Tampere University

Iiro Ahokainen

RECURRENTLY CONNECTED CONTINUOUS-TIME RATE-BASED NEURAL NETWORKS

Bachelor's Thesis
Faculty of Engineering and Natural Sciences
December 2021

ABSTRACT

Iiro Ahokainen: Recurrently connected continuous-time rate-based neural networks
Bachelor's Thesis
Tampere University
Bachelor's Programme in Engineering and Natural Sciences
December 2021

One of the central questions in neuroscience is how neurons and neuron populations communicate with each other. An approach amongst others is to model networks of neurons and populations as a continuous nonlinear dynamical system. Continuous-time rate-based recurrent neural networks (RNNs) offer a way to model neural networks as time-dependent nonlinear dynamical systems. First RNNs were developed several decades ago but recent developments on theoretical frameworks concerning neural manifolds and new optimization techniques motivate to study RNNs in a new perspective.

In this thesis, the theoretical background of RNNs was studied from the neuroscience perspective and the dynamics of RNNs was briefly analyzed with local stability analysis. Several RNNs were studied while the focus was concentrated towards the two most used RNNs; the additive system and the Wilson-Cowan system. The main difference between the models is how fast they respond to postsynaptic currents. It was found that the additive system and the Wilson-Cowan system exhibit same fixed points, and the asymptotic stability near the hyperbolic fixed points is same for both models. This finding explains why the models behave similarly when external inputs are time-invariant. In addition to the additive system and to the Wilson-Cowan system, other RNN variants were briefly studied from a theoretical perspective explaining what is the relation between neuroscience and the RNNs.

Altogether, the results of the thesis help to understand the dynamics of the systems and guide the selection of RNNs on neuroscientifically interesting computational tasks which include modelling of the brain's network dynamics. The RNNs combined with dimension reduction methods can reveal low-dimensional neural manifolds that may arguably explain fundamental brain mechanisms.

Keywords: recurrent neural network, RNN, dynamical system, local stability, Wilson-Cowan system, Hopfield system

PREFACE

This thesis was made as a part of my Bachelor of Science studies at the Faculty of Engineering and Natural Sciences. Motivation for the thesis originated from research done in the Computational Neuroscience Group during summer 2021.

I would like to express my gratitude to Marja-Leena Linne and Mikko Lehtimäki for helping me with the research and with the writing process. In addition, thanks for Prof. Esa Räsänen for providing feedback. Lastly, I want to thank my friends and family for their support during the writing process.

Tampere, 11th December 2021

Iiro Ahokainen

CONTENTS

LIST OF SYMBOLS AND ABBREVIATIONS

E	Reversal potential
G_{syn}	Conductance of synapse
I	Electrical current
K_s	Synaptic kernel
N	Number of units
R	Resistor
V	Membrane voltage
$\Im(z)$	Imaginary part of the complex number z
$\Re(z)$	Real part of the complex number z
Θ	Heaviside step function
δ	Dirac's delta function
λ	Eigenvalue of a Jacobian matrix
$\langle N \rangle$	Average number of connections
ω	Angular frequency
ρ	Neural response function
τ_a	Time constant of the activity dynamics
τ_r	Time constant of the firing rate dynamics
τ_s	Time constant of the synaptic kernel
τ	Time constant of a system
$\mathbf{J}_{WC}$	Jacobian matrix of the Wilson-Cowan model
$\mathbf{J}_{add}$	Jacobian matrix of the additive model
$\mathbf{W}^{ext}$	Connectivity matrix of external inputs
$\mathbf{W}^{rec}$	Recurrent connectivity matrix
θ	Action potential firing threshold
a_e	Activity of the excitatory population
a_i	Activity of the inhibitory population
a	Activity

f'	Derivative of activation function, $\frac{df(x)}{dx} = f'(x)$
f^{-1}	Inverse of activation function
f	Activation function
r_{ext}	External input
r	Firing rate of a neuron
t_r	Numerical value of the absolute refractory period
t_{sp}	Timing of spike
t	Time
v	Presynaptic firing rate
w_{ij}	Synaptic strength from a presynaptic unit j to a postsynaptic unit i
AI state	Asynchronous Irregular state
ARP	Absolute Refractory Period
IF neuron	Integrate-and-Fire neuron
LTM	Long-Term Memory
ODE	Ordinary Differential Equation
RNN	Recurrent Neural Network
STM	Short-Term Memory

1. INTRODUCTION

Modern neuroscience as a field of science has evolved rapidly throughout its short history. The main driver has been the rapid development of technology, especially fast development of electrophysiology to record bioelectrical activity of neurons and neuronal networks and continuous progress on better imaging technologies. As more brain data has been gathered from various species more theoretical frameworks have been developed to understand the data. Due to complexity of the human brain many parallel approaches are needed in order to understand even the very basics of brain dynamics. The brain operates on various spatial and temporal resolutions which makes intuitive understanding of brain dynamics extremely difficult and thus mathematical models are needed. In this work, we will view the brain from a dynamical system point of view, and focus on network dynamics of individual neurons and populations of neurons.

In more detail, focus will be on a particular type of network models, that is, continuous-time rate-based Recurrent Neural Networks (RNNs). First RNNs were developed already several decades ago [1] but interest towards RNNs has been rising due to their high data representative power and better optimizing techniques [2–4]. In short, optimizing RNNs means updating connectivity parameters so that the network will produce a desired response to incoming inputs. RNNs can be seen as a data representation tool and also as a hypothesis generating tool. For example, a common hypothesis is that the RNNs can explain how the brain operates during a working memory task, for example see Ref. [5]. It is up to debate if the RNNs can really offer insight to neuronal mechanisms.

RNNs can be used to model networks where the computational unit is a neuron or networks where the computational unit is a population of neurons. Population of neurons can be modelled as a single computational unit when one makes suitable mean-field approximations. Even networks of mean-field models can be connected recurrently as shown in Figure 1.1. In this work, focus will be on the networks of neurons and the networks of populations that are rate-based and recurrent. We will not focus on RNNs that have spatially dependent simulation variables, rather we will only focus on time-dependent RNNs. Note that many other modelling approaches exist for all the spatial scales shown in Figure 1.1.

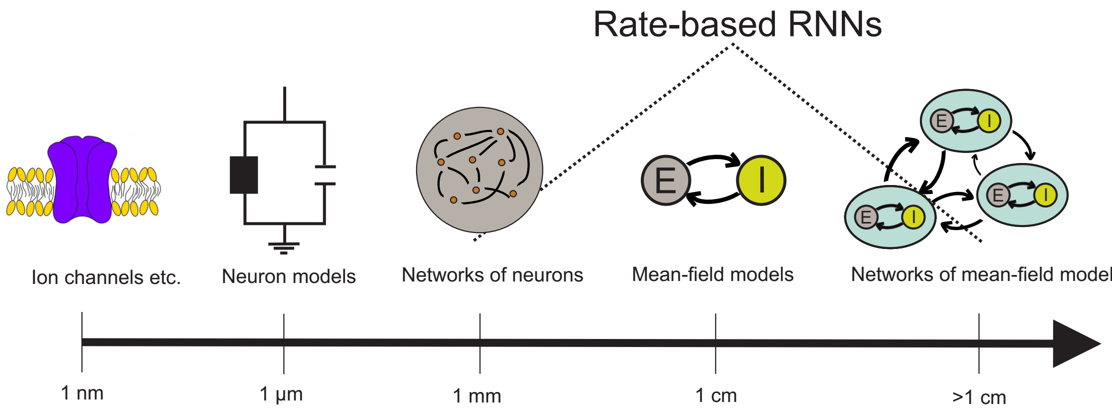

Figure 1.1. *A coarse illustration of different spatial scales of modelling. Left to right: Individual cell components, for example ion channels, can be modelled in numerous ways. These cell components can be connected together to model a single neuron. Neurons on the other hand can be joined together to form networks. The network of neurons can be approximated with one population forming a mean-field model. Here only two populations are connected, excitatory and inhibitory population, but in general the number of populations in the mean-field models can vary from 1 to* N*. The mean-field model can be used as a base unit for a large scale system. Rate-based RNNs can be used to model any of these network models. Picture of the ion channel was adapted from Ref. [6].*

In order to capitalize RNNs on neuroscientifically interesting computational problems one should understand the relevance of these models which is the reason why this study was conducted. The focus will be on examining theoretical motivation behind the RNNs and briefly analysing the dynamics of RNNs. Most of the RNNs are highly simplified systems and it is not trivial to understand how much biology is present in these models which is why the theoretical study is needed. Although we will not focus on the optimization or learning capacity of these models, we will study the basic properties of the dynamics as it lays a profound basis for understanding how these systems can learn and memorize events. The memory of RNNs is closely related to so called neural manifolds which we shall discuss more in Chapter 2. The dynamics is analyzed with local stability analysis while the theoretical background is based on several literature sources. Understanding the theoretical motivation behind the RNNs and the basic stability properties of RNNs lays a profound foundation for the computational use of RNNs.

The thesis is organized as follows: In Chapter 2 the basics of neuronal modelling will be discussed and more concise motivation for the stability analysis is given. In Chapter 3 the neurobiological motivation of RNNs will be discussed broadly while Chapters 4 and 5 focus on numerical methods and stability analysis with examples. Finally, the results of local stability analysis and the implications of theoretical findings will be discussed in Chapter 6.

2. BACKGROUND

2.1 Integrate-and-Fire neuron as low-pass filter

In this section, the characteristics of classical Integrate-and-Fire (IF) neuron are defined and its response to transient input is briefly studied. Furthermore, an example of recurrently coupled spiking network is considered, and a distinction between spiking neural networks and rate-based neural network will be clarified.

Consider an electrical circuit depicted in Figure 2.1. The libid bilayer of a neuronal cell membrane can be modelled as a capacitor C and ion channels are defined by a resistor R and a voltage source E. Parallel to these components is an external input current I_e.

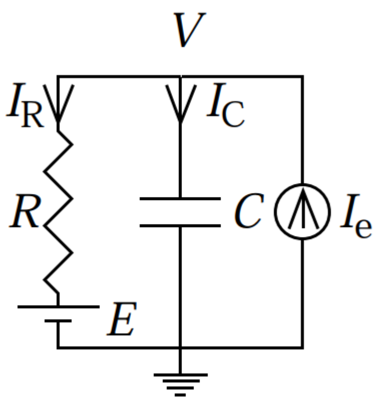

Figure 2.1. *An integrate-and-fire neuron adapted from Ref. [7]. The membrane voltage V depends on the capacitor properties of the cell membrane and on the cell membrane ion channels. In addition, the membrane voltage is altered by the external input.*

Now, considering Kirchhoff's current law for the current through the capacitor $I_C = -I_R + I_e$ and Ohm's law $V - E = RI_R$, the time evolution of the membrane voltage V can be written [7] as

$$C\frac{dV}{dt} = \frac{E-V}{R} + I_e \Leftrightarrow \tau\frac{dV}{dt} = E - V + RI_e, \tag{2.1}$$

where $\tau = RC$ is the time constant of the system. When $V > \theta$, where θ is a threshold characteristic for the neuron, then it is said that an action potential is fired, that is, the cell membrane voltage sharply rises for a small period of time. Then shortly after, the membrane potential is returned to initial resting potential. Real neurons exhibit the so-called Absolute Refractory Period (ARP) after firing an action potential during which the neuron cannot fire even if a strong enough excitation is presented into the cell. Common ARP values range from 0 ms to 5 ms. [8]

Next, the effect of transient external input $I_e = I\cos(\omega t)$ to subthreshold dynamics is considered. Choosing an integrating factor $e^{\int_0^t \frac{1}{\tau}dt}$ and solving Eq. (2.1) for V gives

$$\begin{aligned} V(t) &= e^{-t/\tau}\left[\int e^{t/\tau}\frac{E + RI\cos(\omega t)}{\tau}dt + c\right] \\ &= E + e^{-t/\tau}\int e^{t/\tau}\frac{RI\cos(\omega t)}{\tau}dt + \frac{c}{e^{t/\tau}} \\ &= E + RI\frac{\cos(\omega t) + \tau\omega\sin(\omega t)}{1+\omega^2\tau^2} + \frac{c}{e^{t/\tau}} \end{aligned}$$

Letting the initial term $c/e^{t/\tau}$ to decay close to zero and easily seeing that $\cos(\omega t) + \tau\omega\sin(\omega t) = \sqrt{1+\omega^2\tau^2}\cos(\omega t - \arctan(\omega\tau))$ leads to

$$V(t) \approx E + \frac{RI\cos(\omega t - \arctan(\omega\tau))}{\sqrt{1+\omega^2\tau^2}}. \tag{2.2}$$

From Eq. (2.2) it is clear to deduce that the IF neuron acts as a low pass filter because higher frequency ω leads to diminishing change on the membrane voltage V [7]. This low pass filtering property of IF neuron is an important observation that is needed later in Section 3.1.

2.2 Introduction to spiking neural networks

The IF neuron and any other spiking neuron model can be used as a building block for a recurrently connected network. A straightforward way to formulate a spiking neural network is to replace the external input I_e with a synaptic input I_{syn}. Letting I_{syn} to be a function of incoming spikes, recurrency is obtained. The incoming spikes are sent by the other units in the network which are called presynaptic neurons. The neuron that receives the spikes is called a postsynaptic neuron. There are many ways to define I_{syn} depending on the choices a modeller makes. For example, the synaptic input of one unit in the network can be written as [9]

$$I_{syn}(V) = (E_e - V)G^e_{syn}(t) + (E_i - V)G^i_{syn}(t), \tag{2.3a}$$

$$G^{(e,i)}_{syn}(t) = Q_{(e,i)}\sum_{n=1}^{N} p(n)\Theta(t - t_{sp}(n))e^{-\frac{t-t_{sp}(n)}{\tau_{sp}}}, \tag{2.3b}$$

where $E_{(e,i)}$ is the excitatory/inhibitory reversal potential, $G^{(e,i)}_{syn}$ is the conductance of excitatory/inhibitory inputs, $Q_{(e,i)}$ is an amplitude, and Θ denotes Heaviside step function. Here an exponential decay of a spike at time t_{sp} is chosen for depicting attenuation of the spike through time. In general case, not all neurons N are directly connected to each other. On the contrary, $p(n)$ is set to zero for a majority of neurons while $p(n) = 1$ for the connected neurons [9].

To conclude, spiking neural networks describe the time evolution of neurons' membrane voltages with some connectivity rule that utilizes conductances, for example Eq. (2.3). From now on, spiking networks will not be considered. Instead, the focus is steered towards rate-based RNNs. The main difference between these two approaches is the assumption that all important information is encoded into the rate on which each neuron fires action potentials rather than the explicit time evolution of neurons' membrane voltages and exact timing of spikes. This assumption significantly simplifies the calculation of simulation variable which makes the approach computationally more attractive for tasks where a complete evolution of membrane voltages is not the priority. In the next section, rate variables will be defined in order to help a concise description of continuous-time rate-based RNNs.

2.3 Firing rate of single neuron and activity of population of neurons

We start by defining a quantity called neural response function $\rho(t)$. The neural response function simply expresses timing of spikes t_{sp} with the Dirac's δ function

$$\rho(t) = \sum_n \delta[t - t_{sp}(n)]. \tag{2.4}$$

With the neural response function $\rho(t)$, a trial averaged firing rate of a single neuron can be defined as

$$r(t) = \frac{1}{\Delta t K} \sum_{k=1}^{K} \int_t^{t+\Delta t} \rho_k(t')dt' = \frac{1}{\Delta t} \int_t^{t+\Delta t} \langle \rho(t') \rangle dt', \tag{2.5}$$

where K is a number of trials, $\langle \rho(t) \rangle$ is a trial averaged neural response function and the bin size Δt is chosen so that a maximum of one spike is present at a trial k. Trials are assumed to be independent and repeatable.

In addition to the firing rate of a single neuron, we define a firing rate of a population, which will be referred as an activity. The activity is essentially a trial averaged proportion of neurons in the population that are active at the moment t:

$$\begin{aligned} a(t) &= \frac{1}{\Delta t K}\sum_{k=1}^{K}\sum_{n=1}^{N}\frac{n_{act,k,n}(t;\Delta t)}{N} \\ &= \frac{1}{N}\sum_{n=1}^{N}\frac{1}{\Delta t K}\sum_{k=1}^{K}\int_{t}^{t+\Delta t}\rho_{k,n}(t)dt \\ &= \frac{1}{N}\sum_{n=1}^{N} r_n(t) \end{aligned} \tag{2.6}$$

where $n_{act,k,n}(t;\Delta t)$ is the number of spikes occurring during Δt in a neuron n on a trial k. Definitions of the firing rate of a single neuron and the activity are adapted from Ref. [8]. However, contrary to Ref. [8], trial averaging was included also in the definition of the activity in order to find a connection between the two measurements.

Note that the time bin Δt can vary greatly leading to large range of frequency values in Hertz (Hz), which is a desired attribute in experimental regime but can cause a trouble while building mathematical models for numerical simulations. Therefore, $r(t)$ and $a(t)$ are usually normalized to get values ranging from 0 to 1.

2.4 Stability of dynamical system

As mentioned in the introduction, RNNs can be viewed as dynamical systems. An important part of understanding dynamical systems is the stability of a system. Stability is interesting both from the neuroscience perspective and from the machine learning perspective. In this chapter, the importance of stability is clarified, and later in Chapter 5 some stability properties will be demonstrated with examples. Next, the term 'stability' will be defined and a few words shall be stated about neural manifolds.

2.4.1 Definition of stability

Stability of the system can refer to many subjects depending on the context, and therefore a concise definition of stability is important. When dealing with nonlinear dynamical systems, two different definitions of stability usually come across: the Lyapunov stability and the asymptotic stability. In this thesis focus will be on the stability of fixed points, and the stability of other attractors like limit cycles will not be considered in detail.

A fixed point $\mathbf{z}^*$ of a dynamical system is said to be Lyapunov stable if for every neighborhood L there exists a neighborhood $M \subset L$ for which all solutions $\mathbf{z}(t)$ starting in M remains in L for all times $t \geq 0$ [10]. Notice that the Lyapunov stability does not require that $\mathbf{z}(t)$ should approach to $\mathbf{z}^*$. In contrast to the Lyapunov stability, the asymptotic stability is more stringent condition. When L can be chosen so that $|\mathbf{z}(t) - \mathbf{z}^*| \to 0$ as

$t \to \infty$ for all **z**$(0) \in L$, then **z*** is said to be asymptotically stable [10]. If L contains all the initial conditions **z**(0) that lead to **z***, then L is called a basin of attraction. In this thesis, we are not interested in the Lyapunov stability.

More complicated definitions relate to the limit cycle. The limit cycle can be defined as an isolated periodic orbit, and sometimes the isolation is not included in the definition [10]. The isolated orbit here means that there is no other closed trajectory in the close neighborhood of the solution. Depending on the definition it might be difficult to show that the limit cycle exists. The existence of a limit cycle can be proved in a two-dimensional system for example with the Poincaré-Bendixson theorem [10], but in n-dimensional systems proving the existence of a limit cycle is a notorious task especially if the isolation is required. Therefore, as we will later on refer to a limit cycle, we will neglect the isolation property.

A numerical stability is other form of stability that is present when working with computational models. The numerical stability refers to instabilities that arise from computational inaccuracies. For example, in a chaotic regime of a dynamical system small deviations on initial conditions may lead to unpredictable results. Therefore, the system can become unstable due to insufficient numerical accuracy. The numerical stability will not be discussed in this thesis. Instead, focus will be on the local asymptotic stability of fixed points which will be referred simply as stability or local stability.

2.4.2 Motivation for local stability analysis

An ongoing trend in neuroscience is to understand population activities in terms of manifolds. Neural manifolds are low-dimensional objects that consist of a set of continuous points. In this context, the low dimensionality refers to decreased dimensionality of an original state space. It is argued that the neural manifolds can reveal fundamental circuit mechanisms accounting how a brain operates during a given task. [11]

The asymptotic stability and the Lyapunov stability are closely linked to neural manifolds. Given a high-dimensional space, perturbations in this space should quickly approach to values defined in the neural manifold [11]. In other words, the neural manifold has to act as a low-dimensional attractor. This attractor can be understood as an interplay between stable and unstable fixed points [12]. Therefore, in the perspective of RNNs, local stability analysis of the system remains intriguing. The dynamics of trained RNNs can be understood by studying fixed points and so-called slow points that behave similarly to the fixed points [13]. With the approach from Ref. [13] one can find neural manifolds contained in a high-dimensional state space.

The other motivating factor for studying the local stability of RNNs is the impact on a learning capability of RNNs. Despite the most RNNs are simple by their architecture, they can produce versatile dynamics [14]. These dynamics include oscillatory and chaotic regimes. In some cases chaotic regime weakens the predictive power of RNN especially when the time scale is long. By developing tailored learning methods, one can limit existence of chaotic regimes as done for example in Ref. [4]. Now, we will move forward and start inspecting theoretical background of the most typical RNNs from the neuroscience perspective, and later in Chapters 4 and 5 the asymptotic stability will be discussed.

3. RECURRENT NEURAL NETWORKS

3.1 Derivation of RNNs

In this chapter, continuous-time RNNs will be defined using an approach from Ref. [7]. The derivation below is divided into two parts. In the first part it is shown how presynaptic firing rates are connected to a synaptic current at the soma of postsynaptic neuron, and in the second part the postsynaptic current is connected to the postsynaptic firing rate to complete the model architecture.

Postsynaptic current

Consider a network with computational units described by their firing rates **v** and **r** corresponding to input and output firing rates, respectively (Figure 3.1). Neurons used here are point neurons meaning that they have no spatial structure. Therefore, the term 'soma' (that is the body of a neuron) will be used to mitigate any time lag between synaptic and somatic current that would be present in real neurons. Here the time lag is assumed to be zero and therefore the synaptic currents and the somatic currents are the same. For simplicity, assume only one output neuron.

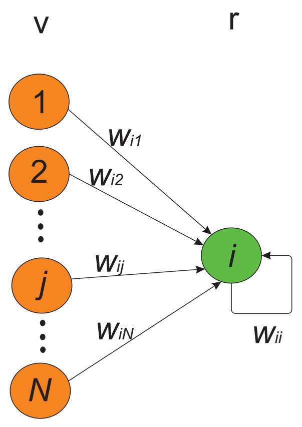

Figure 3.1. *Inputs to a single postsynaptic neuron shown in green from N presynaptic neurons shown in orange. The presynaptic neurons have firing rates* ***v*** *and the postsynaptic neuron has a firing rate* ***r****. In general, there can be many postsynaptic neurons. The connection parameter w_{ij} describes the strength of the connection between two neurons.*

First, we need to define a quantity which connects a presynaptic spike train to a postsynaptic current in the soma of the output unit. We call this quantity as the synaptic kernel $K_s(t)$. The synaptic kernel defined here accounts both for presynaptic and postsynaptic active and passive properties. For our approach, we can choose the synaptic kernel to be a simple decaying exponential as we are not interested in neurons' fine spatial dynamics nor detailed conductance changes in synapses. [7]

Now, assuming that the action potentials are independent of each other, the total synaptic current from one presynapse j over time t is [7]

$$I_{ij} = w_{ij} \sum_{t_{sp}<t} K_s(t - t_{sp}) = w_{ij} \int_0^t K_s(t - t')\rho(t')dt', \tag{3.1}$$

where $t_{sp} > 0$ s is a timing of a spike and w_{ij} is the synaptic weight between the presynapse j and the postsynapse i. The neural response function $\rho(t)$ describes the spike train at the synapse. Notice that here was assumed that only one synapse connects two neurons, or equivalently all synapses from j to i are modelled with a single connection parameter.

Because we are interested in a continuous type model, the spike train will be replaced by a firing rate $v_j(t)$. The replacement can be done without a great harm if the trial-to-trial variation of $\rho(t)$ is small. The variation is small when the number of presynapses is high and the presynaptic spikes are uncorrelated [7]. This is generally a good approximation but if presynaptic neurons fire synchronously, then the firing rate model will be insufficient to model all phenomenons in the network.

Next, summing all inputs gives

$$I_i = \sum_{j=1}^{N} w_{ij} \int_0^t K_s(t - t')v_j(t')dt'.$$

Note that $j = 1, 2, ...i, ..., N$, which means that a direct feedback is allowed. Now, choosing the normalized exponential function $K_s(t) = \exp(-t/\tau_s)/\tau_s$ with a synaptic time constant τ_s and taking time derivative using Leibniz integral rule [15] gives

$$\begin{aligned}
\frac{dI_i}{dt} &= \sum_{j=1}^{N} w_{ij} \frac{d}{dt} \int_0^t K_s(t-t')v_j(t')dt' \\
&= \sum_{j=1}^{N} w_{ij} \left[v_j(t)K_s(t-t)\frac{d}{dt}t + \int_0^t \frac{\partial}{\partial t} K_s(t-t')v_j(t')dt' \right] \\
&= \sum_{j=1}^{N} w_{ij} \frac{v_j(t)}{\tau_s} + \sum_{j=1}^{N} w_{ij} \int_0^t (-\frac{1}{\tau_s^2}) e^{\frac{-(t-t')}{\tau_s}} v_j(t')dt' \\
&= \sum_{j=1}^{N} w_{ij} \frac{v_j(t)}{\tau_s} - \frac{I_i}{\tau_s}. \qquad (3.2)
\end{aligned}$$

Eq. (3.2) now describes the time evolution of postsynaptic current at the soma with respect to presynaptic firing rates. For more lucid description, vectors and dot product may be used:

$$\tau_s \frac{dI_i}{dt} = -I_i + \mathbf{w} \cdot \mathbf{v}.$$

Usually, it is convenient to break the dot product $\mathbf{w} \cdot \mathbf{v}$ to the recurrent connections and to the feedforward connections:

$$\tau_s \frac{dI_i}{dt} = -I_i + \mathbf{w}_{rec} \cdot \mathbf{v}_{rec} + \mathbf{w}_f \cdot \mathbf{v}_f. \qquad (3.3)$$

An illustrative scheme is shown in Figure 3.2. Splitting the dot product into two parts enables including an external input by reinterpreting the feedforward inputs $\mathbf{w}_f \cdot \mathbf{v}_f$ as the external input $\mathbf{w}_{ext} \cdot \mathbf{r}_{ext}$. The external input is vastly used in many computational modelling schemes and can be time-independent or vary in time. Therefore, even if the external inputs are written as $\mathbf{r}_{ext}$, the external inputs are still dependent on time if not otherwise mentioned.

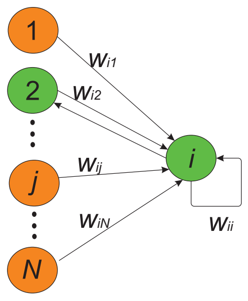

Figure 3.2. *A schematic illustration of a network with feedforward and recurrent connections. Orange nodes have only feedforward connections and green nodes have also recurrent connections.*

All synaptic currents in the network can be presented succinctly by a vector of currents. If N is the number of neurons in the network, then the dot products have to be calculated

N times. Therefore, the vector $\mathbf{w}_{rec}$ is replaced by a recurrent connectivity matrix $\mathbf{W}^{rec}$ and $\mathbf{w}_{ext}$ is replaced with a matrix $\mathbf{W}^{ext}$. Thus, assuming the synaptic time constants are same for every neuron, the synaptic currents are

$$\tau_s \frac{d\mathbf{I}}{dt} = -\mathbf{I} + \mathbf{W}^{rec}\mathbf{v}_{rec} + \mathbf{W}^{ext}\mathbf{r}_{ext}. \tag{3.4}$$

To complete the formulation of the RNNs the connection between the postsynaptic currents and the postsynaptic firing rates will be considered.

From postsynaptic current to postsynaptic firing rate

Next, the connection between the postsynaptic current and the postsynaptic firing rate need to be defined. The simplest assumption is that the postsynaptic firing rate follows changes of the postsynaptic current instantaneously [7]. Therefore, the postsynaptic firing rate can be obtained simply by applying a so-called activation function (or a transfer function or a gain function) f to the postsynaptic current. Typical activation functions are linear functions with a threshold or saturating functions like a sigmoid function [7]. Nevertheless, the firing rate $r = f[I(t)]$ should be non-negative for all values of $I(t)$.

The assumption on instantaneous nature of the connection between the postsynaptic current and the postsynaptic firing rate might not be always appropriate to make. Other approach is to couple the postsynaptic current and the firing rate with the following Ordinary Differential Equation (ODE) [7]

$$\tau_r \frac{dr_i}{dt} = -r_i + f[I_i(t)] \quad \text{or} \quad \tau_r \frac{d\mathbf{r}}{dt} = -\mathbf{r} + \mathbf{f}[\mathbf{I}(t)]. \tag{3.5}$$

Eq. (3.5) is motivated by an observation that the membrane potential of the IF neuron is low pass filtered version of the injected current when the current is sinusoidal as shown in Section 2.1. Finally, the RNN can be fully defined with Eq. (3.4) and Eq. (3.5) after deciding that output firing rates $\mathbf{r}$ corresponds to firing rates of recurrent units $\mathbf{v}_{rec}$. Then one gets a pair of ODEs:

$$\tau_r \frac{d\mathbf{r}}{dt} = -\mathbf{r} + \mathbf{f}(\mathbf{I}(t)) \tag{3.6a}$$

$$\tau_s \frac{d\mathbf{I}}{dt} = -\mathbf{I} + \mathbf{W}^{rec}\mathbf{r} + \mathbf{W}^{ext}\mathbf{r}_{ext}. \tag{3.6b}$$

However, simplified versions are usually preferred and can be obtained by comparing the synaptic time constant τ_s and the firing rate time constant τ_r [7]. If the time constant τ_r is small, then the firing rate approaches its steady state value fast when the synaptic current is constant. Especially, if $\tau_r << \tau_s$, then Eq (3.5) quickly approaches to the steady state value $\mathbf{r}(t) = \mathbf{f}[\mathbf{I}(t)]$ and Eq. (3.6) simplifies to an additive system

$$\tau_s \frac{d\mathbf{I}}{dt} = -\mathbf{I} + \mathbf{W}^{rec}\mathbf{f}(\mathbf{I}) + \mathbf{W}^{ext}\mathbf{r}_{ext}, \quad \mathbf{r}(t) = \mathbf{f}[\mathbf{I}(t)]. \tag{3.7}$$

As a note, the additive system is popularly known as the continuous-time Hopfield system [16]. However, the additive system was studied by Grossberg and Cohen [17] before Hopfield, which is the reason why Eq. (3.7) will be referred simply as the additive system in this thesis.

Continuing, if $\tau_s << \tau_r$, that is when the low pass filtering dynamics are relevant, then Eq (3.4) clearly approaches to

$$\mathbf{I} = \mathbf{W}^{rec}\mathbf{r} + \mathbf{W}^{ext}\mathbf{r}_{ext} \tag{3.8}$$

assuming the time variation of external inputs do not disrupt the steady state which will be generally assumed. This approximation simplifies the recurrent dynamics to

$$\tau_r \frac{d\mathbf{r}}{dt} = -\mathbf{r} + \mathbf{f}(\mathbf{W}^{rec}\mathbf{r} + \mathbf{W}^{ext}\mathbf{r}_{ext}). \tag{3.9}$$

The above system will be referred as the Wilson-Cowan system later on. Reason for the name 'Wilson-Cowan' will be further clarified in Section 3.2.

Here I additionally want to point out that in some sources [18, 19] Eq. (3.7) and Eq. (3.9) are shown to be mathematically equal when the condition Eq. (3.8) is satisfied. However, as shown above, such assumption only holds when $\tau_s << \tau_r$. Therefore, it is not meaningful to make such an assumption and the systems in Eq. (3.7) and Eq. (3.9) should not be considered to be the same. However, later in Section 5.2 it is shown that these two models have same fixed points and their asymptotic stability is the same on those hyperbolic fixed points. This can be intuitively understood by noting that the additive system will satisfy Eq. (3.8) when it is on a fixed point thus making the systems same in this special case.

Biological relevance of additive model and Wilson-Cowan model

The additive system and the Wilson-Cowan system can be derived with different assumptions to those that were used here. Ermentrout's approach [20] is based on separating the properties of the synaptic kernel in the presynaptic properties and in the postsynaptic properties. This separation essentially leads to same equations as derived here, although the time constants are interpreted in a different way. In fact, the synaptic kernel is interpreted as a postsynaptic potential which is a different approach than taken here. Interpreting the synaptic kernel as a postsynaptic potential leads to different simulation variable. Instead of postsynaptic currents one simulates postsynaptic membrane voltages. Therefore, the additive system is called as the voltage-based system in many sources. Also probably due to this discrepancy on the synaptic kernel, the simulation variable in the additive system is sometimes called as 'activity' that is not connected to the definition of activity in Eq. (2.6). The 'activity' in this sense just models abstract internal state of a neuron and not any physical quantity. Because of the ambiguity of the synaptic kernel, it is not always clear how the additive system should be interpreted in a biological context.

Models in Eq. (3.7) and Eq. (3.9) are used in different modelling schemes and it is not always clear what formalism should be preferred. Some insight can be found from [7] while arguably the additive system is more modern and has been more used in the literature [21]. It is up for debate if the low pass filtering is really needed as the synaptic currents already lag behind the presynaptic firing rates. In addition, the IF neurons' response to a step current is fairly rapid [8].

The synaptic kernel, which was chosen to be a simple decaying exponential, could explain why an additional low pass filtering is needed in some cases. Alternatively, one could choose more complex synaptic kernel such as

$$K_s(t) = x(t)u(t)\frac{e^{-t/\tau_s}}{\tau_s}, \tag{3.10}$$

where $x(t)$ and $u(t)$ refer to the synaptic depression and facilitation, respectively, according to the Tsodyks-Markram dynamics [22]. Other synaptic plasticity rules can be used too. In short, the synaptic plasticity refers to the local adaptation of the synaptic strength. Choosing the synaptic kernel in Eq. (3.10) will lead to more complex network model. For example, in the article by Barak and Tsodyks such a model was derived and studied for populations of neurons [23].

The other source of ambiguity is the synaptic weight parameter that is not strictly based on any particular mechanism in the synapse. However, it is possible to modify $\mathbf{W}^{rec}$ so that it accounts for general connectivity paradigms. According to commonly known Dale's principle connections from a neuron to other neurons should be either excitatory or inhibitory. In practice this means constraining each column of $\mathbf{W}^{rec}$ to be either fully positive or fully negative. Other constraints can be also used, for example the proportion of excitatory and inhibitory connections allowed in $\mathbf{W}^{rec}$. Commonly the partition 20/80 % (inhibitory/excitatory) is used. [24]

Generalization to population models

So far we have considered models with computational units being neurons. However, limiting to such models is computationally expensive as every neuron has to be modelled explicitly. Alternatively, when a network of neurons is large, population models are preferred. Arguably, also the assumption we made in Section 3.1 about independency of spike times is better satisfied on a large scale, especially on Asynchronous Irregular (AI) regime where population firing rates tend to be low. Generalization from the cell based RNNs can be easily done by loosely following formalisation presented in Ref. [23].

Consider the ODE Eq. (3.6b) on a unit form:

$$\tau_s \frac{dI_i}{dt} = -I_i + \sum_{j=1}^{N} w_{ij} r_j + \sum_{ext=1}^{N'} w_{i,ext} r_{ext}. \tag{3.11}$$

Now, consider that recurrent (external input) net of neurons is divided to P (P') populations that each has different intrinsic attributes. Neurons in a one population tend to be similar but neurons between the populations are fundamentally different. For example, inhibitory and excitatory neurons are usually separated into different populations [25]. For generality, assume that the neuron i belongs to a population α. Without losing any information Eq. (3.11) can be written as

$$\tau_s \frac{dI_i}{dt} = -I_i + \sum_{\mu=1}^{P} \sum_{j\in\mu}^{N_\mu} w_{ij} r_j + \sum_{\mu'=1}^{P'} \sum_{ext\in\mu'}^{N_{\mu'}} w_{i,ext} r_{ext},$$

where μ is a population index and N_μ is the number of neurons in that population. Next, approximate sums $\sum_{j\in\mu}^{N_\mu} w_{ij} r_j$ and $\sum_{ext\in\mu'}^{N_{\mu'}} w_{i,ext} r_{ext}$ with the average population statistics. Thus, we write with remembering the definition of the activity (Eq. (2.6))

$$\tau_s \frac{dI_i}{dt} \approx -I_i + \sum_{\mu=1}^{P} \langle N_\mu \rangle w_{i\mu} a_\mu + \sum_{\mu'=1}^{P'} \langle N_{\mu'} \rangle w_{i\mu'} a_{\mu'}, \tag{3.12}$$

where $\langle N_\mu \rangle$ is the average number of connections within the population μ, $w_{i\mu}$ is the average connection strength from the population μ to the neuron i, and a_μ is the average firing rate of the population, i.e. the activity. For convenience, $\langle N_{(\mu,\mu')} \rangle$ will be included into the weight parameter [23]. This is generally a good procedure as estimating exact number of neurons on specific populations can be cumbersome, and the same information can be more easily expressed with one parameter.

Now, summing Eq. (3.12) over each neuron in the population α gives

$$I_\alpha = \sum_{i=1}^{N_\alpha} I_i,$$

$$\tau_s \frac{dI_\alpha}{dt} = -I_\alpha + \sum_{\mu=1}^{P} \sum_{i=1}^{N_\alpha} w_{i\mu} a_\mu + \sum_{\mu'=1}^{P'} \sum_{i=1}^{N_\alpha} w_{i\mu'} a_{\mu'},$$

where I_α is the total current coming to the population α. Again, by approximating the sums $\sum_{i=1}^{N_\alpha} w_{i\mu}$ and $\sum_{i=1}^{N_\alpha} w_{i\mu'}$ one gets

$$\tau_s \frac{dI_\alpha}{dt} \approx -I_\alpha + \sum_{\mu=1}^{P} \langle N_{\alpha\mu} \rangle w_{\alpha\mu} a_\mu + \sum_{\mu'=1}^{P'} \langle N_{\alpha\mu'} \rangle w_{\alpha\mu'} a_{\mu'}, \tag{3.13}$$

where $\langle N_{\alpha\mu} \rangle$ and $\langle N_{\alpha\mu'} \rangle$ are the estimated number of connections between the populations. Final step is to include $\langle N_{x,x} \rangle$ in a weight parameter w_{xx} as done previously. With this fix, recurrently connected populations have exactly same equational form as recurrently connected neurons:

$$\tau_s \frac{d\mathbf{I}}{dt} = -\mathbf{I} + \mathbf{W}^{rec}\mathbf{a} + \mathbf{W}^{ext}\mathbf{a}_{ext}, \quad \mathbf{a} = \mathbf{f}[\mathbf{I}(t)], \tag{3.14}$$

when the activities follow populations' total currents instantaneously and

$$\tau_a \frac{d\mathbf{a}}{dt} = -\mathbf{a} + \mathbf{f}(\mathbf{W}^{rec}\mathbf{a} + \mathbf{W}^{ext}\mathbf{a}_{ext}) \tag{3.15}$$

when the activities are low pass filtered. Note that here $\mathbf{W}^{rec}$, $\mathbf{W}^{ext}$ and $\mathbf{I}$ are different than in Eq. (3.6) although the same symbols are used.

When such assumptions on similarity of neurons on a specific population can be made, the number of computational units can be dramatically decreased. For example, if 1000 neurons are to be simulated, instead of simulating each neuron independently, ten populations size of 100 neurons can be simulated if the neurons are roughly characterized with ten different groups. In many modelling tasks this is implicitly assumed as an experimental recording accuracy might only support recording a firing rate of a group of neurons.

We have showed here how recurrently connected network models can be derived assuming independency of spike times and a simple exponential synaptic kernel. The approach [7] which we followed has some caveats; namely neglecting the ARP and not concerning second order moment statistics, that is, the variance of the firing rate. In the next chapter, it is shown how the ARP can be incorporated into the RNNs, and later in Section 3.4 the issue with higher order moment statistics will be discussed.

3.2 Original Wilson-Cowan model

In the last chapter population models were derived after forming RNNs that have a single neuron as the computational unit. Alternative approach is to consider a population as the computational unit right away. Wilson and Cowan derived a population model consisting two recurrently connected populations, from which one is inhibitory and the other is excitatory [25]. The Wilson-Cowan model is considered to be the first population model that included time-evolution of both excitatory and inhibitory populations leading to versatile dynamics, and therefore its contribution to the field is undisputed.

Wilson and Cowan assumed that the activity of the neural network at time $t + \tau$ can be defined by the proportion of sensitive neurons in the population at time t and by the proportion of neurons that receive excitation exceeding an action potential firing threshold per unit time. The sensitivity of neurons refers to the ARP denoted by t_r after an action potential during which neurons cannot be excited. These two conditions are assumed to be independent of each other thus giving the activity of the excitatory population [25]

$$a_e(t+\tau) = [1 - \int_{t-t_r}^{t} a_e(t')dt']f[I(t)], \tag{3.16}$$

where $\int_{t-t_r}^{t} a_e(t')dt'$ is the fraction of neurons on the refractory period and $f[I(t)]$ is population's response function, that is the activation function. The activation function is defined as

$$f[I(t)] = \int_{0}^{I(t)} D(\theta)d\theta,$$

where $D(\theta)$ is a distribution function describing different thresholds in the population and $I(t)$ is the average excitation coming to the population [25]. Assuming that $D(\theta)$ is unimodal, $f(I)$ can be simply taken to be a sigmoid-like function. At this point we do not have tools to define the response function in more detail. However, a recent semi-analytic approach can bring more clarity on this important nonlinearity [26], which will be discussed more in Section 3.4.

Similar to Eq. (3.1), the average excitation can be reasoned to be sum of incoming activities with the exponentially decaying synaptic kernel $K_s(t-t')$. As Wilson and Cowan explicitly considered the system of one inhibitory and one excitatory population, Eq. (3.16) takes the form

$$\begin{aligned} a_e(t+\tau) = {} & [1 - \int_{t-t_r}^{t} a_e(t')dt'] \\ & \times f\left(\int_{-\infty}^{t} K_s(t-t')\left[w_{ee}a_e(t') + w_{ei}a_i(t') + w_{e,ext}a_{e,ext}(t')\right]dt'\right). \end{aligned} \tag{3.17}$$

Here w_{ee} is a positive connection weight from the excitatory population to itself and w_{ei} is a negative connection weight from the inhibitory population to the excitatory population. For the inhibitory population a_i, one could write similar equation. The final step is to approximate the dynamics over a small time interval. Applying temporal coarse graining one finds the following approximations [25]:

$$\int_{t-t_r}^{t} a_e(t')dt' \approx t_r \bar{a}_e(t) \quad \text{and}$$
$$\int_{-\infty}^{t} K_s(t-t')w_{yz}a_z(t')dt' \approx k w_{yz}\bar{a}_z(t).$$

When τ is small applying Euler's discretization leads to a pair of equations which is known as the original Wilson-Cowan model [25]:

$$\tau \frac{da_e}{dt} = -a_e + (1 - t_r a_e) f(w_{ee}a_e + w_{ei}a_i + w_{e,ext}a_{e,ext}), \tag{3.18a}$$
$$\tau \frac{da_i}{dt} = -a_i + (1 - t_r a_i) f(w_{ie}a_e + w_{ii}a_i + w_{i,ext}a_{i,ext}). \tag{3.18b}$$

For simplicity, the constant k was included in the definition of w and bars denoting temporal coarse graining were removed. In fact, the temporally coarse grained version of activity responses better to the definition in Eq. (2.6) as it was defined through experimental settings. In addition, note that the activities were defined as per unit time in Ref. [25], which means that the activities can be also seen as probabilities as they are limited from 0 to 1 Hz. In practice, the maximal frequency can be changed by scaling the activation function.

Note that Eq. (3.18) does not consider spatial dynamics of the populations. Instead, it is assumed that the time spent on sending and receiving action potentials through spatial interactions in the populations is not considerable important factor that would dominate the dynamics. In the proceeding work of Wilson and Cowan [27] spatial relations were considered in addition to the temporal evolution of the activity. However, the model from Ref. [27] is not in the scope of interest so it will not be covered here. Contrary, we will continue to examine the original Wilson-Cowan model and generalize it to fit the formulation used in Section 3.1.

Original Wilson-Cowan model with several populations

Instead of explicitly dividing a mass of neurons into two subpopulations (excitatory and inhibitory) one can think N populations that are either excitatory or inhibitory. This makes modelling work flow more flexible and can expand the use of the model to larger amount of neurons.

Considering N recurrent populations leads to a general form

$$\tau_a \frac{da_j}{dt} = -a_j + (1 - t_r a_j) f(\sum_{k=1}^{N} w_{jk}a_k + \sum_{ext=1}^{E} w_{j,ext}a_{ext}) \tag{3.19}$$

or with the matrix notation

$$\tau_a \frac{d\mathbf{a}}{dt} = -\mathbf{a} + (1 - t_r\mathbf{a})\mathbf{f}(\mathbf{W}^{rec}\mathbf{a} + \mathbf{W}^{ext}\mathbf{a}_{ext}). \tag{3.20}$$

Here it is easy to see that Eq. (3.20) is exactly the same as Eq. (3.15) when the ARP is set to zero. This is in align with the Poisson spike train assumption made in Section 3.1 which neglects the ARP.

3.3 Grossberg's formulation

The previous definitions of RNNs can be found from a few distinct sources. Namely, Grossberg developed such models early on in the late 60's and early 70's (see for example Ref. [28] amongst others collected in Ref. [1]). Grossberg named Eq. (3.7) as the additive Short-Term Memory (STM), while the Wilson-Cowan model is closely related to a shunting STM model [1]

$$\begin{aligned}\frac{da_j}{dt} &= -A_j a_j + (B_j - a_j)[\sum_{k=1}^{N} f_k(a_k)C_{jk}w_{jk}^{+} + I_{ext}^{+}] \\ &\quad - (D_j + a_j)[\sum_{k=1}^{N} g_k(a_k)E_{jk}w_{jk}^{-} + I_{ext}^{-}],\end{aligned} \tag{3.21}$$

where $A - E$ are constants with no particular physical meaning, f_k and g_k are activation functions, and w^{+} and w^{-} are positively valued Long-Term Memory (LTM) traces. These LTM traces are adaptive weights that may evolve through time with respect to learning [1]. Nowadays, instead of using Hebbian learning [29], the LTM traces are usually interpreted simply as trainable weight parameters. The second term in Eq. (3.21) describes positive connections while the third term describes inhibitory connections. I_{ext}^{+} and I_{ext}^{-} describe external inputs in a general level. The model is called the shunting STM because the external inputs are multiplied with the activity. Therefore, an automatic gain control is included in the model.

By setting $B = C = D = E = 1$ and $A = 1/\tau_a$ and considering the same activation function for each unit, Eq. (3.21) simplifies to

$$\frac{da_j}{dt} = -\frac{1}{\tau_a}a_j + (1 - a_j)[\sum_{k=1}^{2N} w_{jk}f(a_k) + \sum_{ext=1}^{E} w_{j,ext}a_{ext}], \tag{3.22}$$

which is somewhat a mix of the additive system and the original Wilson-Cowan system with $t_r = 1$. However, the shunting system has not been used widely, and therefore will not be considered in more detail.

Grossberg's methods differ greatly from the approach of Dayan and Abbott and from the approach of Wilson and Cowan. Grossberg approaches RNNs as psychological machines that evolve through time. Therefore, the early derivation of STM models by Grossberg follows psychological guidelines rather than physiological constraints [28]. Despite different approaches the results stand very similar, which underlies applicability of the RNNs on various workflows.

3.4 Alternative approach: master equation formalism

As shown in Section 3.1 and Section 3.2, the typical RNNs only include the first order moment statistics. However, recent developments in the field aims to create mean-field models that are more detailed and biologically more fascinating. In the work of El Boustani and Destexhe [30] a new population model is derived based on the Markov property. Briefly, it is assumed that during a time step τ a network of neurons can emit a maximum of one spike, and the probability for emitting the spike is only depended on the current state of the network. Therefore, choosing an appropriate time step τ is crucial for this approach. Arguably, choosing the AI state as modelled target state is biologically meaningful as it is reported in adult mammalian awake cortical states [31], thus leading to $\tau = 20$ms as $1/\tau$ gives the maximum activity of neurons [9].

Skipping all arduous derivation, that is present with this stochastic approach, one arrives with a set of mean-field equations [30]

$$\tau \frac{d\mathbf{a}}{dt} = -\mathbf{a} + \mathbf{f} + \frac{1}{2}\langle \mathbf{C}, \mathbf{H_f} \rangle_F, \tag{3.23a}$$

$$\begin{aligned}\tau \frac{dc_{jk}}{dt} = &-2c_{jk} + \delta_{jk}\frac{f_j(1/\tau - f_j)}{n_j} + (f_j - a_j)(f_k - a_k) \\ &+ \sum_{i=1}^{N} c_{ki}\frac{\partial f_j}{\partial a_i} + \sum_{i=1}^{N} c_{ji}\frac{\partial f_k}{\partial a_i}\end{aligned} \tag{3.23b}$$

where $\langle \mathbf{C}, \mathbf{H_f} \rangle_F$ is a vector valued Frobenius inner product of the covariance matrix $\mathbf{C}$ and Hessian matrix $\mathbf{H_f}$ of the activation functions $\mathbf{f}$ with respect to activities a_{jk}. δ denotes Kronecker delta and n_j is the number of neurons is a population j. The time evolution of covariance terms is described by the second equation. Here we immediately see that when the covariances are not concerned, then Eq. (3.23) simplifies to the Wilson-Cowan system.

An important difference here to other models is the definition of the activation functions $\mathbf{f}$. Previously $\mathbf{f}$ was said to be a vector of linear functions with thresholds or a vector of sigmoid-like functions, and f was assumed to be same for every unit in the network. Now, however, f is estimated more carefully from the underlying network and is different for each population. In more detail, a user of the model chooses a spiking neural network on which the estimation of f is based on. In a general form, f is defined as [26]

$$f = \frac{1}{2\tau_V}\text{erfc}(\frac{V_{thr}^{eff} - \mu_V}{\sqrt{2}\sigma_V}), \tag{3.24}$$

where erfc$(\cdot)$ is a complementary Gaussian error function, τ_V is population's average autocorrelation time, μ_V is the average membrane voltage, and σ_V its standard deviation. Each statistical term can be expressed in terms of populations' activities [9]. Effective action potential firing threshold V_{thr}^{eff} is estimated from the underlying single cell model through a parameter fitting procedure [26]. There is no limitations on which spiking model the mean-field model is based on as long as the activation functions can be estimated (see Ref. [9] for various models).

Important consequence of this model fitting procedure is that there is not anymore a connectivity matrix to learn. Instead, all parameters are bound to the underlying properties of the spiking neural network. However, the mean-field models can be recurrently connected when simulating a large scale dynamics, that is, dynamics of the whole brain or brain areas. With this approach the network architecture is usually combination of recurrently connected subpopulations. This means that one population is described usually by one excitatory and one inhibitory subpopulation, so that **a**=$[a_e\, a_i]^T$ describes one population, and then populations are connected together with connection weights. In principle, these connection weights can be trained. However, depending on the research question connection weights might be evaluated in some other way, for example see Ref. [23]. As a conclusion, use cases of Eq. (3.23) remains open and therefore it will not be studied further here. Instead, the focus of local stability analysis will be on the additive system and on the Wilson-Cowan system.

4. METHODS

The additive system (Eq. (3.7) or Eq. (3.14)) and the Wilson-Cowan system (Eq. (3.9) or Eq. (3.15)) will be analyzed in Chapter 5. In this chapter, methods are described for numerical simulations and for local stability analysis.

4.1 Simulations

The additive system and the Wilson-Cowan system can be solved as an initial value problem. In short, a nonlinear dynamical system can be presented as ODEs of the general form

$$\frac{d\mathbf{z}(t)}{dt} = \mathbf{g}(\mathbf{z}(t), \mathbf{r}_{ext}(t), t, \mathbf{W}), \tag{4.1}$$

where $\mathbf{g}(\mathbf{z}(t), \mathbf{r}_{ext}(t), t, \mathbf{W}) : \mathbb{R}^N \rightarrow \mathbb{R}^N$ is a vector of functions, $\mathbf{W}$ refers to parameters, $\mathbf{r}_{ext}(t)$ is time varying external inputs and $\mathbf{z}(t)$ is the vector of simulation variables to be solved through time. Eq. (4.1) can be solved through time given initial values $\mathbf{z}(0)$. Different values for $\mathbf{z}(0)$ were used depending on the simulation (see Section 5.3).

The simulations were carried out with MATLAB R2020b. Differential equations were solved with `ode45` solver that uses the Dormand-Prince method [32]. The Dormand-Prince method is an explicit adaptive step size method that is commonly used for solving nonstiff ODEs. The default relative error tolerance 1e-3 and the absolute error tolerance 1e-6 were used.

4.2 Finding fixed points

Finding fixed points starts by setting $\frac{d\mathbf{a}}{dt} = 0$ (or $\frac{d\mathbf{I}}{dt} = 0$). For example, the two dimensional Wilson-Cowan system with a non-zero ARP becomes:

$$a_i = \frac{f^{-1}(\frac{a_e}{1-t_r a_e}) - w_{ee}a_e - w_{e,ext}a_{e,ext}}{w_{ei}}, \tag{4.2a}$$

$$a_e = \frac{f^{-1}(\frac{a_i}{1-t_r a_i}) - w_{ii}a_i - w_{i,ext}a_{i,ext}}{w_{ie}}, \tag{4.2b}$$

where f^{-1} is the inverse of the activation function f assuming that f is invertible. Clearly, even with only a two dimensional system, analytical solutions are hard to find.

Luckily, fixed points can be found with numerical solvers. Used numerical solvers were `vpasolve` and `fsolve`. Both methods return only a one solution per initialization so many initial guesses are needed. The number of initial guesses was varied from hundreds to thousands. Both solvers gave the same results. However, `fsolve` provides more tools to adjust numerical accuracy why it was favoured. Default algorithm 'trust-region-dogleg' was used with the relative function tolerance, the relative step tolerance and the absolute optimality tolerance each set to 1e-9.

4.3 Linearization

In order to study the asymptotic stability of fixed points linearization is needed. According to the Hartman-Grobman theorem [33] a nonlinear system and a linearized system in a small neighborhood of a hyperbolic equilibrium are topologically equivalent. The hyperbolic equilibrium is a fixed point with eigenvalues λ that have non-zero real parts, and the topological equivalence loosely means that the flow of the linearized system and the flow of the nonlinear system (here **a** or **l**) can be mapped to each other without losing a direction of time [10]. Therefore, the previous statement implies that if all the eigenvalues of a linearized system's Jacobian matrix evaluated at a fixed point have a non-zero real part, then the local stability of the nonlinear system can be understood by linearizing the nonlinear system.

In more detail, ODEs should be linearized around fixed points, and then eigenvalues of the Jacobian matrix evaluated at the fixed points should be inspected. The following rules apply for a fixed point with eigenvalues λ [7]:

- If $\forall\lambda, \Re(\lambda) < 0$, then the fixed point is stable.
- If $\forall\lambda, \Re(\lambda) > 0$, then the fixed point is unstable.
- If $\exists\lambda, \Re(\lambda) < 0$ and $\exists\lambda, \Re(\lambda) > 0$ and $\forall\lambda, \Re(\lambda) \neq 0$, then the fixed point is saddle.
- If $\exists\lambda, \Re(\lambda) = 0$, then nonlinear terms define stability of the fixed point and therefore linearization of the fixed point will not guarantee qualitatively correct results.
- If $\exists\lambda, \Im(\lambda) \neq 0$, then the fixed point has an oscillatory behaviour.

With these rules the asymptotic stability can be understood. We will refer to these rules later in Section 5.3.

5. LOCAL STABILITY ANALYSIS

5.1 Fixed points of two-dimensional system and learning

Stability of a two-dimensional system can be analysed with phase planes [7, 25]. As an example, the phase plane of Eq. (3.18) is shown in Figure 5.1. The vector field in the figure is obtained by calculating the time derivatives of **a** at multiple values with $t = 0$. Fixed points are found by setting $\frac{d\mathbf{a}}{dt} = 0$ and solving for **a** numerically as described in Section 4.2. The fixed points can be also visually found on the intersections of the nullclines (Eq. (4.2)).

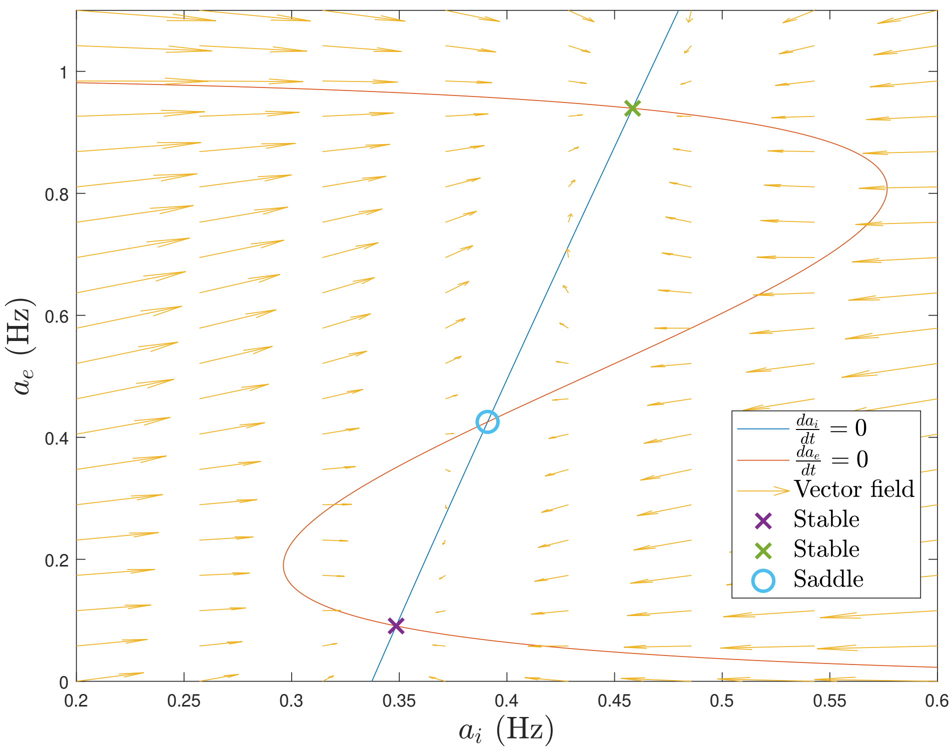

Figure 5.1. *Phase plane and nullclines (Eq.* (4.2)*) of the two dimensional Wilson-Cowan system with a non-zero ARP (see Eq.* (3.18)*). Stability of the fixed points was solved from eigenvalues of the Jacobian matrix evaluated at fixed points. Parameter values used in the simulation were:* $w_{ee} = 6.5$, $w_{ei} = -4$, $w_{ie} = 0.8$, $w_{ii} = -2$, $t_r = 2$ *ms and* $\tau = 10$ *ms. External inputs and external input weights were replaced with constant values -1.5 Hz and 0 Hz, excitatory and inhibitory, respectively. Activation function was chosen to be a sigmoid function* $f = \frac{1}{1+e^{-x}}$. *The code to reproduce this figure is given at Appendix B.*

Stability of the fixed points can be qualitatively understood by studying the vector field. Vectors tend to get smaller near the fixed points implicating the speed of change is getting slow. From the direction of the vectors one can deduce that given initial values $\mathbf{a}(0)$, the result collapses on either stable fixed point when time passes on. Clearly, from Figure 5.1 it can be seen that the middle fixed point is attracting in a one dimension while being repulsive on the other dimension. Therefore, it is a saddle point.

In such parameter regime where multiple stable fixed points exist, consistent initialization of activities is important. For example, if the system shown in Figure 5.1 is used for a classification task, random initialization of activities $\mathbf{a}(0)$ could result to misclassification. More formally, such set of initial values that drive dynamics to a stable fixed point is called the basin of attraction, and when the activities are initialized outside of the target basin of attraction, then misclassification may occur.

When a stable fixed point or multiple stable fixed points exist in a system, the system can be used as a learning machine. Different external inputs, representing for example images, will lead to different fixed points. These fixed points can be memorized and labeled. When a new external input is introduced into the system, a new fixed point is found. By calculating the smallest error between the new fixed point and old fixed points, we can label the new data. If we know that the result we obtained was wrong, we can adjust the synaptic weights so that the system will produce the correct result. This is generally known as a supervised learning. As an example, the system in Figure 5.1 could be taught to memorize a binary cognitive selection task from electrophysiological measurements. As a more concrete example, in Ref. [34] recordings from the prefrontal cortex of two macaque monkeys were used to train the additive system to discriminate between the motion and the colour of a random-dot visual stimulus.

While the phase planes provide a good intuition on how systems behave, more advanced methods are needed when dimensionality of a system is high. In the next sections, behaviour of RNNs will be studied with the linearization around a hyperbolic fixed point.

5.2 Local stability of N-dimensional RNNs

Next, we will focus on the local stability of the additive system and the Wilson-Cowan system with the ARP set to zero. The ARP is set to zero because models with the ARP having other value than zero have seen hardly any use in modern computational tasks. If it is assumed that external inputs $\mathbf{r}_{ext}(t)$ are time-invariant and initial values are in a basin of some attracting stable fixed point, then systems in Eq. (3.7) and Eq. (3.9) will eventually reach to a steady state solution. One can find the steady state solutions and other fixed points by setting $\frac{d\mathbf{r}}{dt} = 0$. For the Wilson-Cowan system (3.9) one finds that

$$\mathbf{r} = \mathbf{f}(\mathbf{W}^{rec}\mathbf{r} + \mathbf{W}^{ext}\mathbf{r}_{ext}) \tag{5.1}$$

and for the additive system

$$\mathbf{I} = \mathbf{W}^{rec}\mathbf{f}(\mathbf{I}) + \mathbf{W}^{ext}\mathbf{r}_{ext}. \tag{5.2}$$

Recapitulating that for the additive system $r_j = f[I_j(t)]$ holds true one can see that Eq. (5.1) and Eq. (5.2) are equivalent. Therefore, locations of the fixed points are the same for both systems.

To solve for Jacobians, one can calculate few elements of it and easily see the general form (see Appendix A for details). For the additive system one finds that

$$\begin{aligned}\mathbf{J}_{add} &= -\mathbf{P} + \mathbf{W}^{rec}\Phi_{add},\\ \Phi_{add,j} &= f'[I_j(t)],\end{aligned}$$

where **P** is $N \times N$ identity matrix and Φ_{add} is a diagonal matrix consisting of derivatives of activation functions. The Jacobian matrix for the Wilson-Cowan system is given as

$$\begin{aligned}\mathbf{J}_{WC} &= -\mathbf{P} + \mathbf{W}^{rec}\Phi_{WC},\\ \Phi_{WC,j} &= f'\left[\sum_{k=1}^{N} w_{jk}r_k(t) + \sum_{ext=1}^{N'} w_{j,ext}r_{ext}(t)\right].\end{aligned}$$

Although the Jacobians may seem as different at the first glance, they are the same when evaluated at the fixed points. This can be easily seen by inserting Eq. (5.2) into the Jacobian of the additive system. Because the Jacobians are the same when evaluated on the fixed point, the asymptotic stability near the hyperbolic fixed point is qualitatively same between the models (see Section 4.3). If external inputs are time-invariant, it is likely that the additive system and the Wilson-Cowan system will reach a stable steady state. Because of these circumstances the additive system and the Wilson-Cowan system evolve similarly through time when external inputs are time-invariant.

5.3 Five-dimensional example

As a demonstration of the RNNs dynamics, networks with 5 units receiving time-invariant external inputs were simulated with respect to time. Both systems have the same connectivity matrix

$$\mathbf{W}^{rec} = \begin{bmatrix} -0.0829 & -0.0029 & 0.1351 & 0.0774 & -0.0966 \\ 0.2379 & -0.2140 & 0.2123 & 0.2141 & -0.3315 \\ 0.1369 & 0.1456 & -0.0858 & 0.1542 & 0.2402 \\ 0.1721 & -0.2237 & -0.0644 & 0.1274 & 0.1631 \\ 0.0096 & -0.2021 & -0.1910 & 0.1389 & -0.1448 \end{bmatrix}$$

and the external inputs

$$\mathbf{W}^{ext}\mathbf{r}_{ext} = \begin{bmatrix} 0.4 \\ 1.0 \\ -3.5 \\ 0.7 \\ 1.2 \end{bmatrix}$$

while the activation function was chosen to be

$$f(x) = \gamma(\tanh(x-2)+1), \tag{5.4}$$

for both models. Here $\gamma = 25$ Hz so that activity is scaled from 0 Hz to 50 Hz. The neuroscientific motivation for choosing such an activation function is to consider the maximal activity, and the activity when zero input is present.

The derivative of the activation function is

$$\frac{df(x)}{dx} = \gamma \text{sech}^2(x-2)$$

and therefore the Jacobian evaluated at the fixed point **I*** is

$$\begin{aligned} \mathbf{J} &= -\mathbf{P} + \mathbf{W}^{rec}\Phi, \\ \Phi_j &= \gamma \text{sech}^2(I_j{}^* - 2). \end{aligned} \tag{5.5}$$

I* can be obtained from the additive model or calculated as $\mathbf{I}^* = \mathbf{W}^{rec}\mathbf{r}^* + \mathbf{W}^{ext}\mathbf{r}_{ext}$, where **r*** is evaluated from the Wilson-Cowan model. The similarity was tested to be true with our example (see Appendix C).

Three fixed points can be found with this setup using methods described in Section 4.2. Fixed points and corresponding eigenvalues are shown in Table 5.1.

Table 5.1. *Fixed points and the local stability of the test system. The eigenvalues are obtained by evaluating the Jacobian matrix Eq.* (5.5) *at the corresponding fixed point. The local stability and oscillatory behaviour are evaluated from the eigenvalues according to the rules described in Section 4.3*

Fixed point 1		Fixed point 2		Fixed point 3	
Location (Hz)	Eigenvalues	Location (Hz)	Eigenvalues	Location (Hz)	Eigenvalues
45.87	-1.640	16.79	-7.779	2.440	0.099+1.198i
50.00	-0.993	32.66	-6.250	6.002	0.099-1.198i
49.97	-1.000	20.32	2.264	0.289	-1.435
0.000	-1.000	0.000	-1.000	7.150	-2.126
0.000	-1.000	0.000	-1.000	2.668	-3.501
Stable		Saddle		Saddle/Oscillatory	

From Table 5.1 one can deduce that if the firing rates/activities are not initialized close to the fixed point 3, then the system will eventually reach the stable equilibrium (fixed point 1). Reasoning behind this is that the fixed point 1 is the only stable fixed point in the system so every initial guess should collapse to it. However, the oscillatory saddle point might act as unstable source for a periodic solution, and therefore some initial guesses might lead to the periodic solution. Figure 5.2 illustrates an initialization where activities are not initialized close to the fixed point 3.

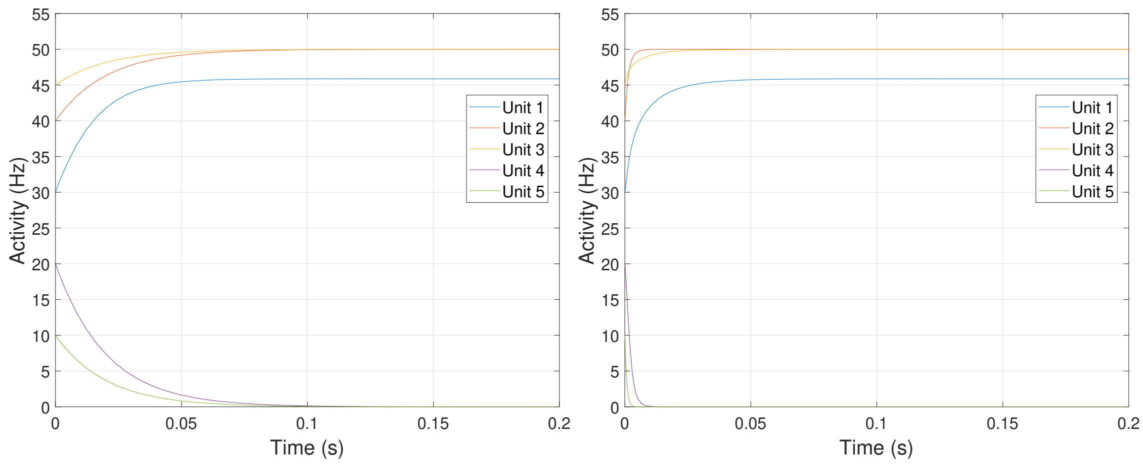

Figure 5.2. *Time evolution of the Wilson-Cowan system with 5 units (left), and time evolution of the additive system with 5 units (right). Both systems have the same parameter values. The time constants were set to $\tau_a = \tau_s = 20$ ms. For the Wilson-Cowan system the activities were initialized to $\boldsymbol{a}(0) = [30\ 40\ 45\ 20\ 10]^T$ Hz, and for the additive system the currents were initialized to $\boldsymbol{I}(0) = \boldsymbol{f}^{-1}(\boldsymbol{a}(0))$, where f^{-1} is the inverse function of Eq.* (5.4).

As discussed in Section 5.2, localization of hyperbolic fixed points and the local stability of those fixed points are the same for the additive system and for the Wilson-Cowan system. This phenomenon can be seen from the simulations in Figure 5.2 as both systems reach the same equilibrium. However, the simulations are not exactly the same. The Wilson-Cowan system reaches the equilibrium much slower than the additive model although the time constants are the same for both systems. This is not surprising as the additive system responses to incoming synaptic current instantaneously while the Wilson-Cowan system is low pass filtered (see Section 3.1).

In the second simulation shown in Figure 5.3, the activities were initialized to the fixed point 3 in order to inspect oscillatory behaviour with the current parameter setting. From Figure 5.3 it can be seen that amplitudes of the oscillations start to increase as time progresses, which can be understood by noting the positive signs of real part of complex eigenvalues (Table 5.1). However, approximately at $t = 2$ s, increase of the amplitudes saturates, while frequencies of the oscillations seem to stay constant. Outward spiraling ends because the activation function f is bounded and therefore a limit cycle is obtained [7].

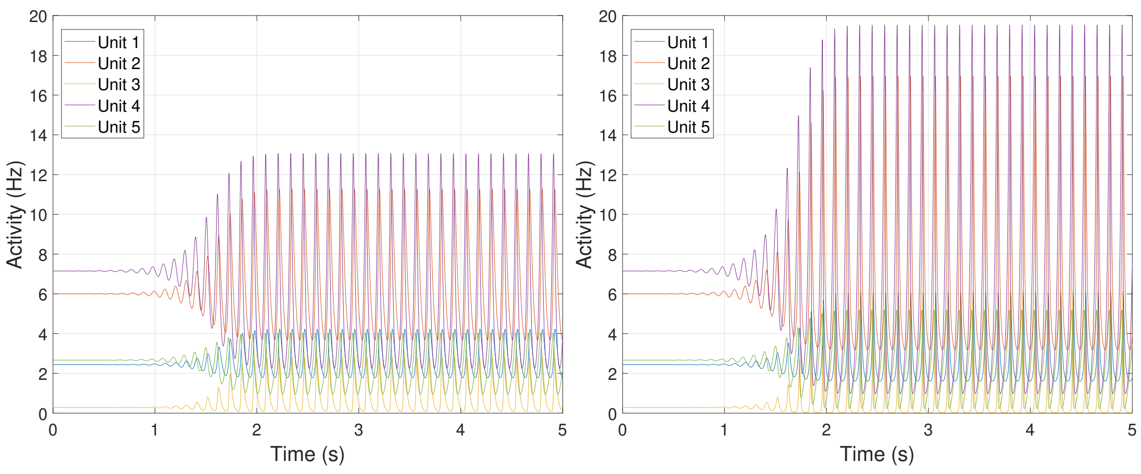

Figure 5.3. *Simulation of the Wilson-Cowan system with 5 units (left) and simulation of the additive system with 5 units (right). Both systems have the same parameter values. The systems in Figure 5.2 are exactly the same as here but now the activities were initialized to the oscillatory saddle point (Table 5.1).*

The amplitudes of the oscillations on the additive system clearly reach higher values than on the Wilson-Cowan system. This observation is consistent with the similarity of fixed points between the systems, because both systems evolve qualitatively in a similar manner near the fixed point. To reproduce the results of this section see Appendix C.

6. DISCUSSION AND CONCLUSIONS

The continuous-time rate-based RNNs are simple but powerful modelling tool [14] that can be used for almost any modelling task. In this work, the RNNs were studied from the perspective of neuroscience. It was shown how the RNNs can be derived using physiological basis of neurons. Two distinct RNNs were obtained; the additive system and the Wilson-Cowan system. The separating factor between these two models is their response to the synaptic current entering into the neuron. The additive system responds to incoming current instantaneously while the Wilson-Cowan system have slower dynamics as demonstrated in Figure 5.2. In addition to these two models, also other not so common RNNs were briefly covered.

It is not clear which model should be used in a given modelling task. The IF neurons have a fast response to a step current input [8] which indicates that the additive model is better fit when the base unit of a network is neuron. In addition, the synaptic inputs already lag behind the firing rates of presynaptic neurons so extra low pass filtering should not be needed. However, the lag from the firing rate of presynaptic neurons to the postsynaptic current is not always enough [7]. As a mention, neurons are not a homogeneous group and therefore lots of information is lost when assumptions are based on a such simple model as the IF neuron, and therefore these observations are only indicative.

If a population of neurons is used as the computational base unit of a network, the selection of a model becomes even more challenging. In such cases it is hard to argue how much neurophysiology is present in the model because of aggressive mean-field assumptions that are needed. To address this gap, a new group of recurrently connected population models have been developed [30] as discussed briefly in Section 3.4. A formalism obtained by the approach can be seen as continuum to work of Wilson and Cowan [25]. Therefore, the low pass filtered version of the RNNs seems to be more firmly backed up than the additive model when the computational base unit of a network is a population of neurons.

In addition to methods introduced in Section 3.4, the RNNs can be molded closer to biological reality by augmenting the synaptic kernel and constraining the parameter space of the connectivity matrix. The synaptic kernel can be modified to account for synaptic plasticity rules, for example the Tsodyks-Markram dynamics [22]. These types of models are still considered as phenomenological models but are certainly more detailed in a biological perspective than the RNNs without synaptic plasticity rules. The other practice to include biological constraints is to consider connectivity rules as done in Ref. [24] and discussed in Section 3.1. How these modifications affect to dimensionality and the dynamics of the system remains as an interesting question.

As we have seen the classical RNNs are biologically ambiguous. However, this does not mean that the RNNs would be insufficient models. The power of the RNNs is within the simplicity of the models. In a network level, the connectivity of neurons, or populations, is the most interesting subject of study. Different connectivity settings can lead to versatile dynamics, and these dynamics can arguably reveal and explain hidden mechanisms on which the brain operates. The dynamics of the RNNs is best understood by the stability analysis which was briefly studied. In addition to methods described here, I advise the reader to familiarize with linear and nonlinear dimension reduction methods as these methods together with the stability analysis form the basis for understanding neural manifolds.

In Section 5.1 the Wilson-Cowan system with a non-zero ARP value was briefly studied with a phase plane analysis. The analysis clarified what is the meaning of fixed points' stability and how stable fixed points are related to the learning capability of RNNs. In addition to methods shown in Section 4.3 and Section 5.1, the asymptotic stability and the Lyapunov stability can be studied with Lyapunov functions [10] as done in Refs. [16] and [17]. However, the use of Lyapunov functions in order to guarantee the stability of fixed points generally requires that $\mathbf{W}^{rec}$ is symmetric [17]. This is not biologically plausible as the symmetry of $\mathbf{W}^{rec}$ violates Dale's principle.

In Section 5.2 it was shown that the additive system and the Wilson-Cowan system without the ARP behave similarly on hyperbolic fixed points. In more detail, it was shown that fixed points have the same coordinates for both systems. In addition, it was noticed that the Jacobian matrices evaluated at the fixed points are the same. These properties were illustrated with an example in Section 5.3. Because of the similarity between the models when the external inputs are time-invariant, choosing between the additive system and the Wilson-Cowan system is quite trivial on learning tasks where it is assumed that dynamics will eventually approach to a stable steady state given the time-invariant external inputs. Note that similarities of basin of attractions between the models was not compared in Section 5.2. Therefore, models might lead to different fixed points with a set of initial values. A separate study is needed for comparing the basin of attractions for these two models.

Because the additive model converges more quickly than the Wilson-Cowan model, the additive model should be used in time-invariant learning tasks. Because the simulation time can be made shorter with the additive system, it provides computationally more efficient learning. However, when considering computational efficacy one must also consider the properties of an optimizer. Optimizing the continuous-time RNNs, that is tuning the connectivity parameters, is a future topic that is not discussed here. In short, optimizing continuous-time RNNs is extremely challenging, especially over long periods of time, which may affect on the decision when choosing the model. Also notice that when the external inputs vary in time, the models may not behave similarly.

To conclude, we have discussed about theoretical motivations behind the continuous-time rate-based RNNs and briefly studied the dynamics of RNNs with time-invariant external inputs. The results of this thesis help to understand the dynamics of the systems, as well as help to choose the right RNN for neuroscientifically interesting computational tasks. These tasks contain modelling of small networks and large population networks revealing low-dimensional neural manifolds that can arguably explain fundamental brain mechanisms.

REFERENCES

[1] S. Grossberg, “Nonlinear neural networks: Principles, mechanisms, and architectures,” *Neural networks*, vol. 1, no. 1, pp. 17–61, 1988.

[2] J. Martens and I. Sutskever, “Learning recurrent neural networks with hessian-free optimization,” *Proceedings of the 28th International Conference on Machine Learning*, pp. 1033–1040, 2011.

[3] B. DePasquale, C. J. Cueva, K. Rajan, G. S. Escola, and L. F. Abbott, “Full-force: A target-based method for training recurrent networks,” *PLOS ONE*, vol. 13, no. 2, pp. 1–18, 2018.

[4] R. Laje and D. V. Buonomano, “Robust timing and motor patterns by taming chaos in recurrent neural networks,” *Nature neuroscience*, vol. 16, no. 7, pp. 925–933, 2013.

[5] W. Chaisangmongkon, S. K. Swaminathan, D. J. Freedman, and X.-J. Wang, “Computing by robust transience: How the fronto-parietal network performs sequential, category-based decisions,” *Neuron*, vol. 93, no. 6, 1504–1517.e4, 2017.

[6] M. Dingman, *Ion channel, neuroscientifically challenged*. [Online]. Available: `https://neuroscientificallychallenged.com/glossary/ion-channel` (visited on Nov. 10, 2021).

[7] L. F. Abbott and P. Dayan, *Theoretical Neuroscience: Computational and Mathematical Modelling of Neural Systems*. MIT Press, 2001.

[8] W. Gerstner, W. M. Kistler, R. Naud, and L. Paninski, *Neuronal Dynamics: From Single Neurons to Networks and Models of Cognition*. Cambridge University Press, 2014.

[9] M. Carlu, O. Chehab, L. D. Porta, D. Depannemaecker, C. Héricé, M. Jedynak, E. K. Ersöz, P. Muratore, S. Souihel, C. Capone, Y. Zerlaut, A. Destexhe, and M. di Volo, “A mean-field approach to the dynamics of networks of complex neurons, from nonlinear integrate-and-fire to hodgkin–huxley models,” *Journal of Neurophysiology*, vol. 123, no. 3, pp. 1042–1051, 2020.

[10] J. D. Meiss, *Differential Dynamical Systems*. Society for Industrial and Applied Mathematics, pp. 105–164.

[11] R. Chaudhuri, B. Gerçek, B. Pandey, A. Peyrache, and I. Fiete, “The intrinsic attractor manifold and population dynamics of a canonical cognitive circuit across waking and sleep,” *Nature Neuroscience*, vol. 22, no. 9, pp. 1512–1520, 2019.

[12] E. Pollock and M. Jazayeri, “Engineering recurrent neural networks from task-relevant manifolds and dynamics,” *PLOS Computational Biology*, vol. 16, no. 8, pp. 1–23, 2020.

[13] D. Sussillo and O. Barak, “Opening the black box: Low-dimensional dynamics in high-dimensional recurrent neural networks,” *Neural Computation*, vol. 25, no. 3, pp. 626–649, 2013.

[14] K.-i. Funahashi and Y. Nakamura, “Approximation of dynamical systems by continuous time recurrent neural networks,” *Neural Networks*, vol. 6, no. 6, pp. 801–806, 1993.

[15] H. Flanders, “Differentiation under the integral sign,” *The American mathematical monthly*, vol. 80, no. 6, pp. 615–627, 1973.

[16] J. J. Hopfield, “Neurons with graded response have collective computational properties like those of two-state neurons,” *Proceedings of the National Academy of Sciences*, vol. 81, no. 10, pp. 3088–3092, 1984.

[17] M. A. Cohen and S. Grossberg, “Absolute stability of global pattern formation and parallel memory storage by competitive neural networks,” *IEEE Transactions on Systems, Man, and Cybernetics*, vol. SMC-13, no. 5, pp. 815–826, 1983.

[18] R. D. Beer, “Parameter space structure of continuous-time recurrent neural networks,” *Neural computation*, vol. 18, pp. 3009–3051, 2006.

[19] K. Miller and F. Fumarola, “Mathematical equivalence of two common forms of firing rate models of neural networks,” *Neural computation*, vol. 24, pp. 25–31, 2012.

[20] B. Ermentrout, “Neural networks as spatio-temporal pattern-forming systems,” *Reports on progress in physics*, vol. 61, no. 4, pp. 353–430, 1998.

[21] D. Sussillo, “Neural circuits as computational dynamical systems,” *Current Opinion in Neurobiology*, vol. 25, pp. 156–163, 2014.

[22] M. Tsodyks, K. Pawelzik, and H. Markram, “Neural networks with dynamic synapses,” *Neural Computation*, vol. 10, no. 4, pp. 821–835, 1998.

[23] O. Barak and M. Tsodyks, “Persistent activity in neural networks with dynamic synapses,” *PLoS Computational Biology*, vol. 3, no. 2, K. J. Friston, Ed., pp. 323–332, 2007.

[24] H. F. Song, G. R. Yang, and X.-J. Wang, “Training excitatory-inhibitory recurrent neural networks for cognitive tasks: A simple and flexible framework,” *PLOS Computational Biology*, vol. 12, no. 2, pp. 1–30, 2016.

[25] H. R. Wilson and J. D. Cowan, “Excitatory and inhibitory interactions in localized populations of model neurons,” *Biophysical Journal*, vol. 12, no. 1, pp. 1–24, 1972.

[26] Y. Zerlaut, B. Teleńczuk, C. Deleuze, T. Bal, G. Ouanounou, and A. Destexhe, “Heterogeneous firing rate response of mouse layer v pyramidal neurons in the fluctuation-driven regime,” *The Journal of Physiology*, vol. 594, no. 13, pp. 3791–3808, 2016.

[27] H. R. Wilson and J. D. Cowan, “A mathematical theory of the functional dynamics of cortical and thalamic nervous tissue,” *Kybernetik*, vol. 13, no. 2, pp. 55–80, 1973.

[28] S. Grossberg, “Embedding fields: A theory of learning with physiological implications,” *Journal of Mathematical Psychology*, vol. 6, no. 2, pp. 209–239, 1969.

[29] D. O. Hebb, *The organization of behavior; a neuropsychological theory.* Wiley, 1949.

[30] S. E. Boustani and A. Destexhe, “A master equation formalism for macroscopic modeling of asynchronous irregular activity states,” *Neural Computation*, vol. 21, no. 1, pp. 46–100, 2009.

[31] A. Destexhe, M. Rudolph, and D. Paré, “The high-conductance state of neocortical neurons in vivo,” *Nature Reviews Neuroscience*, vol. 4, no. 9, pp. 739–751, 2003.

[32] J. Dormand and P. Prince, “A family of embedded runge-kutta formulae,” *Journal of Computational and Applied Mathematics*, vol. 6, no. 1, pp. 19–26, 1980.

[33] P. Hartman, *Ordinary differential Equations*. John Wiley and Sons, 1964.

[34] V. Mante, D. Sussillo, K. V. Shenoy, and W. T. Newsome, “Context—dependent computation by recurrent dynamics in prefrontal cortex,” *Nature*, vol. 503, no. 7474, pp. 78–84, 2013.

APPENDIX A: JACOBIAN MATRICES

Consider a nonlinear dynamical system shown in Eq. (4.1). The Jacobian matrix for the system is given as

$$\mathbf{J}_{add} = \begin{bmatrix} \frac{\partial g_1}{\partial z_1} & \cdots & \frac{\partial g_1}{\partial z_N} \\ \vdots & \ddots & \vdots \\ \frac{\partial g_N}{\partial z_1} & \cdots & \frac{\partial g_N}{\partial z_N} \end{bmatrix}.$$

Now, considering the additive system (3.14) one can calculate on-diagonal and off-diagonal terms. For the additive system $z_j(t) = I_j(t)$ and $g_j(t) = -I_j(t) + \sum_{k=1}^{N} w_{jk} f[I_k(t)] + \sum_{ext=1}^{N'} w_{j,ext} r_{ext}(t)$. Now, for the on-diagonal term one gets

$$\begin{aligned} \frac{\partial g_j}{\partial I_j} &= \frac{\partial}{\partial I_j}\left[-I_j(t) + \sum_{k=1}^{N} w_{jk} f[I_k(t)] + \sum_{ext=1}^{N'} w_{j,ext} r_{ext}(t)\right] \\ &= -1 + \sum_{k=1}^{N} \frac{\partial}{\partial I_j} w_{jk} f[I_k(t)] \\ &= -1 + w_{jj} \frac{\partial f[I_j(t)]}{\partial I_j}. \end{aligned}$$

For the off-diagonal term one gets

$$\begin{aligned} \frac{\partial g_j}{\partial I_l} &= \frac{\partial}{\partial I_l}\left[-I_j(t) + \sum_{k=1}^{N} w_{jk} f[I_k(t)] + \sum_{ext=1}^{N'} w_{j,ext} r_{ext}(t)\right] \\ &= \sum_{k=1}^{N} \frac{\partial}{\partial I_l} w_{jk} f[I_k(t)] \\ &= w_{jl} \frac{\partial f[I_l(t)]}{\partial I_l}. \end{aligned}$$

Therefore, one obtains the following Jacobian:

$$\mathbf{J}_{add} = \begin{bmatrix} -1 + w_{11}\frac{\partial f[I_1(t)]}{\partial I_1} & w_{12}\frac{\partial f[I_2(t)]}{\partial I_2} & \dots \\ w_{21}\frac{\partial f[I_1(t)]}{\partial I_1} & -1 + w_{22}\frac{\partial f[I_2(t)]}{\partial I_2} & \\ \vdots & & \ddots \end{bmatrix}. \tag{A.1}$$

This can be written in a simpler form by using an identity matrix **P** of size $N \times N$ and by recognizing that the synaptic weights form the recurrent connectivity matrix. Thus splitting Eq. (A.1) into three matrices by separating the identity matrix and using matrix product one obtains a final form

$$\begin{aligned} \mathbf{J}_{add} &= -\mathbf{P} + \mathbf{W}^{rec}\Phi_{add}, \\ \Phi_{add,j} &= f'[I_j(t)], \end{aligned}$$

where Φ_{add} is diagonal matrix containing the derivatives of the activation function. For simplicity, we denoted here $\frac{df(x)}{dx} = f'(x)$.

Next, we will calculate the Jacobian for the Wilson-Cowan system. Now we have $z_j(t) = r_j(t)$ and $g_j(t) = -r_j(t) + f\left[\sum_{k=1}^{N} w_{jk}r_k(t) + \sum_{ext=1}^{N'} w_{j,ext}r_{ext}(t)\right]$. With the Wilson-Cowan system we start similarly by calculating an on-diagonal term:

$$\begin{aligned} \frac{\partial g_j}{\partial r_j} &= \frac{\partial}{\partial r_j}\left\{-r_j(t) + f\left[\sum_{k=1}^{N} w_{jk}r_k(t) + \sum_{ext=1}^{N'} w_{j,ext}r_{ext}(t)\right]\right\} \\ &= -1 + \frac{\partial}{\partial r_j} f\left[\sum_{k=1}^{N} w_{jk}r_k(t) + \sum_{ext=1}^{N'} w_{j,ext}r_{ext}(t)\right] \\ &= -1 + w_{jj} f'\left[\sum_{k=1}^{N} w_{jk}r_k(t) + \sum_{ext=1}^{N'} w_{j,ext}r_{ext}(t)\right] \end{aligned}$$

Same process could be repeated for off-diagonal terms. Then, the Jacobian matrix for the Wilson-Cowan system is

$$\begin{aligned} \mathbf{J}_{WC} &= -\mathbf{P} + \mathbf{W}^{rec}\Phi_{WC}, \\ \Phi_{WC,j} &= f'\left[\sum_{k=1}^{N} w_{jk}r_k(t) + \sum_{ext=1}^{N'} w_{j,ext}r_{ext}(t)\right] \end{aligned}$$

as expected. Notice that we excluded the time constants from the derivation of the Jacobians. The time constants could be included but they are not necessary when only qualitatively examining eigenvalues of the Jacobians.

APPENDIX B: PHASE PLANE ANALYSIS

Matlab code for reproducing Figure 5.1. The Wilson-Cowan model is simulated with constant inputs and phase plane is plotted.

```
%% Parameters, all time dependent values are in seconds
clear all
global w1 w2 w3 w4 re ri ext_e ext_i tau_e tau_i
w1 = 6.5;
w2 = -4;
w3 = 0.8;
w4 = -2;
re = 0.002;
ri = 0.002;
ext_e = -1.5;
ext_i = 0;
tau_e = 0.010;
tau_i = 0.010;

% End time seconds
t1 = 0.200;

syms z
syms x

%% Init and run, solve through time
% (I, E)
init = [0, 0];

[t, y] = ode45(@(t, y) WC(t, y, w1, w2, w3, w4, re, ri, ext_e, ext_i, ...
    @g, tau_e, tau_i), [0, t1], init(:));

%% Plot results
clf
fig1 = figure(1);

subplot(1, 1, 1)
plot(t, y(:, :), '-')
ylabel('Activity a(t)')
xlabel("Time (ms)")
grid on
legend("Inhibitory population $a_i$", "Excitatory population $a_e$", ...
    "Interpreter", "latex", "Position", [0.7 0.8, 0.15, 0.05])

%% Solve nullclines
EI = linspace(0.01, 1, 1000);

I_solved = [];
E_solved = [];

t_I = [];
t_E = [];

for i = 1:length(EI)
    I_sol = NULLE(EI(i), w1, w2, @gInv, re, ext_e);
    if isreal(I_sol)
        t_I = [t_I EI(i)];
        I_solved = [I_solved I_sol];
    end


    E_sol = NULLI(EI(i), w3, w4, @gInv, ri, ext_i);
    if isreal(E_sol)
        t_E = [t_E EI(i)];
        E_solved = [E_solved E_sol];
    end
end
%% Plot nullclines
fontsize = 18;

fig2 = figure(2);
clf
subplot(1, 1, 1)
plot(t_E, E_solved, '-')
hold on
plot(I_solved, t_I, '-')
xlabel("$a_i$ (Hz)", "Interpreter", "latex", 'Fontsize', fontsize)
ylabel("$a_e$ (Hz)", "Interpreter", "latex", 'Fontsize', fontsize)


% Vector field
y1 = linspace(0.2, 0.6, 8);
y2 = linspace(0, 1.1, 20);

[mx, my] = meshgrid(y1, y2);

vec1 = zeros(size(mx));
vec2 = zeros(size(mx));
t=0;
for i = 1:numel(mx)
    res = WC(t, [mx(i), my(i)], w1, w2, w3, w4, re, ri, ext_e, ...
        ext_i, @g, tau_e, tau_i);
    vec1(i) = res(1);
    vec2(i) = res(2);
end
qui = quiver(mx, my, vec1, vec2);
set (qui, "Color",  [0.9290 0.6940 0.1250])
set(qui, "MaxHeadSize", 0.1)
qui.ShowArrowHead = "on";
xlim([0.2 0.6])
ylim([0 1.1])
legend("$\frac{da_i}{dt}=0$", "$\frac{da_e}{dt}=0$", "Vector field", ...
    'Interpreter','latex', "Position", [0.73 0.3, 0.15, 0.05], ...
    'Fontsize', fontsize-4)
%% Find fixed points
clear E I
clear sols
clear jacobs
syms E I
assume(E, "real")
assume(I, "real")
dedt=-E+(1-re*E)*g(w1*E+w2*I+ext_e);
didt=-I+(1-ri*I)*g(w3*E+w4*I+ext_i);

sols = [];
jacobs = [];
times = 500;    % Number of initial guesses

for i = 1:times
    fp = vpasolve([dedt==0, didt==0], [E, I], "Random", true);
    sol = [double(fp(1).I); double(fp(1).E)];
    sol = round(sol, 6);

    expi = exp(-(w3*sol(2)+w4*sol(1)+ext_i));
    expe = exp(-(w1*sol(2)+w2*sol(1)+ext_e));
```

```matlab

    p1j = (w2*expe*(1-re*sol(2)))/((expe + 1)^2);
    p2j = (w1*expe*(-re*sol(2) + 1))/(expe + 1)^2 - re/(expe + 1) - 1;
    p3j = (w4*expi*(-ri*sol(1) + 1))/(expi + 1)^2 - ri/(expi + 1) - 1;
    p4j = ((w3*expi)*(1-ri*sol(1)))/(expi + 1)^2;

    jacob = [p3j p4j; p1j p2j];

    if not(isempty(sols))
        if not(ismember(sol, sols))
            dims_sols = length(size(sols));
            if dims_sols == 3
                sols(:,:,size(sols, 3)+1) = sol;
                jacobs{end+1} = jacob;
            else
                sols(:,:,2) = sol;
                jacobs{end+1} =jacob;
            end
        end
    else
        sols = sol;
        jacobs = {jacob};
    end

end

for i = 1:size(sols,3)
    eig_values = eig(full(jacobs{i}));
    label = "Saddle";
    marker = "o";
    if not(isempty(eig_values))
        temp_val1 = round(eig_values(1), 2);
        temp_val2 = round(eig_values(2), 2);
        if temp_val1 == 0 || temp_val2 == 0
            error("NON-hyperbolic fixed point possible. Check numerical accuracy.")
        elseif real(eig_values(1)) < 0 && real(eig_values(2)) < 0
            label = "Stable";
            marker = "x";
        elseif real(eig_values(1)) > 0 && real(eig_values(2)) > 0
            label = "Unstable";
            marker = "s";
        end
        plot(sols(1,1,i), sols(2,1,i), marker, "DisplayName",  ...
            label, "Markersize", 12, 'linewidth', 2)
    end
end
%saveas(fig2, "stability/cw_sig_nullc", "epsc")

%% Gain-function

function gain = g(z)
    gain = 1/(1+exp(-z));
end

%% Gain-function inverse

function gain = gInv(x)
    gain = log(x/(1-x));
end

```

```
%% E nullcline
function nc = NULLE(E, w1, w2, gInv, re, ext_e)
    nc = (-w1*E-ext_e+gInv(E/(1-re*E)))/w2;
end
%% I nullcline
function nc = NULLI(I, w3, w4, gInv, ri, ext_i)
    nc = (-w4*I+gInv(I/(1-ri*I))-ext_i)/w3;
end

%% Wilson-Cowan model
function wc = WC(t, y, w1, w2, w3, w4, re, ri, ext_e, ext_i, g, tau_e, tau_i)
    % wc(1)=didt
    % wc(2)=dedt
    wc(2) = (-y(2) + (1-re*y(2))*g(w1*y(2)+w2*y(1)+ext_e))/tau_e;
    wc(1) = (-y(1) + (1-ri*y(1))*g(w3*y(2)+w4*y(1)+ext_i))/tau_i;

    wc = transpose(wc);
end

```

APPENDIX C: LOCAL STABILITY ANALYSIS

Matlab code for reproducing the simulations in Figure 5.2 and Figure 5.3 and to generate the data in Table 5.1. Downloaded connectivity matrix $\mathbf{W}^{rec}$ is the same matrix as given in Section 5.3.

```
clear all
clf()
% Time constants
tau_s = 0.020;
tau_a = 0.020;

n = 5;       % number of units
t1 = 0.2; %ending time, secs
tr = 0; % absolute refractory period

Iext = [0.4 1 -3.5 0.7 1.2]';
Iext_time = false;

Wrec = load("./Wrec_osc_n5_round.mat", "Wrec");
Wrec = [Wrec(1).Wrec];
syms z
%%

tspan = [0 t1];

% Example initialization for non-oscillatory dynamics
init_frate = [30; 40; 45; 20; 10];

% Example Initialization for oscillatory dynamics
%init_frate = [2.440; 6.002; 0.289; 7.150; 2.668;];

init_curr = finv(init_frate);


[t_add, y_add] = ode45(@(t, y) rnnAdditive(t, y, tau_s, Wrec, Iext, ...
                 n, @f, t1, Iext_time), tspan, init_curr(:));
[t_wc, y_wc] = ode45(@(t, y) rnnWC(t, y, tau_a, Wrec, Iext, n, @f, ...
                t1, Iext_time, tr), tspan, init_frate(:));
%% Plot results
fontsize = 20;

fig1 = figure(1);
subplot(1, 1, 1)
plot(t_add, f(y_add(:, 1:n)), '-')
set(gca,'FontSize',fontsize-4)
grid on
ylabel('Activity (Hz)', 'Fontsize', fontsize)
xlabel('Time (s)', 'Fontsize', fontsize)
ylim([0 55.0])
%ylim([0 20.0])
% legend non-oscillatory
legend("Unit 1", "Unit 2", "Unit 3", "Unit 4", "Unit 5", ...
                        "Position", [0.80, 0.62, 0.05, 0.05], ...
                        'Fontsize', fontsize-4)
% legend for oscillatory
%legend("Unit 1", "Unit 2", "Unit 3", "Unit 4", "Unit 5", ...
%             "Position", [0.18, 0.76, 0.05, 0.05], ...
%             'Fontsize', fontsize-4)
%title("Additive")
%saveas(fig1, "n5/additive_osc", "epsc")
%saveas(fig1, "n5/additive", "epsc")

fig2 = figure(2);
subplot(1, 1, 1)
plot(t_wc, y_wc(:, 1:n), '-')
```

```
set(gca,'FontSize',fontsize-4)
grid on
ylabel('Activity (Hz)', 'Fontsize', fontsize)
xlabel('Time (s)', 'Fontsize', fontsize)
% legend non-oscillatory
legend("Unit 1", "Unit 2", "Unit 3", "Unit 4", "Unit 5",...
            "Position", [0.80, 0.62, 0.05, 0.05], ...
            'Fontsize', fontsize-4)
% legend for oscillatory
%legend("Unit 1", "Unit 2", "Unit 3", "Unit 4", "Unit 5", ...
%            "Position", [0.18, 0.76, 0.05, 0.05], ...
%            'Fontsize', fontsize-4)
ylim([0 55.0])
%ylim([0 20.0])
%saveas(fig2, "n5/wc_osc", "epsc")
%saveas(fig2, "n5/wc", "epsc")

%% Find fixed points

y = sym('y',[n 1]);

dxdtf = @(y)(-y(1:n)+Wrec*f(y(1:n))+Iext)/tau_s; %ADDITIVE
dxdt2f = @(y)(-y(1:n)+(1-tr*y(1:n)).*f(Wrec*y(1:n)+Iext))/tau_a; %WC
models = {dxdtf, dxdt2f};


init_guesses = 500;
sols_add = [];   % fixed points
sols_wc = [];
sols = [];

% fsolve options
options = optimoptions('fsolve', 'FunctionTolerance', 1e-9, ...
        "StepTolerance", 1e-9, "OptimalityTolerance", 1e-9, ...
        "MaxIterations", 10000, "Display", "off");

for m = 1:size(models, 2)
    for i = 1:init_guesses

        % Different inital guesses
        if i > init_guesses/2
            init = randn(n, 1);
        else
            init = -50 + (50+50)*rand(n,1);
        end
        [sol, fval, exitflag, output, jacobian] = fsolve(models{m}, ...
                                                  init, options);

        sol = round(sol, 8);
        if exitflag == 1
            if not(isempty(sols))
                if not(ismember(sol, sols))
                    dims_sols = length(size(sols));
                    if dims_sols == 3
                        sols(:,:,size(sols, 3)+1) = sol;
                    else
                        sols(:,:,2) = sol;
                    end
                end
            else
```

```
                sols = sol;
            end
        end
    end
    if m==1
        sols_add = sols;
    else
        sols_wc = sols;
    end
end

%% Calculate Jacobians and eigenvalues

real_sols_add = f(sols_add);

eigs_add = [];
% Additive model
for i = 1:size(sols_add,3)
    Jadd = -eye(n)+Wrec*diag(Df(sols_add(:, :, i)));
    eig_add = eig(Jadd)
    eigs_add = [eigs_add, eig_add];
end

eigs_wc = [];
% Wilson-Cowan
for i = 1:size(sols_wc,3)
    if sols_wc(:,:,i)>=0 % should not be negative
        internal_sum = Wrec*sols_wc(:,:,i)+Iext;
        Jwc = -eye(n)+Wrec*diag(Df(internal_sum));
        eig_wc = eig(Jwc)
        eigs_wc = [eigs_wc, eig_wc];
    end

end


%% Activation function

function act = f(z)
    alpha = 25;
    act = alpha*(tanh(z-2)+1);
end

%% Activation function inverse

function act = finv(z)
    alpha = 25;
    act = atanh((z-alpha)/alpha)+2;
end
%% Activation function derivative

function act = Df(z)
    alpha = 25;
    act = alpha*(sech(z-2)).^2;

end
%% Additive RNN
function dy = rnnAdditive(t, y, tau_s, Wrec, Iext, n, f, t1, Iext_time)
    % If Iext should be 3D array (depends on time)
    if Iext_time
```

```
        ind = cast(1000*t, 'uint32');
        if ind > t1*1000
            ind = 1000*t1;
        end
        if ind == 0
            ind = 1;
        end
        Iext = transpose(Iext(ind, :));
    end

    dy(1:n) = (-y(1:n)+Wrec*f(y(1:n))+Iext)/tau_s;
    dy = transpose(dy);
end
%% Wilson-Cowan model
function dy = rnnWC(t, y, tau_a, Wrec, Iext, n, f, t1, Iext_time, tr)
    % If Iext should be 3D array (depends on time)
    if Iext_time
        ind = cast(1000*t, 'uint32');
        if ind > t1*1000
            ind = 1000*t1;
        end
        if ind == 0
            ind = 1;
        end
        Iext = transpose(Iext(ind, :));
    end

    dy(1:n) = (-y(1:n)+(1-tr*y(1:n)).*f(Wrec*y(1:n)+Iext))/tau_a;
    dy = transpose(dy);
end

```