necessary variables.

Input weight matrix

While the reservoir is not random (as it is a ring matrix), randomness comes in through the input matrix, ensuring that the projection of the input into the reservoir space is random. The elements of the input matrix are random Gaussian:

$$W_{ij}^{in} \sim \mathcal{N}(0, g^{in})$$

W_{in} ensures that each neuron receives a random mixture of observed signals

$$h_j = \sum_{k=1}^{M} W_{in}^{jk}\, \omega_k$$

For the toy model, g_{in} is the inverse of the square root of the number of inputs.

$$g_{in} = \frac{1}{\sqrt{M}}$$

For the toy model, we set $g_{in} = 0.26$. This keeps the variance of the injected signal constant when the number of input changes. if W_{in} were $O(1)$, then the variance would scale as $Var(h_j) \sim M$. Thus, dividing by $\sqrt{M}$ would keep the variance at 1. For the analysis of fMRI data $g_{in} = 1$, with a dimension of $W_{in} \in \mathbb{R}^{N \times M}$ where M is the number of principal components of the BOLD signal that are retained for analysis.

2.2.b Simulation, training, and validation

Numerical Integration

In general, the reservoir needs to be integrated with a timestep equal to one-hundredth of the sampling interval. We update the states according to a discrete time leaky Euler-style update

$$\mathbf{r}_{t+\Delta t} = (1 - \frac{\Delta t}{\tau})\mathbf{r}_t + \frac{\Delta t}{\tau}(W\mathbf{r}_t + W_{in}\boldsymbol{\omega_t}) \tag{2.5}$$

The homogeneous solution to this behaves like $\mathbf{r}(t) \sim e^{-t/\tau}$ due to the presence of the leaky term $-\mathbf{r}(t)$. Thus, the memory of the initial state decays exponentially with τ: in particular, a small τ implies that the reservoir dynamics will have a short memory. The leakage time τ should be tuned to align with the typical time scale of the system. The magnitude of the discrete update is roughly proportional to $\alpha = \frac{\Delta t}{\tau}$. A small α can help make the system more stable.

Initially, we arbitrarily set $\mathbf{r}_0 = 0$. During training, we discard the initial transient to ensure that any memory of the initial state has vanished. As Δt is the reservoir integration time step, the number of transient steps for any system is calculated as $\frac{k\tau}{\Delta t}$ where k can be the time for the slowest decay rate. In fact, a time around 5-10 time constants is typically sufficient to eliminate the effect of the zero initialization.

Training - open loop

We build the matrix $R \in \mathbb{R}^{N \times T}$ where T is the number of time steps (excluding the transient phase). N is the size of the reservoir. At each time step, we take the input $\boldsymbol{\omega}(t)$ and update the reservoir state using equation 2.2.b.

The reservoir states are stored in the matrix

$$R[:, i] = \mathbf{r}(t)$$

where each column is a nonlinear embedding of past inputs.

The target is the trajectory after the transient steps, i.e:

$$Y \in \mathbb{R}^{M \times T}$$

where M is the system dimension, and T is the number of time steps after the transient phase. Using ridge regression, we learn the coefficients of W_{out} such that it fits to the ground truth trajectory:

$$W_{out}\mathbf{r}(t) \approx \boldsymbol{\omega}(t)$$

The ridge regression step is implemented using the scikit-learn package.

Stochastic extension of the model

The dynamics of a linear RNN are dissipative. If the real parts of the eigenvalues are negative, then all modes decay exponentially. In the autonomous mode, in the absence of any external input, the reservoir's dynamics system dies out. To maintain the system in a non-zero activity state, one can introduce a noise term $\eta(t)$, modifying Eq. (2.1.c) as

$$\tau \frac{d\mathbf{r}(t)}{dt} = \tilde{W}\mathbf{r}(t) - \mathbf{r}(t) + \boldsymbol{\eta} \tag{2.6}$$

The noise addition turns the equation into a stochastic differential one. The noise is an Ornstein-Uhlenbeck process with a small (in the limit, vanishing) correlation time, hence essentially uncorrelated white noise:

$$\mathbb{E}[\boldsymbol{\eta}(t)\boldsymbol{\eta}(t')] = \sigma^2 \, \mathbb{I} \, \delta(t - t')$$

When we solve the SDE we get:

$$\mathbf{r}(t) = \int_{-\infty}^{t} e^{A(t-s)}\boldsymbol{\eta}(s)ds$$

Comparing this to Eq. (2.1) we see how the deterministic input data is replaced by stochastic noise. Since $\boldsymbol{\omega} = W^{out}\mathbf{r}$,

$$\boldsymbol{\omega}(t) = W^{out} \int e^{A(t-s)} \boldsymbol{\eta}(s) ds$$

The reservoir's output is now noise filtered by the learned dynamics of W_{out}. With this additional noise, the system becomes a stationary stochastic process. To simulate the system in the presence of noise, we use the Euler-Maruyama scheme.
The Ornstein-Uhlenbeck process is implemented by sampling $\boldsymbol{\eta}(t) \sim \mathcal{N}(0, \sigma^2)$ where $\boldsymbol{\eta}(t)dt$ is a random increment over time dt. If we divide Eq. (2.2.b) by τ, we obtain:

$$\dot{\mathbf{r}} = \frac{-\mathbf{r} + \tilde{W}\mathbf{r}}{\tau} + \frac{1}{\tau}\boldsymbol{\eta}(t)$$

As commonly done for stochastic differential equations, we can introduce the Wiener increment $dW_t \sim \mathcal{N}(0, dt)$ which corresponds to a Gaussian increment with variance proportional to dt (in the limit of small dt). The noise term can be expressed as:

$$\frac{1}{\tau}\eta(t)dt \Rightarrow \frac{\sigma}{\tau} dW_t$$

So, the equation becomes:

$$d\mathbf{r} = \frac{-\mathbf{r} + \tilde{W}\mathbf{r}}{\tau} + \frac{\sigma}{\tau} d\mathbf{W}_t$$

with $d\mathbf{W}_t = \sqrt{dt}\boldsymbol{\xi}, \quad \boldsymbol{\xi} \sim \mathcal{N}(0, \mathbb{I})$. We use the Euler-Maruyama discretization for Solving stochastic differential equations. The general equation is:

$$\mathbf{r}_{t+dt} = \mathbf{r}_t + f(\mathbf{r}_t)dt + g \cdot \sqrt{dt}\, \boldsymbol{\xi}$$

Finally, the discrete update that we implement in our code is:

$$\mathbf{r}_{t+dt} = \mathbf{r}_t + \frac{dt}{\tau}(-\mathbf{r}_t + \tilde{W}\mathbf{r}_t) + \frac{\sigma}{\tau}\sqrt{dt}\boldsymbol{\xi} \tag{2.7}$$

The variance of noise per step is now $\frac{\sigma^2}{\tau^2}dt$. This ensures that as $dt \to 0$ the noise goes to zero and approximates a continuous system.
With the latent states generated using this scheme, we can now generate the observable $\hat{\boldsymbol{\omega}}_t = W^{out}\mathbf{r}_t$.

2.2.c Prediction

The input data are divided in a training set and a testing/prediction set. The training procedure as described in section 2.2.b, where we drive the reservoir

with the real signal $\boldsymbol{\omega}(t)$, collect states $\mathbf{r}(t)$ and solve for W^{out} by ridge regression, is applied on the training set. Then, the prediction set is used for forecasting. Once W^{out} is fixed, we stop feeding external data as input to the system. The network now evolves in "autonomous mode", receiving as input its own output along with endogenous noise, following the update step described in Eq. (2.2.b). The output is still generated using the same readout map. The last reservoir state obtained during the training phase is used as the initial state for autonomous simulation. In this way, the reservoir state contains the embedded recent history of the signal.

Short-horizon forecast

The model's predictive performance is assessed in two ways. On a short time horizon, we test the model's ability to exactly forecast the upcoming values of the time series.
The model is allowed to run autonomously for a short time window, and we compute two metrics: the correlation between the generated output and experimental data, and the root-mean-square (RMSE) deviation. The correlation checks if the model recreates the temporal pattern correctly, while the RMSE checks whether the signal amplitudes are also well aligned. For each component i in the test interval:

$$\mathrm{corr}_i = \mathrm{corr}(\hat{\omega}_i, \omega_i), \qquad \mathrm{RMSE}_i = \sqrt{\frac{1}{T}\sum_t (\hat{\omega}_i(t) - \omega_i(t))^2}$$

An average performance metric is obtained by averaging over all components.

Long-horizon generative validation

Despite its ability for short-term forecast, the model cannot predict the input time series arbirarily far in the future.
However, in a second validation step we run the reservoir for an time equivalent to the training time to test whether the stochastic autonomous IRNN can reproduce the statistics of brain activity (rather than the exact time series). We assess a degree of statistical fidelity by comparing the Functional Connectivity (correlation matrix) across voxel pairs and Dynamical Functional connectivity (time-evolving FC structure). The details of Functional Connectivity and Dynamical Functional connectivity are presented in the next section.

Noise Amplitude Optimization

The amplitude of endogenous noise has a strong impact on the model's accuracy and predictive ability. We performed a logarithmic sweep over the endogenous noise amplitude and evaluated three quantities: the maximum real part of the autonomous poles, short-horizon prediction accuracy, and functional connectivity (FC) accuracy.

2.2.d Functional connectivity

Functional connectivity is the Pearson correlation between two BOLD signals over a time window:

$$FC_{AB}(t;T) = \mathrm{corr}[X_A(t;T), X_B(t;T)]$$

Here we compute FC with $t = 0s$ and $T = 300s$. So for the long-horizon validation we take the entire 300s simulated segment, compute the voxel-by-voxel Pearson correlation matrix, do the same with the real 300s segment and compare the two matrices.

2.2.e Dynamical Functional Connectivity

We calculate $FC(t)$, a time-varying connectivity matrix. DFC can capture temporal variability. We use a sliding window of 60s, $[t, t+60]$ and compute the functional connectivity in each window $FC_{ij}(t;T) = \mathrm{cor}[X_A(t;T), X_B(t;T)]$ to get one FC matrix per window. We slide this window to get a sequence of FC matrices. We build the Functional Connectivity Dynamics (FCD):

$$FCD(t1, t2) = \text{Similarity between FC(t1) and FC(t2)}$$

by flattening the upper triangle of FC and computing the correlation.

To compare the two FCD of the real and the generated data we compute the Jensen-Shannon Distance, $\mathrm{JSD}(\mathrm{FCD}_{real}, \mathrm{FCD}_{model})$.

2.3 Data

2.3.a Simulated data

Toy model for Koopman analysis

We will first focus on the simple ODE system:

$$\dot{x} = C_x x$$

$$\dot{y} = C_y y + C_{yx}(x \odot x)$$

where x $\in \mathbb{R}^5$, $y \in \mathbb{R}^{10}$. C_x has to be diagonal. We generate this system using RK4.

This system was chosen specifically because it is Koopman-linearizable. A non-linear dynamical system is koopman-linearizable if the Koopman operator is finite-dimensional.
The only non-linear term here is $x \odot x$, so we define a new variable:

$$z = x \odot x$$

This gives us 5 new variables for as there are 5 x variables. To compute the dynamics of z we take the derivative:

$$\dot{z} = 2C_x z$$

So now the new vector is:

$$g = \begin{bmatrix} x \\ z \\ y \end{bmatrix}$$

Which has $5 + 5 + 10 = 20$ dimensions. The matrix:

$$\frac{d}{dt}\begin{bmatrix} x \\ z \\ y \end{bmatrix} = \begin{bmatrix} C_x & 0 & 0 \\ 0 & 2C_x & 0 \\ C_{yx} & 0 & C_y \end{bmatrix}\begin{bmatrix} x \\ z \\ y \end{bmatrix}$$

is completely linear. Therefore this system has a finite-dimensional Koopman

representation. The Koopman generator here is the matrix:

$$K = \begin{bmatrix} C_x & 0 & 0 \\ 0 & 2C_x & 0 \\ C_{yx} & 0 & C_y \end{bmatrix}$$

Hence for this Non-linear toy model we already know the true Koopman operator. The true Koopman eigenvalues are $C_x, 2C_x, \text{eig}(C_y)$. We want to see if the reservoir learns the same eigenvalues by comparing the spectra. Once the reservoir is trained and the output is $\boldsymbol{\omega} = W_{out}\mathbf{r}$, we can compute the eigenvalues and show whether the reservoir acts like a Koopman operator estimator.

To simulate the ODE system, we integrate the equations usinf the Runge-Kutta method. The latter estimates the state at time $t + dt$ by evaluating the derivative function at multiple intermediate points within the integration interval. Given a state $\mathbf{z}(t)$, the RK4 scheme computes four intermediate slopes:

$$k_1 = f(\mathbf{z}_t)$$

$$k_2 = f\left(\mathbf{z}_t + \frac{dt_{gen}}{2}k_1\right)$$

$$k_3 = f\left(\mathbf{z}_t + \frac{dt_{gen}}{2}k_2\right)$$

$$k_4 = f(\mathbf{z}_t + dt, k_3)$$

The updated state is then computed as

$$\mathbf{z}_{t+1} = \mathbf{z}_t + \frac{dt_{gen}}{6}\left(k_1 + 2k_2 + 2k_3 + k_4\right)$$

This formulation provides fourth-order accuracy in the time step dt_{gen}. dt_{gen} is the time step of integration of the system. This is not responsible for any iteration in the reservoir dyamics and its purely for data generation. Determines how coarse or fine our generated data is - used in Toy model and Chaotic.

the matrices C_x and C_y were chosen such that their eigenvalues have small real parts, mostly negative. The non-linear coupling matrix C_{yx} was also scaled to prevent uncontroled growth in the quadratic term.

Table 2.2: Parameters used for the toy dynamical system and reservoir computer experiments.

Parameter	Value
System variables	$x \in \mathbb{R}^5,\ y \in \mathbb{R}^{10}$
C_x	$\mathrm{diag}(-0.08, -0.06, -0.05, -0.04, -0.03)$
C_y	$\mathrm{diag}(-0.09, -0.08, -0.07, -0.06, -0.05, -0.04, -0.035, -0.03, -0.025, -0.02)$
C_{yx}	$0.02\,\mathbf{1}_{10\times 5}$
Integration method	Fourth-order Runge–Kutta (RK4)
Simulation time step	$dt_{gen} = 0.01$
Number of simulation steps	8×10^3
Training split	70%
Reservoir dimension	$N_r = 500$
Input dimension	15
Spectral radius	$\rho = 0.5$
Input scaling	$\sigma = 0.26$
Reservoir time constant	$\tau = 0.1$
Reservoir integration step	$dt = 10^{-4}$
Washout step	2

Non-linear chaotic Systems

As a simple example of chaotic system, we will consider the classical Lorentz system,

$$\dot{x} = \sigma(y - x)$$
$$\dot{y} = x(\rho - z) - y$$
$$\dot{z} = xy - \beta z$$

The Lorenz system is strongly nonlinear. Applying the linear RC method, we expect tiny errors to accumulate, due to its poor ability to model chaos.

2.3.b Real data from fMRI

We analyzed real time series from the work of Kim et al. [17], in which functional magnetic resonance imaging (fMRI) was used to record the brain activity of 16 participants, each of whom underwent several (27) recording runs. In some (20/27) runs, participants were presented with a sequence of visual stimuli divided into different categories (faces, scenes, objects, etc.); in the remaining runs, they were at rest (i.e., in the absence of stimuli). The authors identified a set of 32 areas (mainly occipital and parietal) that exhibited a marked response to specific categories of visual stimuli.

For all details on the experiment as well as data preprocessing, see Kim et al., 2022. The data analyzed here refer to a single brain area during a single resting-state run (300 s of recording) from a single subject. The temporal resolution of the recording was 1 s, and the analyzed area includes a total of 1108 cortical vertices: we therefore have a set of 1108 time series, each consisting of 300 data points.

2.3.c Dimensionality reduction of BOLD signals

fMRI data is usually spatially redundant. Within one cortical area, many of the voxels are not independent measurements are they are often measuring the same underlying population-level processes. They are also blurred by the hemodynamic response and various preprocessing steps that involve smoothing. BOLD is not the underlying phenomena but rather a proxy, so it contains various noise and small voxel-level fluctuations that may not reflect meaningful neural dynamics.
We use Principal component analysis to reduce dimensionality, as many voxel signals are redundant and some of the observed variability can be noise. This way we can compress the signal before feeding into the reservoir. The reservoir will see a denoised representation and the input dimension is reduced making the learning task easier. After applying PCA , for each sessions s we get the eigenvalues

$$\lambda_1^{(s)} \geq \lambda_2^{(s)} \geq \cdots \geq \lambda_{1108}^{(s)} \geq 0$$

if we define

$$r_i^{(s)} = \frac{\lambda_i^{(s)}}{\sum_{j=1}^{m_s} \lambda_j^{(s)}}, \qquad i = 1, \ldots, m_s$$

then r_i^s is the fraction of the total variance captured by the i-th principal component in session s. We can calculate the cumulative explained variance curve for one session as:

$$c_s(k) = \sum_{i=1}^{k} r_i^{(s)}, \qquad k = 1, \ldots, m_s$$

We take the average cumultive explained variance across the sessions. We compute the standard deviation across the sessions to plot the band as $c_{mean}(k) \pm \sigma_c(k)$. Selecting for principal components that explain 99.5% variability, we are left with 45 principal components. This is much larger than the 4 principal components of an individual cortical area of the data obtained in the paper. Again in the original paper, they use 11 cortical areas leading to $4 \times 11 = 44$ principal components. While our 45 components is comparable to this, their 44 is composed of top 4 components of each section leading to less variable data overall.

We split the first 280s as training data and the last 20 seconds as the test time. Fit PCA to the training data and get the main directions of ω_{train}. Project test data onto first k principal coefficients and reconstruct the signal of the last 20 seconds

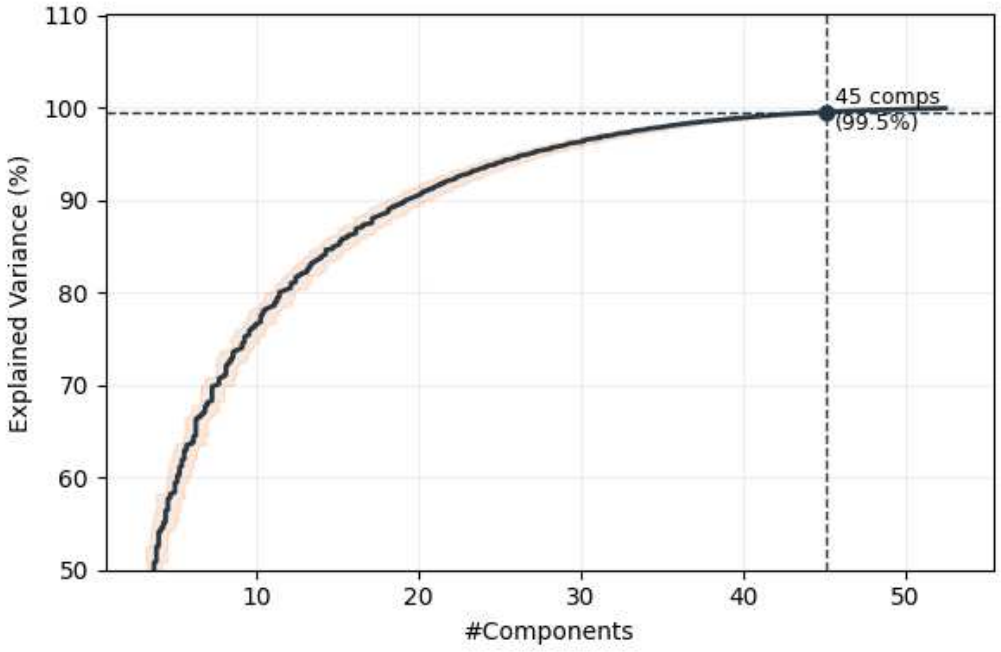


Figure 2.1: The cumulative explained variance as a function of the number of principal components, averaged across 7 sessions. The solid curve represents the mean cumulative explained variance $c_s(k)$, while the shaded region is one standard deviation across the sessions. The plot marks the number of components required to reach 99.5% variance.

$\hat{\omega}^{(k)}$. Compute the residual

$$\Delta\omega = \omega_{test} - \hat{\omega}^{(k)}$$

and see how corrrelated that residual is with the original signal on the unseen last 20s. For each session s, we compute using Pearson Correlation:

$$y^{(s)}(k) = 1 - \mathrm{corr}(\Delta\omega_k^{(s)}, \omega^{(s)})$$

In contrast to the findings reported in the original study, the residual components beyond the 99.5% explained variance threshold do not appear to be purely noise. At approximately 45 principal components, we observe that $1-\mathrm{corr} \approx 0.5$, indicating that the residual signal remains substantially correlated with the original data. This suggests that the discarded components still contain structured information rather than representing purely noise. Consequently, PCA in this setting does not achieve a clear separation between signal and noise.

The residual correlation increases gradually with the number of retained com-

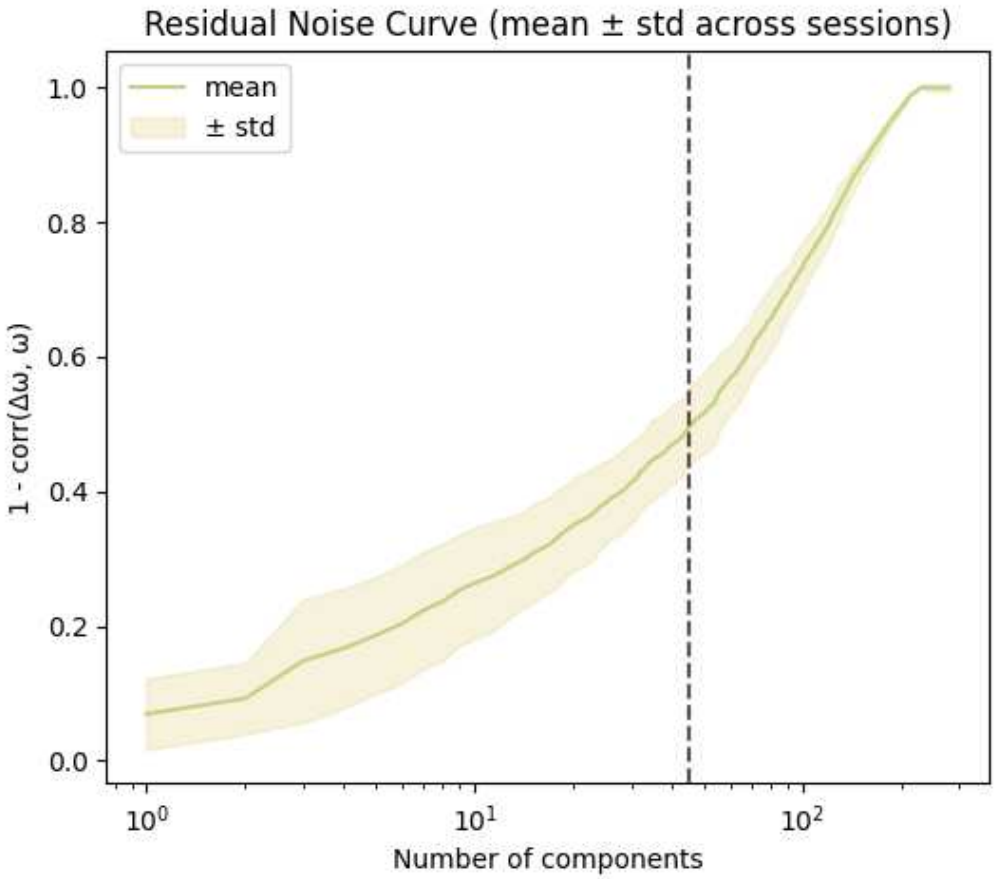


Figure 2.2: The mean residual noise across sessions as a function of the retained explained variance in the PCA reconstruction, The shaded band indicates one standard deviation.

ponents and approaches unity only at much higher dimensionalities. This behavior indicates that the BOLD signal in this dataset is distributed across a higher-dimensional subspace, rather than being concentrated in a small number of dominant modes. As a result, the assumption that the signal can be effectively captured in a low-dimensional representation with noise-dominated residuals does not hold as strongly in this case. From a modeling perspective, this has implications for downstream performance. While increasing the number of components retains more signal, it also reintroduces noise, leading to a trade-off. Empirically, retaining 99.5% of the explained variance provided a better balance for reservoir computing, yielding lower average Jensen–Shannon divergence (JSD) across sessions compared to a more aggressive threshold of 99.9

3 Results

In this Chapter, we report on the application of the linear RNN model described in Sec. 2.1, inspired by reservoir computing, to simulated and real data. Upon discussing a toy model and a simple chaotic system (Sec. 2.3.a), we will focus on the model's ability to capture spatiotemporal dependencies in brain activity time series from fMRI (Sec. 2.3.b).

3.1 Toy system

We begin this chapter by discussing the toy model described in Sec. 2.1. The model involves a 15-dimensional, nonlinear dynamical system. The peculiarity of the system is that it can easily be mapped into a 20-dimensional, linear dynamical system by considering an extended observable space that includes the 15 original variables, and 5 additional observables coinciding with the squares of the first 5 variables. Therefore, the system admits a Koopman operator with a discrete, finite spectrum characterized by 20 non-zero eigenvalues. As the reservoir is trained to learn linear dynamics able to reproduce the input time series, the expectation is that the dynamics of the trained reservoir will approximately match the linear dynamics predicted by the Koopman operator. In particular, the dominant eigenmodes of the dynamics should match the system's Koopan modes.

Accuracy.

We simulated trajectories over a time horizon of $T = 90$ time units, using a fixed integration step of $\Delta t = 10^{-3}$ resulting in 9×10^4 discrete time steps. Trajectories

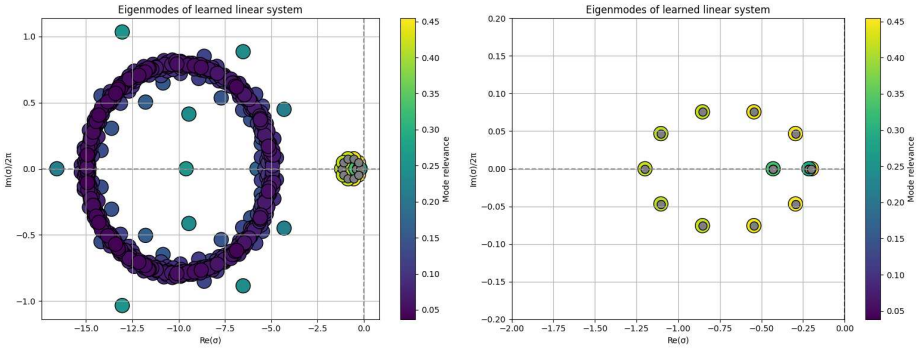


Figure 3.1: (a) Learned eigenvalues of the closed-loop linear system. The color indicates the mode relevance, quantified by the contribution of each mode to the reconstructed observable. A mall subset of highly relevant modes forms a cluster near the 0 on the real axis, while a large number of weakly relevant modes form a broad ring corresponding to intrinsic reservoir dynamics. **(b)** The gray markers denote the theoretical eigenvalues of the underlying toy system, which align with the high relevance learned modes.

were divided into a training and a testing part using a 70-30 split. In the training set, the RC model reached perfect accuracy ($R^2 = 1.0$, $MSE = 2.5 \times 10^{-27}$). We also used the validation set as a prediction set, by initializing the network in the final state of the training set, and using the RC model in the "autonomous" mode, whereby the reservoir's output is fed back into the reservoir as input, in a closed-loop configuration. In this way, we tested the model's ability in truly predicting the time evolution of the system. In the prediction set, the RC model again reached perfect accuracy ($R^2 = 1.0$, $MSE \sim 10^{-31}$). This indicates that the toy system dynamics are simple and stable enough to be captured almost exactly by the learned linear readout.

Mode relevance

Having assessed the model's effectiveness in approximating the toy system's dynamics, we scrutinized the learned linear dynamics more in depth. To this aim, we computed the eigenvalues of the RC model, and compared them with the Koopman spectrum of the system, which can be promptly computed from the linearized version of the toy system. For each mode (eigenvalue), one can compute an index of "relevance" between 0 and 1 weighing the mode's overall contribution to the dynamics (Sec. 2.1.c).

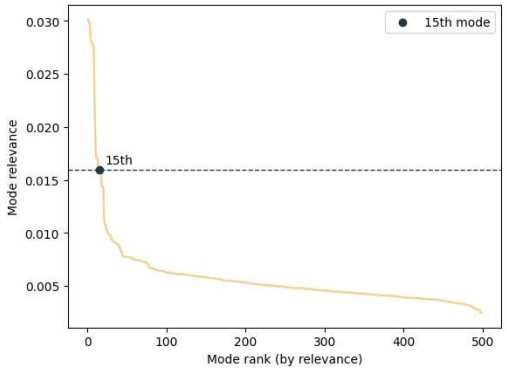


Figure 3.2: Mode relevance ranked over all 500 learned eigenmodes of the closed-loop system. The relevance decays rapidly for the leading modes and then exhibits a long tail of weakly contributing modes, indicating that the learned dynamics are concentrated in a small subset of dominant modes but are not separated by a sharp cutoff. The 15th rank is marked as reference points corresponding to the number of poles of the toy system, respectively

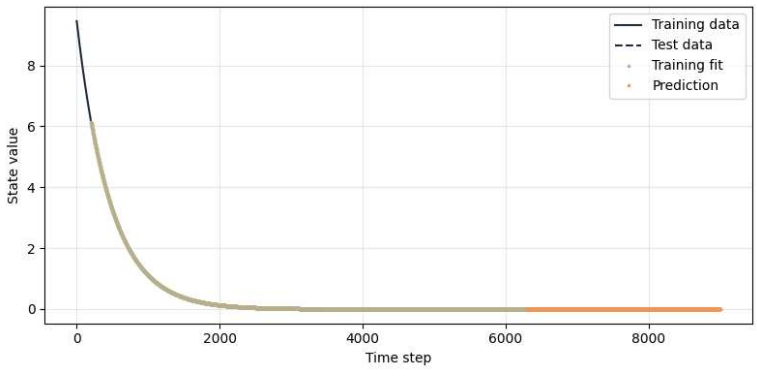


Figure 3.3: Time evolution of a representative state variable of the toy system. Solid lines denote the training data, while dashed lines indicate the ground-truth trajectory in the test region. Markers show the reservoir reconstruction during training (teacher-forced regime) and the autonomous prediction in the test regime. The model accurately reproduces the training dynamics and continues to follow the system trajectory in the test phase. The dynamics exhibit a rapid transient decay followed by convergence toward a stable steady state, which dominates the long-time behavior.

The results are shown in Fig. 3.1a, where the eigenvalues are plotted in the complex plane, with a color informed by the relevance of the corresponding modes. One can immediately notice two structures: a large ring on the left, comprising low-relevance modes, and a smaller cluster on the right, comprising modes with much greater relevance. Thus, only a small subset of the $N = 500$ modes are dynamically important.

The eigenvalues in the large ring have a large negative part, which means that the corresponding modes are strongly damped. These are mostly background reservoir modes that were inherited from the untrained reservoir spectrum (which is perfectly circular, due to the reservoir's ring structure) and are only weakly relevant to the observable dynamics. Eigenvalues in the small cluster have a slightly negative real part (approaching 0 form the left), a small imaginary part and high relevance. Therefore, these are important modes that decay slowly and persist for longer. Their greater relevance means that they dominate the observable reconstruction.
In Fig. 3.1b we offer a closer inspection into the cluster, also displaying the theoretical eigenvalues (gray). It is immediate to notice the alignment of the relevant modes with the theoretical Koopman models of the toy system. Hence, the learned system has recovered with sufficiently high accuracy the effective finite-dimensional linear representation of the underlying nonlinear dynamics.

3.2 Lorenz and Rossler systems

Next, we trained the RC model on two paradigmatic chaotic systems, the Lorenz and Rossler systems (Sec. 2.3.a). Contrary to the previous case, here we expect the RC model to struggle significantly to reproduce the input time series. Chaotic systems correspond to an infinite-dimensional, continuous-spectrum Koopman operator - thus, they cannot be easily approximated by a finite dimensional, discrete spectrum linear system.
We simulated trajectories of length $T = 10^5$ of the Lorenz system. Trajectories were divided into a training and a testing part using a 90-10 split. In the training set, the RC model reached near-perfect accuracy ($R^2 = 1$, $MSE = 9.4 \times 10^{-9}$). However, when using the model in the "autonomous" mode, the performance was dismal ($R^2 = -2.48$, $MSE = 256.37$).
Prediction failure is shown in Fig. 3.5a-c. The predicted trajectory does not follow either lobe of the Lorenz attractor, instead collapsing towards a flat, spiral-like orbit.
In Fig. 3.5d we show the spectrum of the trained reservoir. Unlike the toy model, the learned representation does not yield an extremely sparse spectrum. The dominant dynamics are carried by a band of slowly damped oscillatory modes. This is consistent with the behavior of the autonomous rollout, which converges to a simplified spiral-like structure. Moreover, the Lorenz spectrum includes modes with high relevance but large negative real part, which can introduce stronger damping directions in the learned linear dynamics.

We replicated the same analysis for the Rossler system. We simulated trajectories of length $T = 2000$ steps. In the training set, the RC model reached near-perfect accuracy ($R^2 = 1$, $MSE = 7.87 \times 10^{-5}$), degrading in the testing set ($R^2 = 0.046$, $MSE = 17.82$).

The model's output in the validation phase is shown in Fig. 3.6a. The model is not able to capture the nonlinear geometric structure of the attractor, but instead collapses to a simple spiral like orbit or ring - possibly, the best attempt of a linear system to learn the underlying chaotic dynamics. In Fig. 3.6b we show the spectrum of the trained reservoir. Again, the spectrum indicated the presence of weakly damped rotational/oscillatory components.

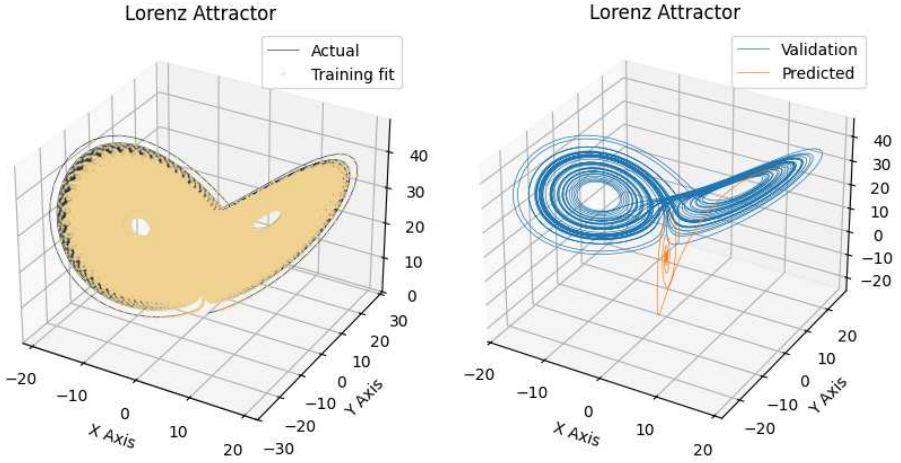


Figure 3.4: Three-dimensional trajectories of the Lorenz system. Left: Training regime (teacher-forced), showing the ground-truth trajectory (line) and the reservoir reconstruction (markers), which closely overlap and indicate near-perfect fitting of the training data. Right: Test regime (autonomous prediction), showing the ground-truth trajectory (blue) and the model prediction (orange).

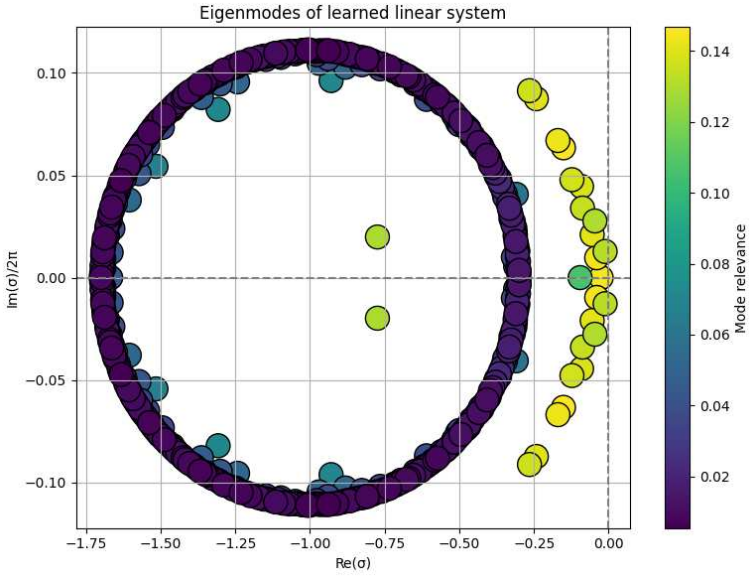


Figure 3.5: Eigenvalue spectrum of the learned closed-loop linear operator for the Lorenz system. Each point represents a learned mode, with color indicating its relevance to the reconstructed observable. A large number of weakly relevant modes form a circular structure corresponding to intrinsic reservoir dynamics, while a subset of highly relevant modes clusters near the real axis.

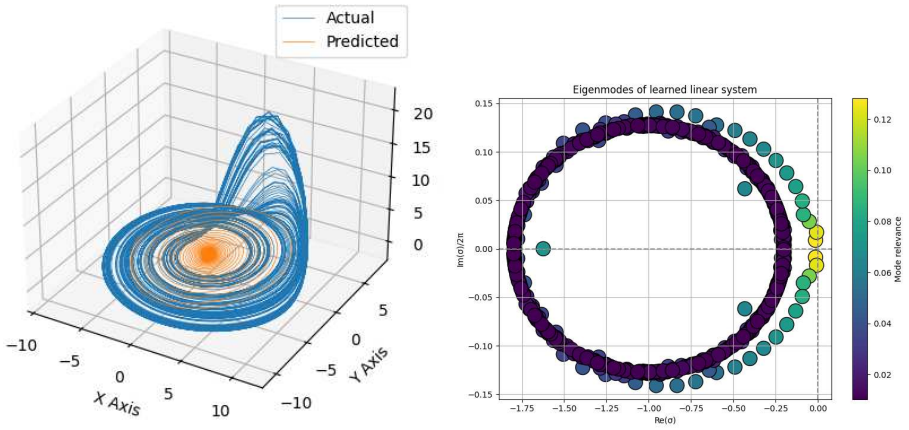


Figure 3.6: Reconstruction and spectral analysis of the Rössler system using the reservoir model. Left: Three-dimensional trajectories showing the ground-truth dynamics (blue) and autonomous prediction (orange). The predicted trajectory captures the overall structure of the attractor but deviates over time due to the chaotic nature of the system. Right: Eigenvalue spectrum of the learned closed-loop linear operator, with color indicating mode relevance.

3.3 Neural time series from fMRI

Having thoroughly analyzed artificial data, we turn our attention to real data from fMRI. Our data consist in BOLD signals of ~ 1000 voxels in visually related areas, coming from seven scanning sessions of the same human participant. As explained in 2.3.c, we trained the RC model using principal components of the voxel-level BOLD signals explaining 99.5% of the variance (on average over sessions, 45 ± 0.8 PCs were needed).

We evaluated the model's performance from its ability to reproduce key features of the original data - specifically, static and dynamic functional connectivity measures. Overall, the model achieved an average functional connectivity (FC) correlation of 0.73 ± 0.07 across sessions, indicating a good reconstruction of the overall connectivity structure.
In contrast, the Jensen–Shannon distance (JSD) between the distributions of dynamic functional connectivity (FCD) in the real data and in the simulated data generated by the model is 0.65 ± 0.07, indicating a substantial mismatch in connectivity dynamics. These results suggest that the model successfully captures the static structure of functional connectivity, but it fails to reproduce the temporal variability of connectivity patterns, generating dynamics that are overly homogeneous across time.

The best performance was obtained at a noise level of $\epsilon = 10^{-6}$, which provides a balance between short-term prediction accuracy and long-term dynamical fidelity.

Noise tuning

We systematically evaluated the effect of injected noise on the learned dynamics by varying its amplitude over several orders of magnitude. As shown in Fig. 3.7, increasing noise leads to a transition from weakly unstable to stable dynamics, as indicated by the maximum real part of the eigenvalues becoming negative.
This stabilization is accompanied by an improved reconstruction of the long-term statistical structure, reflected in higher functional connectivity (FC) similarity. However, increasing noise simultaneously degrades the short-horizon prediction accuracy, as measured by the correlation between predicted and ground-truth trajectories. This reveals a clear trade-off between short-term predictive fidelity and long-term generative accuracy.

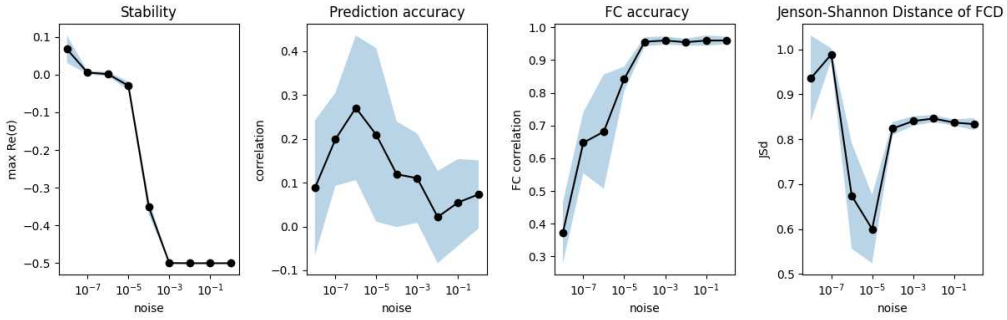


Figure 3.7: Effect of noise amplitude on stability, prediction accuracy, and functional connectivity. Left: Stability of the learned dynamics quantified by the maximum real part of the eigenvalues $\max\mathcal{R}(\sigma)$, where negative values indicate stable dynamics. Middle: Short-horizon prediction accuracy measured by correlation between predicted and ground-truth signals. Right: Functional connectivity (FC) similarity between generated and real data. Increasing noise stabilizes the system (left) and improves FC reconstruction (right), but leads to reduced prediction accuracy (middle), highlighting a trade-off between dynamical stability and short-term predictive fidelity. The last panel highlighs the Jenson-Shannon distance of the Functional connectivity, using windows of 60 seconds at various noise levels. Results are averaged across 7 sessions, with shaded regions indicating variability across the sessions.

Based on this trade-off, we selected a moderate noise level of $\epsilon = 10^{-6}$ subsequent experiments, as it provides a balance between short-term prediction accuracy and long-term fidelity. In addition, a higher noise regime $(\epsilon = 10^{-4})$ was considered to specifically evaluate the model's ability to reproduce long-term dynamics and functional connectivity.

Reconstruction quality

In Fig. 3.8 we show the real and model-reconstructed time series for a single PC. It can be immediately observed that the model achieves a near-perfect fit in the training phases, but its short-term forecasting capability is limited. In Fig. 3.9 we show the real and predicted time series for the dominant components. The model is able to reproduce their general oscillatory structure, indicating that the reservoir captures the underlying dynamics to some extent. However, a consistent phase mismatch between predicted and real signals is observed across several components. The predicted trajectories follow similar frequency patterns, but their peaks are shifted relative to the ground truth. This misalignment leads to reduced correlation values and increased RMSE, despite qualitative similarity in signal structure (Fig. 3.10a-b). This phenomenon is further illustrated in Fig. 3.10c, which shows the temporal evolution of prediction accuracy.

The model achieves high correlation in the short-horizon regime (first few time steps), indicating that it can accurately track the trajectory initially. However, correlation rapidly decays as the prediction horizon increases, eventually approaching zero and even becoming negative at longer times. This behavior reflects the accumulation of phase error over time: small initial misalignments grow progressively, leading to increasing divergence between predicted and true trajectories despite preserving the overall oscillatory structure.There is also variability in performance across components, with the model accurately reconstructing while others exhibit weak or even negative correlation.

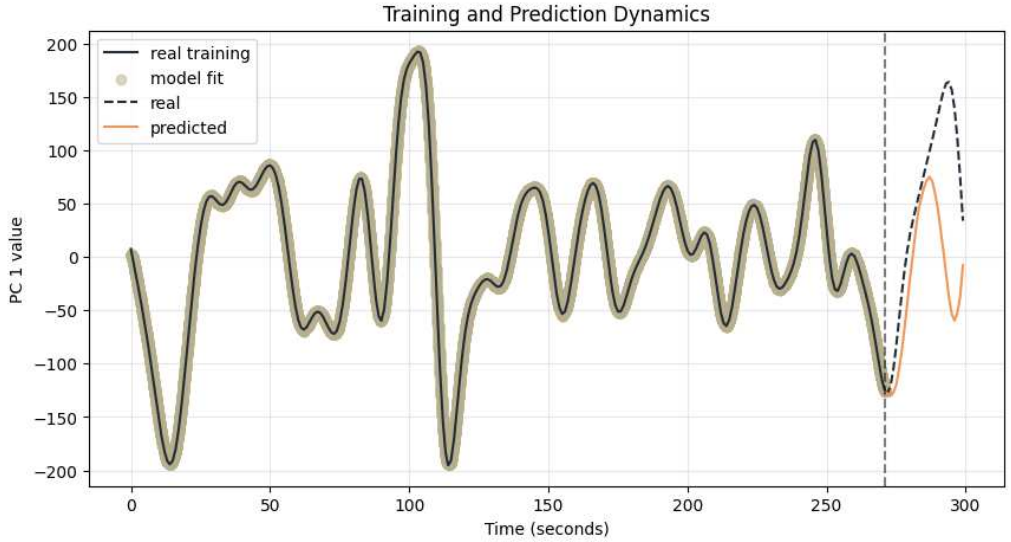
Long horizon result

We generated simulated time series from the model by running it in autonomous mode for an additional $300s$. The reservoir produced stable autonomous dynamics (as per our choice of ϵ). As shown in Fig. 3.12, the model successfully reproduces the overall oscillatory structure of the data. However, the generated trajectories exhibit a gradual increase in amplitude over time, in contrast to the bounded nature of the empirical signals.

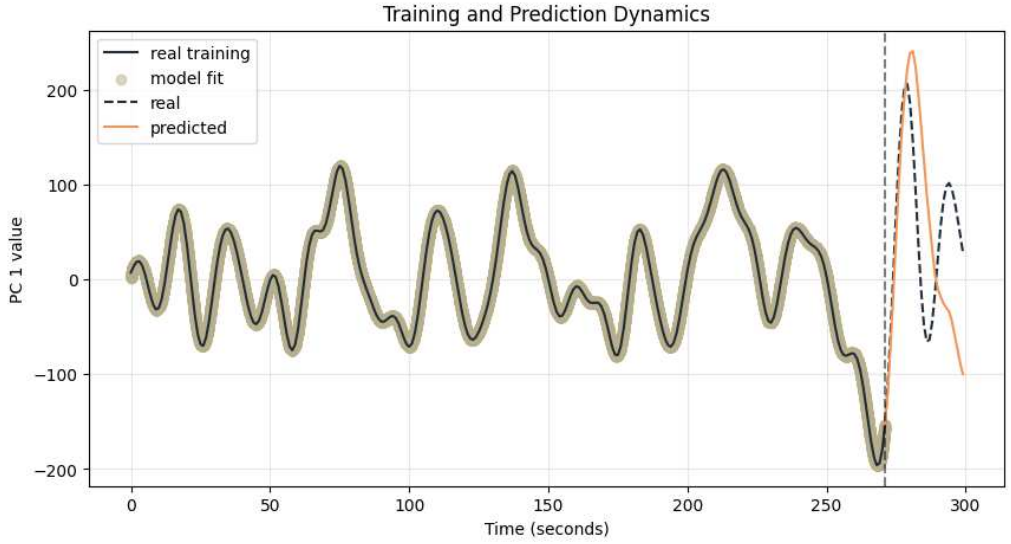
The learned system operates close to the stability boundary, with slightly positive real parts of the eigenvalues leading to slow divergence. The maximum real part of the eigenvalues at the chosen noise level of 10^{-6} is 0.01 ± 0.0056. Although this does not strongly affect short-horizon prediction, it may result in unstable dynamics in the very long term and possibly contribute to discrepancies in statistical measures such as dynamic functional connectivity.

To evaluate the long-term statistical behavior of the model, we compared the functional connectivity (FC) of the generated signals with that of the empirical data. As shown in Fig.3.13, the reservoir is able to reproduce the large-scale correlation structure of the data even when run autonomously for extended time horizons. Despite the divergence observed at the trajectory level, the predicted FC matrices closely resemble the ground-truth connectivity patterns. This is further supported by low Jensen–Shannon divergence (JSD) values, indicating strong agreement between the distributions.

This result suggests that while the model struggles to maintain accurate phase alignment in individual trajectories, it nevertheless captures the underlying statistical dependencies between components.



(a) Session 4



(b) Session 7

Figure 3.8: Training and autonomous prediction dynamics for the first principal component (PC1). The reservoir is trained on the first 271 time points in seconds, resampled at a temporal resolution of $dt = 0.01$. The vertical dashed line marks the transition from teacher-forced training to autonomous prediction. Beyond this point, the predicted trajectory is compared against the ground-truth continuation, showing good short-term agreement followed by gradual divergence.

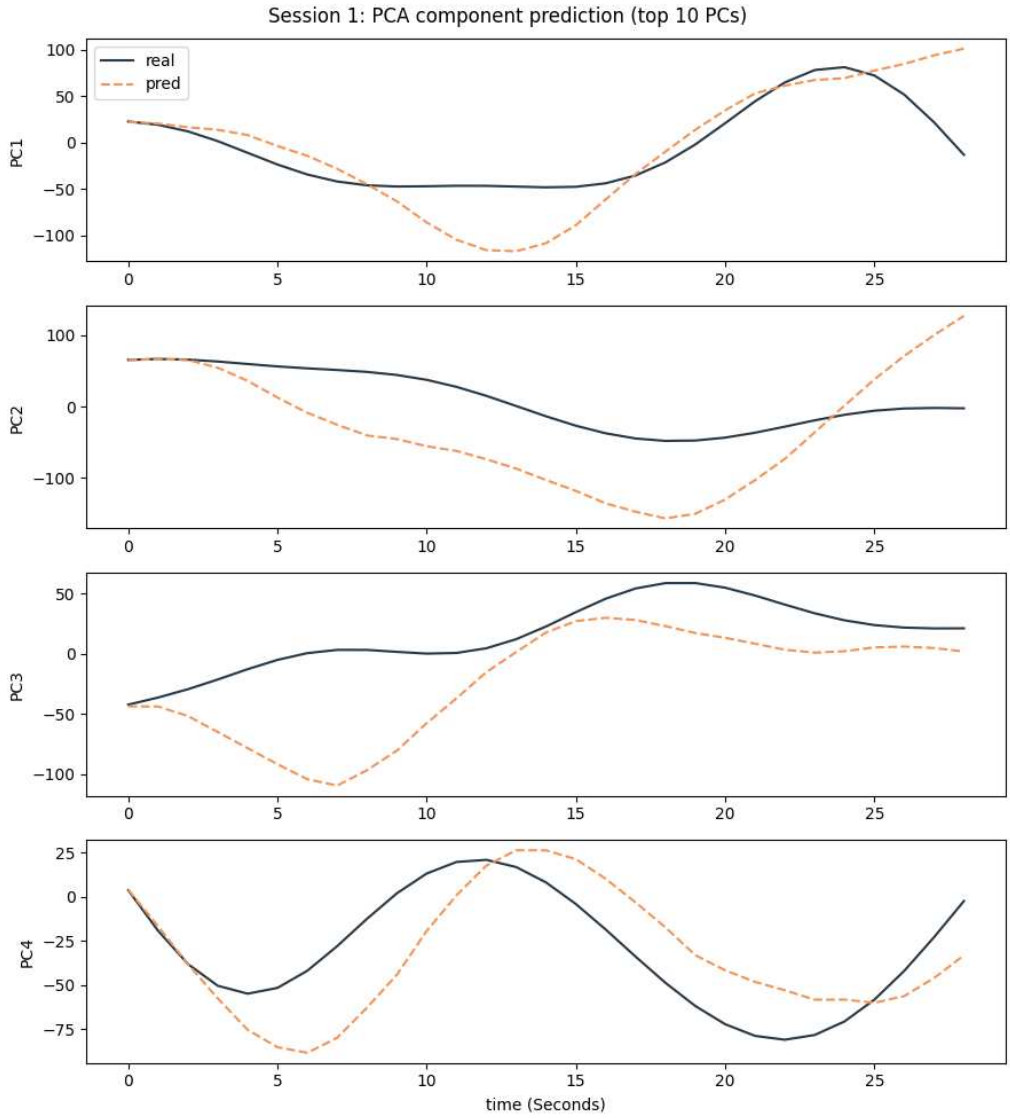
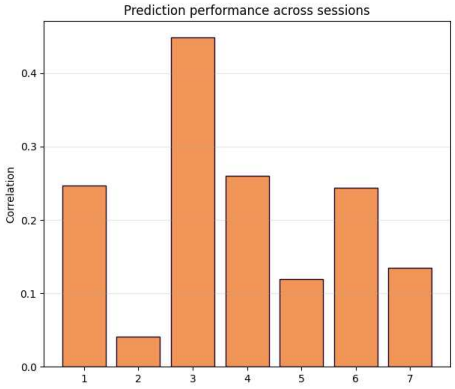
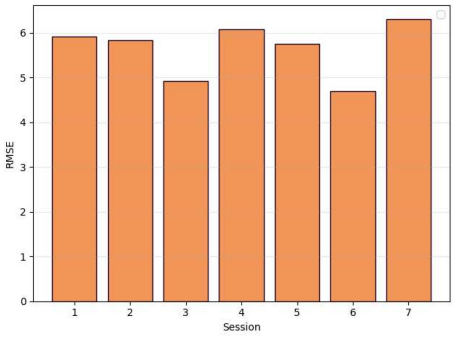


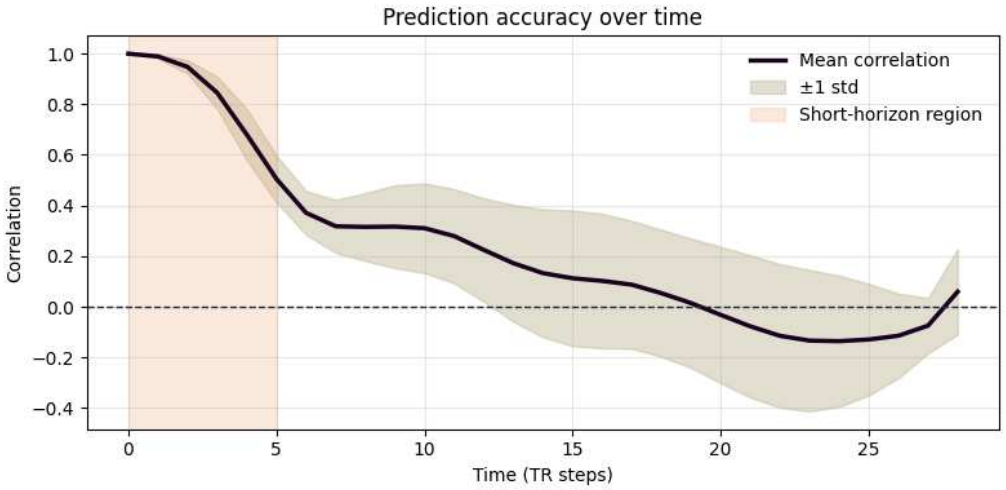
Figure 3.9: The ground-truth trajectories (solid) are compared with model predictions (dashed) over the unseen short horizon of 29 seconds, for the top 4 principal components. The model captures the short-term temporal structure of the dominant components, but deviations increase over time, indicating reduced long-term predictive accuracy.



(a) Prediction correlation across sessions

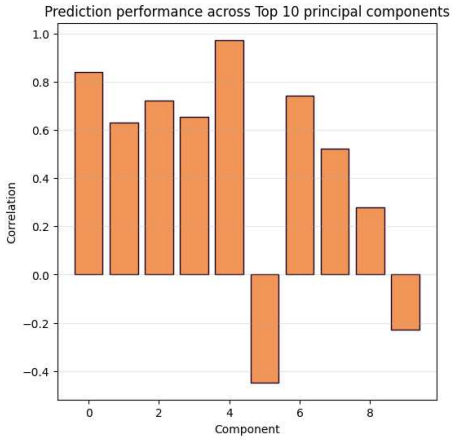


(b) RMSE across sessions

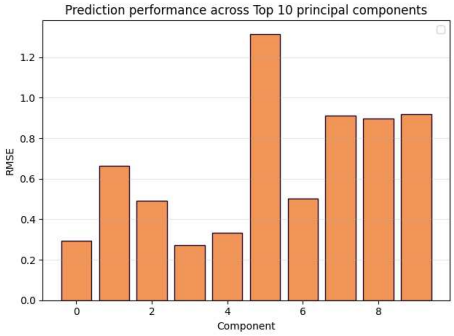


(c) Temporal evolution of prediction correlation (mean $\pm$ std)

Figure 3.10: Evaluation of reservoir prediction performance. (a) Correlation between predicted and true signals across sessions. (b) Root Mean Square Error (RMSE) across sessions. (c) Temporal evolution of prediction accuracy, showing mean correlation and variability over time.



(a) Correlation between predicted and ground-truth signals for the top 10 principal components.



(b) Corresponding RMSE values.

Figure 3.11: The results show strong variability in reconstruction quality across components, with some accurately captured and others exhibiting weak or negative correlation due to phase misalignment.

Dynamical Functional connectivity using Jenson-Shannon Distance

To evaluate the ability of the model to reproduce temporal fluctuations in functional connectivity, we compared the distributions of Functional Connectivity Dynamics (FCD) values obtained from empirical and simulated data. The FCD matrix captures the similarity between functional connectivity patterns computed over sliding time windows, and its distribution reflects the temporal variability of connectivity structure.

As shown in Fig. 3.14, the empirical distribution is concentrated around low-to-moderate correlation values, with a peak near 0.2 and a long tail extending toward higher correlations. However, the model distribution is systematically shifted toward higher correlation values, suggesting that the simulated connectivity patterns are more similar over time compared to the empirical data.

The model matches the overall Functional connectivity very well but fails to reproduce the temporal variability of Functional connectivity. The model's Dynamical Functional Connectivity is concentrated around higher values (0.7 - 0.9) while the data is much lower. Compared to the real signal, the generated one exhibits a reduced FC variability over time.

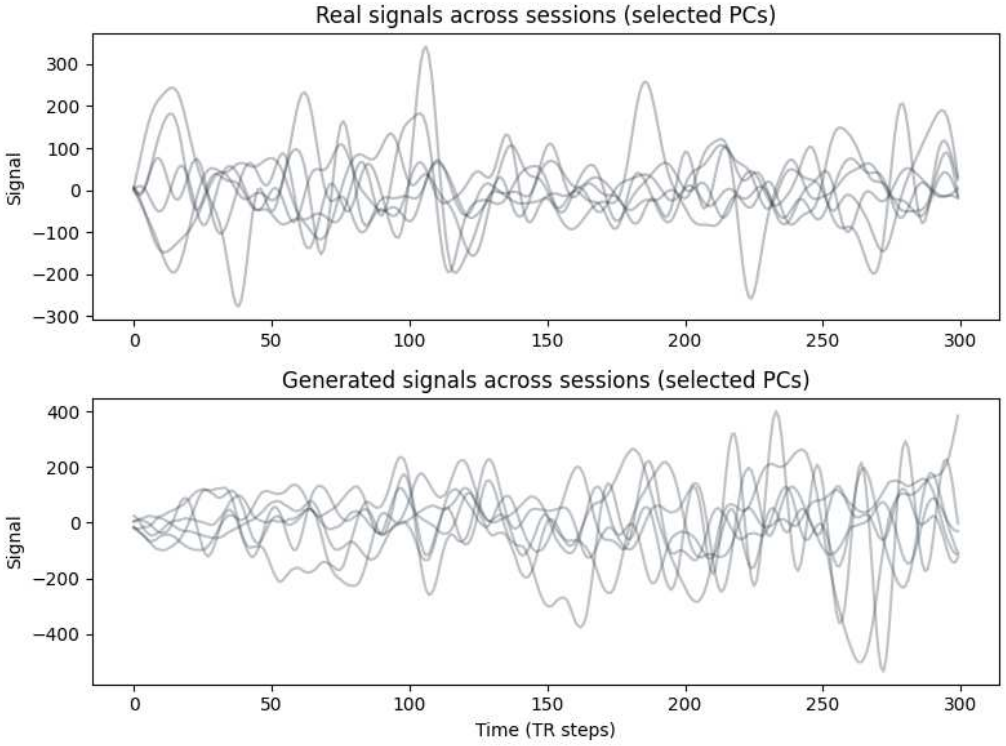


Figure 3.12: Comparison of real and generated signals across sessions for selected principal components. The top panel shows the empirical signals, while the bottom panel shows trajectories generated in autonomous mode after training, with noise amplitude $\epsilon = 10^{-6}$, for an equivalent duration of 300 seconds.

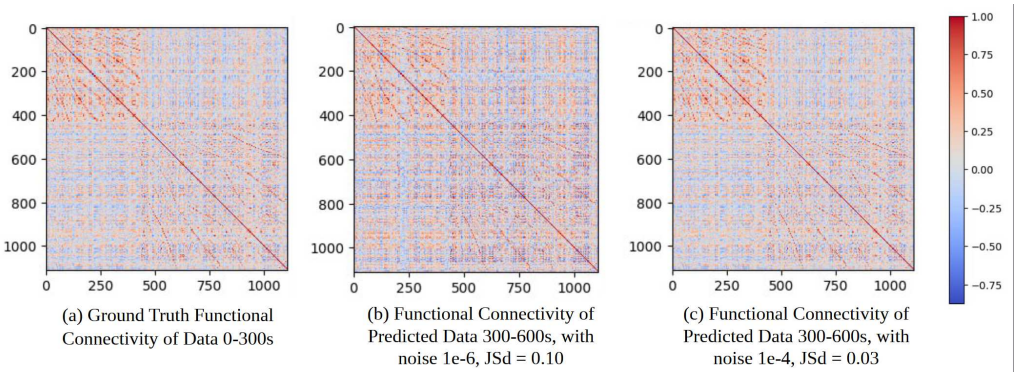
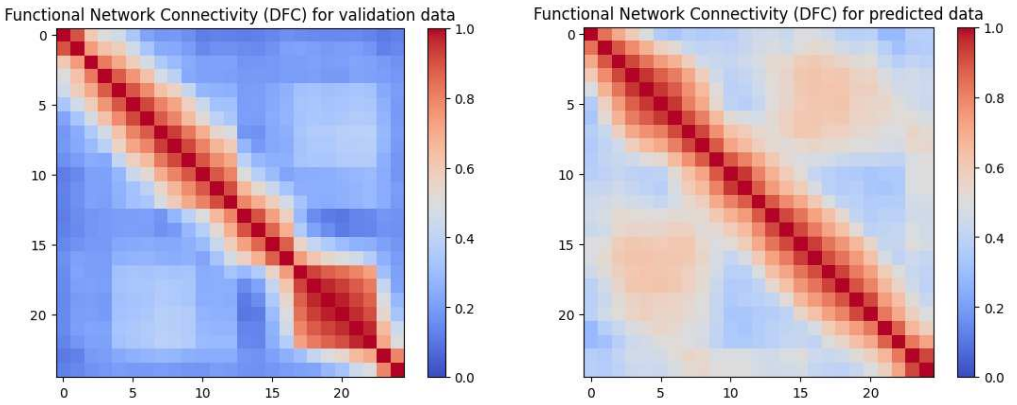


Figure 3.13: Functional connectivity of predicted data of reservoir run autonomously for the same length as data (300-600s).



(a) Dynamical functional connectivity of ground truth data

(b) Dynamical functional connectivity of predicted data

Figure 3.14: Comparison Dynamic functional connectivity (FCD) matrices for empirical and generated data with noise 1e-6. Each matrix represents the similarity between functional connectivity patterns computed over sliding time windows of size = 60s.

Mode relevance

In Fig. 3.16 we show the poles of the trained RC model. Only a subset of the poles moved towards the imaginary axis with a high relevance concentrated in that region. These modes are slow and persistent.

The range of the high-relevance modes lies between 0 and $0.08Hz$. These frequencies are in line with the main frequencies in the BOLD spectrum.

3.4 Discussion

In this chapter, we applied the RC model described in Chapter 2 to simulated time series as well as real brain signals from fMRI.

Our simulation analysis showed that the model can perfectly reconstruct the dynamics of a Koopman-linearizable toy system, whereas it struggles with simple chaotic systems that eschew a description in terms of finite-dimensional, linear dynamics.

Our results on real data (brain signals from fMRI) seem more nuanced. By running the RC model in autonomous mode for a long time window, we can assess the model's ability in reproducing statistical properties of the training signals (the model's direct forecasting ability can be assessed only on a short-horizon basis, for which a ground-truth continuation of the signal is available).

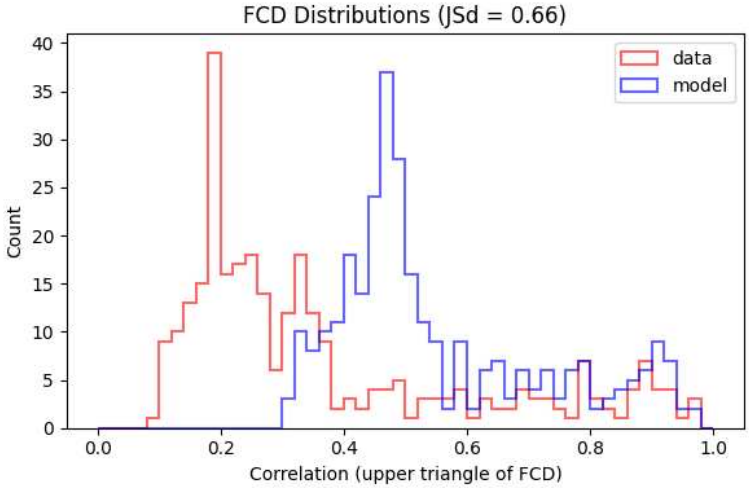
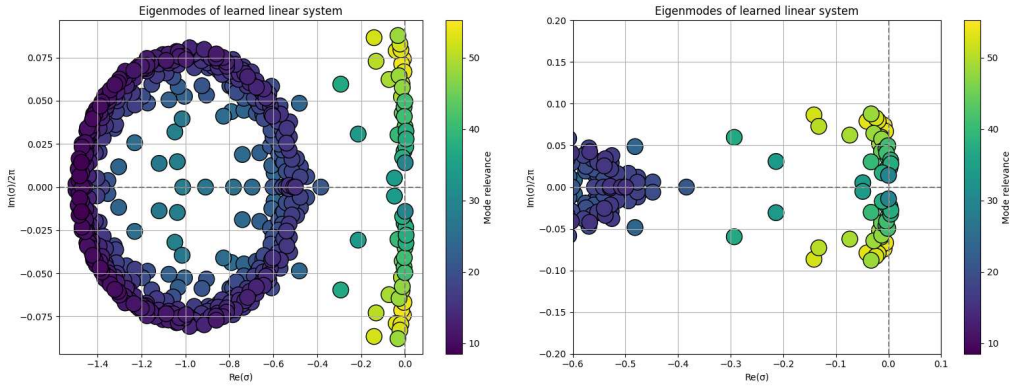


Figure 3.15: Distribution of Functional Connectivity Dynamics (FCD) values for empirical data (red) and model-generated signals (blue). Each distribution is obtained from the upper triangular entries of the FCD matrix, which measures the similarity between functional connectivity patterns across sliding time windows. The Jensen–Shannon distance between the two distributions is 0.66.

The model can generate smooth signals with the right average correlation structure, but not the right degree of dynamic connectivity variation. This does not trivially reflect a sub-optimal choice of the noise amplitude (which was set to 10^{-6} for all analyses). In fact, using a lower or higher amplitude did not lead to better DFC reconstruction (Fig. 3.7). Therefore, improvements may demand further changes, including increasing the size of the reservoir or the amount of available training data.

What we find is a strong sensitivity of the model to hyperparameter selection, particularly to the amplitude of injected noise. While noise is introduced as a stabilizing factor, we see that its role is more nuanced. Specifically, we observe a non-monotonic relationship between noise amplitude and model performance across multiple evaluative criteria.

Functional connectivity accuracy is not strongly sensitive to noise amplitude as it depends on the average covariance over time. At low noise levels the model tends to produce diverging trajectories, which is evident in the stability analysis in figure 3.7. At high noise levels, the noise dominates the dynamics and the model does not learn. Therefore, there is a sweet spot ensuring both stability and reproduction accuracy. Since the optimal noise values may depend on multiple other parameters (e.g., reservoir size and ridge regression tolerance β)



(a) Full eigenvalue spectrum **(b)** Zoomed view near imaginary axis

Figure 3.16: Eigenmode spectrum of the learned linear system. (a) Full spectrum showing separation between fast-decaying and slow modes. (b) Zoomed view highlighting dominant modes near the imaginary axis.

non-monotonically, we could expand upon this direction using a grid search over several parameters. A more extensive parameter search may reveal that the current minimal value (10^{-6}) of the noise amplitude may not be optimal under a different choice of the other parameters.

Overall, we qualitatively reproduced several of the results in the reference paper [7], including: the model's accuracy in reproducing the training data, the model's dependence on external noise, and the model's accuracy in predicting the (static) functional connectivity. However, we obtained a significantly lower performance in short-horizon prediction as well as DFC reconstruction. Actually, a direct comparison between our results and those of Ref. [7] is not warranted, because we did not analyze the same dataset. While we also analyzed fMRI data, the imaging parameters, the region(s) considered, and the number of voxels analyzed are not equivalent to those of the reference paper. Additionally, the two data sets may have undergone slightly different preprocessing. These are inherent differences between our analysis and that in Ref- [7], that cannot be mitigated by parameter optimization. Because of the differences in the training signal, ss expanded upon in section 2.3.c, we had to retain a much larger number of components (~ 45 compared to ~ 4) to explain a sizable fraction of the variance, implying that our system is effectively much higher dimensional. Consequently, the reservoir must reproduce a larger number of modes, including some with relatively low variance. This may considerably alter the temporal variability of correlation structure across windows.

Bibliography

[1] Herbert Jaeger. "The "echo state" approach to analysing and training recurrent neural networks-with an erratum note". In: *Bonn, Germany: German national research center for information technology gmd technical report* 148.34 (2001), p. 13.

[2] Herbert Jaeger and Harald Haas. "Harnessing nonlinearity: Predicting chaotic systems and saving energy in wireless communication". In: *science* 304.5667 (2004), pp. 78–80.

[3] Wolfgang Maass, Thomas Natschläger, and Henry Markram. "Real-time computing without stable states: A new framework for neural computation based on perturbations". In: *Neural computation* 14.11 (2002), pp. 2531–2560.

[4] Heng Zhang and Danilo Vasconcellos Vargas. "A survey on reservoir computing and its interdisciplinary applications beyond traditional machine learning". In: *IEEE Access* 11 (2023), pp. 81033–81070.

[5] Fei Tao et al. "Digital twin modeling". In: *Journal of Manufacturing Systems* 64 (2022), pp. 372–389.

[6] Alexandre Mauroy, Y Susuki, and Igor Mezic. *Koopman operator in systems and control*. Vol. 484. Springer, 2020.

[7] Gabriele Di Antonio et al. "Linearizing and Forecasting: A Reservoir Computing Route to Digital Twins of the Brain". In: *Advanced Science* n/a.n/a (), e17234. DOI: https://doi.org/10.1002/advs.202517234. eprint: https://advanced.onlinelibrary.wiley.com/doi/pdf/10.1002/advs.202517234.

[8] *Infra-Slow EEG Fluctuations Are Correlated with Resting-State Network Dynamics in fMRI - PMC*.

[9] *Ultra-slow oscillations in fMRI and resting-state connectivity: Neuronal and vascular contributions and technical confounds - PMC*. [Online; accessed 2026-04-01].

[10] Duho Sihn, Min-Ki Kim, and Sung-Phil Kim. "Propagation of infra-slow and slow brain activities in electroencephalogram related to behavioral information processing". In: *Neuropsychologia* 220 (2026), p. 109296. ISSN: 0028-3932. DOI: https://doi.org/10.1016/j.neuropsychologia.2025.109296.

[11] *Infraslow EEG oscillations organize large-scale cortical-subcortical interactions during sleep: a combined EEG/fMRI study - PMC*.

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

[13] *Two methods to approximate the Koopman operator with a reservoir computer*. [Online; accessed 2026-03-04].

[14] Matthew O. Williams, Ioannis G. Kevrekidis, and Clarence W. Rowley. *A Data-Driven Approximation of the Koopman Operator: Extending Dynamic Mode Decomposition*. 2014. arXiv: 1408.4408 [math.DS].

[15] *Dynamical Systems of Continuous Spectra | PNAS*. [Online; accessed 2026-03-04].

[16] *Online real-time learning of dynamical systems from noisy streaming data - PMC*. [Online; accessed 2026-03-04].

[17] DoHyun Kim et al. "Spontaneously emerging patterns in human visual cortex and their functional connectivity are linked to the patterns evoked by visual stimuli". In: *Journal of neurophysiology* (2020).