Solutions to Linear Recurrent Neural Networks for Working Memory

Laura Ritter

4th Year Project Report
Computer Science and Mathematics
School of Informatics
University of Edinburgh

2022

Abstract

Working memory as a cognitive process has been the subject of a large amount of research in different fields. From the computational perspective, there are different theories about how it can be achieved in the brain. In particular, two different mechanisms, attractor and non-normal dynamics, have been suggested to model experimentally observed behaviour during working memory tasks. It is unclear how they interact and to what extent they can be considered to produce memory.

The aim of this project is to determine the optimal dynamics for a simple linear recurrent neural network to solve a working memory task. In contrast to previous approaches, the networks are trained directly instead of constructed or linearised from nonlinear networks fitted to experimental data. Furthermore, based on recent findings in the visual system, this is considered for the case of a fixed direction along which the network activity can be read out. While some previous research has aimed to unify the two different mechanisms for working memory, the main focus was on effective integration of inputs. Here, the readout is instead modelled explicitly, providing a different perspective on the mechanisms for working memory.

Instead of attractor or non-normal networks, when optimising for a single delay time, the solutions are oscillatory, but functionally approximating a non-normal solution. Furthermore, when optimising for a range of delay times with exponentially decaying weights, all solutions are strongly non-normal networks. One aspect of the solution shared by all these networks is that amplification of activity plays a major role in stimulus discrimination and maintenance. The obtained non-normal networks unify two existing hypotheses about optimal input integration, while at the same time providing near optimal readout performance.

Research Ethics Approval

This project was planned in accordance with the Informatics Research Ethics policy. It did not involve any aspects that required approval from the Informatics Research Ethics committee.

Declaration

I declare that this thesis was composed by myself, that the work contained herein is my own except where explicitly stated otherwise in the text, and that this work has not been submitted for any other degree or professional qualification except as specified.

(*Laura Ritter*)

Acknowledgements

I would like to thank my supervisor Angus Chadwick for proposing the project and his support throughout the year, in particular for being very approachable and responsive. I am grateful that we were able to discuss a lot of background in depth and resolve any issues encountered in the project. I am very happy that I could work on this interesting project and could learn a lot about an area that was new to me.

I would also like to thank my parents and grandfather for their continued encouragement and always being available to call when I needed it.

A special thank you to Carina for expert rubber ducking  as well as extensive project discussion sessions at breakfast or during the scenic hikes to King's Buildings.

Lastly, thank you to Manos who single-handedly saved me from overworking myself by reliably calling at $9+\varepsilon$.

Table of Contents

Chapter 1

Introduction

Working memory is the ability to retain information about a stimulus for a short period of time without the persistence of the stimulus, often with the aim to process the information further [2]. How this is achieved in the brain is not yet resolved.

The underlying assumption of most attempts to model working memory is that it relies on ongoing population level activity during the delay between stimulus presentation and retrieval [17, 18, 21, 31]. This is required, since the typical membrane time constant lies between 10 and 100ms ([12], Ch.5.2), whereas working memory is considered to last several seconds [22]. Two distinct types of activity have been experimentally observed during working memory tasks: persistent activity which remains stable over the delay period [17, 19], and temporally highly varying activity [13, 30]. The latter is much more common [8, 14]. Both of these types of dynamics have inspired computational models for working memory.

The older, more common model is that of attractor dynamics, modelling persistent activity. Attractors are fixed points in state space which stimulus responses converge to. Once the response has reached an attractor state it can theoretically be maintained indefinitely [21]. However, attractors and their number are a fixed property of the network [20]. Hence, these networks are inflexible with respect to the inputs they can integrate and maintain. They are furthermore required to be normal networks: their connectivity matrix has orthogonal eigenvectors [20, 21].

On the other hand, the hidden feedforward model stipulates that memory maintenance can also be achieved by strongly varying activity [21]. It is much more flexible since it does not require as rigid a network structure: the connectivity matrix of such a network is non-normal, it has non-orthogonal eigenvectors. The model relies on the propagation of a stimulus response along a feedforward chain of patterns which leads to a longer presence of activity in the network [21]. Compared to normal networks, non-normal networks were proven to be capable of much greater memory capacity [20].

Although there have been attempts to argue for one or the other model [31, 35] or to reconcile the two [9, 27, 36], as of now it remains unclear to what extent these different behaviours contribute to working memory and how they interact when they coexist.

The aim of this project is to determine the optimal behaviour of a simplified linear recurrent neural network which has been optimised to perform a pattern discrimination task after a certain delay period. It has been shown that for a similar task, linear recurrent neural networks can find the optimal solution [10]. Therefore, analysing the behaviour of this simple network can shed light on the dynamics which lead to optimal memory performance even in more complex models [36].

Previous research has mostly focused on constructing networks with certain properties that make them well suited for a working memory task [21, 36] or linearising nonlinear systems that were fitted to experimental data [10, 36]. In contrast, in this project, a linear recurrent neural network was optimised directly to solve the task. This provides a novel perspective on the type of the optimal solution. For optimisation, an analytic solution to the network dynamics is used, without requiring the usual training algorithm for recurrent networks, backpropagation through time [40]. The analytic solution has the further advantage that it can be calculated as an average over trials. Thus, the network's behaviour does not have to be simulated, instead a general solution is found.

For this project, it is assumed that the network state is read out by being projected onto a fixed readout vector. Such linear readouts are common in the literature [25, 36, 41] and have been suggested as biologically plausible, due to their simplicity which does not impede their performance [41]. Previous studies of working memory optimised the readout vector for networks fitted to experimental data [36], while the focus lay on how networks can optimally read in information. In contrast, in this project, the network readout is explicitly modelled using a fixed readout direction as proposed for visual attention tasks [28, 32]. Thus, the behaviour of networks explicitly optimised to a fixed readout is analysed to further an understanding of this readout mechanism.

In contrast to previous hypotheses, when optimising for a single delay time the optimal solution found is neither an attractor nor a non-normal network. Instead, a connectivity matrix with complex eigenvalues is learned, resulting in asymptotically stable spiral points, that is, damped oscillations. The network can thus optimally separate the mean activities projected onto the readout for a given delay time. However, the observed slow oscillations mimic the effect of a non-normal network by amplifying the network activity in the correct direction for pattern discrimination.

When optimising for more delay times, both oscillatory and non-normal solutions are found. Moreover, the oscillatory networks approximate non-normal networks even closer due to increased damping after the first amplification of activity.

If additionally the delay times for optimisation are weighted according to slow exponential decay, the optimal networks found are all non-normal. They are characterised by strong hidden feedforward activity and amplification and are near optimally reading the stimulus information into the network.

This report makes the following contributions:

- To the best of our knowledge it constitutes the first attempt to directly optimise a linear recurrent neural network for a working memory task.
- These optimised networks in some cases exhibit very different solutions than

those obtained by direct construction in previous work. Oscillatory dynamics have so far not been considered as plausible models for working memory.

- Training of the networks leads to near optimal input integration.
- To the best of our knowledge it also is the first study of a working memory model using a fixed readout.
- Hence, not only the mechanism for information integration, but also readout is studied. Non-normal networks are found to perform near optimally with this fixed readout.
- The results show that the delay or delays the network was optimised for play a crucial role in determining the resulting type of dynamics.

The findings unify previous theories about optimal input integration in non-normal networks, under the additional assumption of a fixed readout. They provide a single mechanism whereby a stimulus is read in and subsequently amplified to result in a near optimal readout discrimination performance.

After a more detailed introduction this report will present results of the work that has been done:

- An implementation of the analytic solution to a linear recurrent neural network with a binary decision rule, using single delay, cumulative and weighted cumulative loss functions.
- Large-scale experiments conducted with these implementations.
- Analyses of the behaviour of the resulting networks depending on parameter settings under the different decision and loss functions with respect to
  - the method by which they solve the problem,
  - their amplifying properties and how they are used,
  - how learned solutions differ depending on the parameters the networks were optimised for, characterising regularities in those solutions,
  - read-in and read-out performance.
- Visualisations of the results of the analyses.
- An interpretation of the results in the context of neuroscientific literature.

Chapter 2

Background

2.1 Artificial Neural Networks in Neuroscience

Neuroscientific research today is mostly using nonlinear neural networks to model cognitive phenomena as experiments have shown that the majority of neural processing exhibits nonlinear behaviour. However, some explanations of experimentally observed phenomena relying purely on linear models have been proposed. These are simpler to analyse and thus offer a better understanding of the processes defining neural activity.

Usual nonlinear feedforward neural networks are composed of multiple layers of neurons, the connections between which are described by weight or connectivity matrices. Additionally, each neuron has some nonlinear activation function which governs its activity depending on input it receives from other neurons or from outside the network.

Considering a network without such nonlinear activation functions makes it simpler to describe and thus a starting point for the analysis of more complex systems. The total output of the network is defined by the input to the network which is transformed solely by the connectivity matrices in every layer. These can be summarised as just one matrix, the product of all the connectivity matrices between different layers. Thus, in the case of continuous time models, the network can be fully described by the equation

$$\frac{d\mathbf{x}}{dt} = \mathbf{A}\mathbf{x}(t) + \mathbf{B}\mathbf{u}(t) + noise, \tag{2.1}$$

where the connectivity matrix $\mathbf{A}$ determines the change in network activity $\mathbf{x}(t)$ over time. The elements of $\mathbf{x}(t)$ each represent the activity of one neuron in the network. The columns of $\mathbf{B}$ describe the directions in which the inputs $\mathbf{u}$ influence the system [24]. For a stable system, the matrix $\mathbf{A}$ requires eigenvalues with negative real parts.

A similar equation describing firing rate models often additionally includes a time constant τ and a term $-\mathbf{x}(t)$ representing the decay of network activity towards a steady state [12, 23, 26, 37] so that the eigenvalues of the connectivity matrix A are not constrained to be negative, but to not exceed unity in order to ensure stability.

$$\tau\frac{d\mathbf{x}}{dt} = -\mathbf{x}(t) + \mathbf{A}\mathbf{x}(t) + \mathbf{B}\mathbf{u}(t) + noise. \tag{2.2}$$

Networks of the form 2.2 are common in the literature. Letting the time constant $\tau = 1$ and the decay term be absorbed into the connectivity matrix A, they are equivalent to the model presented in Equation 2.2. The models differ in a shift by 1 in eigenvalues. An eigenvalue of 1 for model 2.2 is equivalent to 0 in model 2.1. The latter (2.1) will be used throughout this project. When referencing publications using model 2.2, the eigenvalues reported there will be shifted to agree with the model used in this project.

The dominant neural network architecture for modelling working memory and many other cognitive tasks is a recurrent neural network (RNN). It is thought to capture the connectivity in the corresponding brain regions in the cerebral cortex well, since cortical neurons also exhibit strong recurrent connections [26]. Additionally, it is well suited to modelling time-dependent processes which require memory of previous states [4]. Sufficiently large (nonlinear) RNNs can model the behaviour of any nonlinear dynamical system well [16]. In contrast to the feed-forward network, an RNN does not only have connections in one direction from the input layer via hidden layers to the output layer; instead, there can also exist backwards connections. That is, in an RNN, a neuron can receive input from any neuron in the network [4].

Naturally, this complicates the analysis of such networks. This is true especially because not only the network activity when the solution to some task is found is of interest, but also how the activity of the network changes over time to arrive at the solution. Training RNNs using supervised learning to reproduce experimental results or to generate hypotheses about how a neural network solves a problem is a common approach in neuroscientific research [4, 34, 37].

Especially hypothesis generation requires a thorough understanding of the RNN's behaviour and after the training phase. To this end, the network is interpreted as a dynamical system instead of as a function producing certain outputs. Nowadays it is becoming more feasible to experimentally record the activity of many neurons. Following this development, explanations of responses to stimuli are increasingly sought at the level of the neural population rather than of the single neuron. The description of neural activity as a dynamical system has therefore been expanded to reflect the assumption that computations are carried out by interactions of neurons, instead of just analysing the dynamics of single neurons [28, 37]. There is a vast amount of literature about dynamical systems theory from other scientific disciplines which has been used to describe the learning behaviour of (recurrent) neural networks.

2.2 Models of Working Memory

2.2.1 Attractor Model

There are competing models of working memory, some of which do not necessarily make use of nonlinear networks. The most well-established model of working memory is the attractor model [21]. It stipulates that working memory is achieved by maintaining neural response activity to a stimulus in a fixed point of the network beyond the natural decay time of neural responses [25]. This is achieved by positive feedback: certain patterns of neural activity reinforce themselves, and are thus prolonged, theoretically

indefinitely. Here, patterns refer to collections of active neurons and their firing rates.

A network with attractors filters out irrelevant input patterns or noise and thus converges quickly to specific attracting pattern of activity (see Figure 2.2 Left). The self-feedback required is mathematically described by the eigenvalues (denoted λ) of the connectivity matrix. They determine the extent to which a pattern of activity (represented by the corresponding eigenvector **e**) reinforces itself [21]. Attractor networks are **normal**: they have orthogonal eigenvectors reinforcing themselves independently. Generally, activity patterns are associated with a **time constant** describing the speed of their exponential decay. Together, activity patterns and their time constants are referred to as **modes** [10]. Here, the **eigenmodes** are discussed, whose decay time constant is the reciprocal of the associated eigenvalue. An eigenvalue of 0 represents indefinitely sustained activity of a pattern, eigenvalues between -1 and 0 correspond to activity which decays more slowly than the intrinsic neural decay, while eigenvalues less than -1 lead to faster decay than the intrinsic timescale.

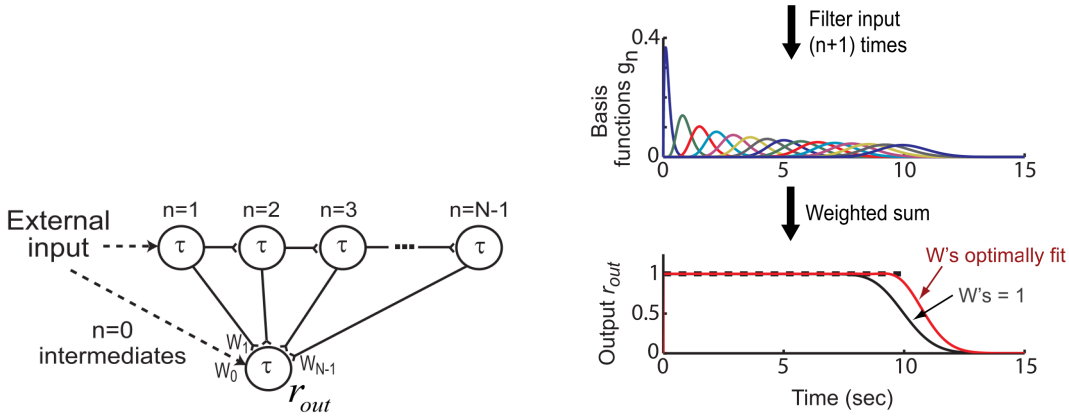
One problematic aspect of modelling working memory by only such self-feedback interactions is that if positive eigenvalues occur, the network activity increases exponentially. The need to avoid this unnatural behaviour complicates tuning the network weights to achieve persistent activity [21]. A different problem is that simple attractor networks can only remember one input at a time, so longer sequences cannot be held in memory [20]. Moreover, networks with only self-interacting patterns are very inflexible: they can only integrate and sustain those patterns which agree with their connectivity structure. Depending on how attractors are arranged in state space, the network is flexible in terms of the strength, but not location of the attractor, allowing only to adaptively discount stronger noise [29]. Moreover, some flexibility in the selection of the attracting pattern is possible, e.g. by re-orientation of a selection vector onto which input is projected [25]. Nevertheless, the potential attracting patterns are a fixed property of the network's connectivity and limited in number.

2.2.2 Hidden Feedforward Model

Interacting activity patterns provide a more flexible approach. In order for eigenmodes to interact, they need to be non-orthogonal. The corresponding networks are then called **non-normal** (see Figure 2.2 Middle). In this case, the Schur decomposition of the connectivity matrix gives further information about the behaviour of the network.

The **Schur decomposition** of a matrix A finds a set of orthonormal (orthogonal and unit length) basis vectors O, such that $A = OTO^{-1}$, where T is an upper triangular matrix. By construction, one of the vectors of O is an eigenvector. Thus, the Schur decomposition identifies both self-interacting patterns (corresponding to diagonal values of T) and those which interact with other patterns in a feedforward manner (corresponding to off-diagonal values of T).

However, the Schur decomposition is not unique, and the choice of decomposition not obvious. In [21], an early study of non-normal networks in working memory, this issue is circumvented by constructing a connectivity matrix from a chosen decomposition: instead of using the decomposition to analyse a network, the constructed network is



(a) A feedforward network. Paths of different lengths through the chain to neurons at the top transfer external input to the output at the bottom.

(b) Response to a single pulse at $t = 0$. Above: Each temporal basis function represents a different step in filtering through the chain on neurons in 2.1a, the peaks of the functions shift successively to the right. Below: A weighted sum of the basis function approximates a step function, i.e. prolonged activity up to time 10.

Figure 2.1: Figures from [21].

used to characterise how networks with a certain Schur decomposition behave.

The feedforward model of working memory considers a chain of neurons where each neuron's activity provokes activity of the next neuron and of the final neuron (Figure 2.1a). The final neuron's activity is prolonged due to the pathways of different lengths that reach it, due to the intrinsic delays of neurons along the path. This cumulative activity at the final neuron can be viewed as persistent activity (Figure 2.1b). Despite this kind of memory occurring in a feedforward chain, it can also be achieved by non-normal recurrent networks in the form of **hidden feedforward** activity. In that case, instead of a chain of single neurons, a feedforward chain of overlapping activity patterns encodes the information about a stimulus [21].

This kind of model has much fewer limitations on the kind of inputs it can effectively integrate and the activity it can sustain. Furthermore, it is not susceptible to unintended exponential growth and generally simpler to tune. There is experimental evidence for both attractor and transient hidden feedforward dynamics during a working memory task, often both occur at the same time [36].

2.2.3 Memory Loading as an Attempt to Reconcile Attractor and Non-normal Dynamics

While both attractor and strong transient non-normal dynamics can be observed experimentally during a working memory task, there is no consensus on how to embed both phenomena into a unified theory of working memory. Thus, it is common to focus on one of the two models. Often, transient dynamics have been considered to be a side effect without functional relation to memory maintenance. However, there are also

attempts to reconcile these two models while ascribing a specific function to transient dynamics [36], even if it is not memory maintenance as in [21]. For instance, in a model where memory maintenance is achieved solely by attractor dynamics, the transient dynamics observed during the delay period in a working memory task are thought to serve as a mechanism to optimally load information into the network so that it can be more effectively maintained [36].

It was shown in numerical simulations using two-neuron linear RNNs and subsequently larger networks, that there exist input directions which optimise memory performance late in the delay period. These are largely orthogonal to the attractor, contradicting the previous assumption that optimal loading occurs when the input stimulus is aligned well with the attractor so the network simply has to perform pattern completion by aligning its activity completely with the attractor.

With optimal information loading the input is first amplified using transient non-normal dynamics before it is aligned with the attractor. Since this results in activity further along the (correct) attractor, the network activity more likely converges to the correct attractor. This results in poor memory performance after a short delay, but a significant improvement over direct alignment with the attractor after a longer delay period. Such networks display the initial strong transient dynamics which converge to stable activity which were observed experimentally. Here, transient activity serves to amplify and subsequently align the input and is thus considered a memory loading mechanism [36].

Much of the analysis relies on calculating the most amplifying mode as in [23], an approach that does not take into account the noise of the network. So, while the networks were trained including a noise term, they were evaluated assuming noiselessness. This somewhat inconsistent approach is presumably owed to the particularly good analytical tractability of the calculation of the most amplifying mode if noise is neglected.

2.2.4 Illustration of the Types of Dynamics Discussed in this Project

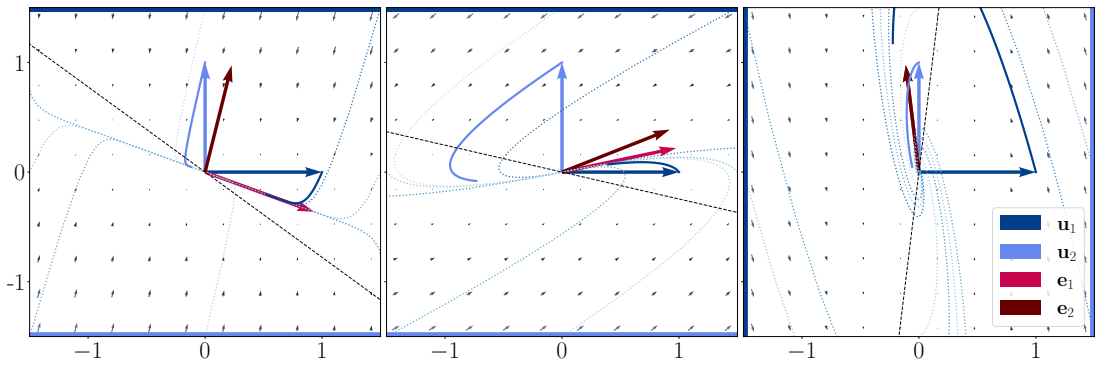


Figure 2.2: Flow fields of connectivity matrices. Left to right: attractor (orthogonal eigenvectors), non-normal (non-orthogonal eigenvectors), oscillatory dynamics (complex conjugate eigenvectors). The dashed black line marks the decision boundary, coloured edges indicate which stimulus should lie on that side of the decision boundary to be labelled correctly. Faint blue dashed lines show trajectories for different initial conditions. The (projection onto $\mathbb{R}$ of) the eigenvectors in red, dominant eigenvector brighter.

2.3 Training and Readout of Neural Networks

RNNs are usually quite hard to train, since even small amounts of mistuning can lead to drastic unwanted changes in behaviour [21]. Usually, especially in the case of nonlinear networks, supervised learning is achieved by using backpropagation through time and thus gradient descent on the loss. Even the simpler linear feedforward networks can exhibit surprising amounts of nonlinearity in their dynamics [33]. However, every RNN is equivalent to a very deep feedforward network. Thus, even in small RNNs the problems that occur in training of normal feedforward networks are aggravated [37].

Linear RNNs however do not necessarily need to be trained via backpropagation, since their formulation can be condensed to a single weight matrix. This reduces the amount of parameters with respect to which gradient descent has to be performed. The lack of nonlinear activation functions thus allows the network to be trained directly, as it will be done in this project.

In [33], the dynamics of learning of feedforward networks with standard backpropagation are studied. They define a system of interactions between input and output modes which can be described by a set of differential equations. Despite the linearity of the network itself these differential equations are nonlinear. This complexity is reflected in the learning dynamics observed in practice, for example the loss remaining constant for many iterations of the gradient descent algorithm and on the other hand periods of rapidly diminishing loss [33]. The behaviour of the common squared loss function on multilayer linear feed-forward networks explains this: it has been proved that this loss function has a unique (local and global) minimum or a unique connected set of minima and saddle points which correspond to plateaus of the loss [3].

The aim of the training in the case of a working memory task is to be able to reconstruct a stimulus which had been applied to the network after some delay. To this end, the network's response to the stimulus needs to be decoded. There are several possibilities of how this readout can operate. The simplest is to take a weighted sum of the neurons' activities, a **linear readout**. It can be represented as a vector in the state space of the network onto which the activity is projected, yielding a single real number as output. Due to their simplicity and effectiveness, linear readouts have been suggested as biologically plausible [41]. It is not resolved if such a readout for a particular brain area is flexible over time (e.g. [6]) or fairly stable [28, 32]. Computational models have been built using both assumptions. In the context of modelling, the subsequent question arises whether the readout should be trained together with the network, not trained, or be the only part of the model that is trained (reservoir computing [37]). In the context of this project, the readout will be kept fixed and only the network will be trained with the readout as training parameter.

2.4 Network Analysis

As already mentioned, various tools from different disciplines are used to analyse neural networks, for instance observability and controllability Gramians originating in control

theory. Given a state space of the network model of the form

$$\frac{d\mathbf{x}}{dt} = \mathbf{A}\mathbf{x}(t) + \mathbf{B}\mathbf{u}(t) + noise \quad \mathbf{y}(t) = \mathbf{C}\mathbf{x}(t) + noise \tag{2.3}$$

with parameters as in Equation 2.1 and $\mathbf{C}$ a readout, the observability Gramian

$$\mathbf{Q} = \int_0^\infty e^{t\mathbf{A}^\intercal}\mathbf{C}^\intercal\mathbf{C}e^{t\mathbf{A}}dt \tag{2.4}$$

relates $\mathbf{A}$ and $\mathbf{C}$ to describe how well the network state can be inferred from observing just input $\mathbf{u}$ and output $\mathbf{y}$. That is, it can be used to determine which kinds of inputs the network activity is particularly sensitive to. This could provide explanations for selective interactions of different neural circuits in the brain. The controllability Gramian analogously relates matrices $\mathbf{A}$ and $\mathbf{B}$. It encodes the preferred directions of the network activity in state space [24]. The main use of the observability Gramian in this project will be to calculate the most amplifying direction in a network, as defined in [23, 36].

The short-term memory capacity of a neural circuit can be assessed using Fisher information. RNNs can store information about their inputs or previous states without needing attractors, as in hidden feedforward architectures [21]. Such memory of previous states is implicit, since the current state is to some extent dependent on previous states. The Fisher Memory Matrix (FMM) introduced in [20] measures to what extent transient memory traces of earlier inputs influence the current state of the network. The FMM's diagonal elements $J_{k,k}$ which the authors call Fisher Memory Curve (FMC) represent the Fisher information in the current network state about a stimulus at time $t-k$. The off-diagonal elements represent the degree to which stimuli entering the network at different times $t-k$ and $t-l$ interfere. Thus, the FMC constitutes a **signal-to-noise ratio (SNR)** of past signals incorporated into the network state compared to the SNR of the current input [20].

In this project, the main tool to analyse networks are decompositions of the connectivity matrix or related matrices. The properties of the eigendecomposition, the Schur decomposition and the singular value decompositions are defined and discussed throughout this chapter together with their interpretation concerning a particular phenomenon. One important measure unifying these decompositions describes the ratio of hidden feedforward to total network connectivity. Henrici's departure from normality can be calculated as $f^A = \frac{\Sigma_i(\sigma_i^2 - |\lambda_i|^2)}{\Sigma_i\sigma_i^2}$. The numerator equals the sum of squares of the off-diagonal elements in the Schur decomposition, σ_i denotes singular values, λ_i eigenvalues. The modulus around the eigenvalues is necessary if they are complex [10, 26].

Otherwise, the Schur decomposition, though important in many common analyses will not be considered in the analysis of the results of this project. This is since by construction one of the Schur vectors is equal to an eigenvector for each Schur decomposition. As the networks in this project are only two-dimensional, if $\mathbf{e}_1$ is the right eigenvector associated with the dominant eigenvalue and $\mathbf{e}_1, \mathbf{s}_1$ is the corresponding Schur basis, then $\mathbf{s}_1$ is collinear with the left eigenvector $\mathbf{e}_{l_2}$. This is because the Schur basis is orthonormal and any pair of right and left eigenvectors corresponding to different eigenvalues are orthogonal [10]. Thus, in the two-dimensional case, the Schur decomposition does not yield any additional interpretable directions in state space.

2.5 Non-Normal Dynamics

2.5.1 Properties of Non-Normal Networks

Non-normal networks tend to exhibit more interesting and varied behaviour than normal networks. For instance, as mentioned above, hidden feedforward activity in RNNs is only possible if activity patterns interact, i.e. are non-orthogonal [21].

Accordingly, the analysis of the FMC of normal networks yields that their memory capacity is limited to unity, either more distant or more recent signals can be kept in memory, but there is a tradeoff between the two. In non-normal networks, the directions in state space corresponding to singular vectors are more apt at retaining information about the network history than others. Certain classes of non-normal networks are shown to have extensive memory, defined as being proportional to the number of neurons in the network. This can for instance be achieved in networks exhibiting transient exponential amplification of signal (and noise) for $\mathrm{O}(N)$ time steps [20].

Not all non-normal networks have this amplifying property. A criterion for amplifying linear RNNs can be derived from the **propagator** of the network. For a network as in Equation 2.1, the propagator is defined as e^{At}, the exponential of the connectivity matrix multiplied with the delay time. The two necessary and sufficient conditions for a network to be able to amplify some activity patterns are non-normality and at least one singular value of its propagator being greater than unity. In this case, the corresponding right singular vectors in the singular value decomposition of the propagator constitute amplified directions [7]. As in [20], the number of trajectories it is possible to encode in a network depends linearly on the network size. Interestingly, transient amplification occurs in many cases in networks which are close to instability.

Amplification is insufficient for memory maintenance: the duration of transient amplification is still determined by the dominant eigenvalue of the network's connectivity matrix which sets the network's timescale. Furthermore, not only the signal but also the input noise can be amplified, depending on its direction, thus amplification does not directly imply better decodability. However, it may serve to stabilise SNR or improve decodability across noisy readout weights [7].

2.5.2 Balanced Amplification

Balanced amplification is also exclusively a feature of special non-normal connectivity matrices. It is a mechanism for selective amplification of patterns in networks with excitatory-inhibitory structure. It results from strong excitatory and inhibitory connections in a recurrent network. There, small changes in activity in the difference between excitatory and inhibitory activity can provoke large changes in the sum of excitatory and inhibitory firing. These sum and difference patterns can be described in terms of activity vectors or modes. The sum mode is an eigenvector of the connectivity matrix and the difference mode the corresponding Schur mode. Thus, responses to small fluctuating spatial patterns of unbalancing regional activity in either direction depend on the relation between the pattern structure and network connectivity.

Balanced amplification constitutes an alternative to the classic Hebbian amplification.

The latter stipulates that neurons with similar activity excite each other, while those with opposite activity inhibit each other, leading to amplification and reproduction of certain patterns, together with a lengthening of decay constants. This relates to the attractor model of working memory in that if a pattern is selectively reproduced faster than its intrinsic delay, it will persist longer than others. Balanced amplification on the other hand does not change the decay time of amplified patterns. Hence it does not of itself contribute to prolonged activity [26]. Nevertheless, amplification of a pattern could lead to a better retainment of memory traces [20].

2.5.3 Non-Normality for Discrimination Tasks

Non-normal dynamics can also serve to improve stimulus discrimination. Let $\mathbf{u}_1(t), \mathbf{u}_2(t)$ be vectors of inputs to the network's neurons at time t. Then an optimal **input linear discriminant** is a vector $\mathbf{w}_{LD}$ such that projecting $\mathbf{u}(t)$ onto $\mathbf{w}_{LD}$ by calculating their scalar product $d(t) = \mathbf{w}_{LD}^{\mathsf{T}}\mathbf{u}(t)$ yields the greatest difference in values for the distinct stimuli and discounting the largest amount of noise. In the following equation for the input linear discriminant, the matrix $\Sigma_{\mathbf{n}}$ represents the input noise covariance.

$$\mathbf{w}_{LD}(t) = \Sigma_{\mathbf{n}}^{-1}(\mathbf{u}_2(t) - \mathbf{u}_1(t)), \tag{2.5}$$

If constant input with temporally uncorrelated noise drives the network into a fixed activity pattern, then a linearisation of the dynamical system around this fixed point can serve as an approximation. This linearisation can be analysed to find a description in terms of Schur or eigenmodes with associated time constants τ. Each of these integrates the projection of the network activity onto the mode for time τ.

A signal is considered to be read in along the **left eigenmodes** of a system. Thus, a network optimally integrating information for the discrimination task has all left eigenmodes aligned with the input linear discriminant. Their time constants τ are equal and longer than the time the stimulus is present. In other words, increased memory of the network response to a stimulus improves discrimination. Normal networks' eigenmodes are orthogonal, so at most one can be aligned with $\mathbf{w}_{LD}$. However, non-normal networks do not have this constraint. Several of their slowly decaying eigenmodes can be aligned with the input linear discriminant, improving performance. Indeed, both nonlinear and linear networks can achieve this optimal solution.

Experimental evidence suggests that this is achieved during learning by realigning slowly decaying left eigenmodes, and not by slowing the dynamics of already better aligned left eigenmodes or increasing non-normality of the network [10].

Chapter 3

Methods

3.1 Network, Task and Inputs

3.1.1 Network Equation

A common simplification in the analysis of neural network is to reduce the number of neurons or units in the system [21, 26, 36]. Using only two units can nevertheless reveal some of the properties of the dynamical system on a larger scale and has the advantage of making a system visualisable. In particular, one can inspect the trajectories of the network's responses to a stimulus once it has converged to a solution.

Furthermore, the restriction to linear networks means that an analytic solution of these networks can be found. This makes simulations unnecessary, the network state at time t and the derivatives used for the optimisation can simply be calculated using matrix operations. Additionally, this analytical approach allows averaging over trials to arrive at a general solution for given parameters. Linear networks have been found to predict the behaviour of nonlinear networks for the same task well, so linear networks are not necessarily an oversimplification [36].

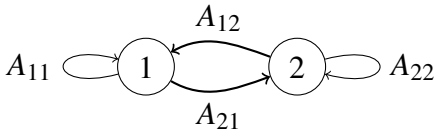


Figure 3.1: A two unit network, the labels correspond to the entries in a 2×2 connectivity matrix $A = \begin{bmatrix} A_{11}A_{12} \\ A_{21}A_{22} \end{bmatrix}$, the effective weight of the connection.

In this project, a two stimulus discrimination task is considered. An RNN is trained to reconstruct after some delay which one of two stimuli it was presented with. The delay, readout direction and stimuli are parameters for the training of the network. The RNN is described by the differential equation

$$\dot{\mathbf{x}}(t) = A\mathbf{x}(t) + \mathbf{u}(s,t), \tag{3.1}$$

where A is the connectivity matrix and $\mathbf{x}(t)$ the network state at time t. The connectivity matrix A is required to be stable, that is, to have nonpositive eigenvalues. If this

condition is violated, then due to the nature of the solution to the above differential equation,

$$\mathbf{x}(t) = e^{A(t-t_0)}\mathbf{x}_0 + \int_{t_0}^{t} e^{A(t-\tau)}\mathbf{u}(s,\tau)d\tau,$$

the network activity would blow up. This is due to the matrix exponential term, the equivalent of the propagator described in Section 2.5.1, becoming large.

No further restrictions are placed on the network. Hence, the single units in the network should not be interpreted as neurons. This is because each matrix entry is allowed to become positive or negative under the constraint of stability. Thus the units do not satisfy Dale's law and cannot be divided into excitatory and inhibitory neurons. Rather they can be viewed as averages over mixed populations of excitatory and inhibitory neurons. Similarly, the entries of the connectivity matrix could be interpreted as an effective interaction weight between the populations, encompassing both the actual connection strength and a nonlinearity.

Note that in the following, $\Sigma_{\mathbf{n}}$ and $t_s = t_0 = 0$. Further note that evaluation times will be denoted by t, whereas the time a network was optimised for will be denoted by t_d.

3.1.2 Inputs

The input to the network is a vector $\mathbf{u}(s,t) = \mathbf{u}_s\delta(t-t_s) + \mathbf{n}(t)$ representing the direction in state space the stimulus drives the network into. Here, $s \in \mathbb{R}$ is a label for the stimulus. It is the value the projection of the network response onto the readout should take when reconstructing which stimulus the network saw. The stimulus is applied only once and with an infinitesimally brief duration, at time t_s. While $\mathbf{u}_s$ is the fixed direction in state space associated with stimulus label s, the actual input to the network is disturbed by temporally uncorrelated Gaussian noise $\mathbf{n}(t)$. Taken together, Equation 3.1 corresponds to the first of the equations in 2.3 with B the identity matrix and the noise term incorporated into $\mathbf{u}(s,t)$.

Assuming further that the initial condition of the network is independent of the input noise, it is possible to neglect the initial condition. Networks are considered to be in a stable state with respect to the input noise when the stimulus arrives.

3.2 Decision and Loss Functions

3.2.1 Decision Rule used with a Single Delay Loss Function

The performance of the network in this memory task is tested by attempting to reconstruct which stimulus had been applied from the network state at time $t_d > t_s$. To this end, the network state is projected onto a weight vector $\mathbf{w}$, thus providing a linear readout. The result will be used in a decision function $d(\mathbf{x}(t_d))$ which determines the real value associated with the current network state which can then be compared to the label s of the stimulus.

In the main results, a binary decision function will be considered: $d(\mathbf{x}(t_d)) = \Theta(\mathbf{w}^\intercal\mathbf{x}(t_d))$ where Θ is the Heaviside step function. Hence, the network is considered to have solved

the task if the readout of the activity after some delay has the correct sign. Differences in magnitude between the readout and the stimulus label are ignored. The rule is thus explicitly constructed for two stimulus discrimination tasks.

A square loss function, $L = \sum_s \langle (s - d(\mathbf{x}(t_d)))^2 \rangle$, is used with this decision function. Note that the loss function is calculated as an average ($\langle \cdot \rangle$) over trials. In addition to the advantage of a general solution this simplifies the expression and thus also helps avoid the otherwise necessary simulation of the network and training using backpropagation.

3.2.2 Parameter Restrictions

The binary decision rule requires some restrictions on the inputs. Since the vectors $\mathbf{u}_s$ for $s \in \{0,1\}$ and $\mathbf{w}$ are kept as unit vectors, they are best described in terms of their anticlockwise angle from the positive x-axis. In the following, unless stated otherwise, the vector $\mathbf{u}_1$ will be fixed as $[1,0]$, whereas $\mathbf{u}_2 = [\cos(\theta), \sin(\theta)]$ and $\mathbf{w} = [\cos(\varphi), \sin(\varphi)]$ with $\theta, \varphi \in [0, 2\pi]$.

Due to the restriction of parameters, this binary approach also has the advantage of making the results comparable, and the losses interpretable. The loss is the sum over the input stimuli of the probability of the network not recognising a stimulus correctly. Thus, there is a clear definition of the network performing at chance level, in the two stimuli case, this is equivalent to a loss of 1.

3.2.3 Cumulative Loss Function - Training for Multiple Delays

With the binary decision rule for single delays, activity was always optimised for a specific point in time. However, usually the exact delay at which information is required to be available is not known in a working memory task. It therefore makes sense to optimise for a delay period instead of just a single delay. In most cases, the solution with the binary rule had already produced sensible losses during the part of the delay period close to the delay the network was optimised for. However, there is a tradeoff between low losses at the optimised delay and high losses at earlier times. The latter are not a realistic model of how working memory is assumed to perform.

The loss function can be extended to account for more than a single delay time. Since the analytical tractability of the integral over delay time of the binary decision rule loss function is limited, it is approximated by an average over losses at some number of delay times. The number of evaluation points of the loss function was chosen empirically to be 25, retaining accuracy while reducing computational cost.

However, in some cases this discretisation of the optimisation range leads to suboptimal solutions being chosen. These have a high oscillation frequency, optimising just for the evaluation points as shown in Figure 3.2. If the aim is to retain a memory for an extended period of time, the performance should not drop for a slight deviation from the delay or delays the network was optimised for. Thus, the loss function requires adjustments, such as a penalty for rapid oscillations as determined by the imaginary part of the network's eigenvalues. Rarely, such fast oscillations occurred also with the

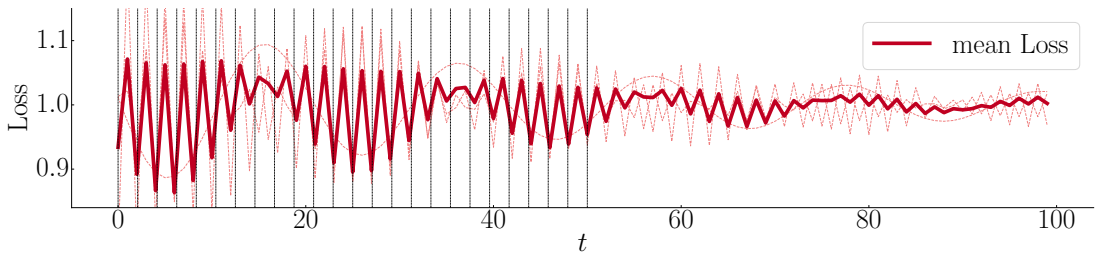


Figure 3.2: Networks with very high oscillation frequency simply optimising for the evaluation points marked by vertical lines.

general binary decision rule, so the adaptation of the loss function by a penalty for loo large imaginary parts was applied there.

3.2.4 Cumulative Loss Function with Exponential Weighting

Since it is natural for memory to be present closer to the time of stimulus presentation and to subsequently decay, it makes sense to weight evaluation points differently. This can be achieved by either sampling evaluation points in the delay interval accordingly, or to calculate a weighted average with weights from a decaying function. The latter approach is chosen here.

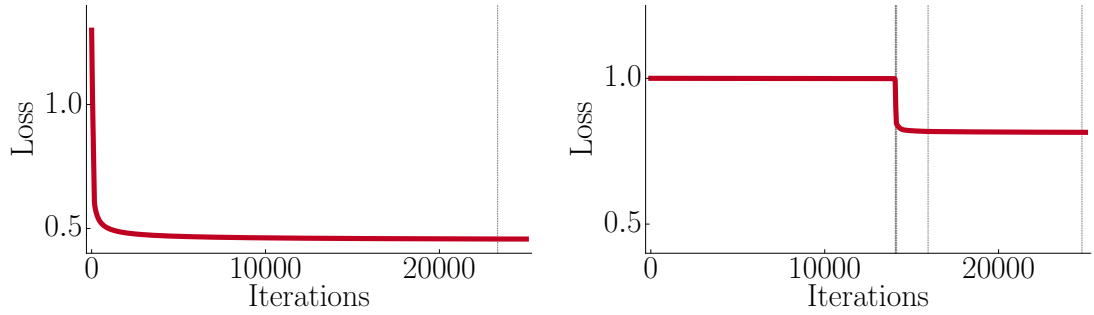
The last adaptation of the loss function includes an exponentially decaying weight $e^{-\lambda t_d}$ for all evaluation points t_d. In addition, jitter is added to the evaluation points to discourage quick oscillations optimising for the evaluation points only. However, this is usually insufficient for the purpose, and an additional penalty is necessary.

Other decreasing functions could also be used for weighting, but the exponential has the advantage of adding only a single parameter, the decay constant λ. Furthermore, there is evidence that memory does indeed decay exponentially [42].

Naturally, the smaller the decay constant λ, the lower the losses for longer delays. In the following, $\lambda = 0.01$ was chosen, since when optimising for a maximal delay of 50, convergence occurs at a similar delay time compared to the previous rules. If the network timescale is interpreted as a neuronal timescale (approximately 20ms), then a delay of 50 is equivalent to one second. While this is not a long time, similar results can be achieved for longer delays. The delay $t_d = 50$ is chosen here to achieve comparability. However, when choosing $t_d = 500$ similar results require a smaller decay constant of $\lambda = 0.001$. This results in higher losses throughout: as noted in [20], there is a tradeoff between better memory performance at early and late delays.

3.3 Training the Network

The network is then trained via gradient descent on the loss. Solving the differential equation 3.1 describing the RNN, and expanding the loss function, it is possible to determine the derivative of the loss function with respect to the connectivity matrix A in terms of the derivatives of mean $\langle \mathbf{x}(t) \rangle$ and covariance matrix $\Sigma_{\mathbf{x}(t)}$ of the network



(a) $t_d = 1$ (b) $t_d = 10$

Figure 3.3: Learning dynamics of the 2 neuron network when initialised with $A = -\mathbf{1}$ with $\theta \approx 4.599$, $\varphi \approx 3.761$. The learning rate is allowed to decrease if the loss would otherwise increase. Vertical dashed lines indicate where the learning rate has been decreased. 3.3a The network converges to a local minimum. As the loss approaches the minimum its change slows even before the learning rate is decreased. 3.3b Convergence to the local minimum is only achievable with small learning rates. The dynamics include long plateaus and sudden decreases of the loss function which in many instances coincide with the decrease in learning rate.

state (see Appendix B). Then, the matrix A is updated using the usual gradient descent algorithm

$$A \leftarrow A - \eta \frac{\partial L}{\partial A},$$

where the learning rate η is some small constant.

It is important that the matrix A remains stable after the update, otherwise the covariance matrix $\Sigma_{\mathbf{x}(t)}$ may lose the property of positive definiteness and thus become invalid.

3.3.1 The Loss Landscape

The optimisation of recurrent neural networks is not a simple task. The loss function has numerous local minima which can exhibit different dynamics. In addition the network tends to become unstable. Stability and the local minimum found depend on the initialisation of the connectivity matrix A and the chosen learning rate.

The loss need not decrease at a steady rate across iterations, but there may be ”long plateaus of little apparent improvement followed by almost stage-like transitions to better performance”[33]. These rather extreme learning dynamics are observed with a minority of parameter settings. Figure 3.3b shows one example. However, for the majority of settings resulting in stable networks, convergence to a local minimum is much more regular and the learning dynamics resemble what is shown in Figure 3.3a.

3.3.2 Methods of Optimisation

These learning dynamics render the problem of optimisation a difficult one. An adaptive learning rate which is halved every time the loss would increase can be used. In some

cases, this lets the network converge where it had become unstable or simply had not converged to the local minimum. Nevertheless, this remedy is unsatisfactory, since for some parameter choices larger initial learning rates lead to convergence, whereas some smaller ones lead to instability, complicating the choice of initial learning rate. This is aggravated by the fact that having decreased the learning rate the network may require more iterations to converge.

The number of iterations must thus also be carefully chosen, since it is the only stopping mechanism for the gradient descent algorithm. This is due to learning dynamics as in Figure 3.3b. Convergence and a plateau of loss are indistinguishable from the point of view of the local change in loss or its derivative, therefore assumed convergence cannot be used as a stopping criterion. Instead, an empirical upper limit for the number of training iterations can be set by training networks with different parameters for a large number of iterations for a particular learning rate. The limit is chosen as a number of iterations after which all but perhaps a small fraction can be assumed to have converged.

Furthermore, the local minimum a network converges to may not be the global minimum and the dynamics of the network may differ significantly between different minima. To find a candidate for the global minimum it would be necessary to initialise the algorithm with different connectivity matrices. This, in combination with small learning rates and many iterations of the algorithm significantly increases the computational resources required and renders large-scale analyses infeasible.

Hence, instead of using the algorithm coded from scratch, the computationally more efficient library function `scipy.optimize.basinhopping` is used for the majority of the computations to determine the optimal behaviour of the network with greater certainty. This algorithm uses different initialisations obtained as an offset from an original one and subsequently uses a local minimisation algorithm, `scipy.optimize.minimize` which was found to work best with conjugate gradient algorithm `CG`. The latter is a class of algorithm using only first derivatives, also used in [40].

By initialising `scipy.optimize.basinhopping` itself with several different matrices chosen at random and accepting the result yielding the lowest loss, it is possible to achieve better solutions than previously. Due to the large number of initial conditions covered and an internal adaptive learning rate this increases the probability of finding a global minimum for a given set of parameters. Indeed, for most initial conditions this procedure finds similarly valued local optima, that is, many quite different connectivity matrices that lead to almost identical losses.

Chapter 4

Results

4.1 Binary Decision Rule

4.1.1 Solution Method - Training and Trained Network

There are several hypotheses about training strategies of the network to improve discriminability of the input stimuli after some delay.

1. Training can achieve a separation of the mean response for each stimulus so that the response patterns for the stimuli become increasingly distinct.
2. It can amplify the mean response to the stimulus vectors. While this alone is insufficient for prolonging activity, see the discussion in Sections 2.5.1, 2.5.2, if the network is specifically adapted for two stimulus directions, it may learn to selectively amplify those directions more than the noise which may be oriented in any direction in state space.
3. Both these ways of the network acting on the activity means are insufficient for discriminability of the network readout. Despite a large mean separation, the responses to stimuli will be labelled incorrectly if they lie on the wrong side of the decision boundary defined by the readout vector. Thus, it is necessary to not only move their means apart, but also into the correct directions in state space with respect to the decision boundary.
4. Training can reduce the effect of noise by shaping the adjacency matrix such that the network covariance projected onto the readout $\mathbf{w}^\intercal \Sigma_{\mathbf{x}(t)} \mathbf{w}$ is decreased. This can be done since the network covariance is the solution to a Lyapunov equation dependent on only the noise covariance $\Sigma_{\mathbf{n}}$ and the connectivity matrix A, see Appendix B.7.

Figure 4.1 shows a representative example of the change in mean and variance projection across training, together with the associated change in SNR and loss. Considering the training is necessary to judge how the network is adapted, since the variance depends directly on the connectivity matrix and thus does not change across delay time. While mean separation can be observed during a trial, the variance cannot change during a

trial since the network is driven into a stable state with respect to the input noise before the stimulus is applied. Note that for the production of this figure, the standard gradient descent algorithm from Section 3.3 with an adaptive learning rate was used in order to be able to access intermediate results.

The results indicate that mean separation and not variance reduction leads to the increase in SNR observed. The mean activities projected onto the readout diverge, but variance increases monotonically. However, the further increase in variance does not influence the loss or SNR which remain stable. Furthermore, the mean activities take on different signs and thus lie on opposite sides of the decision boundary. Amplification also occurs, although this is not as obvious from Figure 4.1 and will be discussed in more detail in Section 4.1.3. It can be seen here only by observing that the projection $\mathbf{w}^\intercal \langle \mathbf{w}_0 \rangle$ decreases to a value slightly below -1, but readout and stimulus vectors only have length 1. Thus, some amplification has occurred. In fact, this amplification is related to the increase in variance observed throughout training.

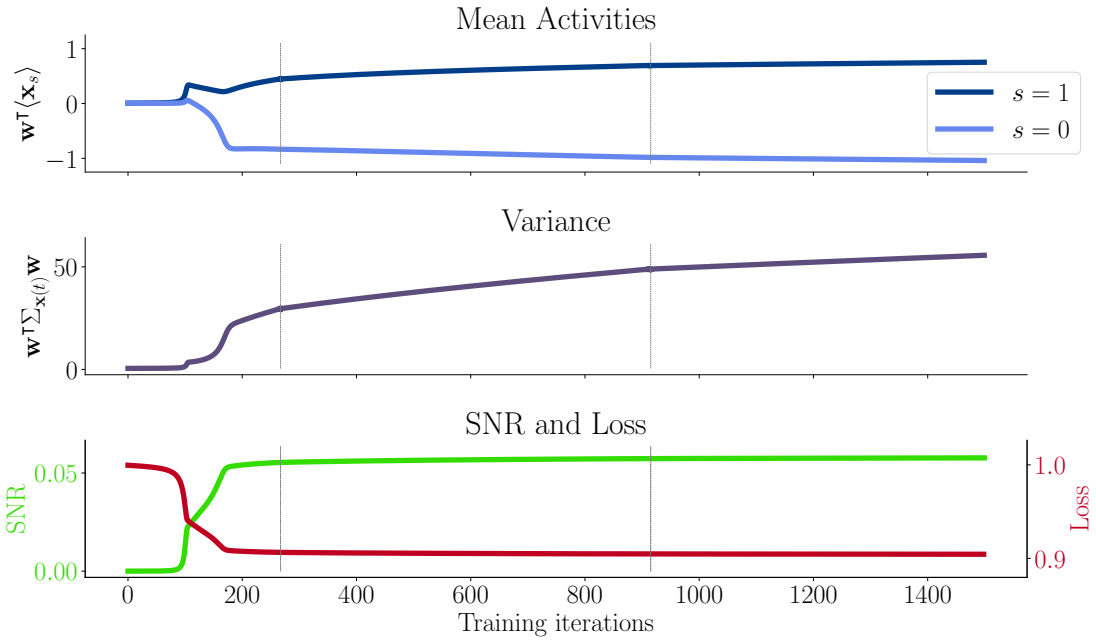


Figure 4.1: Different metrics across training iterations. Vertical dashed lines indicate a decrease in learning rate.

4.1.2 Dynamics of the Solution

The most striking result is that with a binary decision rule, the most common type of dynamics is neither attractor, nor non-normal dynamics. The majority of the best performing networks exhibit oscillatory dynamics where the eigenvalues and eigenvectors of the connectivity matrix are complex conjugates.

The loss function of even just a two unit network may have many local minima with different behaviours. However, out of those, the ones with oscillatory dynamics exhibited by far the smallest losses. Interestingly, the oscillations are in most cases not fully used

by the network when integrating input up to the delay time it was optimised for. In this way the network's effect is similar to that of a typical non-normal network.

The observed oscillatory networks have a similar effect to the one in Figure 4.2. Clearly, for most delays which differ from the delay t_d the network was optimised for, the network performs worse, even below chance level (Figure 4.2d). As the delay approaches t_d, performance improves. Interestingly, none of these networks optimised with random parameters for a specific t_d actually exhibited their optimal performance at that specific t_d, rather shortly before, as in Figure 4.2b.

Furthermore, the network only uses about a quarter of an oscillation up to time t_d. This makes sense if the objective is to achieve amplification of the network activity in the correct direction in state space. The oscillation is damped, which is observed in Figure 4.2d. Hence, the largest amplification can only occur during the first half of the first oscillation.

Figure 4.2 also shows nicely how the trajectories of the means agree with the shape of the covariance matrix of the network. In most networks there is an alignment of the direction of largest variance and largest amplification of the mean vector. However, in this case, the direction of the mean vector when it is most amplified is not orthogonal to the decision boundary.

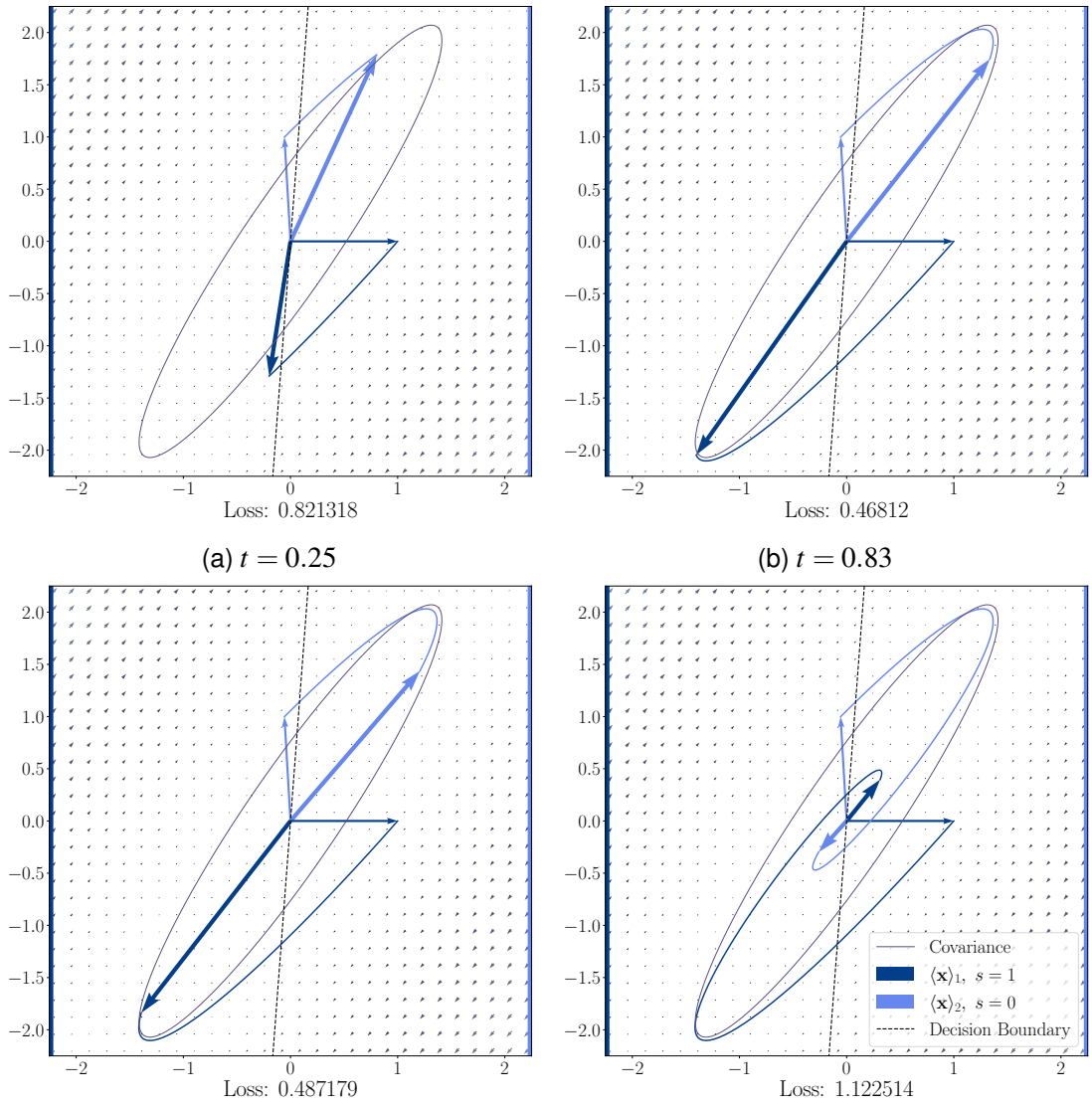
This would be unexpected if the only goal was to ensure that the mean activity lies as far on the correct side of the boundary as possible to avoid mislabelling due to noise in particular trials. However, the alignment of the direction of greatest amplification and variance would then mean that the direction of greatest variance was aligned with the readout direction as well. This would in turn lead to a deterioration of readout accuracy and thus may counteract the benefits of the amplification. In fact, in most cases, the direction of least variance is well aligned with the read-in direction. This makes sense, since the noise present in the network is input noise, so minimising its impact can lead to an improvement in performance. Furthermore, this agrees with the findings in [10].

Figure 4.4 shows ten networks and their performance across time. Again, the point of greatest mean separation occurs slightly before the time t_d. Unsurprisingly, the point of greatest mean separation is also that of maximal SNR and minimal loss. There is a large variance in performance of networks optimised for different parameters. Many exhibit amplification of the projection onto the readout of up to 15 (without amplification, the maximum would be 1).

4.1.3 Amplification

Amplification of activity is thought to support working memory due to an increase or stabilisation of the SNR [7, 20], or in particular improving memory loading [36]. Indeed, the obtained networks are amplifying networks. There can be two uses of amplification.

1. If the stimulus responses are already located on the correct sides of the decision boundary, amplification can be used to move them apart, and thus also further away from the decision boundary.



(a) $t = 0.25$ (b) $t = 0.83$

(c) $t_d = 1$ (d) $t = 3$

Figure 4.2: The effect of the network with connectivity matrix $A = [[-5.5239, 3.9512], [-6.4182, 4.0049]]$ which was optimised for $t_d = 1$ shown with different delays. Figure 4.2b shows the result near the optimum. The decision boundary is shown as a dashed line, thin blue arrows are the stimulus vectors, wide ones the corresponding mean activity at time t. The covariance matrix is shown in purple for $\frac{3\sigma}{8}$. The coloured edges show which stimulus would need to lie on which side of the decision boundary in order to be assigned the correct label. Most oscillatory networks, especially for longer delays, produce much greater amplification than the one above. This does not change the solutions qualitatively, but they cannot be visualised as clearly.

2. The amplification can be used to move stimulus responses to the correct side of the decision boundary. This is used in cases like in Figure 4.5a. There, the response is amplified along the decision boundary, almost parallel to it, in order to cross it eventually. After having crossed it, there additionally is a great mean separation which would not have been possible, had the responses been moved

across the boundary directly.

A comparison (Figure 4.3) shows that the most amplifying direction for linear attractor networks in [23, 36] as derived from the observability Gramian experiences the greatest amplification in terms of norm. The amplified direction found through SVD as defined in [7] is also greatly amplified. Due to the oscillatory behaviour of the network, there exist several increases in vector norm, instead of a single increase followed by exponential decline as in [7]. The same ordering persists when projected onto the readout (dashed lines). This suggests that the main use of amplification in the networks under consideration is mean separation along the readout direction, not cases like in Figure 4.5a, where amplification occurs mostly orthogonally to the readout. Furthermore, this shows that while the network is optimised for two stimuli, neither of them experiences the maximal possible amplification, not even in the direction of the readout. Depending on the orientation of the stimuli, the network may represent a balance of amplifying both stimulus responses strongly, instead of just one of them optimally.

When training the network with the standard gradient descent algorithm described in Section 3.3, a substantial number of networks were found to be optimal close to instability. It was observed in [7] that also non-normal amplifying networks constitute a class of networks close to unstable networks in the space of connectivity matrices. This makes sense intuitively, since in unstable networks at least some directions experience unbounded amplification. Within the class of non-normal networks, amplifying networks have strong non-normality [7].

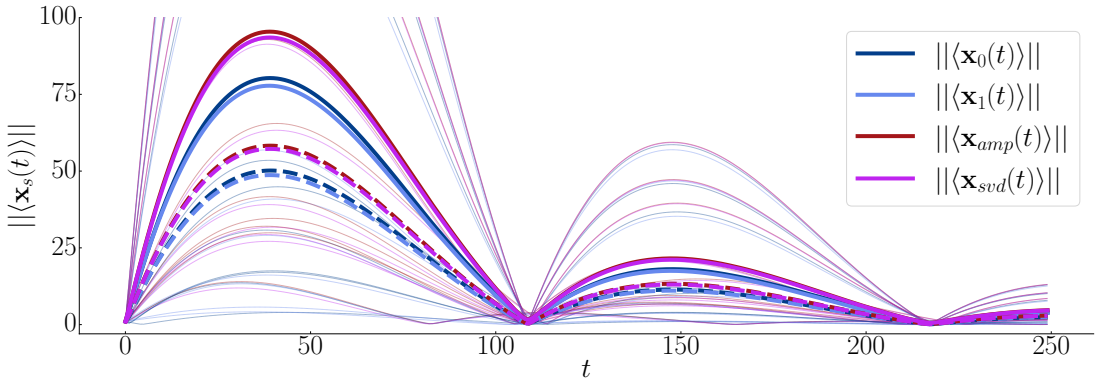


Figure 4.3: Norm of the mean vectors for the input stimuli and the most amplifying mode as calculated in [23, 36] and an amplified direction found through SVD [7] for 10 networks optimised tor $t_d = 50$ and the average of the results for these networks. Dashed lines indicate the projections onto the readout.

4.1.4 Comparing the Solution Methods of Different Types of Dynamics

Recall that the time constant of a mode is defined as the reciprocal of the eigenvalue's real part. In oscillatory networks, oscillation frequency is determined by the imaginary part of the eigenvalues. The closer it is to 0, the slower the oscillation. As noted before,

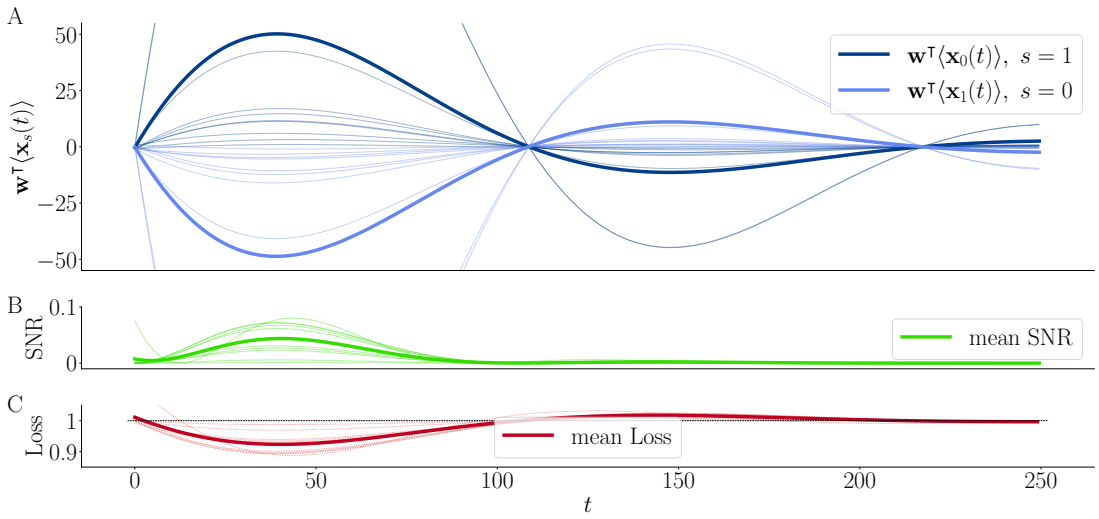


Figure 4.4: Networks trained for the same parameters as in 4.3. Thick lines indicate average behaviour. It is clear that the network is well-adapted to a delay of 50, since the optimal discrimination, highest SNR and lowest loss are all reached briefly before $t_d = 50$. **A**: Projection of the mean activity vector onto the readout for the different input directions. Around $t_d = 50$ there is a large separation between the projections, which lie on the correct side. This supports the hypothesis that learning is achieved by mean separation. After $t = 100$ the projections of the mean activities will be labelled wrongly. **B**: SNR and mean thereof, same networks, determined using the inputs they were optimised for. **C**: Loss and mean thereof, mirroring the SNR.

since the oscillatory networks use only approximately a quarter of their oscillation, their effect is similar to that of a non-normal network. This is clearly visible in Figures 4.5a and 4.5b. In both cases, the flow field is parallel to the dominant eigenvector, the slow mode (for the non-normal network in Figure 4.5b) or the real part of the eigenvectors (for the oscillatory network in Figure 4.5a). The flow is oriented in opposite directions on the different sides of the dominant eigenvector so that the mean activity is amplified in opposite directions, increasing mean separation.

The difference lies in the asymptotic behaviour. The non-normal network converges to the slow mode and activity returns along the slow mode towards the origin. The oscillatory network keeps spiralling around the origin, though with decreasing amplitude. This damped oscillation guarantees that activity does not grow indefinitely. In networks without oscillation, convergence to some mode directed back towards the origin is used to avoid runaway network activity.

Indeed all networks shown have at least one relatively slow direction. The attractor network 4.5d exhibits the slowest mode of all, almost by one order of magnitude. Nevertheless, sustained activity along a slow direction is insufficient for the task, since in this case discrimination is impaired as both stimuli are projected onto the slow mode on the same side of the decision boundary. The difference to the other types of networks is also visible in the flow field, it is the only one where the fastest flow is directed orthogonally onto the slow mode which is reached after only a short delay. Subsequently the response remains on the slow mode, in this case on the same side

for both stimuli. Thus, the attractor network performs worse than even the non-normal network whose slow direction has the shortest time constant out of all networks shown but separates the means using amplification.

The time constants for the oscillatory networks 4.5a, 4.5c are very similar, nevertheless, their behaviour differs strikingly, due to the difference in the imaginary part of the eigenvalue which is responsible for the oscillation. While the network in Figure 4.5c is also a oscillatory network and performs well for $t_d = 50$, it oscillates several times up to the delay it was optimised for. Thus, it is not a plausible solution, since it only performs well at a few points during the delay period. Here, a certain delay needed to be chosen for the optimisation, but in general it is to be assumed that not only one delay time is known in advance to be relevant, so the memory should be available throughout the delay period, albeit with some decline in availability over time. Indeed, all other kinds of networks described here have this property. This is why in all other analyses the case of rapid oscillations is excluded. It is discussed here as a demonstration that not all networks having good losses - the loss is better than that of the non-normal network - and long time constants are in fact plausible solutions.

There are fundamental differences of the solutions of non-normal and oscillatory networks which cause oscillatory solutions to be better when optimising just for a single delay time, especially if the delay is long. Non-normal networks use a fast mode to improve loss immediately and a slow mode to maintain this improvement for as long as possible, whereas for oscillatory networks improvement cannot be sudden unless deterioration is also sudden since both are determined by the oscillation frequency.

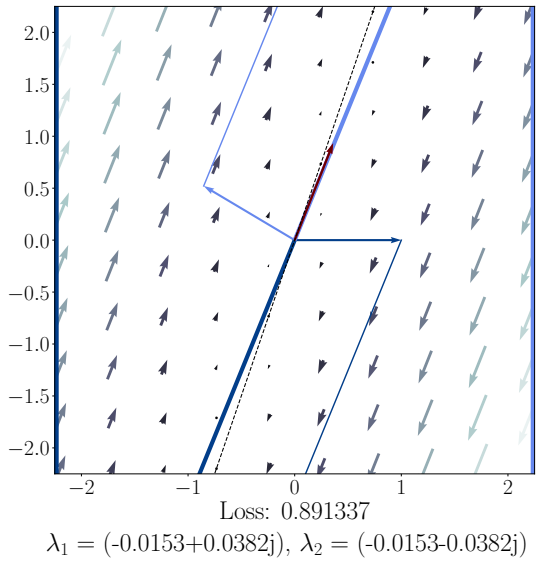
Thus, when optimising for just a single delay, oscillations are preferable, as their point of optimal discrimination can be moved to late delays by slowing the oscillation. Non-normal or attractor networks on the other hand would require either

1. slower convergence onto the slow, maintaining mode, achieved by both eigenvalues being close to 0, or
2. that the maintaining mode is much slower than all those exhibited here. Even then it is not clear if the SNR can be maintained, since there is additional input noise also at later delays.

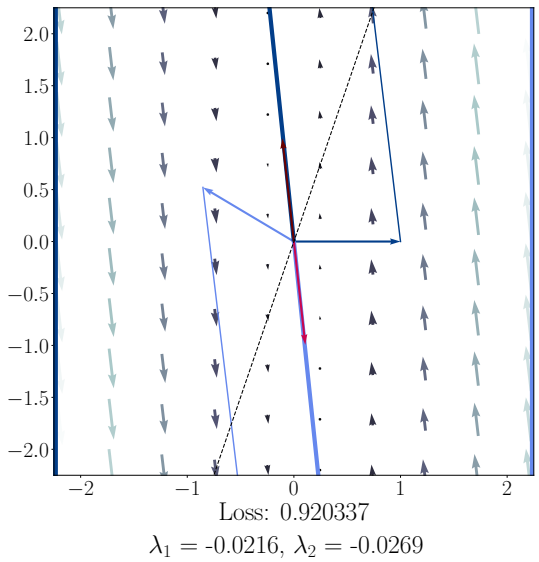
In the latter case, despite the slowness of decay of the signal, noise may dominate if the signal is not instead amplified further, as in oscillatory networks. Of course, noise can also be amplified, but it does not have a fixed direction. It will therefore not always be oriented in a direction of large amplification, unlike the signal. In any case, constructing such a non-normal or attractor network would presumably require a very fine tuning of the network weights, which may not be achievable with the optimisation methods used.

4.1.5 Dependence on Angles between Input Stimulus Vectors and Readout Vector

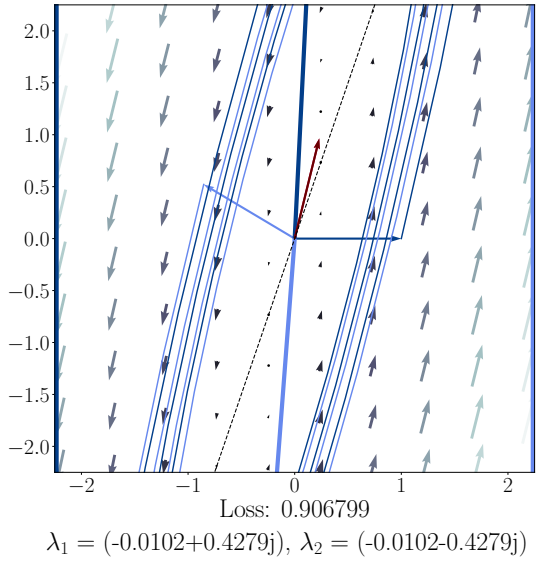
Except in the pathological case of equal stimulus vectors, the optimal solution found is complex. Hence, the question of how the oscillatory networks differ is reduced to the



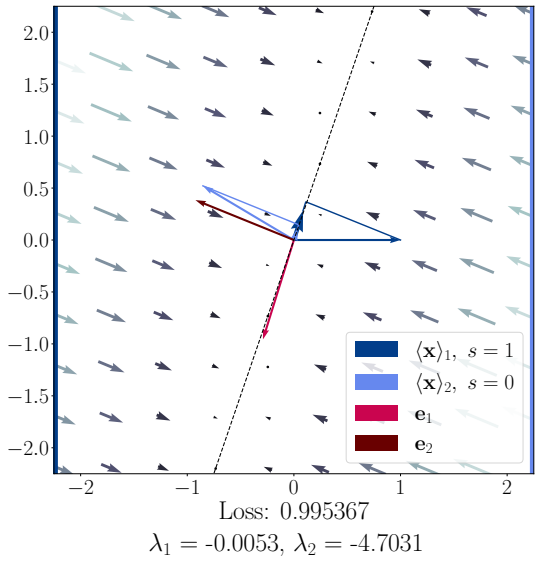
(a) The network performing optimally, a slow oscillation.



(b) A non-normal network, with near parallel eigenvectors oriented in opposite directions.



(c) A network with too rapid oscillations.



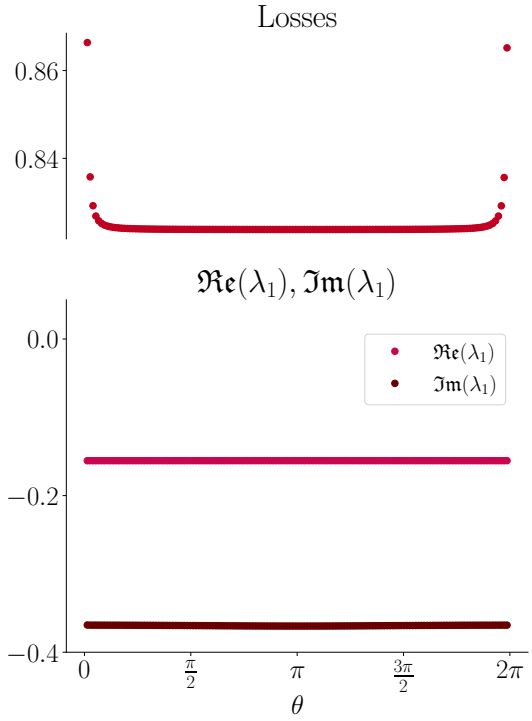
(d) An attractor network.

Figure 4.5: Three different networks for the same parameters. The decision boundary is shown as a dashed black line, arrows indicate the flow field. Noted underneath are the eigenvalues of the connectivity matrix, where λ_i corresponds to eigenvector $\mathbf{e}_i$ in the phase plots.

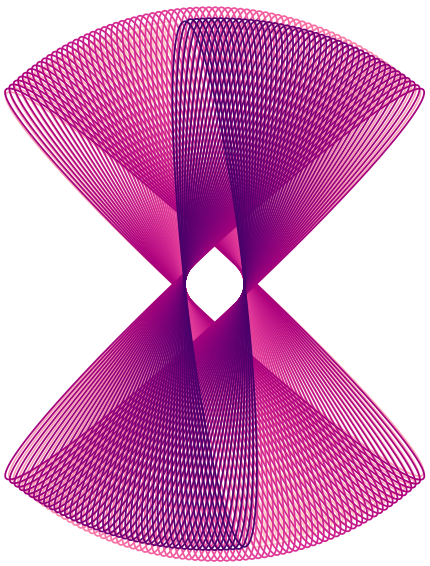
frequency and amplitude of the oscillation and its direction in state space with respect to the decision boundary.

The solution of the network largely depends on the angle between the readout vector $\mathbf{w}$ and the linear discriminant $\mathbf{w}_{LD}$. The latter is fully dependent on the input stimulus vectors and the noise covariance matrix (Equation 2.5). Figure 4.6 illustrates the dependence. There are only small changes in the eigenvalues of the network and apart from the orientation, the network covariances also do not differ much. The same is true

for most losses, once the stimulus vectors are at some small angle from each other, at least when their length is adapted so as to keep $\mathbf{w}_{LD}$ a unit vector.



(a) Eigenvalues of the learned connectivity matrix and corresponding losses as a function of $\mathbf{u}_2$ varying (with adapted stimulus vector length, so as to keep $\mathbf{w}_{LD}$ at unit length), only complex eigenvalues are shown. Except where the input stimulus vectors are equal, the eigenvalues are complex conjugates. It suffices to consider λ_1. Though not exactly equal, they remain very similar.



(b) Covariance ellipses of the corresponding networks, that is, only those for complex eigenvalues are shown. The small changes in eigenvalues is mirrored in the small differences in the axis lengths of the covariance matrices. Apart from this, they differ only in their angle about the origin.

Figure 4.6: Eigenvalues and covariance of 150 networks with the angles between the input stimulus vectors changing, the angle between $\mathbf{w}$ and $\mathbf{w}_{LD}$ is kept constant.

4.2 Binary Cumulative Loss Function

4.2.1 Behaviour across Time

Compared to the behaviour with the binary rule alone, the binary cumulative loss function results in the optimal loss occurring at an earlier delay. This makes sense since earlier delays are now also optimised for. Interestingly, this is not the only effect. Non-normal networks appear to be able to compete with oscillatory networks when optimising for more delays. In the visualisation of the loss (Figures 4.7B), the non-normal networks can be identified by their strictly monotonic behaviour after some initial displacement from the value they converge to. Non-normal and oscillatory

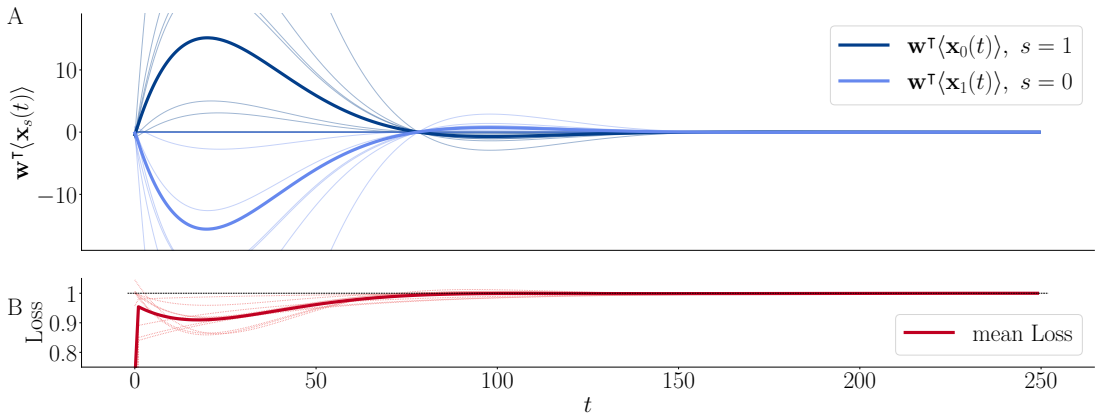


Figure 4.7: A: Projections of mean activity vectors for the same parameters as in 4.4. Notably, the point of greatest mean separation has shifted towards an earlier delay. Furthermore, greater damping occurs so that activity converges after $t = 150$. B: Corresponding losses. Monotonic behaviour shortly after 0 is that of non-normal networks.

networks appear comparable in terms of the scale of the loss.

Furthermore, the occurring oscillations are much more strongly damped. While the amplification during the first half of the oscillation has a comparable magnitude to those found previously (Figure 4.4A), it is much reduced during the second half where vectors are labelled incorrectly. Damping is so strong that after one full oscillation the projections are close to zero, a marked change from the continued oscillations when optimised for just one delay time (Figure 4.4A). This is accompanied by a different shape of the projection curves in Figure 4.7A.

4.2.2 Damped Oscillations Closely Approximate Non-normal Dynamics

The projections in Figure 4.7 are far less symmetric than in Figure 4.4. After reaching the point of maximal mean separation they flatten significantly and thus prolong the period in which the stimulus vectors are still labelled correctly. Notably, this is not an artefact of the averaging process, since it also occurs in the individual curves for different networks as indicated by the faint lines. With early strong amplification and subsequent slowed decrease, the oscillatory networks approximate standard non-normal networks even better in their behaviour across time than those in Figure 4.4. Though some oscillation remains (see Figure 4.7 around $t = 100$), the projection curve is approaching the behaviour exhibited by non-normal networks, although with much stronger amplification and a longer period of increasing amplification (compare Figure 4.9).

Figure 4.8 shows why this is the case. In non-normal networks trajectories converge onto the slow mode and return towards the origin along that slow direction. Oscillatory networks mimic this behaviour by moving the trajectories similarly close to the real projection of the complex eigenvectors. When optimising for a single delay, the

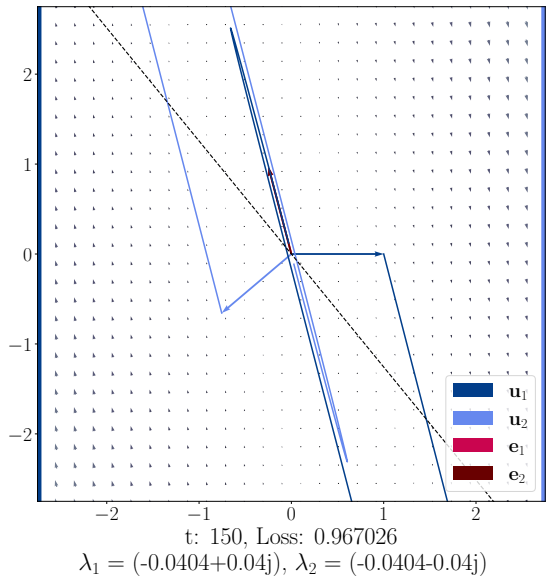


Figure 4.8: One of the oscillatory networks from Figure 4.7 at a delay of $t = 150$. Here the trajectories approach the direction determined by the real projection of the eigenvectors much faster, in the second half of the first oscillation the trajectories are already almost entirely aligned with the real projection of the eigenvector. Compare Figures 4.5c, 4.2d for less damped oscillatory networks where this alignment is not taking place.

trajectories in oscillatory networks are also oriented in the direction of this mode, but shifted away from the origin (see Figure 4.5c). Here, they approach the origin much faster and align almost completely with this mode. This alignment is already almost achieved in the second half of the first oscillation (see also C.1).

Both real and imaginary part of the eigenvalues are close to zero. Thus, in the first half of the first oscillation the slowness of the oscillation maintains the amplification in the correct direction. Subsequently, the trajectories align strongly with the slow mode to avoid strong amplification in the incorrect direction. Since the trajectories are moving slowly along the mode, the turning point of the oscillation back towards the origin occurs while they are still close to the origin. The greater the alignment, the less amplification occurs.

4.3 Binary Cumulative Loss Function with Exponential Weighting

4.3.1 Dynamics and Behaviour across Time

The most striking result is that even with a very small decay constant the solutions change dramatically from the previously observed oscillations. With exponential weighting, the optimal networks are non-normal and amplifying for all choices of parameters. Indeed, the most amplifying direction is the one defined in [36] (Figure C.4) and it orthogonal to fast mode (see C.3). This makes sense, since amplification occurs along the fast mode. However, the amplification is much weaker than in oscillatory networks (compare Figures 4.4A, 4.7A, 4.9A).

The lowest loss the network can reliably achieve (i.e. the loss is not just very low at a single time as in Figure 4.7) is slightly better than when optimising for a single delay or unweighted delays (compare Figures 4.4C, 4.7B, 4.9B). Nevertheless, the losses at time $t_d = 50$, the last evaluation point of the loss function, remain comparable to those

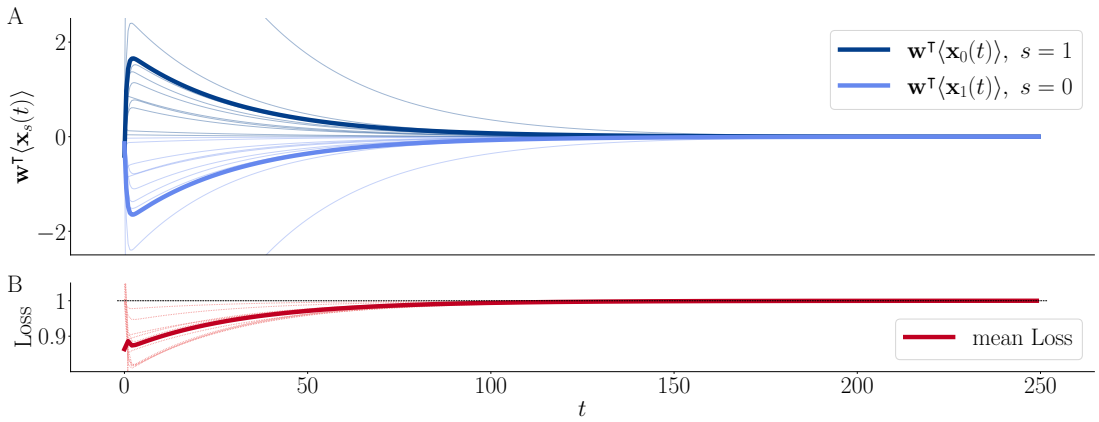


Figure 4.9: **A**: Evaluation points weighted according to exponential decay with decay constant 0.01. All resulting networks are non-normal. **B**: The corresponding losses are better than for the previous loss functions in Figures 4.4, 4.7 for shorter delays but increase more quickly and monotonically afterwards, subsequently converging to 1.

obtained previously. Since there are no oscillations, the performance never drops below chance level. That is, the losses converge to 1 from below (see Figure 4.9B).

Since all the resulting networks are non-normal, it is natural to ask how these solutions can be characterised in a more fine-grained manner. To this end, the connectivity matrices can be decomposed in several ways, revealing different aspects of the solution.

4.3.2 Degree of Non-Normality and Amplification

The eigendecomposition is useful to characterise the self-interacting modes of the network. Figure 4.10A shows the angle between the unit length right eigenvectors. This angle is in all cases close to either 0 or π, i.e. indicating strong non-normality. Moreover, the eigenvectors further approach collinearity as the input linear discriminant and the readout align. It was verified via the results for different parameters that the alignment of $\mathbf{w}$ and $\mathbf{w}_{LD}$ is indeed the cause of this behaviour (see Figure C.5).

The increase in non-normality is reflected in the amount of amplification the network produces. In normal networks, there is no amplification and the mean activity for some stimulus is moved orthogonally directly onto the slow mode. With increasing non-normality, the convergence onto the slow mode happens in an increasingly non-orthogonal way. That is, the trajectories are moved towards the slow mode at a more acute angle. This is done by moving away from the origin, resulting in amplification.

This interpretation is supported by the results of the singular value decomposition of the propagator of the network. Figure 4.10D shows a marked increase in one of the singular values for the same parameters where non-normality increased, a sign that the network is more amplifying [7]. Indeed, all of these non-normal networks are amplifying (they all have a singular value greater than unity), reinforcing the hypothesis that amplification is central to the solution.

Furthermore, as opposed to the singular values of the propagator, the changes of the

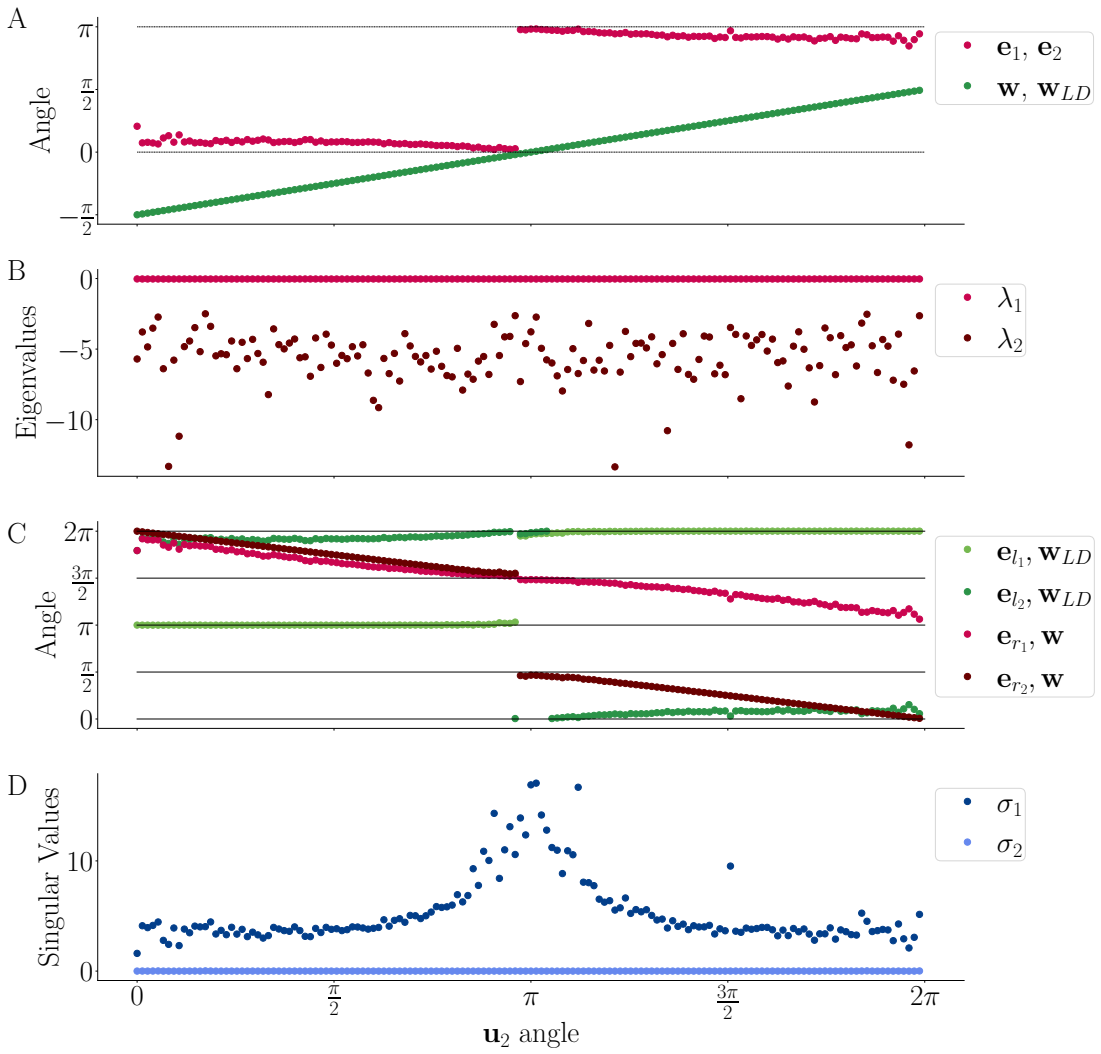


Figure 4.10: 150 networks optimised up to a delay of $t_d = 50$ for $\mathbf{u}_1 = [1,0]$, $\mathbf{w} = [-1,0]$ and $\mathbf{u}_2$ varying from 0 to 2π. **A**: Angle between right eigenvectors, measuring degree of non-normality, and angle between $\mathbf{w}$ and $\mathbf{w}_{LD}$. **B**: Eigenvalues of the connectivity matrix. The dominant eigenvalue also varies, but only on a very small scale and without obvious regularity. **C**: Projection of the left eigenvectors onto $\mathbf{w}_{LD}$ and of the right eigenvectors onto $\mathbf{w}$. **D**: Singular values of the propagator e^A of the network. Singular values greater than unity indicate amplification.

dominant eigenvalue are very small and do not show an obvious pattern. This is mirrored on a larger scale by the other eigenvalue, suggesting that small changes in the dominant eigenvalue could translate to larger changes in the other eigenvalue. The dominant eigenvalues are all extremely similar. In fact, the same approximate dominant eigenvalue is obtained for networks optimised for a maximum delay of 5 instead of 50.

The acute angle between the lines defined by left eigenvectors is always equal to the acute angle between the lines defined by right eigenvectors. Thus, the angle between the left eigenvectors is not displayed. Since the right eigenvectors are very close, so are the left eigenvectors.

4.3.3 Alignment of the Left Eigenvectors with the Input Linear Discriminant Allows Near Optimal Integration

The left eigenvectors are thought of as input analysing, whereas the right eigenvectors are considered output analysing. Recall that the linear discriminant represents the direction yielding the greatest difference between the projections $\mathbf{w}_{LD}^{\intercal}\mathbf{u}_1$ and $\mathbf{w}_{LD}^{\intercal}\mathbf{u}_2$. The dominant left eigenvector is almost perfectly aligned with the linear discriminant, albeit changing orientation once. The resulting good alignment of both left eigenvectors with the linear discriminant and the varying alignment of the right eigenvectors with the readout could suggest that the network is solving the problem more in terms of optimal loading than readout optimisation (Figure 4.10C). This is not entirely true, as Section 4.3.5 shows. Instead, optimal loading *requires* a close alignment of the left eigenvectors with $\mathbf{w}_{LD}$, since the optimal read-in direction $\mathbf{w}_{LD}$ is static and depends only on input parameters. There is more flexibility for the readout direction.

4.3.4 No Alignment of the Right Eigenvectors with the Readout

Since input that is integrated along a left eigenvector is amplified along the corresponding right eigenvector [10], one would expect the readout to be optimal if it was aligned with the right eigenvectors. This is not the case, as shown in Figure 4.11. In fact, the dominant right eigenvector is close to orthogonal to the optimal readout (the output linear discriminant) throughout. The right eigenvectors' angle to the actual readout varies between orthogonality and alignment (Figure 4.10C).

Given a fixed readout, a fixed relationship between right eigenvectors and readout is not even desirable, since it reduces the scope for network optimisation. While this lack of flexibility is a sufficient argument, it is worthwhile to dwell on what would happen in the case of alignment since it elucidates the general behaviour of non-normal networks.

The direction of the fast mode largely determines the flow field. The greatest flow is parallel to the fast mode, but flowing in opposite directions. Thus, if both stimulus vectors lie on the same side of the fast mode, the response trajectories move in the same direction. If they lie on different sides, their trajectories move in opposite directions.

Furthermore, independent of the stimulus vectors' locations, in a non-normal network the responses need to be moved in opposite directions for optimal performance. This is the case even if both are located on the correct side of the decision boundary: Were they moved in the same direction, one response must lie on the wrong side of the decision boundary upon convergence to the slow mode, unless the slow mode is aligned with the boundary. But then the projection of both responses onto the readout is 0 since the boundary is orthogonal to the readout.

The slow mode avoiding orthogonality to the readout can be observed in Figure 4.10C (bright red line). This happens for the same reason: with great amplification even a slow mode that is almost aligned with the decision boundary performs well, one that is perfectly aligned can never do so.

Consider now the following problematic cases of fixed alignment of readout and right eigenvectors:

1. *Slow mode aligned with readout, stimulus vectors on the same side of the decision boundary*: A large amount of amplification is required in order for one response to be moved across the boundary and remain there for a significant amount of time. This requires a small angle between fast and slow modes. On the other hand, the fast mode needs to be located between the stimulus vectors. This will not be possible in most cases, despite some possible adaptation of the fast mode.
2. *Fast mode aligned with readout*: The fast mode cannot be adapted to lie between stimulus vectors, and the network performance largely depends on the angles between stimulus vectors and readout.

Clearly, fixing this alignment does not lead to a satisfactory solution.

4.3.5 Evaluation of the Fixed Readout

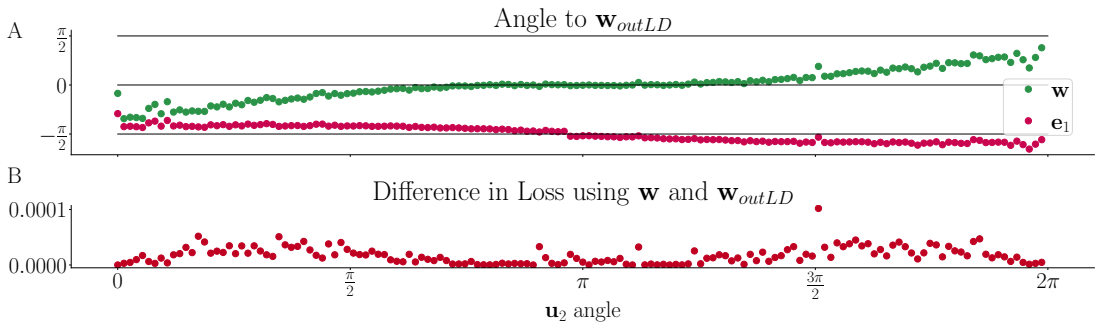


Figure 4.11: **A**: Angle between the output linear discriminant and $\mathbf{w}$, $\mathbf{e}_1$. **B**: Difference between the losses obtained with $\mathbf{w}$ and $\mathbf{w}_{outLD}$ at $t_d = 50$.

It is possible to assess how well the networks are adapted to the fixed readout by calculating the optimal readout, the output linear discriminant $\mathbf{w}_{outLD} = \Sigma^{-1}_{\mathbf{x}(t)}(\langle \mathbf{x}_1(t) \rangle - \langle \mathbf{x}_2(t) \rangle)$. Figure 4.11 shows the angle between the $\mathbf{w}_{outLD}$ and the actual readout $\mathbf{w}$. Thus, for a little less than half the parameter space, the networks adapted so that the fixed readout direction $\mathbf{w}$ is optimal.

The dependence of the output linear discriminant on delay time is insignificant. For networks trained for delays up to $t_d = 50$, $\mathbf{w}_{outLD}$ remains stable after two time steps.

Figure 4.11 explains the non-normality of the optimised networks, but in particular the strong non-normality where $\mathbf{w}$ and $\mathbf{w}_{LD}$ are aligned. If it was becoming more normal at this point of alignment, it could align both the left eigenvectors with the input linear discriminant and the right eigenvectors with the output linear discriminant (which in that case would be orthogonal to $\mathbf{w}_{LD}$). This would achieve the least information loss during integration and readout *if the readout was also aligned with the output linear discriminant*. But this is not the case, since $\mathbf{w}$ and $\mathbf{w}_{LD}$ are aligned.

Strong non-normality on the other hand effects a shift in the optimal readout $\mathbf{w}_{outLD}$ to align closer with the actual readout. This can be done since contrary to $\mathbf{w}_{LD}$, $\mathbf{w}_{outLD}$ depends on the network dynamics.

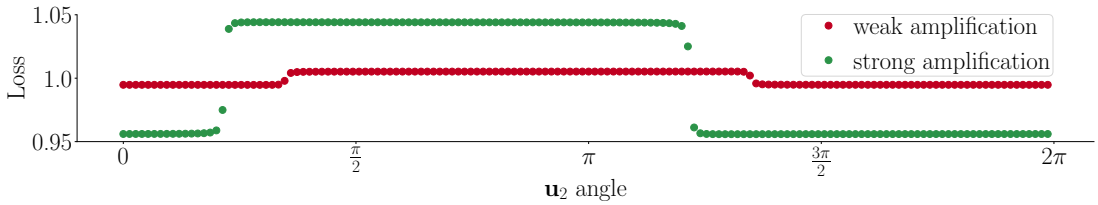


Figure 4.12: Comparison of the losses for a strongly and a weakly amplifying network on the same stimuli, $\mathbf{w}$ varies from 0 to 2π.

Thus, while the fixed readout may not be optimal for the network, the network adapts so that the readout becomes near optimal for the changed dynamics. Figure 4.11 shows that the differences in loss between the actual and the optimal readout is very small, even where they are not aligned. While this is shown for $t_d = 50$, similarly small differences (< 0.00025) are observed for early delays after the convergence of $\mathbf{w}_{outLD}$ as well.

Figure 4.12 shows a comparison of a strongly and a weakly amplifying network for the same stimuli. The readout is varied around the unit circle. The losses remain similar on either a high or a low level, each for half of the parameter range. However, the losses for strongly amplifying networks depart further from chance level. Thus, strong non-normal networks can achieve better performance by adapting so that their optimal readout falls anywhere within the well-performing range of readouts. This explains the near optimal performance of the fixed readout even when it is not well aligned with the optimal readout.

The above analysis reveals how non-normal networks are optimised for the task. Optimal information integration needs to be achieved by aligning the left eigenvectors closely with the input linear discriminant. However, one of them is not perfectly aligned, allowing some freedom in the exact dynamics. Thus, the dynamics can be adapted so that the optimal readout of the resulting network lies in the range of very similarly well performing readouts. In this way, the networks achieve a balance between read-in and read-out accuracy.

Chapter 5

Discussion

5.1 Limitations of the Approach

5.1.1 Limitations of the Optimisation Process

Even with the exponentially weighted cumulative decision rule, the optimisation method is constrained to a specific time window. However, the exponential decay ensures that a long range of delays can be covered with very long time windows, where late delays will be discounted almost completely. Nevertheless, integrating over the range of delays $[0, \infty)$ would be preferable.

Furthermore, the optimiser used also places constraints on the found solutions. While many different initialisations often lead to final losses which are very close together, these local minima are not necessarily close in the space of connectivity matrices. Other local minima with an even better performance are thus eminently possible. This is a known phenomenon, which has inspired investigation into methods of regularisation in order to obtain the biologically most plausible networks through training [34].

Observations showed that for specific parameters the optimiser can find different types of solutions, but with different performance. For different initialisations, one type of solution was consistently superior to others (oscillatory with the one-delay binary rule, non-normal with the exponentially weighted cumulative binary rule). This indicates that this type of solution, though not necessary any of the particular local minima, is indeed optimal for the problem.

Nevertheless, it is possible that the optimal solution is of a different type but simply difficult for the optimiser to find. However, if such a solution exists, it is likely unstable in the sense that a small perturbation of the weights vastly decreases performance. This is because the optimiser uses the analytical first derivative to find solutions, so if the solution has some regularity it is likely that it would be found by the optimiser.

However, the figures in Section 4.3.2 indicate that for some parameters the optimal solution is easier to find than for others. Where in Figure 4.10 the projections vary continuously for some parameter ranges it is reasonable to assume that these indeed represent an optimal solution. In parameter ranges exhibiting a large amount of variation,

the algorithm likely failed to find the optimal solution. Nevertheless, the overall regularity of the solutions allows to infer properties of the optimal solution even though the algorithm may not have found it.

5.1.2 Limitations of the Model

In terms of the neuroscientific interpretation of the results, the two greatest limitations are the abstraction and dimensionality of the networks. The abstraction to linear models is not necessarily a hindrance, since it has been often observed that the behaviour of linear networks translates well to nonlinear networks [36]. However, in order to compare to previous results or experimental data, it may be beneficial to introduce some structure to the connectivity matrix. It is unclear if restricting it to represent excitatory and inhibitory populations would also restrict the type of solution. Low dimensionality means that some aspects of the solution which may be important could not be analysed properly. These include:

1. The Schur basis of the network which in two dimensions consists only of left and right eigenvectors. Even just in three dimensions it could add new insights by highlighting an important direction in state space, since there is an infinite number of vectors orthogonal to any one direction.

2. The oscillations found with the early decision rules and loss functions are by nature two-dimensional. This is because they are defined by complex conjugate eigenvalues and eigenvectors. A short exploration has shown that with the single delay binary rule oscillations also occur in higher dimensional networks. Approximately half of the eigenvectors tend to be non-real. It is known that non-normality does not necessarily restrict to two-dimensional subspaces. It remains to be seen if these oscillations in higher dimensions are decoupled or if they interact across subspaces.

3. The memory capacity of a network is usually judged in relation to the network's size. To be able to judge if the oscillatory and non-normal solutions perform differently or are able to integrate a greater variety of inputs than attractor networks, one would need to consider larger networks to see how well each type of network scales. So far, only a two stimulus discrimination task has been considered. To see how many patterns a network can be trained for effectively, more stimuli would need to be used for training. However, with the present binary rule this would complicate stimulus labelling. Furthermore, with the above model, attractor networks and oscillatory networks trained for two stimuli generalise equally to other directions. It is not clear if the same would be the case in higher dimensions.

Furthermore, even in two dimensions the stimulus labels influence the results to a strong degree. Where highly non-normal solutions are required to move responses across the decision boundary, much simpler dynamics are needed if the same stimuli exchange labels so that they lie on the correct side of the decision boundary in the first place.

5.2 Summary of the Findings

The findings suggest that for different kinds of tasks marked by the delay after which stimulus reconstruction is tested, a different kind of dynamics is optimal. In general, the network solves the problem by creating dynamics that both adjust the direction of the response in state space and create a separation of the mean responses to optimally decode the network activity and discount the noise. Crucial for both these effects is the network's ability to amplify responses. Amplification is caused by either strong (damped) oscillations or strongly non-normal activity. Slow oscillations have a similar effect to that of non-normal networks, but are trading good performance at a single, late delay for worse performance at earlier delays.

Among non-normal networks, there are no qualitative differences between networks optimised for different parameters. The good alignment of the left eigenvectors and the direction of least variance with the input linear discriminant serve to minimise the amount of information loss and disturbance. Strong non-normal dynamics adjust the network response so that the optimal readout lies in the range of very similarly well performing readouts. Strong amplification serves to improve performance for this range of good readouts. Using adaptation to both stimuli and readout, non-normal networks achieve a balance between optimal read-in and read-out accuracy.

The findings support the hypothesis that non-normal dynamics and the resulting selective amplification play an important role in the maintenance of activity which is largely presumed to be the underlying mechanism of working memory. Indeed, these non-normal dynamics are essential to yield optimal performance in the presence of a fixed readout.

5.3 Comparison to Previous Work

The results of the four different ways of optimisation represent a gradual move towards more realistic and plausible models. Over the course of the adaptations of the loss function for the binary decision rule the networks successively approach the type of non-normal networks commonly discussed in the literature on working memory. Oscillatory networks may not constitute a plausible solution since they do not conform to the common assumption that memory performance is best directly after the stimulus is applied and declines afterwards [42]. However, non-normal networks also exhibit a later onset and increase of activity, a behaviour also observed experimentally [39].

As noted by Goldman et al. [21], highly non-normal networks have many non-real ε-pseudoeigenvalues [38]. These are the eigenvalues of matrices a small perturbation away from the actual connectivity matrix and characterise the behaviour of the transient response. This reinforces the strong link between highly non-normal and oscillatory networks. Not only can their effect on the stimulus vectors be similar, they are close in the parameter space of connectivity matrices. In [21], hidden feedforward networks are compared to a class of networks with complex eigenvalues. However, the latter are not considered as potential solutions to a working memory task and their dynamics are not analysed further.

Since the literature is largely concentrating on the analysis of normal versus non-normal networks, in the following, unless otherwise specified, reference will be made to the results of the cumulative binary rule with exponential weights.

In [20], it was proved that only non-normal networks can have an extensive memory capacity. In this project, capacity itself is not considered, since the network analysed is small and capacity is usually a property of interest when considering scaling issues. However, the results of this project also suggest that the optimally performing networks are non-normal.

The resulting networks for any of the loss functions are not fully functionally feed-forward networks as considered by Goldman et al. [21], since they neither have both eigenvalues equal to -1 nor overlapping eigenmodes. Instead, they constitute functionally mixed networks.

Chadwick et al. [10] analysed a similar two stimulus discrimination task to the one considered in this project. There, the stimuli were a consistent, noisy stream of inputs with different means. The results were similar to those by Goldman et al. [21] in that the optimal networks for the task were found to be strongly non-normal with equal eigenvalues and overlapping left eigenvectors. Additionally, the left eigenvectors were aligned with the input linear discriminant and thus the direction of highest SNR. Thus, these networks achieved optimal input integration. They found that the response information of the network scales linearly with the number of neurons and the time constant of the network. Hence, equal eigenvalues of zero maximise the response information. In this project, the same alignment of left eigenvectors is observed: due to the strong non-normality of the network, the left eigenvectors are at a very acute angle. One of them is aligned near perfectly with the linear discriminant. However, a marked difference is the very robust observation in this project that the time constants of these (left) eigenmodes are far from equal. One remains close to zero, thus having a long timescale, the other is more negative.

These differing time constants should not be optimal according to the analysis of Chadwick et al. [10], but they do agree with the findings of Stroud et al. [36]. Stroud et al. defined a 'linear attractor network' of the form $\dot{\mathbf{x}} = A\mathbf{x}$ (compare Equation 2.1) by requiring that the connectivity matrix A has a single eigenvalue $0 > \lambda_1 \gg -1$, all others $\lambda_i < \lambda_1$. They constructed and subsequently analysed these networks to find the task performance for different input directions, including the most amplifying direction.

Proceeding in the opposite direction and optimising networks for a working memory task, the analysis above shows that for the exponentially weighted binary rule the same type of network emerges. The non-normal networks obtained through the optimisation exhibit the same structure of the eigenvalues.

For the analysis of experimental data, Stroud et al. [36] fitted linear networks to experimental data and adapted the readout to be optimal for the network. They variously optimised the readout for a single or several fixed delays, obtaining similar results. This is a marked difference in approach to training the network for a fixed readout as presented here. Nevertheless, the results agree when the network is optimised for several fixed, weighted delays.

Furthermore, the definition of the most amplifying mode in [36] for 'linear attractor networks', appears to extend to oscillatory networks as well. Despite the fact that in the calculation of the most amplifying direction the input noise is neglected, in the networks with input noise that were considered, this direction was indeed the most amplifying. This remained true for the results of all versions of the binary rule, non-normal and oscillatory alike.

The networks examined in [26], where balanced amplification was proposed were designed to be only weakly non-normal. The ratio of feedforward to total connectivity in their networks is stated to be $f^A = 0.55$, much less than for any of the networks obtained in this project. Much of the analysis of feedforward activity from a mode with small differences in excitatory and inhibitory activity to one where their sum is increased does not apply here, since the networks considered in this project do not have an excitatory-inhibitory structure. However, their main analysis of the two-step feedforward dynamics still applies: activation along a second Schur mode (the dominant left eigenvector) causes activation along the fast mode (the non-dominant right eigenvector). This mechanism causes amplification.

Furthermore, while it has been suggested that an improvement in stimulus representation and not a change in readout may be the most effective way to integrate new input (in an attention task [32]), in the task considered here, a fixed readout appears to put some constraints on the network. This is because the performance depends largely on the training parameters, i.e. the angle between input linear discriminant and readout. While the trained networks may be optimal for a task with a fixed readout, it is likely that the fixed readout is not optimal for the task. Nevertheless, non-normal networks can approximate optimal performance rather closely.

In [27] the authors conclude that a fixed readout can only be optimal for stable, that is, attractor models. For non-normal networks it is dependent on the delay (see the equation of the output linear discriminant in Section 4.3.5). However, the fact that the learned networks were all non-normal despite a fixed readout suggests that non-normal networks overcome this issue. Indeed, the results of this project show that the optimal readout of the obtained strongly non-normal networks does not significantly depend on delay time and remains stable after a short delay. Thus, the non-normal networks obtained here may be particularly well adapted to a fixed readout.

Thus, the results unify elements of those of Stroud et al. [36] and Chadwick et al. [10]. The eigenvalues of the solution agree with those observed in Stroud et al. [36], while the alignment of eigenvectors agrees with Chadwick et al. [10]. The amplification mechanism is very similar to that described in [26] for excitatory-inhibitory networks. Combining elements of previous theories about optimal input integration and selective amplification yields a mechanism adapted specifically for the fixed readout, two stimulus discrimination task.

5.4 Future Work

There remain some unanswered questions about the behaviour of the non-normal networks in Section 4.3.2 and Figure C.5.

- Given that it does not appear to have an effect on the flow field when the slow mode changes sign, why does do so and subsequently remains consistent?
- Why are the dominant eigenvalues so similar for different delay times?
- How could the effect of the stimulus labelling be mitigated?
- What is the effect of a change in noise covariance?

An avenue for future work will be to extend the present analysis to larger networks. The regularities described in Section 4.3 could form the basis for predictions about properties of non-normal networks based on the relationship between input linear discriminant and readout. These could be tested against experimental data. To this end, it would be useful to also consider less abstract models, for instance constraining them to an excitatory-inhibitory structure. Furthermore, a comparison of the dynamics obtained using a fixed readout versus a variable readout may be an interesting direction to consider further.

Bibliography

[1] Awad Al-Mohy and Nicholas Higham. Computing the fréchet derivative of the matrix exponential, with an application to condition number estimation. *SIAM J. Matrix Analysis Applications*, 30:1639–1657, 01 2008.

[2] Alan Baddeley. Working memory. *Science*, 255(5044):556–559, 1992.

[3] Pierre Baldi and Kurt Hornik. Neural networks and principal component analysis: Learning from examples without local minima. *Neural Networks*, 2(1):53–58, 1989.

[4] Omri Barak. Recurrent neural networks as versatile tools of neuroscience research. *Current Opinion in Neurobiology*, 46:1–6, 2017. Computational Neuroscience.

[5] Richard H. Bartels and G. W. Stewart. Solution of the matrix equation ax + xb = c. *Commun. ACM*, 15:820–826, 1972.

[6] Daniel Birman and Justin L. Gardner. A flexible readout mechanism of human sensory representations. *Nature Communications*, 10:3500, 2019.

[7] Giulio Bondanelli and Srdjan Ostojic. Coding with transient trajectories in recurrent neural networks. *PLOS Computational Biology*, 16(2):1–36, 02 2020.

[8] Carlos D Brody, Adrián Hernández, Antonio Zainos, and Ranulfo Romo. Timing and neural encoding of somatosensory parametric working memory in macaque prefrontal cortex. *Cerebral cortex*, 13(11):1196–1207, 2003.

[9] Sean Edward Cavanagh, John Towers, Joni D. Wallis, Laurence Tudor Hunt, and Steven Wayne Kennerley. Reconciling persistent and dynamic hypotheses of working memory coding in prefrontal cortex. *Nature Communications*, 9, 2018.

[10] Angus Chadwick, Adil Khan, Jasper Poort, Antonin Blot, Sonja Hofer, Thomas Mrsic-Flogel, and Maneesh Sahani. Learning shapes cortical dynamics to enhance integration of relevant sensory input. 2021.

[11] Biswa Nath Datta. *Numerical Methods for Linear Control Systems*. Academic Press, San Diego, 2004.

[12] Peter Dayan and L. F. Abbott. Theoretical neuroscience: Computational and mathematical modeling of neural systems. 2001.

[13] Sam A. Deadwyler and Robert E. Hampson. Temporal coupling between subicular and hippocampal neurons underlies retention of trial-specific events. *Behavioural Brain Research*, 174:272–280, 2006.

[14] Daniel Durstewitz and J. Seamans. Durstewitz d, seamans jk. beyond bistability: biophysics and temporal dynamics of working memory. neuroscience 139: 119-133. *Neuroscience*, 139:119–33, 05 2006.

[15] Daniel Fischer. Gaussian integrals over a half-space. `https://math.stackexchange.com/questions/556977/gaussian-integrals-over-a-half-space?rq=1`, Nov 2013, accessed 09 Jan 2022.

[16] Ken-ichi Funahashi and Yuichi Nakamura. Approximation of dynamical systems by continuous time recurrent neural networks. *Neural Networks*, 6(6):801–806, 1993.

[17] S. Funahashi, C. J. Bruce, and P. S. Goldman-Rakic. Mnemonic coding of visual space in the monkey's dorsolateral prefrontal cortex. *Journal of Neurophysiology*, 61(2):331–349, 1989.

[18] J.M. Fuster. Unit activity in prefrontal cortex during delayed-response performance: neuronal correlates of transient memory. *Journal of Neurophysiology*, 36(1):61–78, 1973.

[19] Joaquin M. Fuster and Garrett E. Alexander. Neuron activity related to short-term memory. *Science*, 173(3997):652–654, 1971.

[20] Surya Ganguli, Dongsung Huh, and Haim Sompolinsky. Memory traces in dynamical systems. *Proceedings of the National Academy of Sciences*, 105(48):18970–18975, 2008.

[21] Mark S. Goldman. Memory without feedback in a neural network. *Neuron*, 61(4):621–634, 2009.

[22] P.S. Goldman-Rakic. Cellular basis of working memory. *Neuron*, 14(3):477–485, 1995.

[23] Guillaume Hennequin, Tim P. Vogels, and Wulfram Gerstner. Optimal control of transient dynamics in balanced networks supports generation of complex movements. *Neuron*, 82(6):1394–1406, 2014.

[24] Ta-Chu Kao and Guillaume Hennequin. Neuroscience out of control: control-theoretic perspectives on neural circuit dynamics. *Current Opinion in Neurobiology*, 58:122–129, 2019.

[25] Valerio Mante, David Sussillo, Krishna V. Shenoy, and William T. Newsome. Context-dependent computation by recurrent dynamics in prefrontal cortex. *Nature*, 503:78–84, 2013.

[26] Brendan K. Murphy and Kenneth D. Miller. Balanced amplification: A new mechanism of selective amplification of neural activity patterns. *Neuron*, 61(4):635–648, 2009.

[27] John D. Murray, Alberto Bernacchia, Nicholas A. Roy, Christos Constantinidis, Ranulfo Romo, and Xiao-Jing Wang. Stable population coding for working memory coexists with heterogeneous neural dynamics in prefrontal cortex. *Proceedings of the National Academy of Sciences*, 114(2):394–399, 2017.

[28] A. M. Ni, D. A. Ruff, J. J. Alberts, J. Symmonds, and M. R. Cohen. Learning and attention reveal a general relationship between population activity and behavior. *Science*, 359(6374):463–465, 2018.

[29] Matthew F. Panichello, Brian DePasquale, Jonathan W. Pillow, and Timothy J. Buschman. Error-correcting dynamics in visual working memory. *Nature Communications*, 10:3366, 2019.

[30] Eva Pastalkova, Vladimir Itskov, Asohan Amarasingham, and György Buzsaki. Internally generated cell assembly sequences in the rat hippocampus. *Science*, 321(5894):1322–1327, 2008.

[31] Mitchell R. Riley and Christos Constantinidis. Role of prefrontal persistent activity in working memory. *Frontiers in Systems Neuroscience*, 9, 2016.

[32] Douglas A. Ruff and Marlene R. Cohen. Simultaneous multi-area recordings suggest that attention improves performance by reshaping stimulus representations. *Nature Neuroscience*, 22:1669–1676, 2019.

[33] Andrew M. Saxe, James L. McClelland, and Surya Ganguli. Exact solutions to the nonlinear dynamics of learning in deep linear neural networks. 2014.

[34] H. Francis Song, Guangyu R. Yang, and Xiao-Jing Wang. Training excitatory-inhibitory recurrent neural networks for cognitive tasks: A simple and flexible framework. *PLOS Computational Biology*, 12(2):1–30, 02 2016.

[35] Eelke Spaak, Kei Watanabe, Shintaro Funahashi, and Mark G. Stokes. Stable and dynamic coding for working memory in primate prefrontal cortex. *The Journal of Neuroscience*, 37:6503 – 6516, 2017.

[36] Jake P. Stroud, Kei Watanabe, Takafumi Suzuki, Mark G. Stokes, and Máté Lengyel. Optimal information loading into working memory in prefrontal cortex. 2021.

[37] David Sussillo. Neural circuits as computational dynamical systems. *Current Opinion in Neurobiology*, 25:156–163, 2014. Theoretical and computational neuroscience.

[38] Lloyd N. Trefethen and Mark Embree. *Spectra and Pseudospectra: The Behavior of Nonnormal Matrices and Operators*. Princeton University Press, 2005.

[39] Kei Watanabe and Shintaro Funahashi. Neural mechanisms of dual-task interference and cognitive capacity limitation in the prefrontal cortex. *Nature Neuroscience*, 17:601–611, 2014.

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

[41] Jacob L. Yates, Leor N. Katz, Aaron J. Levi, Jonathan W. Pillow, and Alexander C. Huk. A simple linear readout of MT supports motion direction-discrimination performance. *Journal of Neurophysiology*, 123(2):682–694, 02 2020.

[42] Shaowu Zhang, Fiola Bock, Aung Si, Juergen Tautz, and Mandyam V. Srinivasan. Visual working memory in decision making by honey bees. *Proceedings of the National Academy of Sciences*, 102(14):5250–5255, 2005.

Appendix A

Continuous Decision Rule - A Flawed Representation of the Problem

A.1 A First Attempt

The continuous decision rule is very similar to the binary rule, however, it lacks the Heaviside step function: $d_{cont}(\mathbf{x}(t_d)) = \mathbf{w}^\intercal \mathbf{x}(t_d)$. Instead, it is simply the projection onto the linear readout.

A.2 Dynamics of the Solutions

When all networks are initialised with the same matrix, in this case the negative identity $A = -\mathbf{1}$, the optimal dynamics found with the simple gradient descent algorithm differ between attractor, non-normal and oscillatory dynamics, each characterised by the angle between the eigenvectors of the resulting connectivity matrix.

Attractor networks have real, approximately orthogonal eigenvectors, non-normal networks have real, non-orthogonal eigenvectors, and oscillatory networks have a set of complex conjugate eigenvectors. Figure 2.2 illustrates the different kinds of dynamics. There, they were obtained by keeping the stimulus vectors stable and varying only the angle ϕ of the readout vector $\mathbf{w}$. This suggests that all else being equal, the relationship between the angles of the vectors $\mathbf{u}_1, \mathbf{u}_2$ and $\mathbf{w}$ determines the behaviour of the network.

A.3 Magnitude of the Vectors and Arbitrary Choice of Stimulus Labels s

The major disadvantage of using the continuous decision rule is that both the magnitude of input and weight vectors and the magnitude of the arbitrarily chosen labels s have a large impact on the behaviour of the network. This is caused by the way the loss function is calculated. Since it depends on the difference $(s - d_{cont}(\mathbf{x}(t_d)))^2$, the network will attempt to change its state and thus the input representation in such a way as to minimise

this difference. Thus, when s is chosen to be large compared to the input vectors' magnitude, the network needs more time and more extreme dynamics to minimise this difference than when s is smaller. Given that s is simply an arbitrary label attached to the input vectors, using this loss function does not reasonably quantify the network's performance.

An extreme case illustrates this well: letting the stimulus labels be -1 and 1 and the output of the continuous decision function at some time t_d be -0.01 and 0.5 respectively, it is reasonable to interpret this as the network having solved the input discrimination task after a delay of t_d. That is, it was possible to reconstruct from the network state, which input had which label, simply by considering the signs of the outputs of the continuous decision function. However, the network will attempt to change its activity further if that results in reducing the squared loss. This may be of interest if there is a larger set of labels, or even continuous labels, but in this simple task, such adaptation is unnecessary and leads to extreme dynamics.

Appendix B

Mathematical Derivations[1]

B.1 Network Model

The dynamics of the network under consideration are described by the differential equation

$$\dot{\mathbf{x}} = A\mathbf{x} + \mathbf{u}(s,t) \tag{B.1}$$

which has as parameters the stable connectivity matrix A and the stimulus-dependent input $\mathbf{u}(s,t)$. The network state at time t is represented by $\mathbf{x}(t)$, the stimulus is labelled s.. The network will be optimised by adapting the connectivity matrix A based on the input. The input

$$\mathbf{u}(s,t) = \mathbf{u}(s)\delta(t - t_s) + \mathbf{n}(t) \tag{B.2}$$

in turn consists of a stimulus-dependent vector indicating the direction in state space the stimulus drives the network into which only appears once at time t_s, and temporally uncorrelated Gaussian white noise $\mathbf{n} \sim \mathcal{N}(0, \Sigma_{\mathbf{n}})$.

Equation B.1 can be solved by using an integrating factor to arrive at

$$\mathbf{x}(t) = e^{A(t-t_0)}\mathbf{x}_0 + \int_{t_0}^{t} e^{A(t-\tau)}\mathbf{u}(s,\tau)d\tau \tag{B.3}$$

for some initial condition $\mathbf{x}_0$ at time t_0, assuming that $\mathbf{x}_0 \sim \mathcal{N}(0, \Sigma_{\mathbf{x}_0})$ is independent of the input noise $\mathbf{n}$.

Assuming that the network is driven into a stationary state with respect to the input noise before the input arrives, that is $t_0 \to \infty$, this initial condition may be neglected. Considering this and inserting Equation B.2 describing the input into Equation B.3 gives

$$\mathbf{x}(t) = e^{A(t-t_s)}\mathbf{u}(s) + \int_{-\infty}^{t} e^{A(t-\tau)}\mathbf{n}(\tau)d\tau. \tag{B.4}$$

Stimulus-conditioned mean and covariance are then

$$\langle \mathbf{x}(t) \rangle = e^{A(t-t_s)}\mathbf{u}(s) \tag{B.5}$$

[1] By Angus Chadwick

and

$$\Sigma_{\mathbf{x}(t)} = \langle (\mathbf{x}(t) - \langle \mathbf{x}(t) \rangle)(\mathbf{x}(t) - \langle \mathbf{x}(t) \rangle)^\intercal \rangle = \int_{-\infty}^{t} e^{A(t-\tau)} \Sigma_{\mathbf{n}} e^{A^\intercal (t-\tau)} d\tau = \int_{0}^{\infty} e^{A\tau} \Sigma_{\mathbf{n}} e^{A^\intercal \tau} d\tau. \tag{B.6}$$

B.2 Derivatives for Network Optimisation

To perform gradient descent on the square loss function, it will be necessary to compute the derivatives of both mean $\langle \mathbf{x}(t) \rangle$ and covariance $\Sigma_{\mathbf{x}(t)}$ of the network activity. The derivative of the mean $\frac{d\langle \mathbf{x}(t) \rangle}{dA_{ij}} = \frac{de^{A(t-t_s)}}{dA_{ij}} \mathbf{u}(s)$ can be calculated numerically using a scaling and squaring algorithm [1], implemented e.g. in `linalg.expm_frechet`. The derivative of the covariance requires the solution of two Lyapunov equations which can be solved by the Bartels-Stewart algorithm [5] using library routines like `scipy.linalg.solve_continuous_lyapunov`. The first Lyapunov equation is the result of integrating Equation B.6 by parts to arrive at

$$A\Sigma_{\mathbf{x}(t)} + \Sigma_{\mathbf{x}(t)} A^\intercal + \Sigma_{\mathbf{n}} = 0. \tag{B.7}$$

In this way, $\Sigma_{\mathbf{x}(t)}$ can be calculated from A and $\Sigma_{\mathbf{n}}$. Note that since A is stable and $\Sigma_{\mathbf{n}}$ is symmetric positive definite by virtue of being a covariance matrix, $\Sigma_{\mathbf{x}(t)}$ is symmetric positive definite as well and thus a valid covariance matrix, see Theorem 7.2.3 in [11]. Furthermore, this Lyapunov equation can be differentiated with respect to A_{ij} to obtain

$$A\frac{d\Sigma_{\mathbf{x}(t)}}{dA_{ij}} + \frac{d\Sigma_{\mathbf{x}(t)}}{dA_{ij}} A^\intercal + E_{ij}\Sigma_{\mathbf{x}(t)} + \Sigma_{\mathbf{x}(t)} E_{ji} = 0, \tag{B.8}$$

where E_{ij} has as (i,j)th entry a 1 and zeros everywhere else. Since this is another Lyapunov equation it can be solved in terms of A and $E_{ij}\Sigma_{\mathbf{x}(t)} + \Sigma_{\mathbf{x}(t)} E_{ji}$ to obtain $\frac{d\Sigma_{\mathbf{x}(t)}}{dA_{ij}}$.

B.3 Continuous Decision Rule

Using a linear continuous decision rule $d_{cont}(\mathbf{x}(t_d)) = \mathbf{w}^\intercal \mathbf{x}(t_d) + c$ with fixed $\mathbf{w}$ and c, a value is assigned to the network state at time t_d. This can be compared to the stimulus label s with a square loss function $L = \sum_s \langle (s - d_{cont}(\mathbf{x}(t_d)))^2 \rangle$. In this case, the gradient with respect to A_{ij} can be calculated in a simply from the two derivatives derived in Section B.2, since the loss function can be rewritten as

$$\begin{aligned} L &= \sum_s \langle (s - c - \mathbf{w}^\intercal \mathbf{x}(t_d))^2 \rangle \\ &= \sum_s \left[(s-c)^2 - 2(s-c)\mathbf{w}^\intercal \langle \mathbf{x}(t_d) \rangle + \mathbf{w}^\intercal \langle \mathbf{x}(t_d)\mathbf{x}^\intercal(t_d) \rangle \mathbf{w} \right] \\ &= \sum_s \left[(s-c)^2 - 2(s-c)\mathbf{w}^\intercal \langle \mathbf{x}(t_d) \rangle + (\mathbf{w}^\intercal \langle \mathbf{x}(t_d) \rangle)^2 + \mathbf{w}^\intercal \Sigma_{\mathbf{x}(t_d)} \mathbf{w} \right]. \end{aligned}$$

Thus, its derivative is

$$\frac{\partial L}{\partial A_{ij}} = \sum_s \left[2(s-c)\mathbf{w}^\intercal \frac{\partial \langle \mathbf{x}(t_d) \rangle}{\partial A_{ij}} + 2 \cdot \left(\mathbf{w}^\intercal \frac{\partial \langle \mathbf{x}(t_d) \rangle}{\partial A_{ij}} \right) \cdot \mathbf{w}^\intercal \langle \mathbf{x}(t_d) \rangle + \mathbf{w}^\intercal \frac{\partial \Sigma_{\mathbf{x}(t_d)}}{\partial A_{ij}} \mathbf{w} \right].$$

B.4 Binary Decision Rule

Wrapping the previous decision rule into a Heaviside step function

$$\Theta(x) = \begin{cases} 0 & \text{if } x \leq 0 \\ 1 & \text{otherwise} \end{cases}$$

results in the binary decision rule $d_{bin}(\mathbf{x}) = \Theta(\mathbf{w}^\intercal \mathbf{x} + c)$. Since the stimulus labels s were arbitrary, they can be chosen from $\{0,1\}$. Keeping the square loss function requires the calculation of an expectation over trials, thus, with the binary decision rule this results in

$$\begin{aligned} L =& \sum_s \langle (s - d_{bin}(\mathbf{x}(t_d)))^2 \rangle \\ =& \sum_s \int_{-\infty}^{\infty} d\mathbf{x}(t_d) p(\mathbf{x}(t_d)|s) (\Theta(\mathbf{w}^\intercal \mathbf{x}(t_d) + c) - s)^2 \\ =& \sum_s \int_{-\infty}^{\infty} d\mathbf{x}(t_d) p(\mathbf{x}(t_d)|s) (\Theta(\mathbf{w}^\intercal \mathbf{x}(t_d) + c)^2 - 2s\Theta(\mathbf{w}^\intercal \mathbf{x}(t_d) + c) + s^2). \end{aligned}$$

Given that $\Theta(\mathbf{w}^\intercal \mathbf{x}(t_d) + c) \in \{0,1\}$, $\Theta(\mathbf{w}^\intercal \mathbf{x}(t_d) + c)^2 = \Theta(\mathbf{w}^\intercal \mathbf{x}(t_d) + c)$ and it is only necessary to consider the case where $\Theta(x) > 0$, thus restricting the bounds of the integral. Furthermore, considering only two stimuli where without loss of generality $s_1 = 1$ and $s_2 = 0$,

$$\begin{aligned} L =& \sum_s \int_{\mathbf{w}^\intercal \mathbf{x}(t_d) + c > 0} d\mathbf{x}(t_d) p(\mathbf{x}(t_d)|s) - 2s \int_{\mathbf{w}^\intercal \mathbf{x}(t_d) + c > 0} d\mathbf{x}(t_d) p(\mathbf{x}(t_d)|s) + s^2 \\ =& 1 + \sum_s (1 - 2s) \int_{\mathbf{w}^\intercal \mathbf{x}(t_d) + c > 0} d\mathbf{x}(t_d) p(\mathbf{x}(t_d)|s) \\ =& 1 + p(\mathbf{w}^\intercal \mathbf{x}(t_d) + c > 0 | s = 0) - p(\mathbf{w}^\intercal \mathbf{x}(t_d) + c > 0 | s = 1) \\ =& p(\mathbf{w}^\intercal \mathbf{x}(t_d) + c > 0 | s = 0) + p(\mathbf{w}^\intercal \mathbf{x}(t_d) + c < 0 | s = 1) \end{aligned} \tag{B.9}$$

and the loss function reduces to the sum of the probabilities of making a wrong decision for either of the two stimuli s.

Taking Equation B.9 and noting that given a continuous covariance function of $e^{A(t-\tau)}\mathbf{n}(\tau)$ in Equation B.4 the network state follows a normal distribution $\mathbf{x}(t_d) \sim \mathcal{N}(\langle \mathbf{x}(t_d) \rangle, \Sigma_{\mathbf{x}(t_d)})$, it is possible to write

$$\begin{aligned} &p(\mathbf{w}^\intercal \mathbf{x}(t_d) + c > 0 | s = 0) \\ &\qquad = \frac{1}{(2\pi)^{N/2} \sqrt{|\Sigma_{\mathbf{x}(t_d)}|}} \int_{\mathbf{w}^\intercal \mathbf{x}(t_d) + c > 0} d\mathbf{x}(t_d) e^{-(\mathbf{x}(t_d) - \langle \mathbf{x}(t_d) \rangle)^\intercal \Sigma^{-1}_{\mathbf{x}(t_d)} (\mathbf{x}(t_d) - \langle \mathbf{x}(t_d) \rangle)}. \end{aligned} \tag{B.10}$$

Then, using a series of coordinate transforms, the probability in Equation B.10 can be expressed as

$$p(\mathbf{w}^\intercal \mathbf{x}(t_d) + c > 0 | s = 0) = 1 - \Phi\left(- \frac{\mathbf{w}^\intercal e^{A(t-t_s)} \mathbf{u}(0) + c}{\sqrt{\mathbf{w}^\intercal \Sigma_{\mathbf{x}(t_d)} \mathbf{w}}} \right),$$

where $\Phi(r) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{r} e^{-r^2/2}$ is the cumulative distribution function of the Gaussian distribution [15]. Similarly,

$$1 - p(\mathbf{w}^\intercal \mathbf{x}(t_d) + c > 0 | s = 1) = \Phi\left(-\frac{\mathbf{w}^\intercal e^{A(t-t_s)} \mathbf{u}(1) + c}{\sqrt{\mathbf{w}^\intercal \Sigma_{\mathbf{x}(t_d)} \mathbf{w}}} \right).$$

It follows that

$$L = 1 - \Phi\left(-\frac{\mathbf{w}^\intercal e^{A(t-t_s)} \mathbf{u}(s=0) + c}{\sqrt{\mathbf{w}^\intercal \Sigma_{\mathbf{x}(t_d)} \mathbf{w}}} \right) + \Phi\left(-\frac{\mathbf{w}^\intercal e^{A(t-t_s)} \mathbf{u}(s=1) + c}{\sqrt{\mathbf{w}^\intercal \Sigma_{\mathbf{x}(t_d)} \mathbf{w}}} \right). \quad \text{(B.11)}$$

Since $\Phi'(r) = \frac{1}{\sqrt{2\pi}} e^{-r^2/2}$ is a Gaussian function, it is simple to calculate the derivative of the loss using the chain rule.

$$\begin{aligned}
\frac{\partial L}{\partial A_{ij}} =& -\Phi'\left(-\frac{\mathbf{w}^\intercal e^{A(t-t_s)} \mathbf{u}(s=0) + c}{\sqrt{\mathbf{w}^\intercal \Sigma_{\mathbf{x}(t_d)} \mathbf{w}}} \right) \frac{\partial}{\partial A_{ij}} \left[-\frac{\mathbf{w}^\intercal e^{A(t-t_s)} \mathbf{u}(s=0) + c}{\sqrt{\mathbf{w}^\intercal \Sigma_{\mathbf{x}(t_d)} \mathbf{w}}} \right] \\
&+ \Phi'\left(-\frac{\mathbf{w}^\intercal e^{A(t-t_s)} \mathbf{u}(s=1) + c}{\sqrt{\mathbf{w}^\intercal \Sigma_{\mathbf{x}(t_d)} \mathbf{w}}} \right) \frac{\partial}{\partial A_{ij}} \left[-\frac{\mathbf{w}^\intercal e^{A(t-t_s)} \mathbf{u}(s=1) + c}{\sqrt{\mathbf{w}^\intercal \Sigma_{\mathbf{x}(t_d)} \mathbf{w}}} \right] \\
=& -\Phi'\left(-\frac{\mathbf{w}^\intercal e^{A(t-t_s)} \mathbf{u}(s=0) + c}{\sqrt{\mathbf{w}^\intercal \Sigma_{\mathbf{x}(t_d)} \mathbf{w}}} \right) \\
&\cdot \left(-\frac{\mathbf{w}^\intercal \frac{\partial e^{A(t-t_s)}}{\partial A_{ij}} \mathbf{u}(s=0)}{\sqrt{\mathbf{w}^\intercal \Sigma_{\mathbf{x}(t_d)} \mathbf{w}}} + \frac{\mathbf{w}^\intercal e^{A(t-t_s)} \mathbf{u}(s=0)}{(\mathbf{w}^\intercal \Sigma_{\mathbf{x}(t_d)} \mathbf{w})^{\frac{3}{2}}} \mathbf{w}^\intercal \frac{\partial \Sigma_{\mathbf{x}(t_d)}}{\partial A_{ij}} \mathbf{w} \right) \\
&+ \Phi'\left(-\frac{\mathbf{w}^\intercal e^{A(t-t_s)} \mathbf{u}(s=1) + c}{\sqrt{\mathbf{w}^\intercal \Sigma_{\mathbf{x}(t_d)} \mathbf{w}}} \right) \\
&\cdot \left(-\frac{\mathbf{w}^\intercal \frac{\partial e^{A(t-t_s)}}{\partial A_{ij}} \mathbf{u}(s=1)}{\sqrt{\mathbf{w}^\intercal \Sigma_{\mathbf{x}(t_d)} \mathbf{w}}} + \frac{\mathbf{w}^\intercal e^{A(t-t_s)} \mathbf{u}(s=1)}{(\mathbf{w}^\intercal \Sigma_{\mathbf{x}(t_d)} \mathbf{w})^{\frac{3}{2}}} \mathbf{w}^\intercal \frac{\partial \Sigma_{\mathbf{x}(t_d)}}{\partial A_{ij}} \mathbf{w} \right)
\end{aligned}$$

The derivatives required are calculated using the formulae derived in Section B.2.

Appendix C

Supplementary Figures

C.1 Supplementary Videos

The supplementary videos are:

1. Binary_Exponential_wpi.mp4: one input ($\mathbf{u}_1$) is fixed at $[1,0]$, $\mathbf{w} = [-1,0]$. The second input is varying. Networks trained with exponential decay constant 0.01 and 25 decision times in the range $[0,50]$. Eigenvectors named in order of dominance.
2. Binary_Exponential_w34pi.mp4: same as 1), but the angle of $\mathbf{w}$ to the positive x-axis is $\frac{3\pi}{4}$.
3. Binary_Exponential_w34pi.mp4: same as 2), but only left eigenvectors are shown. One is always collinear with the input linear discriminant.
4. Damped_oscillation_big.mp4: Shows a damped oscillation obtained by using the binary rule with several evaluation points.
5. Damped_oscillation_zoom.mp4: The same as 4) but zoomed in. It is visible that the responses align quickly with the eigenvectors. Note that although the animation starts at time 0, the response quickly leaves the frame so that it appears as if there was no change in the beginning.

The plotting code for phase-planes was adapted, but heavily modified from that used in the Honours Differential Equations course.

C.2 Quantifying the Effect of Feedforward Connectivity

The proportion of feedforward connectivity compared to the total connectivity in the network is high throughout but increases towards the point of alignment of $\mathbf{w}$ and $\mathbf{w}_{LD}$ where the network activity is almost entirely determined by its hidden feedforward connectivity.

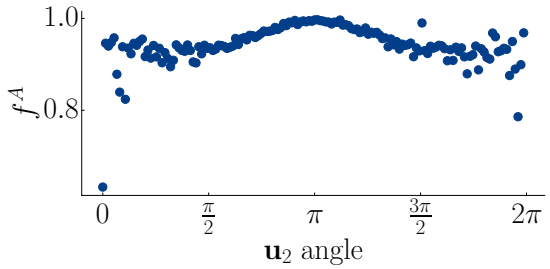


Figure C.1: The proportion of feed-forward connectivity in the networks. The maximum attained is approximately 0.9971.

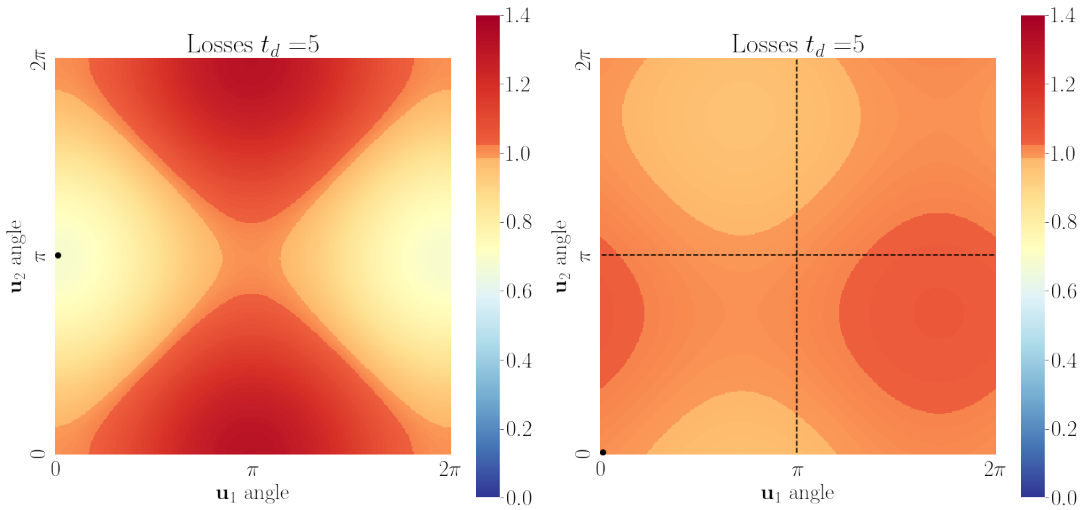
C.3 Generalising to other Stimulus Directions

In the main text, the concern has been how the network solves the task for the inputs it is optimised for. A comparison has also been drawn with some selected directions in state space suggested as particularly well-performing in some respect in the literature. However, since one of the central advantages of non-normal networks are their flexibility compared to attractor networks, the question is to what extent the oscillatory networks found with this setup share this property of being able to integrate a larger amount of the parameter space.

Figure C.2 shows examples for this. In the standard case in Figure C.2a, the network performs well for approximately half the input angles. Where the two stimulus vectors are equal, it performs at chance level, a region separating the contiguous parameter regions in which the network performs above or below chance level respectively. The results show a symmetry of areas of good and bad performance: For the parameters with high losses, a relabelling would approximately lead to the complementary low loss.

Figure C.2b shows a somewhat pathological example since during training both stimulus vectors were equal, so that the same direction in state space was associated with two different labels. This can only be solved by chance performance for these particular inputs. The network has adapted to perform at roughly chance level for the majority of inputs, although interestingly, the contiguous regions of performance above and below chance level do still exist, albeit reduced both in size and deviation from chance performance.

This does not represent an improvement of flexibility compared to an attractor network which exhibits the same patterns. The same is true for non-normal networks which in theory should generalise much better than attractor networks. However, this may be due to the low dimensionality of the network, since memory is usually considered as a function of network dimension [20].



(a) Optimised for $\mathbf{u}_1$, $\mathbf{u}_2$ aligned but in opposite directions. The readout has angle 0.

(b) Optimised for $\mathbf{u}_1$, $\mathbf{u}_2 = [1,0]$. The readout has angle π (marked with dashed lines) and is thus also aligned but pointing in the opposite direction.

Figure C.2: Performance of networks optimised for $t_d = 5$ for parameters marked with a black dot. The readout is kept fixed in the position the network was optimised for. Red colours indicate performance below chance level, orange around chance level, other colours above chance level.

C.4 Most Amplifying Directions

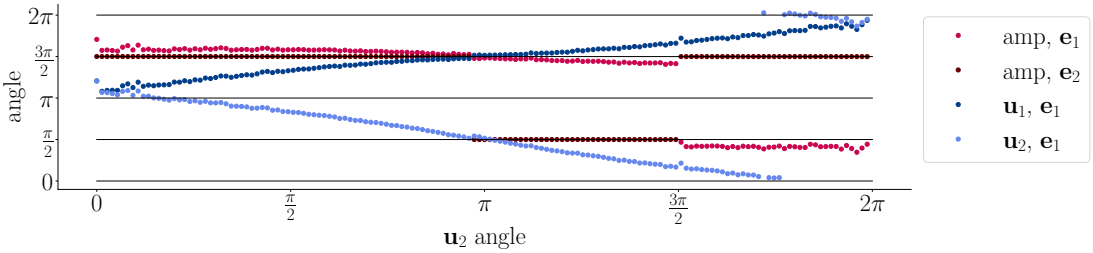
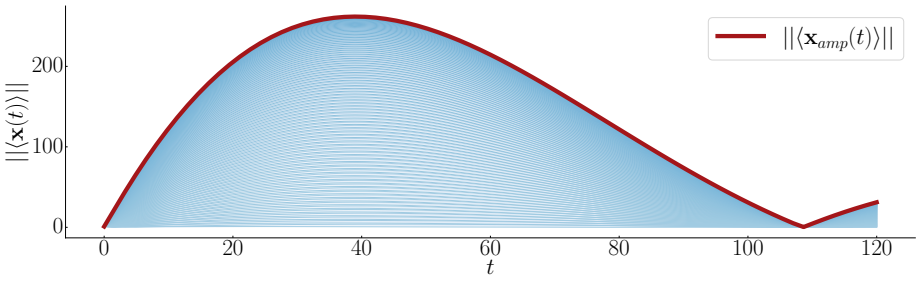
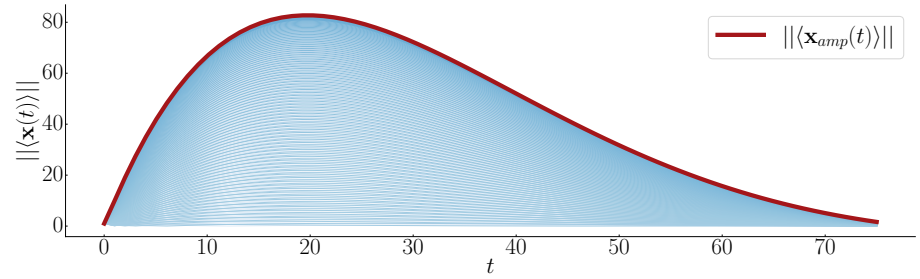


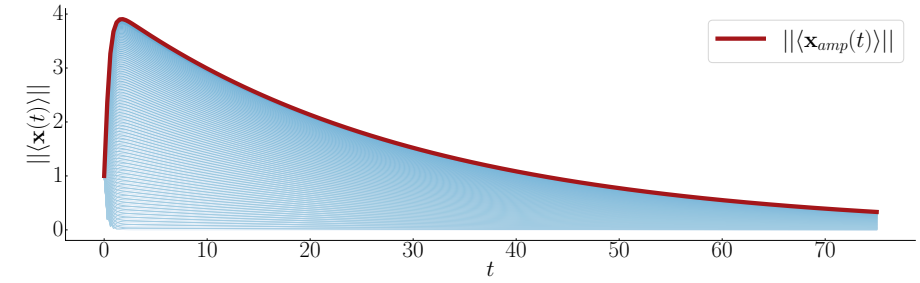
Figure C.3: The most amplifying mode *amp* is orthogonal to the fast mode $\mathbf{e}_2$. The stimulus vectors do not consistently align with it.



(a) Simple binary decision rule, optimising for a single delay.



(b) Binary decision rule, optimising for several delays.



(c) Binary decision rule, optimising for several delays with an exponential weighting.

Figure C.4: Amplification of stimulus vectors for 360 different angles in blue. The most amplifying vector as calculated in [36] dominates all of them.

C.5 Eigen- and Singular Value Decomposition

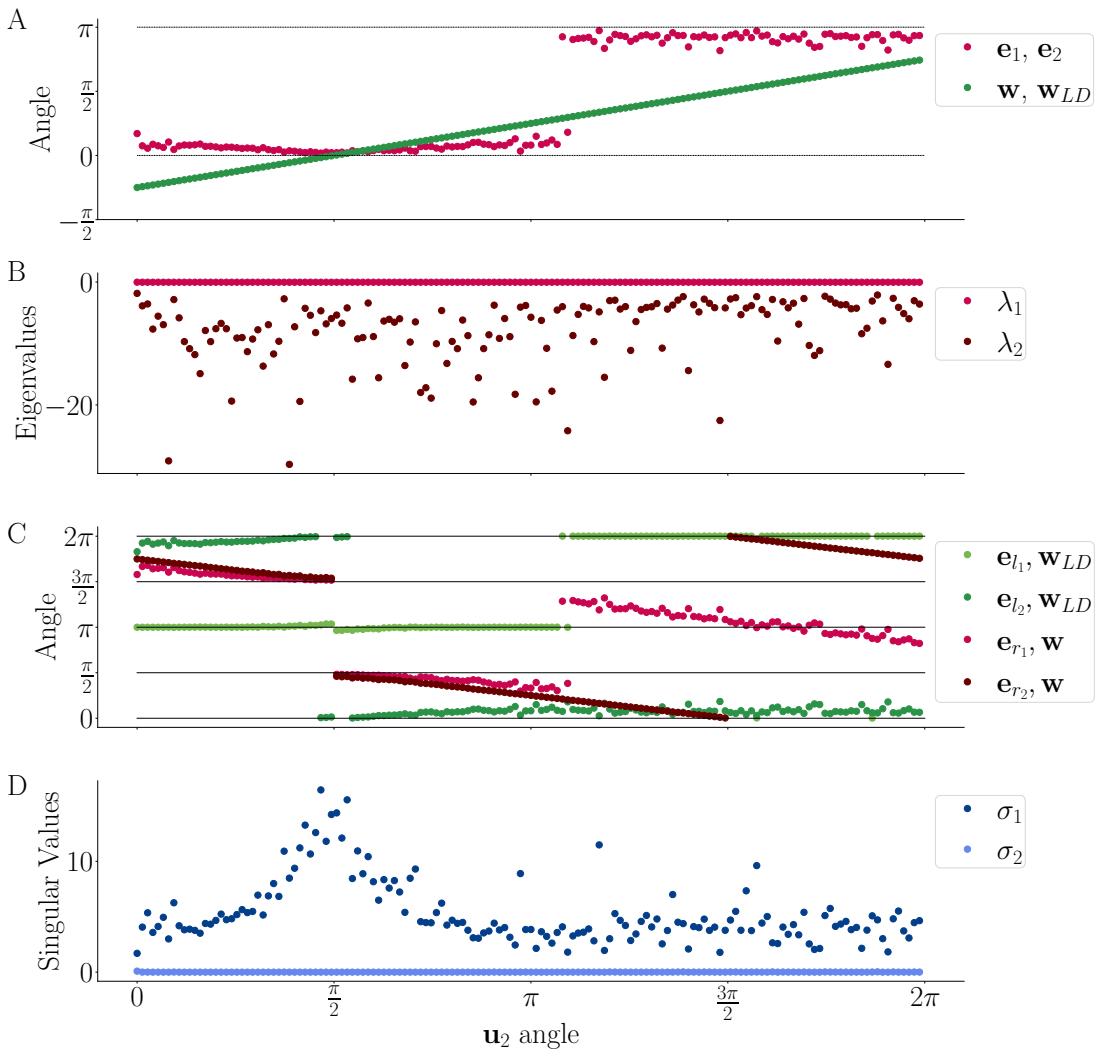


Figure C.5: Same setup as for Figure 4.10, but with the angle of $\mathbf{w}$ $\phi = \frac{3\pi}{4}$. Thus, $\mathbf{w}$ and $\mathbf{w}_{LD}$ are aligned at $\frac{\pi}{2}$. This is to demonstrate that indeed the angle between $\mathbf{w}$ and $\mathbf{w}_{LD}$ is crucial to the behaviour of the network. Here, the second jump occurs as the angle between $\mathbf{w}$ and $\mathbf{w}_{LD}$ is equal to the sum of the angle between these two vectors at 0 and $\frac{\pi}{2}$.