Neural Dynamics on Graphs

Schriftliche Prüfungsarbeit
für die Master-Prüfung des Studiengangs
Angewandte Informatik
an der
Ruhr-Universität Bochum

vorgelegt von

Oliver Lomp
Matrikelnummer: 108005204827

Abgabedatum: 1. Dezember 2010

Erstprüfer: Rolf Würtz
Zweitprüfer: Christian Igel
Betreuer: Guillermo Sebastián Donatti

Abstract. Traditional neural models of the human brain fail to reproduce its complex behavior. In contrast to these models, neural dynamics define their state not only depending on current inputs, but also on previous ones, providing a framework that is able to model these properties more accurately. Neural fields describe the distribution of the activation of neuron populations in cortical tissue based on neural dynamics. Its neurons are positioned along a continuous feature dimension that can be used to solve tasks in computer vision. The computational requirements of neural fields are high, and applying them to the high-dimensional features used by many other computer vision algorithms is infeasible. The aim of the present work is to develop and analyze a model based on neural fields that can be used in connection with such high-dimensional data, the so-called neural field graph model. Its basis is a graph structure that can represent a clustering of such a high-dimensional feature space. The graph's nodes are assigned an activation value that is governed by the same principles as the neurons in the neural field. Several challenges arise from this approach. The synaptic interactions among the neurons in a neural field graph are influenced by the topology of the graph structure, and a method for balancing these interactions is given. Methods for the detection of the centers of active regions in the graph are provided. The limitations of these methods are examined, and experiments indicate that they fulfill their purpose in most cases. The neural field model appears to present a successful approach to applying neural dynamics on a graph structure. Some limitations remain that should be overcome in future work.

Contents

Preface

Notation

In the present work, vectors are printed in bold-face lower case letters (e.g., $\boldsymbol{y} = (y_1, y_2, \ldots, y_n)^T$). Special vectors are the $\mathbf{1}$-vector with $\mathbf{1}_i = 1$ and the analogously defined $\mathbf{0}$-vector.

Matrices are denoted by bold, capital letters (e.g., $\boldsymbol{M}$), and their elements are indexed as $(M)_{r,c}$, with r being the row index and c being the column index. In some cases it is necessary to index a whole row- or column-vector of a matrix. For the k-th row of a matrix A, this is given by $(A)_{k,\cdot}$, and for the k-th column it is denoted by $(A)_{\cdot,k}$. Diagonal matrices are given by

$$diag\,(x_1, x_2, \ldots, x_n) = \begin{pmatrix} x_1 & 0 & \cdots & 0 \\ 0 & x_2 & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & x_n \end{pmatrix}.$$

The identity matrix is given as $\boldsymbol{I} = diag\,(1, 1, \ldots, 1)$.

The identifier $\mathbb{R}$ is used to denote the set of real numbers, and $\mathbb{R}_+$ is used to denote a real number in the interval $(0, \infty)$. Similarly, the set $\mathbb{N}$ denotes the set of natural numbers. For any element $c = a + i \cdot b$ in the set of complex numbers $\mathbb{C}$, the real and imaginary parts are denoted as $a = \mathcal{R}\,(c)$ and $b = \mathcal{I}\,(c)$, respectively.

When dealing with differential equations, we usually specify them in the form $\dot{x} = f(x)$. In the present work, x is often a function depending on a time scale (i.e., $x(t)$), leading to the notation $\dot{x}(t) = f(x(t))$. Given such a function, its first derivate evaluated at a is denoted as $f'(a) = \frac{df(x(t))}{dx(t)}\Big|_{x(t)=a}$.

Introduction

Cognitive processes in the human brain have a number of time-dependent properties. Some of these properties can be characterized as incorporating the history of the cognitive system (i.e., previous states influence the current one). Humans, for instance, are capable of perceiving motion when presented with static images in rapid succession. This capacity is called the perception of apparent motion, and the motion quartet is a psychophysical experiment examining this phenomenon. In it, two dots are positioned on the diagonally opposing sides of an imaginary rectangle. In rapid succession, the dots change their positions to the free corners of the rectangle, causing motion to be perceived in either horizontal or vertical direction. The determination of the perceived direction is influenced by a number of factors (e.g., the aspect ratio of the rectangle, previous stimuli, etc.). However, once a detection is established, it usually stays stable for longer periods of time, even if the decision is ambiguous.

Classical models of human cognition, such as artificial neural networks, cannot reproduce the time-dependent properties of the human brain. Neural dynamics are more suited for this task because they are based on dynamical systems that describe the state of a system by specifying its change over time. They are used in the neural field model developed by Amari (1977). In this model, neurons are positioned along a continuous feature dimension and assigned to layers based on their type. The neurons interact with their neighbors with a strength that depends on an interaction kernel, describing the weight of the connection between them based on the difference of two neurons' positions, and a nonlinearity that allows only active neurons to influence others. The neural field model can be used to reproduce several capabilities that are advantageous for models of cognition inspired by the human brain, including the time-dependent ones. Nevertheless, there is a downside to neural fields: their simulation. This process is computationally complex, in particular when the dimensionality of the input data is high. Many computer vision algorithms that use a different approach, such as Slow Feature Analysis (Wiskott, 2003), or Elastic Bunch Graph Matching (Wiskott et al., 1997, 1999), use high-dimensional feature sets to represent objects. Solutions for dealing with the high dimensionality are often either inherent to such algorithms, or the problem is solved in another way (e.g., in Slow Feature Analysis, a hierarchical structure can be used to greatly reduce the impact of dimensionality (Wiskott and Sejnowski, 2002)).

Another possible solution to the problem of dimensionality is to sub-sample the feature space and subsequently only model relevant parts of it. One method of achieving this is to use an algorithm like the Growing Neural Gas (Fritzke, 1995). The output of this algorithm is a graph structure that clusters the input space while preserving its topology. Based on the success of these approaches, the present work aims to replace the continuous feature space in neural fields with a graph structure. The derived model is called the neural field graph model. Instead of directly using the information from the feature space for modeling interaction between neurons, it relies on the information encoded in the underlying graph's topology by determining the strength of the interaction based on the distance of neurons in the graph. Several concerns arise from this approach. One is that the topology around each neuron may be different due to the way the feature space is clustered, and neurons may behave differently based on their location in the graph structure. Depending on the purpose of the neural field graph, this may be undesirable. A method for correcting this imbalance is proposed based on the number of influences each neuron receives. Another problem is the detection of the centers of active regions (peaks) in the neural field graph. In neural fields, peak centers can be detected by using information about the active neurons' locations in the feature space. Because this information is missing in neural field graphs, different methods are devised. Similar to the neural interaction, these solutions are based on information encoded in the neural field graph. One method selects peak centers based on the minimum of the neurons' distance sums in an active cluster, the other two use maxima in the activation values.

After the formalization of the model and the proposed solutions to these problems in Section 1–6 of Chapter 4, the model is examined analytically in Section 7 of the same chapter, and empirically in Part 4. A goal of the examination in Chapter 5 is to determine what properties of the neural field model are still present in the neural field graph model. Additionally, the proposed methods for balancing synaptic weights and detecting peak centers are tested in Chapter 7 to determine whether or not they fulfill their intended purpose.

Part 1

Dynamic Field Theory

CHAPTER 1

Dynamical Systems

Classic models of cortical and neuronal information processing, such as neural networks (e.g., the multi-layer perceptron introduced by Rosenblatt (1958)), are not dependent on a time scale and usually calculate their output instantaneously.[1] As argued in the following sections, however, human cognition takes place over time (Schöner, 2007; Schneegans and Schöner, 2008; Erlhagen and Jancke, 2004). Therefore, models of neuronal information processing that incorporate time-dependent properties, such as the history of the cognitive system (i.e., its previous states and detection decisions) are desirable in various research fields concerned with modeling processes in the human brain (Erlhagen and Bicho, 2009).

Dynamical systems offer a framework for creating models that incorporate these properties. In Chapter 1, this fact is exploited to develop a simplified model of the cognitive information processing involved in the detection of motion conveyed by visual stimuli. This model is intended as an example used as a basis for introducing the concepts of dynamical systems. Furthermore, it is developed with the aim of solving the motion correspondence problem introduced in Section 1. An instance of the motion correspondence problem is given by the motion quartet, a psychophysical experiment that investigates special properties of human perception (Gilroy et al., 2001). The results of these experiments reveal a number of the time-dependent properties of human perception and cognitive processes that are not replicated in classical models. These properties form the requirements for the model of motion perception developed in Section 7. Building towards the formalization of this model, the basic concepts of dynamical systems, a theoretical framework often used in physics for describing the state changes of a physical system over time, are discussed in Section 2–6. Basic concepts of dynamical systems include the analysis of a system's stability and of a process called bifurcation, in which the nature of this stability changes either qualitatively or quantitatively. Bifurcations, also called instabilities, play an important role in the use of dynamical systems for cognition because they can be used to describe how a system reacts to a change in its input. Finally, Section 7 details the model that solves a simplified version of the motion correspondence problem, showing an example of how dynamical systems and the concepts presented in the previous sections can be applied to model human cognitive processes with some of the desired time-dependent properties.

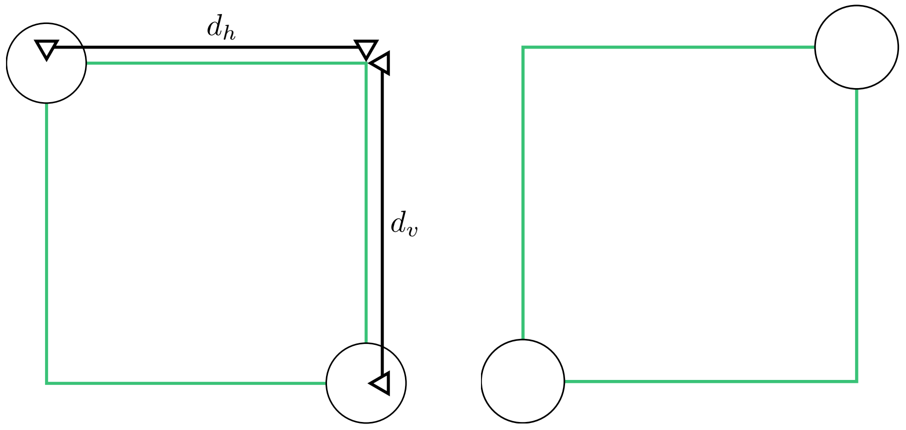
1. The Motion Correspondence Problem

The formal definition of the motion correspondence problem (Hibbard et al., 2000) is the following:

Definition 1.1 (*Motion correspondence problem*)**.** Given two successive visual stimuli, establish motion paths that consistently link image features in one stimulus to image features in the other one if they originate from the same physical object in space.

[1]The time needed for calculating the output is disregarded here, because it differs conceptually from a time scale.

Figure 1.1 The two frames of the motion quartet are shown in Figure 1.1a and Figure 1.1b (the superimposed rectangle is not shown during the experiments). These frames are presented to participants in rapid succession over longer periods of time, inducing a perceived motion of the dots in either horizontal or vertical direction. d_h and d_v indicate the horizontal and vertical distance between the corners of the imaginary rectangle and are used to calculate its aspect ratio as $\frac{d_h}{d_v}$.



(a) First frame. **(b)** Second frame.

One instance of the motion correspondence problem, called *stroboscopic alternative motion*, is introduced by von Schiller (1933). In experiments on the *motion quartet* (Gilroy et al., 2001) — an experiment also proposed as part of the original work on stroboscopic alternative motion — participants are presented with two dots on the diagonally opposing sides of an (imaginary) rectangle. In rapid succession, the dots change their places to the empty corners of the rectangle, leading to a perceived motion in either horizontal or vertical direction. The motion quartet is illustrated in Figure 1.1. After alternating presentation of the two frames comprising the motion quartet over a prolonged period of time, the test subjects are asked to specify which motion direction they perceived during the trial. The results derived from their responses expose several properties of human perception and the cognitive processes involved in solving the motion correspondence problem in the human brain.

One result of this experiment is that the perceived motion direction depends on the aspect ratio of the imaginary rectangle. The motion direction that is perceived by the test subjects usually adheres to the *minimal mapping solution* introduced by Ullman (1979). This solution entails selecting those motion paths that minimize the distance traversed by the percepts (i.e., the dots in the motion quartet). However, the *minimal mapping solution* does not always accurately model human responses, as shown by further experiments detailed by Gilroy et al. (2001), during which the aspect ratio of the rectangle (i.e., the ratio of horizontal to vertical distance $\frac{d_h}{d_v}$) is gradually changed over time, going either from being significantly greater than one to being less or vice versa.[2] When the aspect ratio changes, a phenomenon called *hysteresis* (Hock et al., 1993) can be observed. If the aspect ratio of the initial stimulus induces horizontal motion to be perceived and is subsequently varied by an increase of the horizontal distance between the dots, then horizontal motion is

[2]Note that due to effects of the *vertical-horizontal illusion* (Fineman, 1996) causing horizontal distances to be perceived shorter than vertical ones, the point of perceived equidistance is not actually the one where $d_h = d_v$. However, this effect is disregarded in the text.

perceived even when the vertical distance is slightly (but perceivably) smaller than the horizontal distance, thus violating the *minimal mapping solution*. To effect a change of the perceived motion direction, the difference between the two distances has to be increased significantly beyond the point of equidistance.

Another property of the cognitive processes involved in solving the motion correspondence problem is the stability of the solution. Once a solution is established, random switches (i.e., a sudden switch of the perceived motion direction) can occur for a stimulus with an unchanged aspect ratio (Gilroy et al., 2001). However, they occur infrequently and the solution is usually stable over periods of time stretching significantly more than a single successive presentation of the two frames.

These observations indicate that the cognitive processes involved in solving the motion correspondence problem depend on the history of the system (preference to perceive the motion direction selected before) as well as the present stimulus and thus are not instantaneous. The minimal mapping solution fails to take these properties into account, therefore, a different model that does account for such time-dependent properties is desirable.

2. Modeling Neural Activity with Dynamical Systems

Dynamical systems provide a theoretical framework that is frequently used in physics to model the state of systems that change over time based on their current and previous state, such as a swinging pendulum or the location and movement of a rigid body in space (Schneegans and Schöner, 2008). They are also a well-studied field in mathematics, providing tools for analyzing their stability, as well as the derivation and stability of their corresponding numerical solutions (Ascher and Petzold, 1998). Dynamical systems are therefore well suited for creating models of human cognition with an emphasis on time dependent properties (Spencer et al., in press, 2010). The general concepts behind dynamical systems are introduced over the course of the following sections, followed by an investigation of a solution of the motion correspondence problem based on them in Section 7.

Following Ascher and Petzold (1998), dynamical systems model the state of a system at a given time[3] $t \in \mathcal{T} \subseteq \mathbb{R}$, with n ($n \in \mathbb{N}$, $n > 0$) dynamic variables described by $u : \mathcal{T} \mapsto \mathbb{R}^n$. The state of the system at time t, denoted by $\boldsymbol{u}(t)$, depends on its previous states. Therefore, instead of directly specifying the relationship for $\boldsymbol{u}(t)$, a dynamical system is characterized by a rate of change that depends on the system's current state, denoted by $\dot{\boldsymbol{u}}(t) = \frac{d}{dt}\boldsymbol{u}(t)$ and given in the form

$$\tau\dot{\boldsymbol{u}}(t) = f\big(\boldsymbol{u}(t)\big)\,, \tag{1.1}$$

where $f : \mathbb{R}^n \mapsto \mathbb{R}^n$ and $\tau \in \mathbb{R}_+$, with $\tau > 0$, is a timescale that determines how fast (or slow) the state of the system changes in relation to the time t. In the present work, Equation 1.1 and similar equations describing the rate of change of a dynamical system are also referred to as the system's *dynamics*. In addition to these dynamics, the *initial state* (also known as the *initial value*) of the system, from which all subsequent states arise has to be specified. Without loss of generality,[4] we assume that $\mathcal{T} \subset [0, \infty)$, henceforth referring to the initial state of a dynamical system as $\boldsymbol{u}(0) \in \mathbb{R}^n$.

In some cases, it is possible to derive an equation for $\boldsymbol{u}(t)$ independent of the rate of change from a system's dynamics. This is called the solution of the system

[3]In general, the variable the dynamical system depends on is not necessarily time, but for the purpose of the present work it is more descriptive to think of it in this way (see also Ascher and Petzold, 1998).

[4]An existing time interval can be transformed to one starting from zero by means of an offset.

and, as an example, the solution for the dynamical system described by

$$\tau \dot{u}(t) = -u(t)\,, \tag{1.2}$$

together with an initial value $u(0) = \lambda$ can be derived using an integrating factor as given in (Bronstein et al., 2008). In order to apply this method, Equation 1.2 is rewritten to the standard form

$$\frac{du(t)}{dt} + \frac{1}{\tau}u(t) = 0\,. \tag{1.3}$$

This leads to the integrating factor μ for a constant c_1,

$$\mu = e^{\int \frac{1}{\tau}dt} = e^{\frac{t}{\tau}+c_1}\,. \tag{1.4}$$

Inserting this factor into the equation provided by Bronstein et al. (2008), the general solution for the system given by Equation 1.2 is found to be

$$u(t) = \frac{1}{\mu} \cdot c_2 = c_2 \cdot e^{-\frac{t}{\tau}} e^{-c_1}\,. \tag{1.5}$$

Empirical data is fitted by setting $c_1 = 0$ and $c_2 = \lambda$ (e.g., the one presented in Figure 1.4b), thus a solution of the system is given as

$$u(t) = \lambda \cdot e^{-\frac{t}{\tau}}\,. \tag{1.6}$$

3. Approximating Solutions of Dynamics

In practice, solutions for dynamical systems are often hard or even impossible to determine analytically. Therefore, they are usually approximated using numerical methods. This section provides a brief overview of two such numerical solvers, the forward and the backward Euler methods. In the remainder of the present work, applying one of these methods for the numerical solution of a dynamical system is also referred to as its *simulation*.

For the description of these numerical methods, the dynamical system being solved is given in the more general form presented by Ascher and Petzold (1998),

$$\frac{d\boldsymbol{u}(t)}{dt} = f(t, \boldsymbol{u}(t))\,, \tag{1.7}$$

where $\boldsymbol{u}(t) \in \mathbb{R}^n$, with $n \in \mathbb{N}$ and $n > 0$, represents the dynamic variables and $t \in \mathbb{R}$, with $0 \leq t \leq s \in \mathbb{R}$ and $f : \mathcal{T} \times \mathbb{R}^n \mapsto \mathbb{R}^n$. It is also assumed that the initial value of the dynamical system is known to be $\boldsymbol{u}(0) = \boldsymbol{c}$. In terms of the real-time simulation of a dynamical system, s is the time step (i.e., the length of time for which to simulate the system).

3.1. The Forward Euler Method.

In this method, the time step s is divided into N sub-steps

$$0 < t_0 < t_1 < \ldots < t_{N-1}\,, \tag{1.8}$$

with $t_{N-1} = s$. Given these time steps, the dynamical system is approximated using the values

$$\boldsymbol{u}_0, \boldsymbol{u}_1, \ldots, \boldsymbol{u}_{N-1}\,, \tag{1.9}$$

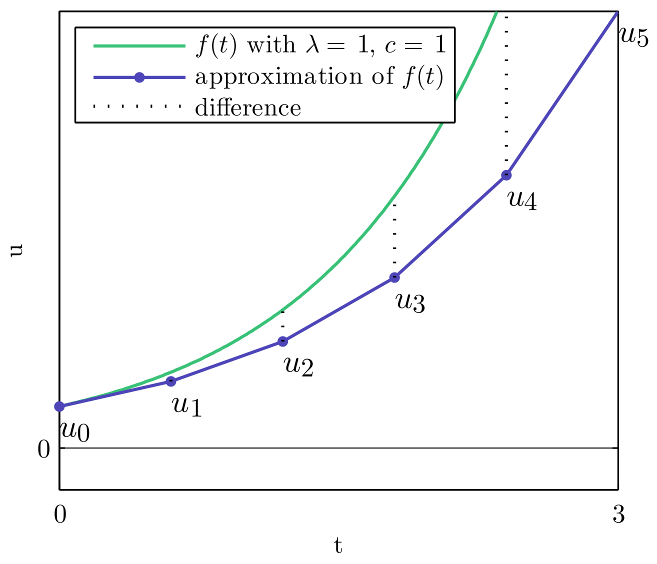
where $\boldsymbol{u}_i \in \mathbb{R}^n$, $\boldsymbol{u}_i \approx \boldsymbol{u}(t_i)$, and $\boldsymbol{u}_0 = \boldsymbol{c}$. Given the Taylor series of the dynamics $\boldsymbol{u}(t)$ about the point t_{i-1} with $t_i - t_{i-1} = h_i$ for $i \in \{1, \ldots, N-1\}$

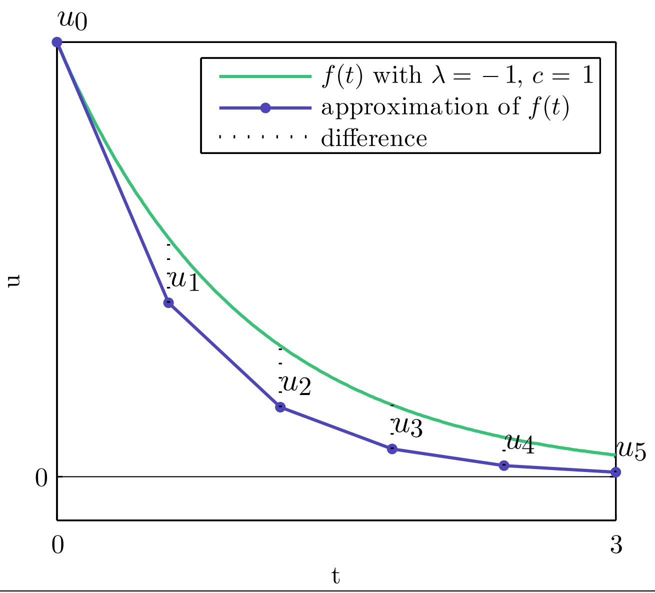
$$\boldsymbol{u}(t_i) = \boldsymbol{u}(t_{i-1}) + h_i \frac{d}{dt}\boldsymbol{u}(t_{i-1}) + \frac{1}{2}h_i^2 \frac{d^2}{dt^2}\boldsymbol{u}(t_{i-1}) + \ldots\,, \tag{1.10}$$

the terms of quadratic and higher order are dropped, leading to the equation

$$\boldsymbol{u}(t_n) \approx \boldsymbol{u}(t_{i-1}) + h_i \frac{d}{dt}\boldsymbol{u}(t_{i-1})\,, \tag{1.11}$$

Figure 1.2 Two examples for the forward Euler approximation. The plots show the approximation of the dynamics $\frac{d}{dt}f(t) = \lambda f(t)$ and a plot of the solution $f(t)$ of the dynamics, $f(t) = ce^{\lambda t}$. The dots mark the substeps u_n of the forward Euler method that is used for approximating the dynamics.





in which $\frac{d}{dt}\boldsymbol{u}(t_{i-1})$ can be substituted using Equation 1.7, yielding

$$\boldsymbol{u}(t_i) \approx \boldsymbol{u}_{i-1} + h_i f(t_{i-1}, \boldsymbol{u}_{i-1}) \,. \tag{1.12}$$

Equation 1.12 can then be used to iteratively calculate the values for $\boldsymbol{u}_i$, finally resulting in the approximation $\boldsymbol{u}_{N-1} \approx \boldsymbol{u}(s)$. Two numerical examples for approximations with the forward Euler method are shown in Figure 1.2.

3.2. The Backward Euler Method. Geometrically speaking, the forward Euler method (Ascher and Petzold, 1998) uses the slope of the dynamics at the point t_{i-1} and follows the resulting tangent to estimate the value for $\boldsymbol{u}(t_i)$. In the backward Euler method, the slope is determined directly at the point t_i. Thus, Equation 1.12 is rewritten to

$$\boldsymbol{u}_i = \boldsymbol{u}(t_i) \approx \boldsymbol{u}_{i-1} + h_i f(t_n, \boldsymbol{u}_i)\,. \tag{1.13}$$

In contrast to the forward Euler method, the values for the $\boldsymbol{u}_i$ in Equation 1.13 cannot be calculated directly because the unknown variable appears on both sides of the equation. Instead, the result is determined by solving the nonlinear system

$$\boldsymbol{g}(\boldsymbol{u}_i) = \boldsymbol{u}_i - \boldsymbol{u}_{i-1} - h_i f(t_i, \boldsymbol{u}_i) = 0\,. \tag{1.14}$$

This system can be solved by using Newton's method (Bronstein et al., 2008) to calculate

$$\boldsymbol{u}_i^{\nu+1} = \boldsymbol{u}_i^{\nu} - \left(\boldsymbol{I} - h_i \frac{\partial}{\partial \boldsymbol{u}} f(t_i, \boldsymbol{u}_i^{\nu})\right)^{-1} \left(\boldsymbol{u}_i^{\nu} - \boldsymbol{u}_{i-1} - h_i f(t_i, \boldsymbol{u}_i^{\nu})\right) \tag{1.15}$$

iteratively for $\nu = 1, 2, \ldots$, starting from an initial guess, e.g., $\boldsymbol{u}_i^0 = \boldsymbol{u}_{i-1}$. This iteration continues until, for some ν_{max}, the system satisfies

$$|\boldsymbol{u}_i^{\nu_{max}} - \boldsymbol{u}_i^{\nu_{max}-1}| \leq \epsilon\,, \tag{1.16}$$

for a given $\epsilon \in \mathbb{R}_+$. Examples of approximations using the backward Euler method are shown in Figure 1.3.

4. Stability of Dynamical Systems

Inspired by models in (Schöner, 2007), we can now use dynamical systems to create an exemplary model of a neuron's activation $u(t) \in \mathbb{R}$ as

$$\tau \dot{u}(t) = -u(t) + h + s(t)\,, \tag{1.17}$$

where $s : \mathcal{T} \mapsto \mathbb{R}$ is the neuron's input and $h \in \mathbb{R}$, with $h \leq 0$, is called the *resting level*. Figure 1.4 shows a plot of the system's dynamics and records of its state taken during simulations[5] starting from different initial values, in the absence of input (i.e., $s = 0$) and for $h = 0$. Figure 1.5 shows the behavior of the system during simulations with varying input and resting levels.

Figure 1.4b suggests that the state of the dynamical system specified by Equation 1.17 converges to a certain state, in this case 0. Figure 1.5 further indicates that the state to which the system converges is dependent on its resting level and the input. These states are called *fixed points* (Weisstein, 2010b) or *stationary points*. In a fixed point, the rate of change $\dot{u}(t)$ is zero, causing the system to stay in it until an outside influence is applied. Thus, according to Spencer and Schöner (2003), fixed points are those system states that satisfy

$$\dot{u}(t) = 0\,. \tag{1.18}$$

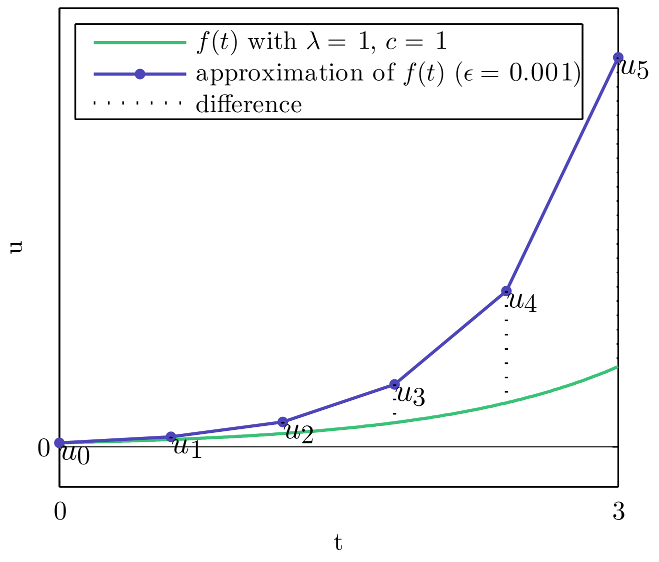
In the case of the dynamical system specified by Equation 1.17, the fixed points can be determined as the zero crossings of the dynamics plot given in Figure 1.4a. Accordingly, they can be calculated as

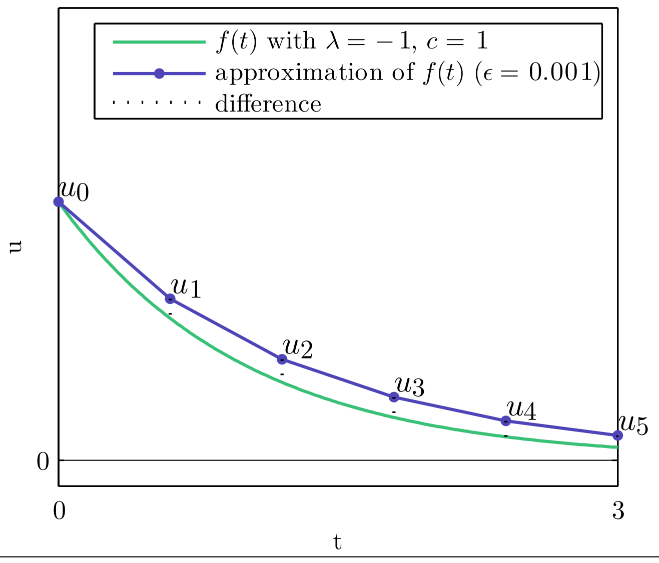
$$-\,u_0(t) + h + s(t) = 0 \Leftrightarrow u_0(t) = h + s(t)\,, \tag{1.19}$$

where $u_0(t)$ is the system's state representing a fixed point. Equation 1.19 confirms the initial observations that the point to which the system converges (i.e., the fixed point) depends on the input and the resting level. By means of such fixed points that relocate depending on the input level, dynamical systems can "react" to external stimuli, making them available as tools for the modeling of, among other things, cognitive processes.

[5]Details on how to simulate dynamical systems by approximating their solutions are given in Section 3; a forward Euler approach is used in all simulations as a default method.

Figure 1.3 Two examples for the backward Euler approximation. The plots, like in Figure 1.2, show the solution $f(t) = ce^{\lambda t}$ and the approximation for the corresponding dynamics $\frac{d}{dt}f(t) = \lambda f(t)$ (see Figure 1.2). The approximation is generated with the backward Euler method using Newton's method (see Section 3.2) for solving the nonlinear system. The dots mark the sub-steps of the approximation of the dynamics.





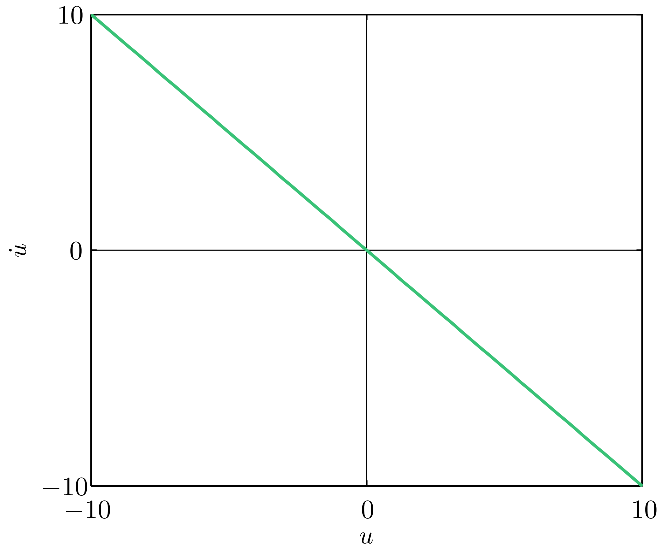
Consider now the dynamical system characterized by the following rate of change:

$$\tau \dot{u}(t) = u(t) - u^3(t) \tag{1.20}$$

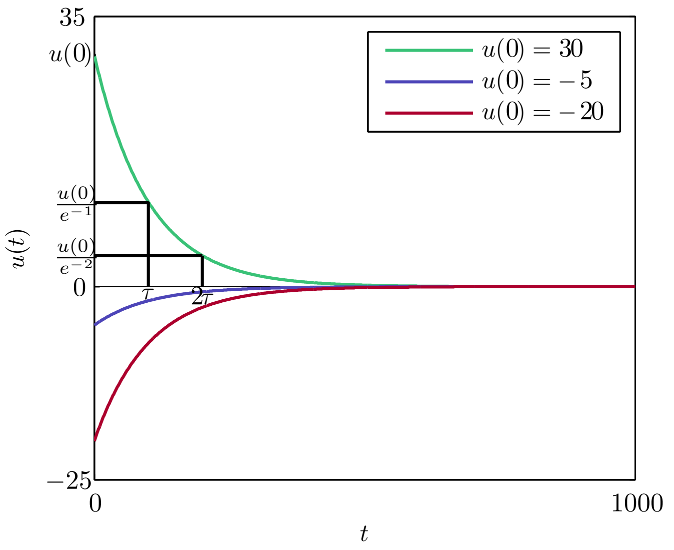
The system's fixed points can again be calculated as described above, leading to the three different fixed points marked in Figure 1.6a:

$$u_{0,1} = 0, u_{0,2} = 1 \text{ and } u_{0,3} = -1\,. \tag{1.21}$$

Figure 1.4 Plots for the dynamical system $\tau\dot{u}(t) = -u(t) + h + s(t)$ given in Equation 1.17 for $h = 0$ and $s(t) = 0$.



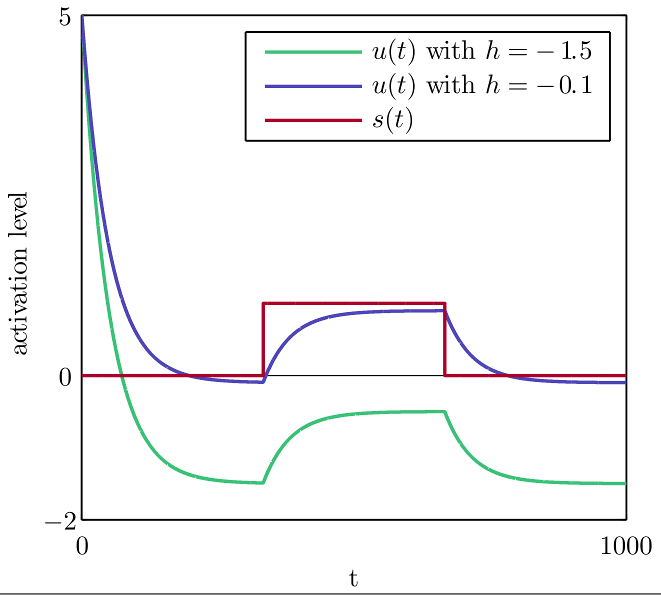
(a) The plot of the system's dynamics. The timescale is selected to be $\tau = 1$.



(b) The time course of the system's state in simulations starting from different states. The timescale used for the simulation is $\tau = 100$.

The plot of the system's behavior from different initial states is shown in Figure 1.6b. On the one hand, it indicates that the states of systems that start close to the fixed point $u_{0,1}$ converge to one of the other fixed points, despite being significantly closer to $u_{0,1}$. On the other hand, the system's state seems to always converge to either $u_{0,2}$ or $u_{0,3}$, depending on which one is closer. This behavior suggests that there are different kinds of fixed points. The first kind (which $u_{0,1}$ belongs to) is called *repellor*, and the second kind (which $u_{0,2}$ and $u_{0,3}$ belong to) is called *attractor* (Milnor, 2006).

Figure 1.5 Simulation of the dynamical system given by Equation 1.17 with varying input $s(t)$ and resting level h. The plot suggests that the point to which the system's state converges over time depends on the input and resting level.



Attractors are stable solutions of a dynamical system, meaning two things (cf. *Lyapunov stable attracting set* in (Auslander et al., 1964; Milnor, 2006; Spencer and Schöner, 2003)):

(1) If a system's state is in the attractor region, it remains in that region as long as the parameters of the system do not change.

(2) Systems with a state close to the attractor region converge into it over time.

Note, that in most — if not all — cases investigated in the present work, the attractor region is reduced to a single point. All fixed points that are not stable by the definitions above are considered unstable. The union of all system states that fall into case 1 and 2 (i.e., those regions from which all solutions converge to a given stable fixed point) are also called the attractor's *basin of attraction* (Ott, 2006).

How can we tell if a fixed point is an attractor or a repellor for a given dynamical system? The answer for the general case of dynamical systems with more than one variable is non-trivial; one method involves the linearization of the dynamics around the fixed points (Murray et al., 1994) and is used in Part 3, Section 7. However, a formal discussion exceeds the scope of the present work and therefore an intuitive description is given below for the case of dynamical systems restricted to one variable.

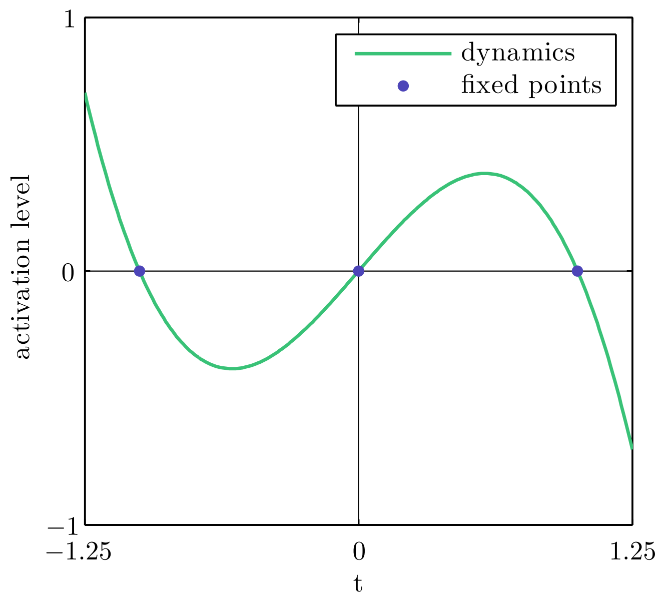
In the previous example (Equation 1.20), the slope of the dynamics is given by

$$\frac{d}{du}\dot{u}(t) = \frac{d}{du}\left(u(t) - (u(t))^3\right) = 1 - 3(u(t))^2\,. \tag{1.22}$$

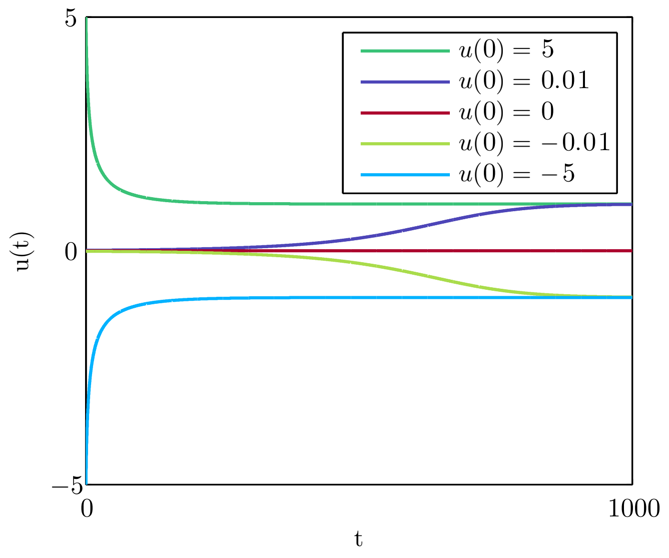
At the fixed points, this equation yields

$$\left.\frac{d\dot{u}(t)}{du}\right|_{u_{0,1}} = 1,\ \left.\frac{d\dot{u}(t)}{du}\right|_{u_{0,2}} = -2 \ \text{ and } \left.\frac{d\dot{u}(t)}{du}\right|_{u_{0,3}} = -2\,. \tag{1.23}$$

Figure 1.6 Plots for the dynamical system $\tau\dot{u}(t) = u(t) - (u(t))^3$. The fixed points $u_{0,1} = 0$, $u_{0,2} = 1$ and $u_{0,3} = -1$ are marked in the dynamics plot (Figure 1.6a). Figure 1.6b shows several simulations of the system, indicating that the states of systems with $u(0) \approx u_{0,1}$ and $u(0) \neq u_{0,1}$ are "pushed away" from the fixed point. Only the system starting with $u(0) = u_{0,1}$ remains in the fixed point.



(a) A plot of the dynamics. The fixed points are marked with dots.



(b) The development of the system's state recorded over time starting from several initial values.

Here, the repellor $u_{0,1}$ has a positive slope, while the attractors $u_{0,2}$ and $u_{0,3}$ have negative slopes. This is no coincidence: A negative slope at an attractor a means that system states with $u(t) < a$ undergo a positive rate of change, i.e., $u(t') = u(t) + \delta$ with $\delta \in \mathbb{R}_+$ and $t' > t$, thereby decreasing the difference between the fixed point and the system's state (because $a - u(t') = a - (u(t) + \delta) < a - u(t)$), while systems with $u(t) > a$ undergo a negative change, also reducing the distance $a - u(t)$, which asymptotically leads to a distance of $a - u(t) \approx 0$. For repellors, the situation is exactly the opposite (i.e., the difference between the system state and the repellor increases, leading to the behavior observed in Figure 1.6b).

5. Bifurcations

Bifurcations – also called *instabilities* (Schöner, 2007) – are described by a change of the parameters of a dynamical system, leading to a subsequent change in the markup of the system's fixed point states (Blanchard et al., 2006). They can be subdivided into two categories: quantitative and qualitative changes (Simmering and Spencer, 2008; Spencer and Schöner, 2003). The former either results in the formation of new fixed points or the disappearance of existing ones when one or more parameters of the dynamical system are changed gradually, while the latter means that an attractor becomes a repellor (or vice versa) when the system undergoes similar changes of the parameters.

Bifurcations can play an important role in models of cognitive processing described by dynamical systems (Spencer and Schöner, 2003). For instance, switches in the perception of horizontal or vertical motion during the presentation of the motion quartet (see Section 1) can be thought of as bifurcations: when the aspect ratio of the imaginary rectangle is changed sufficiently, a model could be adjusted so that an attractor for the opposite motion direction vanishes (quantitative change), or goes from being stable to being unstable (qualitative change), leading to the detection of the correct motion direction (Hock et al., 2003).

In Section 5.1 and Section 5.2, qualitative and quantitative changes are discussed briefly based on a subset of the different normal forms of bifurcations given in Guckenheimer and Holmes (1997).

5.1. Quantitative Changes: the Saddle-Node Bifurcation. Consider the dynamical system

$$\dot{u}(t) = f(u(t)) = \mu - (u(t))^2\,, \tag{1.24}$$

with the parameter $\mu \in \mathbb{R}$, representing the normal form of the *saddle-node bifurcation*. With the methods introduced previously, the fixed points can be calculated with respect to μ as

$$u_{0,1} = \sqrt{\mu} \text{ and } u_{0,2} = -\sqrt{\mu}\,, \tag{1.25}$$

for values of $\mu > 0$. For $\mu = 0$ only one fixed point is present:

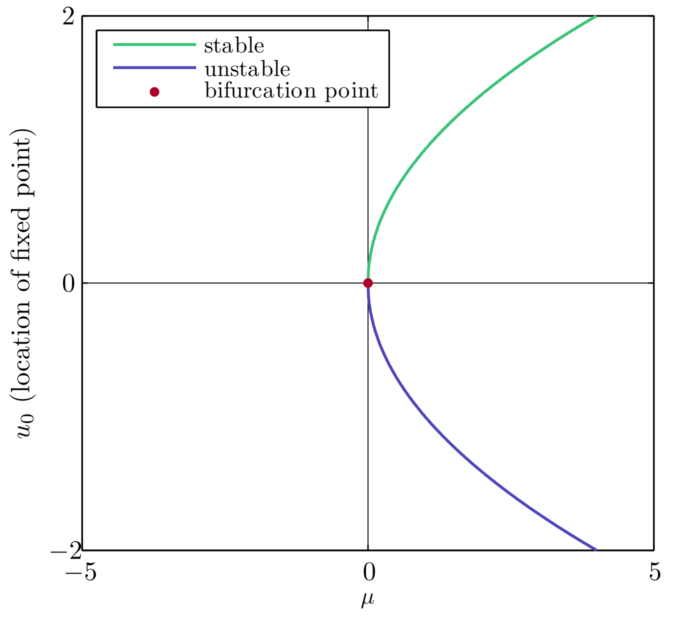
$$u_{0,3} = 0\,. \tag{1.26}$$

To examine the stability of the fixed points, the slope of the dynamics at the fixed points is calculated as well

$$\begin{array}{rrcl} & f'(u) = \frac{d}{du(t)} f(u(t)) & = & -2u(t)\,, \\ \Rightarrow & f'(u_{0,1}) & = & -2u_{0,1} = -2\sqrt{\mu}\,, \\ & f'(u_{0,2}) & = & -2u_{0,2} = 2\sqrt{\mu}\,, \\ & f'(u_{0,3}) & = & 0\,. \end{array} \tag{1.27}$$

Equation 1.27 and Equation 1.25 show that there are two fixed points for $\mu > 0$, the attractor $u_{0,1}$ and the repellor $u_{0,2}$. However, for $\mu = 0$, only the unstable fixed point $u_{0,3}$ is present, and for $\mu < 0$, no more fixed points exist. This is illustrated in Figure 1.7. When simulating the system over time, bifurcations, or more specifically, a quantitative change of the fixed points, may occur if the parameter μ is varied. For example, if μ starts out greater than zero and is then gradually decreased, the two fixed points that are initially present would move closer to each other. When

Figure 1.7 Stability plot for the saddle-node bifurcation. The system exhibits an attractor and a repellor for $\mu > 0$. For $\mu = 0$, only one fixed point exists, and none are present for $\mu < 0$. A system for which the parameter μ is varied over time thus displays a bifurcation if μ gradually goes from being greater than zero to being less than zero, or vice versa.



μ reaches zero, the attractor and repellor both merge into one fixed point that vanishes once μ becomes less than zero.

5.2. Qualitative Change: the Transcritical Bifurcation. Another example of a bifurcation is the *transcritical bifurcation*. It can be observed in the dynamical system given by the following equation:

$$\dot{u}(t) = f(u(t)) = \mu u(t) - (u(t))^2\,, \tag{1.28}$$

where $\mu \in \mathbb{R}$. An analysis similar to the one performed in Section 5.1 reveals the following fixed points

$$u_{0,1} = 0 \text{ and } u_{0,2} = \mu \tag{1.29}$$

for $\mu \neq 0$, and a single fixed point

$$u_{0,3} = 0 \tag{1.30}$$

for $\mu = 0$. In order to determe the stability of the fixed points, we find that

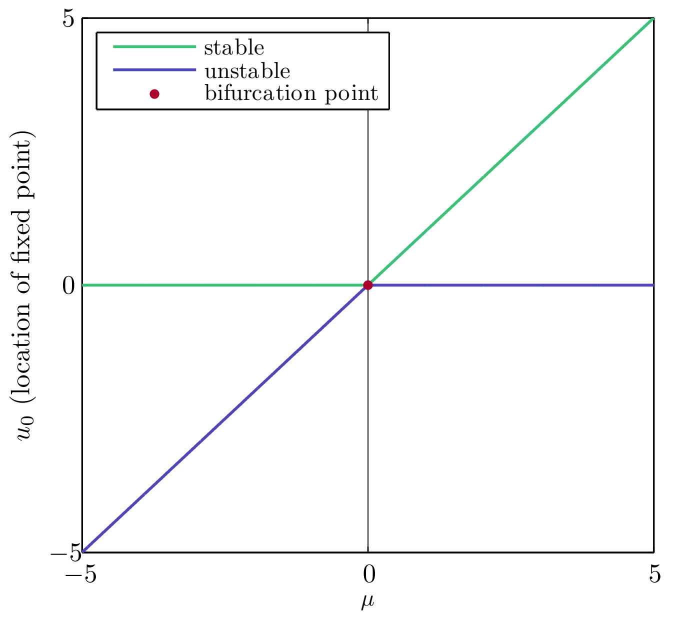
$$f'(u_{0,1}) = \mu\,, \quad f'(u_{0,2}) = -\mu \quad \text{and} \quad f'(u_{0,3}) = 0\,. \tag{1.31}$$

Therefore, $u_{0,1}$ is stable for $\mu < 0$ and instable for $\mu > 0$ (i.e., a qualitative change of the dynamics occurs during the bifurcation). For $u_{0,2}$, a qualitative change occurs as well, with conditions opposite to those of $u_{0,1}$, while $u_{0,3}$ is unstable. Figure 1.8 shows a plot of the fixed points depending on the parameter μ.

6. Noise

Consider again the time course for the dynamical system $\tau\dot{u}(t) = u(t) - (u(t))^2$ shown in Figure 1.6b. It indicates that the system with the initial value 0 remains in the repellor. This behavior is grounded in the fact that the system's rate of change in this state is $\dot{u}(t) = 0$. Therefore, it does not leave the fixed point.

Figure 1.8 Stability plot for the transcritical bifurcation. Two fixed points are present in all cases but $\mu = 0$, in which they overlap and form a single fixed point. Their stability depends on the selection of μ and can change, e.g., if μ changes from being greater than zero to a negative value.



However, when designing a dynamical system for cognitive information processing, one usually wishes it to converge to an attractor and prevent it from lingering in an unstable state. Therefore, in many cases a noise term is added to a system. Following the example in Section 5.2, the system would be

$$\tau\dot{u}(t) = u(t) - (u(t))^2 + \eta\,, \tag{1.32}$$

where η is a random variable with $\eta \sim \mathcal{N}(0, \sigma^2)$. The value σ^2 depends on the design of the dynamical system and is usually determined empirically. The noise term may also be present in the system already when an input is inherently noisy, in which case an additional noise term may not be necessary. The simulation of the system in Equation 1.32, shown in Figure 1.9, demonstrates that systems starting in the repellor no longer remain there, due to the rates of change around the repellor leading the system's state away from it. However, the state of the systems still converges to one of the attractors.[6] Which attractor is finally reached by those systems depends on the values of the random variable.

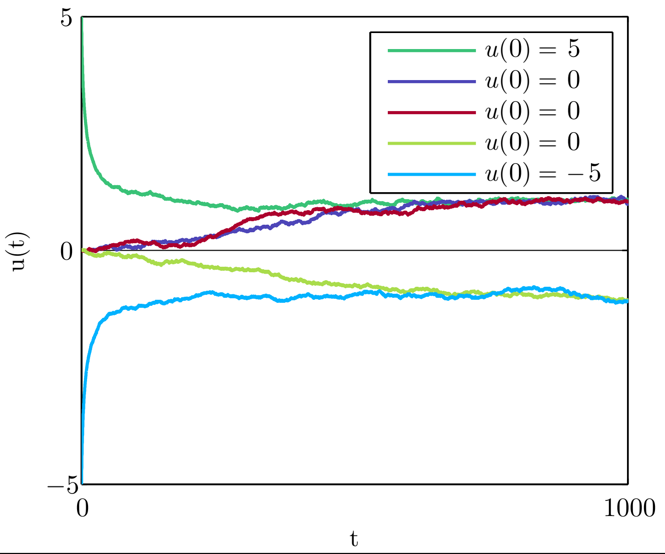
7. Dynamics and the Motion Quartet

With the devices introduced in sections Section 2–6, we can once again turn our attention towards the initial problem posed by the motion quartet. This section develops a less complex version of the model for solving the motion correspondence problem presented in (Hock et al., 2003).

This model solves a simplified motion correspondence problem: it contains only two neurons, one for perceived motion in horizontal direction and another for vertical one. Additional properties of the perceived stimulus are dropped to reduce the model's complexity (e.g., the starting point and the exact direction of

[6]Due to the zero-mean noise term, convergence means that the expected value of a system converging to an attractor a is given by $E\{u(t)\} = a$ for some $t > t_{conv}$.

Figure 1.9 Simulations of the system detailed in Figure 1.6 with an additive noise term $\eta \sim \mathcal{N}(0,5)$. With this addition, systems with an initial value of 0 randomly converge to one of the attractors. Without the noise terms, these systems would remain in their initial state, the repellor (see Figure 1.6b).



the perceived motion). Each neuron's activation is modeled as a dynamical system described by the dynamics

$$\begin{aligned} \tau \dot{u}_h(t) &= -u_h(t) + r + \varsigma \cdot s_h(t) - w_{hv} \cdot f(u_v(t)) + \eta_1 \text{ , and} \\ \tau \dot{u}_v(t) &= -u_v(t) + r + \varsigma \cdot s_v(t) - w_{vh} \cdot f(u_h(t)) + \eta_2 \text{ ,} \end{aligned} \tag{1.33}$$

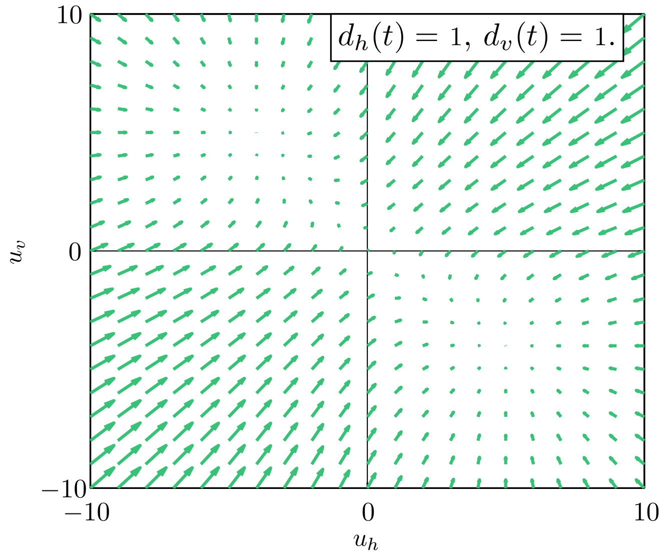
where u_h is the dynamic variable for the horizontal motion and u_v the one for vertical motion. $r \in \mathbb{R} < 0$ is the resting level, while w_{hv}, $w_{vh} \in \mathbb{R}$, $\varsigma \in \mathbb{R}_+$ and $f : \mathbb{R} \mapsto \mathbb{R}$ is the sigmoidal function (described in detail in Section 2.1 of Chapter 2). η_1 and η_2 are normally distributed noise terms as described in Section 6. The input is determined based on the aspect ratio of the motion quartet presented to the system: let $d_h(t)$ be the horizontal distance between the dots in the motion quartet at any given time and $d_v(t)$ the vertical distance. Then the inputs are defined according to

$$s_h(t) = \frac{d_v(t)}{d_h(t)}, \ s_v(t) = \frac{d_h(t)}{d_v(t)} \, . \tag{1.34}$$

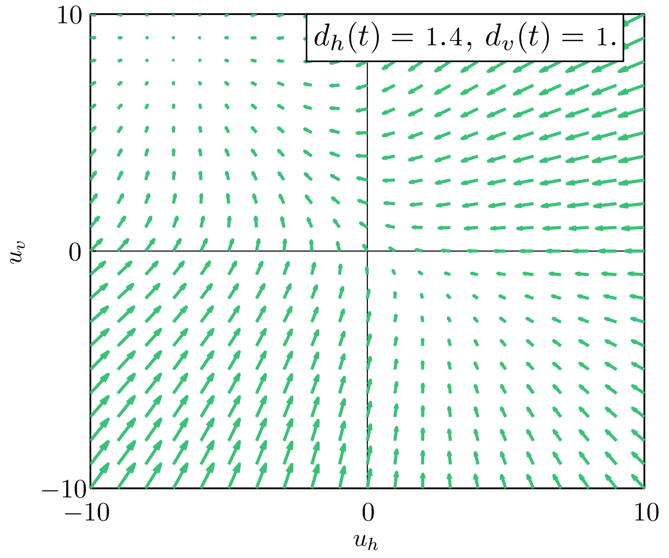
The model's output is a detection of either no, horizontal, or vertical motion. Motion in one direction is perceived if the state of one of the dynamical systems in the model exceeds a threshold, while no motion is perceived if none of the systems' states are above threshold. A detection of both motion directions at the same time would indicate a failure of the model to solve the (simplified) motion correspondence problem, while the detection of no motion should only occur briefly during an initial reaction time when a change of the stimulus occurs.

Figure 1.10 shows exemplary plots of the dynamic field given different inputs, and Figure 1.11 shows a simulation of the system. The simulation suggests that the (simplified) motion correspondence problem is indeed solved by this system: at most one perceived motion is selected at any time during the simulation. Additionally, properties of human perception, including those introduced in Section 1, can be observed: in r_1, a reaction time can be observed (i.e., the detection of the motion does not happen at the same time the stimulus is presented). Due to a smaller

Figure 1.10 Dynamics of the model of the motion-correspondence solution given by Equation 1.33. The parameters for the system are chosen as follows: $r = -5$, $\tau = 100$, $w_{hv} = 9$, $w_{vh} = 9$, $\beta = 1$ and $\varsigma = 10$. The components in the vectors, displayed as arrows, indicate the rate of change starting from the arrow's origin. The horizontal component of the vectors indicates the rate of change of u_h and the vertical component of u_v.

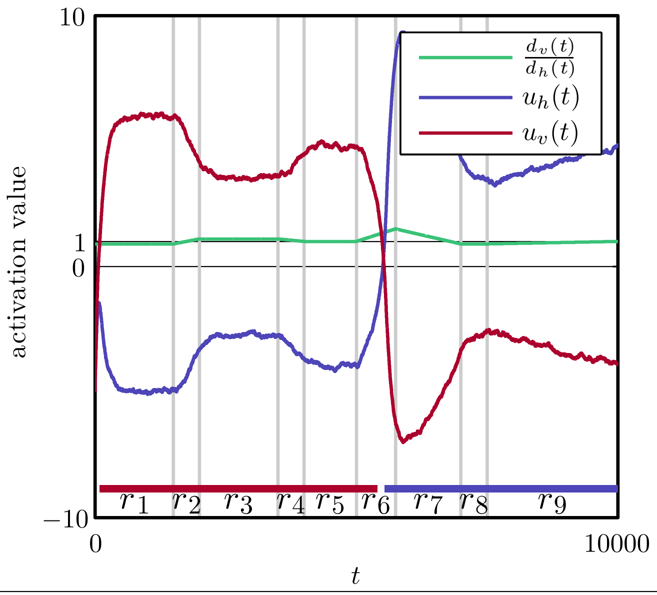


(a) Vector field of the dynamics for a stimulus with an aspect ratio of 1. The system exhibits two fixed point attractors, one in the second quadrant and one in the fourth, where the length of the vectors approaches zero.



(b) Vector field of the dynamics for a stimulus with greater horizontal distance. Only the fixed point in the second quadrant remains, the fixed point in the fourth quadrant has vanished (i.e., a bifurcation occurred). However, the rate of change near the vanished fixed point still shows remnants of its influence, indicated by the reduced length of the vectors in that region.

Figure 1.11 Simulation of the model of the motion-correspondence solution given by Equation 1.33. The parameters for the system are detailed in Figure 1.10. The colored bars at the bottom mark which motion direction is perceived by the system. No bar symbolizes that no motion direction was detected. The threshold for the detection of motion is set to 1.



vertical distance, the neuron for vertical motion reaches above-threshold activation eventually. In r_3, hysteresis appears to be present: the vertical distance is now greater than the horizontal one. However, instead of a change in the perceived motion, there is only a decay of the activation u_v that is not sufficient to move the activation below the threshold. Only when there is a stronger change in the aspect ratio in r_6 does the detection change.

CHAPTER 2

Neural Fields

Section 7 in Chapter 1 demonstrates how a model for two neurons with mutual interaction can be realized using dynamical systems. The same principles can be extended to a continuous space in which neurons are identified by their spatial coordinate $\boldsymbol{x} \in X$, with $X \subseteq \mathbb{R}^n$. Their activation is consequently given as a function $u : X \times \mathcal{T} \mapsto \mathbb{R}$, and the connections between the neurons (i.e., the equivalent to w_{hv} and w_{vh} in the motion detection model) could be realized as a convolution with an interaction kernel described by another continuous function.

The neural field model, introduced by Amari (1977), is such a model. Its original purpose is to describe the behavior of neurons in cortical tissue from the human brain. However, it is also used in the fields of autonomous robotics and embodied cognition and for models of cognitive processes (Bicho et al., 2000; Jancke et al., 1999; Erlhagen and Schöner, 2002). It is based on organizing the different types of neurons into m layers, each of which contains only neurons of the same type. Neurons in the layers are characterized by their spatial location $\boldsymbol{x}$,[1] along a continuous feature dimension as introduced above, as well as their activation u at time t. The change of the activation is specified by means of a dynamical system, similar to the one used for the model given in Equation 1.33. For a neuron in layer i, this dynamical system can be divided into the following additive components:

$$\tau_i \frac{\partial}{\partial t} u_i(\boldsymbol{x}, t) = -u_i(\boldsymbol{x}, t) + interaction_i(\boldsymbol{x}, t) + resting\ level_i + input(\boldsymbol{x}, t)\,, \quad (2.1)$$

where τ_i is the timescale associated with the neuron's layer, $interaction_i$ (described in detail in Section 2) specifies how the activations of neurons in the field influence those of others, and the $resting\ level_i$ (see Section 1) specifies the activation level in the absence of *input* (also described in Section 1). A neuron in this model is usually considered active when its activation exceeds a preset threshold. In the present work this threshold is set to zero, unless stated otherwise.

Over the course of the following sections, the components of Equation 2.1 are explained based on the work in Amari (1977). Finally, the complete equation is presented in Section 3, followed by some exemplary neural fields that highlight some of the capabilities of the neural field model.

1. Resting Level and Input

The input of a neuron at the position $\boldsymbol{x}$ can change over time. In the absence of any interaction, the field equation approximately[2] reduces to the neuron model $\dot{u}(t) = -u + h + s(t)$ (described by Equation 1.17 in Section 4 of Chapter 1). While

[1]The model given by Amari (1977) does not explicitly specify the feature dimension as a vector. In practice, multidimensional fields are often composed of one-layer networks with activations and features represented as vectors, rather than having one layer for each feature dimension (Erlhagen and Schöner, 2002). Even in Amari (1977), neurons are depicted on a two-dimensional plane.

[2]As is detailed in Section 2, a sigmoidal function is used for the interaction. Therefore, there is always some residual interaction in the neural field, but it approaches zero fast and is thus disregarded in the discussion.

positive input raises the activation level of the field, negative input lowers it. For neural fields, the input of a neuron $\boldsymbol{x}$ is denoted as $s(\boldsymbol{x}, t)$ and usually satisfies $s(\boldsymbol{x}, t) \geq 0$, where $s(\boldsymbol{x}, t) = 0$ means no input.

The resting level, denoted by $h_i \in \mathbb{R}$, specifies the level of activation to which the neurons relax in the absence of any input or interaction. The value of h_i is usually selected to be negative to avoid above-threshold activation in a field when no input is present.

2. Interaction

The model for solving the motion correspondence problem presented in Chapter 1 already contains an instance of interaction between neurons: the term $-w_{hv} \cdot f(u_v(t))$ in Equation 1.33 influences the rate of change of $u_h(t)$ if $f(u_v(t))$ is sufficiently different from zero. Depending on the selection of w_{hv}, this interaction can shift the attractor of the system below the threshold, thereby suppressing the activation of the neuron. In the motion-correspondence problem, this interaction behavior called *inhibition* plays an important role: with appropriate selection of the weights w_{hv} and w_{vh}, inhibition can induce that at most one neuron is active at any given time. Along with inhibition, neural fields also contain the opposite form of interaction called excitation. Excitation, in the above example, would mean a positive influence on the activation of $u_h(t)$ (i.e., instead of $u_h(t)$ being lowered when $u_v(t)$ is active, it would be raised). The realization of the concepts of excitation and inhibition in neural fields are described in Section 2.1–Section 2.3.3.

2.1. The Nonlinearity. Only active neurons interact with other neurons. In the previous example (see Section 7), a sigmoidal function is used to determine if a neuron is active. However, other possibilities exist, and the function determining the strength of the interaction based on a neuron's activity is termed *the nonlinearity*. Throughout the present work, the nonlinearity in the neural field is denoted by f. Usually, it f maps from a real value representing the activation of a neuron to the interval $[0, 1]$, where 0 (or value close to it) is assigned to activation values that are considered inactive, while 1 (or a slightly smaller value) is assigned to values considered active.

Amari (1977) suggests using the step function as nonlinearity:

$$f(u) = \begin{cases} 0: & u \leq 0 \\ 1: & u > 0 \end{cases} . \tag{2.2}$$

With the step function, neurons are considered to fire at their maximum rate if their activation reaches a value greater than zero. Otherwise, they are considered completely inactive (see Figure 2.1a).

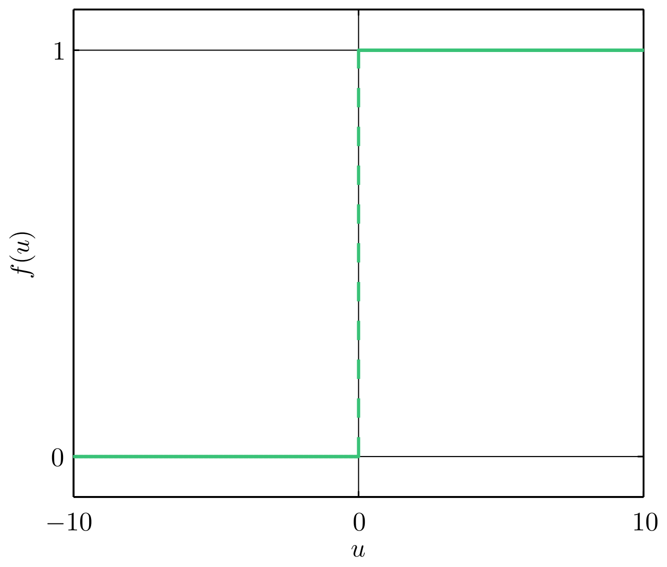
Another possibility for the nonlinearity is the sigmoidal function shown in Equation 2.3.

$$f(u) = \frac{1}{1 + e^{-\beta u}} . \tag{2.3}$$

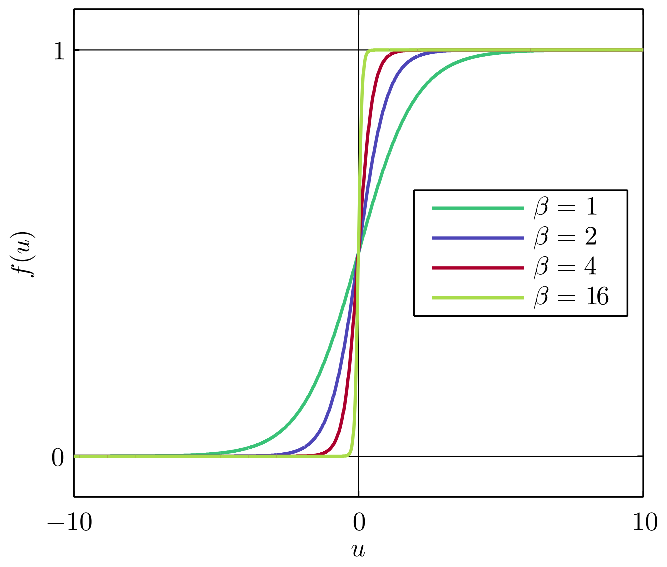
The parameter $\beta \in \mathbb{R}$, with $\beta > 0$, controls the steepness of the transition, with higher values resulting in a steeper transition, as is demonstrated by Figure 2.1b. Asymptotically, this function behaves similar to the step function, but when the activation of a neuron is close to the threshold value, there is a continuous transition from inactivity to the full firing rate (or vice versa).

2.2. Inhibition and Excitation. Inhibition and excitation in neural fields are modeled with an interaction kernel. This kernel describes the influence of a neuron $\boldsymbol{x}$ in layer i on another neuron $\boldsymbol{x}'$ in layer j and takes the form of a function

Figure 2.1 The two different nonlinearities introduced in Section 2.1 are shown in Figures 2.1a and 2.1b.



(a) A plot for the step function introduced in Equation 2.2.

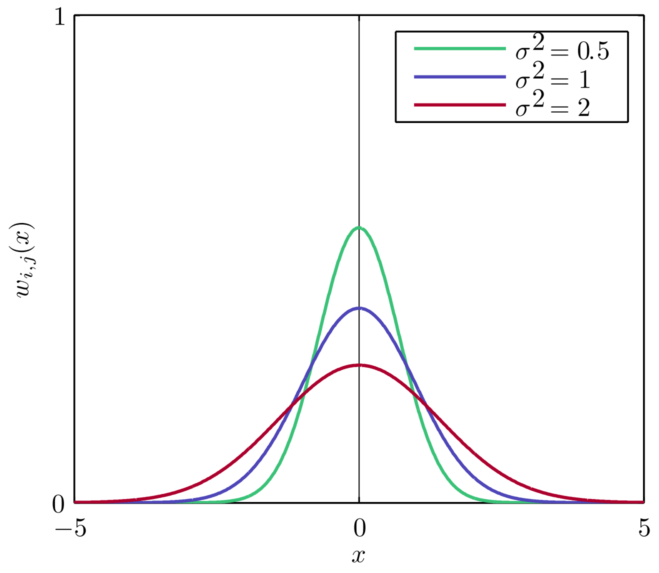


(b) Plot for the sigmoidal function described by Equation 2.3. Different values for β are used, indicating that a higher value for β results in a steeper slope around $u = 0$.

$w_{i,j}(\boldsymbol{x}, \boldsymbol{x}'; t - t')$. Thus, the interaction component of the neural field equation is given by

$$interaction_i(\boldsymbol{x}, t) = \sum_{j=1}^{m} \int w_{i,j}(\boldsymbol{x}, \boldsymbol{x}'; t - t') f\big(u_j(\boldsymbol{x}', t')\big) d\boldsymbol{x}' dt' \,, \tag{2.4}$$

Figure 2.2 Plot for the one-dimensional Gaussian interaction kernel for different values of σ.



where f represents the nonlinearity as described in Section 2.1 and m denotes the number of layers present in the neural field. The time difference $t - t'$ is used to model the time that activation takes to spread in cortical tissue.

As with the weights $-w_{hv}$ and $-w_{vh}$ of the model described in Section 7 of Chapter 1, negative values in the interaction kernel lead to inhibition by lowering the activation level of the attractor. Positive values have the opposite effect; they raise the attractor level and consequently lead to excitation. Note that neurons can also influence themselves in the case where $i = j$ and $\boldsymbol{x} = \boldsymbol{x}'$; this is called *self-excitation*: once a neuron becomes active, it further raises its own activation.

2.3. Interaction Kernels. Neural fields used in the present work are assumed to be homogeneous, meaning that the interaction between two neurons no longer depends on their exact positions, but only on their difference. The interaction kernel is also assumed to be symmetric and the activation of neurons is taken to spread instantaneously throughout the neural field (i.e., $t - t' = 0$). These assumptions allow for a simplification of Equation 2.4 to:

$$interaction'_i(\boldsymbol{x}, t) = \sum_{j=1}^{m} \int w_{i,j}(\boldsymbol{x} - \boldsymbol{x}') f\big(u_j(\boldsymbol{x}', t)\big) d\boldsymbol{x}' \,. \tag{2.5}$$

2.3.1. *The One-Dimensional Gaussian Interaction Kernel.* This kernel is defined by the probability density function of a normal distribution with mean zero and variance σ^2 (Bronstein et al., 2008):

$$w_{i,j}(x) = \varphi_{\mu=0,\sigma}(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{\frac{-x^2}{2\sigma^2}} \,. \tag{2.6}$$

Since the Gaussian interaction kernel only generates values of zero or greater (see Figure 2.2), it has no inherent inhibition. However, inhibition can be achieved by subtracting a scalar value as described in Section 2.3.4.

2.3.2. *The Mexican Hat Interaction Kernel.* Often, interaction kernels have positive values near 0 and negative values further away, meaning that neurons close to each other have an excitatory connection while neurons further away have an

Figure 2.3 Plot for the Mexican hat interaction kernel for different σ_1 and σ_2.

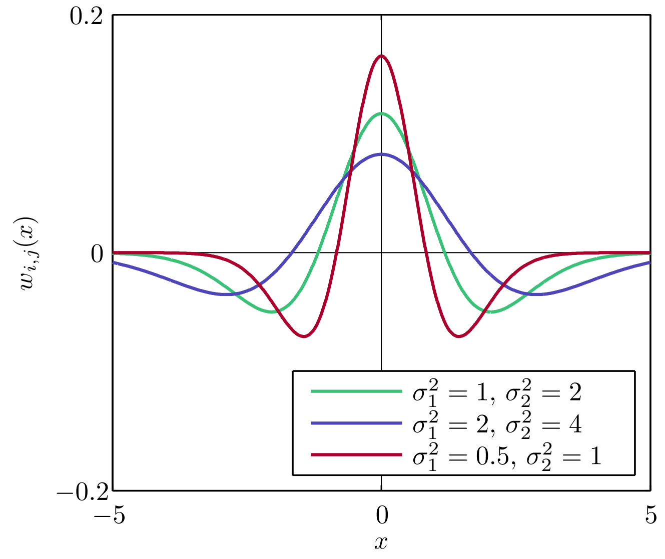
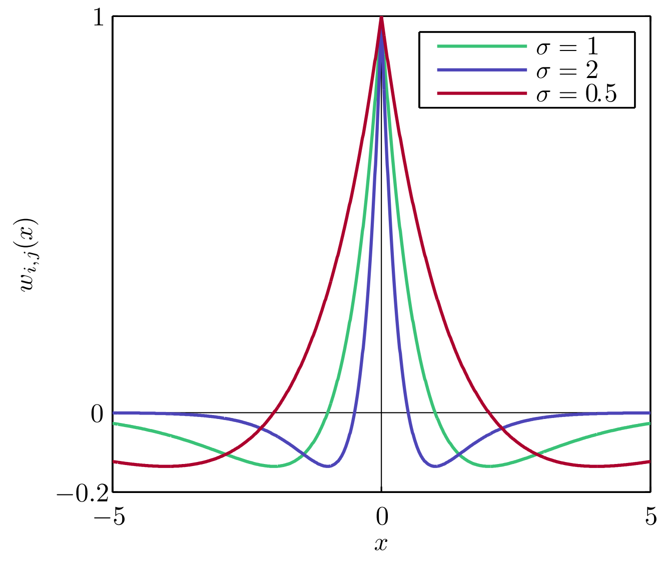


Figure 2.4 Plot for the modified wizard-hat interaction kernel for different values of σ.



inhibitory one. An interaction kernel that realizes these properties is the Mexican hat interaction kernel, introduced by Amari (1977). It is modeled after typical interactions found in cortical tissues (Coombes, 2006). The kernel is given by

$$w_{i,j}(x) = \varphi_{0,\sigma_1}(x) - \varphi_{0,\sigma_2}(x)\,, \tag{2.7}$$

where $\sigma_1 < \sigma_2$. The two parameters can be used to adapt the reach of the inhibitory (σ_1) and excitatory (σ_2) connections among the neurons. Equation 2.7 is plotted in Figure 2.3 for different values of σ_1 and σ_2.

2.3.3. *The Wizard-Hat.* An interaction kernel called wizard-hat is proposed by Coombes and Schmidt (2010). It is given by

$$w_{i,j}(x) = (1 - |x|)e^{-|x|} . \tag{2.8}$$

In the present work, a slightly modified version of Equation 2.8 is used that allows for scaling of the width of the interaction kernel with the parameter $\sigma \in \mathbb{R}_+$:

$$w_{i,j}(x) = (1 - |\sigma x|)\, e^{-|\sigma x|} . \tag{2.9}$$

A plot of this function is shown in Figure 2.4.

2.3.4. *Global Inhibition.* When it is present, all neurons receive inhibitory influence in addition to the influence given by the interaction kernel. This is realized by substituting the function for the interaction kernel with

$$w'_{i,j}(d) = w_{i,j}(d) - \gamma , \tag{2.10}$$

where $\gamma \in \mathbb{R}_+$ represents the strength of the global inhibition and $w_{i,j}(d)$ is an interaction kernel that is usually zero (or very close to zero) outside of a region $[-\sigma_I, \sigma_I]$. When some neurons in a neural field with global inhibition become active, the activity levels of all other neurons are lowered. If the amount of global inhibition is chosen appropriately, only a small region of the neural field becomes active, even if other regions receive a similar amount of input.

3. The Neural Field Equation

Summing up the concepts introduced in Section 1–2, the complete dynamical system for a neuron $\boldsymbol{x}$ in layer $i \in \{1, \dots, m\}$ of a neural field is given by (Amari, 1977):

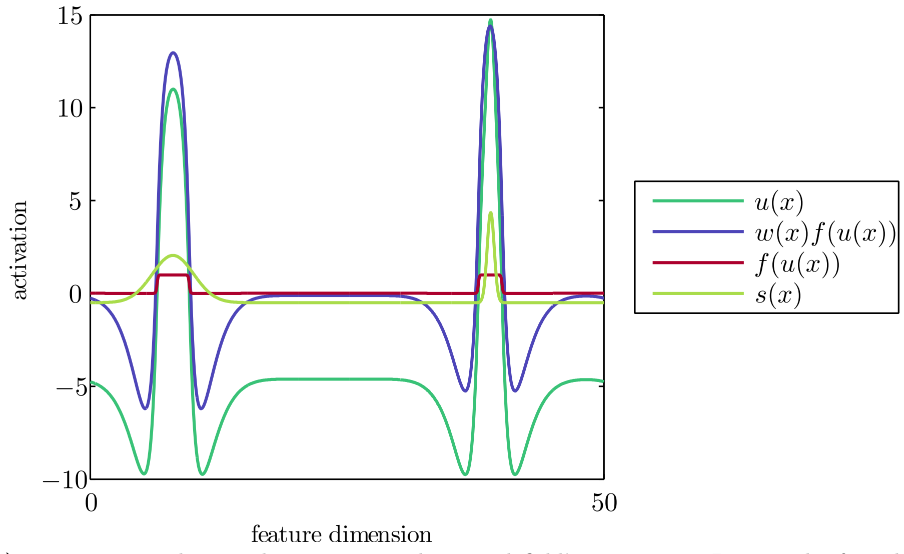
$$\tau_i \frac{\partial u_i(\boldsymbol{x}, t)}{\partial t} = -u_i(\boldsymbol{x}, t) + \sum_{j=1}^{m} \int w_{i,j}(\boldsymbol{x} - \boldsymbol{x}') f\big(u_j(\boldsymbol{x}', t)\big) d\boldsymbol{x}' + h_i + s(\boldsymbol{x}, t) . \tag{2.11}$$

In Figure 2.5a, the capability to form so-called peaks (i.e., regions of above-threshold activation, can be seen). Multiple peaks in the neural field's activation level u form based on peaks in the input. These input peaks could represent objects tracked along a feature dimension, while the center of the peak in the activation of the field represents the estimated position of an object in the feature space. Despite the fact that one of the input peaks is broader and the other one stronger, both result in peaks in the neural field's actiation that show only small differences in width due to the inhibitory sections of the wizard-hat interaction kernel.

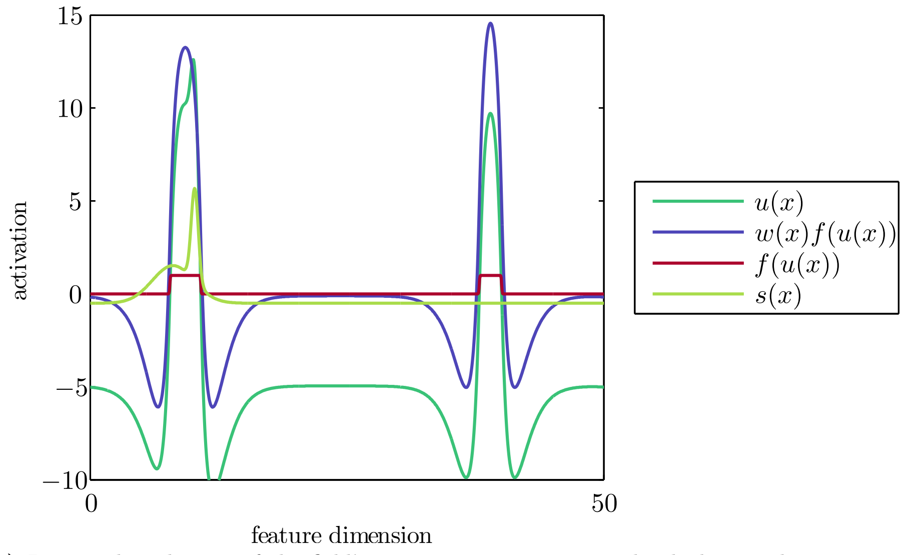
Figure 2.5b shows properties called *merging* and *self-sustained peaks*. The former occurs when two input peaks overlap: instead of two peaks in the field's activation level, only one peak forms. The latter is the residue of a previous input peak: even though no more input is present, a peak may still remain active and stable.

Figure 2.6 illustrates the effects of global inhibition: even though two peaks similar in shape and strength are present in the input, only one of them generates above-threshold activation in the corresponding neurons. Global inhibition lowers the other peak (and the activation of most other neurons), resulting in the field "selecting" one of the two peaks.

Figure 2.5 Plots of a one-layer neural field using the modified wizard-hat interaction kernel. The parameters for the neural field varied during the experiments are the timescale τ, the slope of the sigmoidal function β, the resting level of the field, h, the global inhibition γ and the width of the interaction kernel σ.

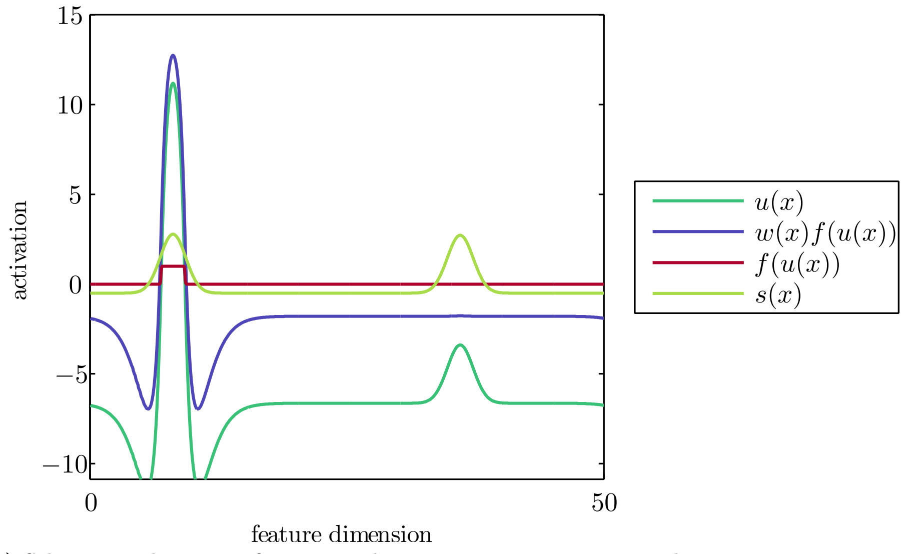


(a) Two input peaks are also present in the neural field's activation. Despite the fact that their widths are different, both exhibit only a small difference in their width in the neural field's activation. The parameters for the neural field are: $\tau = 10$, $\beta = 1$, $h = -4$, $\gamma = 0.001$ and $\sigma = 1$.

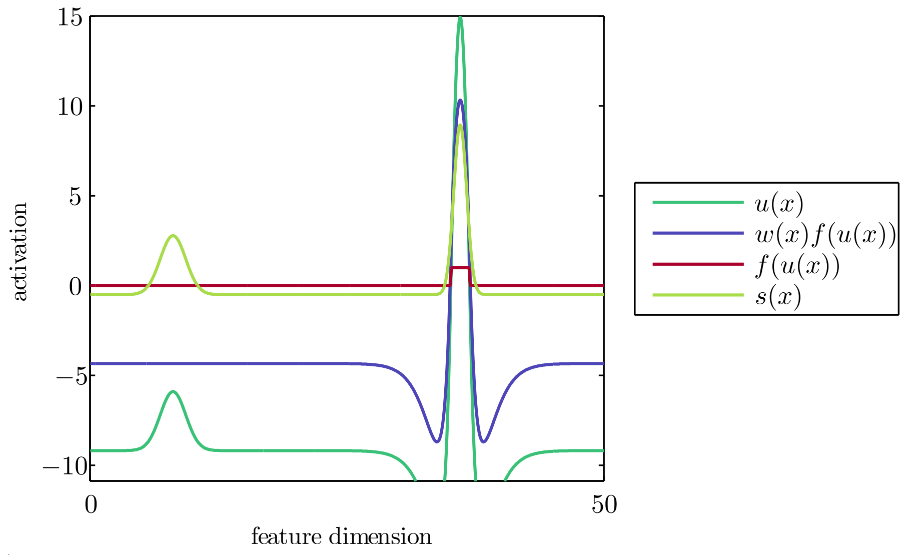


(b) Due to the selection of the field's parameters, an input peak which is no longer present, leaves a residual self-sustained peak in the neural field's activation u. Two overlapping input peaks are merged into one peak in the neural field. Its parameters are $\tau = 10$, $\beta = 1.8841$, $h = -4.3448$, $\gamma = 0.001$, and $\sigma = 1$.

Figure 2.6 Plots displaying the ability of neural fields to perform selection. The setup and parameters of the field are the same as described in Figure 2.5. The main influence in selection is the global inhibition which suppresses other peaks if an above-threshold peak is already present in the field's activation u.



(a) Selection with $\tau = 10$, $\beta = 1.8841$, $h = -4.3448$, $\gamma = 0.038204$ and $\sigma = 1$.



(b) Selection with the parameters $\tau = 10$, $\beta = 1.8841$, $h = -4.3448$, $\gamma = 0.11812$ and $\sigma = 1$.

Part 2

Graph Theory

CHAPTER 3

Conventions and Definitions

Although the reader is assumed to be familiar with the basic concepts of graph theory, some are briefly described in this section in order to present the notation used throughout the present work. For more in-depth information about the topic, various sources can be found in the references (Bondy and Murty, 1976; Chartrand, 1985; Diestel, 2006).

1. Nodes, Edges and Graphs

Definition 3.1 (*Directed graph*)**.** Following Chartrand (1985) and Seidel (1992), a *directed graph* is defined as a pairing $G = (V, E)$ where V is called the set of nodes given as $V = \{n_1, n_2, \ldots, n_N\}$, and E is the relation $E \subseteq V^2$, where $V^2 = V \times V$. The elements in E are called edges, and each one describes a connection between two nodes. In contrast to the literature, E may be reflexive for some of the graphs in the present work (i.e., edges of the form (v, v) with $v \in V$ are allowed).

Definition 3.2 (*Undirected graph*)**.** A graph $G = (V, E)$ is undirected if the relation E is symmetric (i.e., $(u, v) \in E \Leftrightarrow (v, u) \in E$). In this case, $(u, v) \in E \vee (v, u) \in E$ is shortened to $\{u, v\} \in E$ or $uv \in E$.

Definition 3.3 (*Subgraph*)**.** Let $G = (V, E)$ be a graph as defined above. Then $G' = (V', E')$ is a subgraph of G if and only if G' is a graph and $V' \subseteq V \wedge E' \subseteq E$.

Definition 3.4 (*Incident edges*, *incident nodes* and *indegree*)**.** Let n be a node in a directed graph $G = (V, E)$. Then, the incident edges of the node n are given by $\zeta(n) = \{(u, n) \in E\}$. The size of this set, $indeg(n) = |\zeta(n)|$, is called the indegree of the node n. Similarly, the incident nodes of n are given as $\Gamma_{in}(n) = \{u \big| (u, n) \in E\}$.

Definition 3.5 (*Neighborhood*)**.** The neighborhood $\Gamma_G(n)$, or for short $\Gamma(n)$, of a node n in the graph $G = (V, E)$ is defined as all nodes connected to it with an edge: $\Gamma(n) = \{n_j \big| (n, n_j) \in E \vee (n_j, n) \in E\}$.

2. Graphs with Associated Costs

In some graphs, costs are associated with the edges. For example, These costs can represent the time it takes for a cargo to traverse the route between two points in a transport network, where the points are represented as the nodes of the graph and the possible routes between them are represented as edges. A *graph with associated costs* is given in the form $G = (V, E, c)$, where $c : V^2 \mapsto \mathbb{R}$ is the cost function. In the present work, any $(i, j) \notin E$ is assigned the cost $c(i, j) = \infty$ by convention, while for any $v \in V$, if $(v, v) \in E$, the cost is given as $c(v, v) = 0$.

3. Paths

Definition 3.6 (Path)**.** Given a graph $G = (V, E)$, a *path* is defined as a subgraph $P = (V_P, E_P)$, with $V_P = \{n_1, \ldots, n_k\} \subseteq V$ and $E_P \subseteq E$ of the form $E_P = \{n_1n_2, n_2n_3, \ldots n_{k-1}n_k\}$, with $n_i \in V_P$. The node n_1 is called the origin of the path while n_k is called the terminus (Bondy and Murty, 1976).

The path P in Definition 3.6 is also denoted by $P = \langle n_1, n_2, \ldots, n_k \rangle$ specifying only the ordered vertices of the path, implying an edge between the first and second node, the second and third, and so on. Alternatively, if the intermediate nodes are undetermined, the path is denoted by $n_1 \xrightarrow{P} n_k$.

Definition 3.7 (Costs of a path)**.** According to Cormen et al. (2001), the costs $c(P)$ of a path $P = \langle n_1, n_2, \ldots, n_k \rangle$ on a graph $G = (V, E, c)$ with associated costs c are defined as $c(P) = \sum_{i=2}^{k} c(n_{i-1}, n_i)$. If no cost function is associated with a graph, the implicit cost function

$$c(u,v) = \begin{cases} 0 & : \quad (v,v) \in E \\ 1 & : \quad (u,v) \in E, \ u \neq v \\ \infty & : \quad (u,v) \notin E \end{cases} \tag{3.1}$$

is assumed.

Definition 3.8 (Cycle and negative Cycle)**.** A path C in a graph G with $C = \langle n_1, n_2, \ldots, n_k, n_1 \rangle$ and $k \geq 3$ is called a cycle. If the costs of the path C are negative, it is also called a negative cycle.

3.1. Shortest Paths.

Definition 3.9 (Shortest path costs)**.** The *shortest path costs* for two nodes u, v in a graph G are defined as

$$\delta(u,v) = \begin{cases} \min\left\{c(P) \middle| u \xrightarrow{P} v\right\} & : \quad \text{there is a path from } u \text{ to } v \text{ in } G \\ \infty & : \quad \text{otherwise.} \end{cases} \tag{3.2}$$

There are two forms of the problem of finding shortest paths (Cormen et al., 2001): for a single source and target, or for all pairs in the graph.

Definition 3.10 (*Single-source shortest path*)**.** For a graph $G = (V, E)$, find a path P^* with costs $c(P^*)$ that fulfills $u \xrightarrow{P^*} v$ and $c(P^*) = \delta(u,v)$ for two given nodes $u, v \in V$.

Definition 3.11 (*all-pairs shortest paths*)**.** In a graph $G = (V, E)$, find all paths P^*_{uv} with costs $c(P^*_{uv})$, such that $u \xrightarrow{P^*_{uv}} v$ and $c(P^*_{uv}) = \delta(u,v)$ for all pairs of nodes $u, v \in V$.

There are various algorithms for solving the *single-source shortest path* problem (e.g., Dijkstra's algorithm, or the Bellman-Ford algorithm). In the present work, only the *all-pairs shortest paths* problem is of relevance. Therefore, these algorithms are not discussed further.

3.2. The Floyd-Warshall Algorithm.

The *all-pairs shortest paths* problem on a directed graph with associated costs $G = (V, E, c)$ is often solved by means of the *Floyd-Warshall algorithm.* This algorithm runs in $\mathcal{O}\left(|V|^3\right)$ time and is known to be correct for all graphs that do not contain negative cycles (Cormen et al., 2001). The basis for the algorithm is a recursive restatement of the *all-pairs shortest paths* problem.

First, the intermediate vertices of a path $P = \langle v_1, v_2, \ldots, v_{l-1}, v_l \rangle$ are defined as all vertices on the path except the origin and the terminus (i.e., all vertices in the set $\{v_2, \ldots, v_{l-1}\}$ are the intermediate vertices of P). Assuming, without loss of generality, that $V = \{1, \ldots, n\}$, the *all-pairs shortest paths* problem can be solved by iteratively calculating Equation 3.3 for all pairs $i, j, k \in V$:

$$(D)^{(k)}_{i,j} = \begin{cases} c(i,j) & : \quad k = 0 \\ \min\left\{(D)^{(k-1)}_{i,j}, (D)^{(k-1)}_{i,k} + (D)^{(k-1)}_{k,j}\right\} & : \quad k \geq 1 \end{cases}, \tag{3.3}$$

where $\boldsymbol{D}^{(k)}$ is a matrix of path costs (or lengths), and each element $(D)_{i,j}^{(k)}$ specifies the length of the shortest path leading from the node i to the node j using only the intermediate vertices $\{1, \ldots, k\}$. Thus, $\boldsymbol{D}^{(n)}$, also referred to as the *distance matrix*, specifies the shortest path costs for all pairs of nodes.

Part 3

The Model

CHAPTER 4

The Neural Field Graph Model

This model contains two main components: the first one is an undirected graph obtained by sub-sampling an input space or clustering a high-dimensional data set using an algorithm like the Growing Neural Gas (Fritzke, 1995). The second one is an activation value associated with each node in the graph. The change of the activation values over time is modeled with a dynamical system based on an adapted version of the one used by the neural field model presented in Chapter 2. However, the interaction component of the adapted dynamical system is no longer determined by the distance in a feature space, as is the case in the neural field equation, but rather by the topology of the graph, coupled with a weight function that is based on sampling an interaction kernel. From such a neural field graph, an extended one is constructed on which all calculations for the simulation of the dynamics are performed in order to reduce the computational complexity.

This chapter begins with the formalization of these concepts, followed by a description of the process of sampling the synaptic weights from the edges of the neural field graph, and adapting them to the potential imbalance caused by differences in the topology surrounding each neuron. Here, the concept of the extended neural field graph is formalized, including a more efficient version of the equation used for the update of the activation values.

Once the model is established, its asymptotic stability is analyzed, starting with the general case. Only a few general statements are found, therefore, some more specific cases are examined in order to generate an understanding of the influence and effects of the parameters of the model, comprised mainly by the weights between neurons, the global inhibition and the input and resting level.

Finally, efficient versions of the algorithms involved in constructing and simulating neural field graph and its extended counterpart are specified, together with an analysis of their complexity.

1. Adaptation of the Neural Field Equation

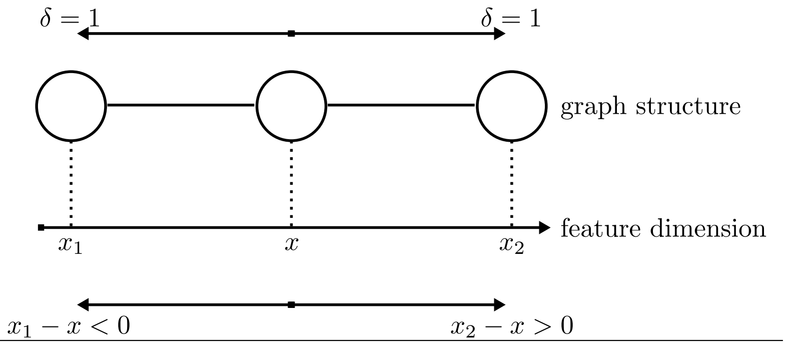
Definition 4.1 (Neural field graph)**.** Let $G = (V, E)$ be an undirected graph and $u : V \times \mathcal{T} \mapsto \mathbb{R}$. Then a neural field graph is given as $F = (G, u)$. The graph nodes are given as $V = \{v_{i_1,1}, v_{i_2,2}, \ldots, v_{i_N,N}\}$, with $i_n \in \{1, \ldots, m\} = M$ and $n \in \{1, \ldots, N\}$. The set E represents the edges in the graph according to the definition in Chapter 3.

The function u specifies the activation state of a node at time t. Each node together with its activation value forms a neuron in the neural field graph, thus edges in a neural field graph are also referred to as synapses. The first index $i \in M$ of a neuron $v_{i,j}$ specifies its layer, while the second index $j \in \{1, \ldots, N\}$ identifies the neuron. For notational convenience, the following definition is made

$$layer(v_{i,j}) = i. \tag{4.1}$$

Formally, a layer in a neural field graph is defined as follows:

Figure 4.1 Demonstration of implicit symmetry. The three neurons in the neural field graph depicted below represent features located at x_1, x, and x_2 (such a relation of neurons to features is not generally present, nevertheless, it is assumed to be known for this example). In a neural field, the interaction kernel would be sampled for negative and positive values when the interaction between x and x_1 and between x and x_2 is determined. In the neural field graph, only positive distances occur. Therefore, the neurons corresponding to x_1 and x_2 would both produce interaction sampled from positive feature difference values due to the definition of the sampling in Equation 4.3.



Definition 4.2 (Layer)**.** Given a neural field graph $F = (G, u)$, a layer is a subgraph $L_m = (V_m, E_m)$ of $G = (V, E)$, where $m \in M$, $V_m = \{v \in V \big| layer(v) = m\}$ and $E_m = \{\{k, l\} \subseteq E \big| k, l \in V_m\}$.

Then, the neural field equation presented in Chapter 2 can be restated for the activation of a neuron v on layer i in the neural field graph as

$$\tau_i \frac{\partial}{\partial t} u(v,t) = -u(v,t) + \sum_{l=1}^{m} \sum_{v' \in V_l} \hat{w}_{i,l}(v', v) f\big(u(v',t)\big) + h_v + s(v,t), \tag{4.2}$$

where $\hat{w}_{i,l} : V \times V \mapsto \mathbb{R}$ is the function used for sampling the interaction kernel (detailed in Section 2) and f is the nonlinearity. Similar to the model presented by Amari (1977), h_v represents the resting level of the neuron v, while $s(v,t)$ denotes its input at time t. In the present work, the nonlinearity is assumed to be a sigmoidal function because of its analytical properties, but other functions with similar behavior may be used instead.

2. Sampling the Interaction Kernel

The interaction kernel $w_{i,j}$ is sampled to determine the strength of the connection between two neurons. In the neural field model (see Chapter 2), the strength depends on the difference between the positions of the neurons in the feature space. Such a space is no longer present in the neural field graph model. Consequently, it does not rely on a feature difference, but rather on the distance of neurons in the graph's topology. Formally, the weight between two neurons v_1 and v_2 with $layer(v_1) = i$ and $layer(v_2) = j$ is determined as

$$\hat{w}_{i,j}(v_1, v_2) = \hat{w}_{i,j}(\delta(v_1, v_2)) = \mu \cdot w_{i,j}(\sigma \cdot \delta(v_1, v_2)), \tag{4.3}$$

where $\sigma \in \mathbb{R}_+$ scales the interval between samples, $\mu \in \mathbb{R}_+$, and δ describes the costs of the shortest path in accordance with Definition 3.9. Note, that the cost function for the graph is assumed to be the implicit cost function described by Equation 3.1. Because $\delta \geq 0$, it follows that only the positive side of the interaction kernel is sampled. Implicitly, this is similar to the assumption of a symmetric interaction kernel made by Amari (1977), as explained in Figure 4.1.

The factor μ is introduced to compensate for the loss of interaction due to the proposed sampling method. For example, if a Gaussian kernel is used, then it is known that

$$\int_{-\infty}^{\infty} w_{i,j}(x)dx = 1\,. \tag{4.4}$$

Thus, μ might be selected so that the following equation is fulfilled:

$$\mu \cdot w_{i,j}(0) + 2 \cdot \sum_{d=1}^{\infty} \mu \cdot w_{i,j}(\sigma \cdot d) = 1. \tag{4.5}$$

3. Global Inhibition

Recall that in neural fields, global inhibition is achieved by substituting the interaction kernel with $w'(x) = w(x) - \gamma$. Similarly, global inhibition in a neural field graph is realized by setting $\hat{w}'_{i,j}(v_1, v_2) = \hat{w}_{i,j}(v_1, v_2) - \gamma$. For the interaction component of Equation 4.2, this means that the following restatement can be made for a neuron v on layer i:

$$\begin{aligned}
\mathit{interaction}_i(v) &= \sum_{l=1}^{m} \sum_{v' \in V_l} \hat{w}'_{i,j}(v', v) f\big(u\,(v', t)\,\big) \\
&= \sum_{l=1}^{m} \sum_{v' \in V_l} \Big(\hat{w}_{i,l}(v', v) - \gamma\Big) f\big(u\,(v', t)\,\big) \\
&= \sum_{l=1}^{m} \sum_{v' \in V_l} \Big(\hat{w}_{i,l}(v', v) f\big(u\,(v', t)\,\big) - \gamma f\big(u\,(v', t)\,\big)\Big) \\
&= \sum_{l=1}^{m} \sum_{v' \in V_l} \hat{w}_{i,l}(v', v) f\big(u\,(v', t)\,\big) \underbrace{- \sum_{l'=1}^{m} \sum_{v' \in V_{l'}} \gamma f\big(u\,(v', t)\,\big)}_{\text{global inhibition term } \gamma_i(t)}\,.
\end{aligned} \tag{4.6}$$

Because each neuron in the neural field graph is assigned to exactly one layer, and both sums in the global inhibition term iterate over each neuron in every layer, it follows that the sums iterate over all neurons. Additionally, the terms in the sums do not depend on the layer, but only on the current neuron of the iteration. Thus, the global inhibition term $\gamma_i(t)$ in Equation 4.6 can further be simplified to

$$\begin{aligned}
\gamma_i(t) &= -\sum_{l'=1}^{m} \sum_{v' \in V_{l'}} \gamma f\big(u\,(v', t)\,\big) \\
&= -\gamma \sum_{v \in V} f\big(u\,(v, t)\,\big)\,.
\end{aligned} \tag{4.7}$$

Because $\gamma_i(t)$ no longer depends on the layer i, a global term $\gamma_g(t) = \gamma_i(t)$ is defined. During the update of the activation of a neuron in the neural field graph, the interaction no longer needs to be calculated by sampling the weight function for all neurons in the graph, but instead only for the ones that have a distance less than a predetermined δ_{max} selected according to

$$\delta_{max} = \underset{d \in \mathbb{N}}{argmin} \left\{ |\hat{w}_{i,j}(d')| < \epsilon \;\forall\; d' > d \right\} \tag{4.8}$$

for a given small, positive value $\epsilon \approx 0$. Thus, the global inhibition can be calculated separately from the general interaction with Equation 4.7.

4. Constructing Extended Neural Field Graphs

From a given neural field graph $F = (G, u)$ with $G = (V, E)$, an *extended neural field graph* F^e (or *extended graph*) is constructed in order to lower the computational requirements of the simulation of the graph's dynamics. It is defined as

$$F^e = \big(G^e, u, w\big)\,, \tag{4.9}$$

where

$$G^e = (V, E^e) \tag{4.10}$$

is a directed graph extending G and $u : V \times \mathcal{T} \mapsto \mathbb{R}$. As before, u represents the activation of the neurons. The extended set of synapses is constructed as

$$E^e = \big\{(v_2, v_1)\big| v_1, v_2 \in V \wedge \delta(v_1, v_2) \leq \delta_{max}\big\}\,, \tag{4.11}$$

where $\delta(v_1, v_2)$ is the length of the shortest path between v_1 and v_2 in G (see Chapter 3) and δ_{max} limits the distance for the spread of activity (barring the spread via global inhibition). The function $w : E^e \mapsto \mathbb{R}$ stores the extended synaptic weights between two neurons which is calculated according to the sampling method described in Section 2:

$$w(v_{j,l_2}, v_{i,l_1}) = \hat{w}_{l_1,l_2}(v_{i,l_1}, v_{j,l_2})\,. \tag{4.12}$$

The global inhibition is not included here and is calculated separately with the method given in Section 3.

Finally, the dynamical system for the activation of a neuron v on the layer i is given by

$$\tau_i \frac{\partial}{\partial t} u\,(v, t) = -u\,(v, t) + \sum_{v' \in \Gamma_{in}(v)} w(v', v) f\big(u\,(v', t)\,\big) + h_v + s\,(v, t) + \gamma_g(t)\,. \tag{4.13}$$

Note, that with this dynamical system only the incident neighbors $\Gamma_{in}(v)$ of a neuron v in G^e have to be considered. Additionally, the term $\gamma_g(t)$ is always the same across all neurons, and therefore only has to be calculated once for each update. If $\delta_{max} = \infty$, then it is equivalent to the one in Equation 4.2.

5. Balancing the Weights

Consider two nodes n_a and n_b in an extended neural field graph; let n_a be influenced by six inhibitory connections from other nodes and n_b only by three. Additionally, let the synaptic weights of all those influences have the same value w. If all neurons influencing n_a and n_b were active, this would create an imbalance: n_a would receive inhibition of strength $6 \cdot w$ (neglecting for a moment that it would actually be a little less due to the sigmoidal nonlinearity), while n_b would receive only $3 \cdot w$. Depending on the purpose of the neural field graph, this may either be desired or lead to inappropriate behavior. For example, if the selection of a neuron is desired and all neurons should react to input in the same manner, then the latter is the case: neurons with a higher amount of inhibitory connections would be at a disadvantage (i.e., they would potentially need more input to be selected as opposed to those with fewer connections of this kind).

This imbalance may be corrected by modifying the weights accordingly. Continuing the above example, the inhibition passed to n_b should be doubled in order to achieve equal amounts of inhibition for both neurons. To formalize this concept for cases with more than two neurons, the weight-based indegree of a neuron v_1 for any given weight class ω is defined as

$$indeg_{\omega,l_1}(v_1) = \big|\big\{(v_2, v_1) \in E^e \big| w(v_2, v_1) = \omega \wedge layer(v_2) = l_1\big\}\big|\;, \tag{4.14}$$

and the upper limit of the weight-based indegree between layers l_1 and l_2 as

$$\lambda_{\omega,l_1,l_2} = \max\left\{indeg_{\omega,l_2}(v)\middle|v \in V, layer(v) = l_1\right\} . \tag{4.15}$$

Then the *synaptic imbalance* of a neuron v on the layer l_v, stemming from neurons on layer l, can be determined as

$$a_{\omega,l}(v) = \frac{indeg_{\omega,l}(v)}{\lambda_{\omega,l_v,l}} \tag{4.16}$$

for a given weight class ω. Each weight in the extended graph can then be balanced by setting

$$w'(v_1, v_2) = \frac{w(v_1, v_2)}{a_{w(v_1,v_2),l_1}(v_2)} = \frac{w(v_1, v_2) \cdot \lambda_{w(v_1,v_2),l_1,l_2}}{indeg_{w(v_1,v_2),l_1}(v_2)} \tag{4.17}$$

as the synaptic weight for all $(v_1, v_2) \in E^e$. Equation 4.17 requires that the condition $indeg_{\omega,l}(v) \neq 0$ is fulfilled, which is true because at least one synapse that meets the prerequisites exists, namely the one being balanced.

6. Peak Detection

In neural fields, peaks in the activation and their centers can be detected using information on the location of neurons in the feature space. For the neural field graph model, such information is not present. Instead, it uses the information encoded in the neural field graph's topology. Therefore, the concept of an *active subgraph* is introduced in order to delimit connected regions in the graph that are considered a peak.

Definition 4.3 (Active subgraph)**.** Let $F = (G, u)$ be a neural field graph with the neurons V and the synapses E. An active subgraph F_A of F is defined as a subgraph $F_A = (V_A, E_A)$ of G with

$$V_A = \left\{v \in V\middle|u(v) > 0\right\} \tag{4.18}$$

and

$$E_A = \left\{(v_1, v_2) \in E\middle|v_1, v_2 \in V_A\right\} . \tag{4.19}$$

An active subgraph can be divided further into one or more connected components.[1] Each component is denoted by F_A^i with $i \in \{1, \ldots, n_a\}$ and represents at least one peak. The first two peak detection methods presented below further restrict this by assuming that each component of the active subgraph represents exactly one peak in the activation function.

Given the preceding definitions, the following criteria can be used to find the neurons p_i representing the center of the peaks described by the component F_A^i of an active subgraph F_A.

6.1. Distance-Minimizing Center. For this criterion, the neuron p_i is selected according to

$$p_i = \underset{p \in F_A^i}{argmin} \left\{ \sum_{v \in F_A}^{i} \delta(p, v) \right\} . \tag{4.20}$$

For each component of the active subgraph, this criterion selects the neuron minimizing the distance to all other neurons within the same component.

[1]The case in which no neurons are active is disregarded here because no peaks need to be detected in it.

6.2. Activity-Maximum. This criterion selects the neuron p_i as the one with the maximum activation inside the component of the active subgraph:

$$p_i = \underset{p \in F_A^i}{argmax} \left\{ u\left(p,t\right) \right\}. \tag{4.21}$$

6.3. Local Activity Maximum. The criteria described in Section 6.1 and Section 6.2 both select exactly one neuron per component of the active subgraph. However, experiments described in Part 4 suggest that this is not always desirable. Therefore, a third criterion is introduced in this section.

Rather than searching for the global optimum in each component like the *activity-maximum* criterion, the local activity maximum criterion is inspired by the notion of local maxima in analysis. It operates on all neurons of the active subgraph and does not distinguish between its components. Let n be an active neuron. Then n is considered a peak center (or part thereof) exactly if

$$u(n,t) \leq u(n',t) \ \forall \ n' \in \Gamma_G(n)\,. \tag{4.22}$$

Note that the equality is allowed here because otherwise plateaus (i.e., neurons for which one or more neighbors have the same activation value while the remaining neighbors have a smaller value) would not be selected. However, this also means that in such a case, all neurons in this plateau are selected as peak centers. If this is not desired, then the distance-minimizing center (see Section 6.1) could be selected among all the neurons that are part of a plateau.

7. Stability Analysis

Let $N = |V|$ be the number of neurons in a given extended neural field graph, and the neurons be numbered arbitrarily as $\{1, \ldots, N\}$. For the stability analysis, the dynamical systems used for the update of the neurons' activation values are rewritten to one system in vector form:

$$\boldsymbol{\tau} \circ \frac{\partial \boldsymbol{u}(t)}{\partial t} = -\boldsymbol{u}(t) + \boldsymbol{W} \cdot f\big(\boldsymbol{u}(t)\big) + \boldsymbol{h} + \boldsymbol{s}(t)\,. \tag{4.23}$$

In Equation 4.23, $\boldsymbol{u}(t) = (u_1(t), \ldots, u_N(t))^T$ contains the activation value for each neuron, $\boldsymbol{h} = (h_1, \ldots, h_N)^T$ represents the resting levels, the function f represents the component-wise application of the nonlinearity and $\boldsymbol{\tau} = (\tau_1, \ldots, \tau_N)^T$ denotes the time scales. Note, that the time scales in this form are given per neuron instead of per layer, thus they have to be mapped accordingly. For example, for a neuron v assigned to the index i, the corresponding entry in the vector of time scales is $\tau_i = \tau_l$, where τ_l is the timescale of layer $l = layer(v)$ in the neural field graph. Furthermore, the entries of the weight matrix are given as

$$(W)_{i,j} = w(v_j, v_i) - \gamma, \tag{4.24}$$

with $w(j,i)$ being the (possibly balanced) synaptic weight between the neurons assigned the indices j and i, and as before, the global inhibition strength γ. On the left-hand side of Equation 4.23, component-wise multiplication is used for the time scale:

$$(\boldsymbol{x} \circ \boldsymbol{y})_i = x_i \cdot y_i. \tag{4.25}$$

For the analysis, the input is assumed to be constant over time (i.e., $\boldsymbol{s}(t) = \boldsymbol{s}$), and $\boldsymbol{c} = \boldsymbol{h} + \boldsymbol{s}$, leading to the basic system equation

$$\boldsymbol{\tau} \circ \frac{\partial \boldsymbol{u}(t)}{\partial t} = -\boldsymbol{u}(t) + \boldsymbol{W} \cdot f\big(\boldsymbol{u}(t)\big) + \boldsymbol{c}\,. \tag{4.26}$$

7.1. Properties of the Sigmoidal Function. The derivative of the nonlinearity is required for discussing the solutions of the system given by Equation 4.26 and their stability. Because the sigmoidal function is used, it is given by

$$\begin{aligned}\frac{d}{dx}f(x) &= \frac{d}{dx}\frac{1}{1+e^{-\beta x}}\\ &= \frac{\beta e^{-\beta x}}{1+2e^{-\beta x}+e^{-2\beta x}}\\ &= \frac{\beta e^{-\beta x}}{e^{-\beta x}\cdot(e^{\beta x}+2+e^{-\beta x})}\\ &= \frac{\beta}{\underbrace{1+e^{\beta x}}_{1/f(-x)}+\underbrace{1+e^{-\beta x}}_{1/f(x)}}\,. \end{aligned} \tag{4.27}$$

The sigmoidal function is also symmetric:

$$\begin{aligned} f(-x) &= \frac{1}{1+e^{\beta x}} = \frac{1}{(1+e^{-\beta x})e^{\beta x}}\\ &= \frac{e^{-\beta x}}{1+e^{-\beta x}} = \frac{1+e^{-\beta x}-1}{1+e^{-\beta x}}\\ &= \underbrace{\frac{1+e^{-\beta x}}{1+e^{-\beta x}}}_{1} - \underbrace{\frac{1}{1+e^{-\beta x}}}_{f(x)}\\ \Leftrightarrow \qquad f(-x) &= 1-f(x)\,. \end{aligned} \tag{4.28}$$

With this result, Equation 4.27 can be rewritten further:

$$\begin{aligned}\frac{d}{dx}f(x) &= \frac{\beta}{\frac{1}{f(-x)}+\frac{1}{f(x)}}\\ &= \frac{\beta}{\frac{1}{1-f(x)}+\frac{1}{f(x)}}\\ &= \frac{\beta}{\frac{f(x)+(1-f(x))}{(1-f(x))\cdot f(x)}}\\ &= \beta(1-f(x))f(x)\,. \end{aligned} \tag{4.29}$$

This solution is also shift-invariant (Oppenheim and Schafer, 1975):

$$\begin{aligned}\frac{d}{dx}f(x+a) &= \left.\frac{df(x)}{dx}\right|_{x=x+a}\cdot\left(\frac{d}{dx}(x+a)\right)\\ &= \beta f(x+a)(1-f(x+a))\,, \end{aligned} \tag{4.30}$$

where $a \in \mathbb{R}$ is the parameter representing the shift.

7.2. Stationary Solutions. To determine the stationary solution $\boldsymbol{u}_0(t)$, zero change in all dimensions is assumed for Equation 4.26:

$$\begin{aligned} \boldsymbol{\tau}\circ\frac{\partial \boldsymbol{u}_0(t)}{\partial t} &= \boldsymbol{0}\\ \Rightarrow \qquad \boldsymbol{0} &= -\boldsymbol{u}_0(t)+\boldsymbol{W}\cdot f(\boldsymbol{u}_0(t))+\boldsymbol{c}\\ \Leftrightarrow \qquad \boldsymbol{u}_0(t) &= \boldsymbol{W}\cdot f(\boldsymbol{u}_0(t))+\boldsymbol{c}\,. \end{aligned} \tag{4.31}$$

During the present work, Equation 4.31 is also referred to as the *basic fixed point equation* of the neural field graph model.

According to Equation 4.31, the space of possible fixed points is limited by the following considerations: f always maps to values in the interval $[0,1]$. Thus, values of $f(\boldsymbol{u}_0)$ are contained in an N-dimensional hypercube given by

$$P_f(\boldsymbol{v}) = \sum_{j=1}^{N} v_j \cdot \boldsymbol{e}_j , \tag{4.32}$$

with vectors $\boldsymbol{v} \in [0,1]^N$ and $e_j \in \mathbb{R}^N$ defined as

$$(\boldsymbol{e}_j)_k = \begin{cases} 1 & : \quad j = k \\ 0 & : \quad j \neq k \end{cases} . \tag{4.33}$$

Therefore, any stationary point $\boldsymbol{u}_0$ that satisfies Equation 4.31 is always located inside the linearly transformed hypercube offset by $\boldsymbol{c}$:

$$P'_f(\boldsymbol{v}) = \sum_{j=1}^{N} v_j \cdot \boldsymbol{W} \cdot \boldsymbol{e}_j + \boldsymbol{c} = \boldsymbol{W} \cdot \boldsymbol{v} + \boldsymbol{c} . \tag{4.34}$$

7.3. Stability of a Stationary Point. Suppose that $\boldsymbol{u}_0(t) = \boldsymbol{u}_0$ is a stationary point.[2] Then, we can look at small deviations from $\boldsymbol{u}_0$ given by

$$\boldsymbol{\xi}(t) = \boldsymbol{u}(t) - \boldsymbol{u}_0 \Leftrightarrow \boldsymbol{u}(t) = \boldsymbol{\xi}(t) + \boldsymbol{u}_0 . \tag{4.35}$$

The derivative of $\boldsymbol{\xi}(t)$ with respect to t is

$$\frac{\partial}{\partial t}\boldsymbol{\xi}(t) = \frac{\partial}{\partial t}\boldsymbol{u}(t) . \tag{4.36}$$

Inserting Equation 4.35 and 4.36 into Equation 4.26 shows that

$$\boldsymbol{\tau} \circ \frac{\partial}{\partial t}\boldsymbol{u}(t) = \underbrace{-\boldsymbol{\xi}(t) - \boldsymbol{u}_0 + \boldsymbol{W} \cdot f(\boldsymbol{u}_0 + \boldsymbol{\xi}(t)) + \boldsymbol{c}}_{g(\boldsymbol{\xi}(t))} . \tag{4.37}$$

A Taylor expansion of g around $\boldsymbol{\xi}(t) = \boldsymbol{0}$ yields

$$g(\boldsymbol{\xi}(t)) = g(\boldsymbol{0}) + \boldsymbol{J}g(\boldsymbol{0}) \cdot \boldsymbol{\xi}(t) + O(\boldsymbol{\xi}^2(t)), \tag{4.38}$$

where $\boldsymbol{J}g(\boldsymbol{0})$ is the Jacobian matrix of g evaluated at $\boldsymbol{0}$. Because $\boldsymbol{u}_0$ fulfills Equation 4.31, $\boldsymbol{W} \cdot f(\boldsymbol{u}_0)$ can be derived from the equation as

$$\boldsymbol{W} \cdot f(\boldsymbol{u}_0) = \boldsymbol{u}_0 - \boldsymbol{c} . \tag{4.39}$$

Thus, $g(\boldsymbol{0})$ is

$$\begin{aligned} g(\boldsymbol{0}) &= -\boldsymbol{u}_0 + \boldsymbol{W} \cdot f(\boldsymbol{u}_0) + \boldsymbol{c} \\ &= -\boldsymbol{u}_0 + (\boldsymbol{u}_0 - \boldsymbol{c}) + \boldsymbol{c} \\ &= \boldsymbol{0} . \end{aligned} \tag{4.40}$$

According to McMillen (2008), the entries of the Jacobian matrix in Equation 4.38 are given by

$$(Jg)_{i,j} = \frac{\partial g_i}{\partial \xi_j(t)} . \tag{4.41}$$

[2] Although it is possible that the stationary point changes over time (e.g., if bifurcations and other transient states occur because of changing input), these cases are not treated here because they exceed the scope of the present work.

For $i = j$, Equation 4.41 becomes

$$\begin{aligned}(Jg)_{i,i} &= \frac{\partial g_i}{\partial \xi_i(t)} \\ &= -1 + \frac{\partial}{\partial \xi_i(t)} \sum_{k=1}^{l} (W)_{i,l} \cdot f\big(u_{0,l} + \xi_l(t)\big)\,. \end{aligned} \tag{4.42}$$

Using the shift-invariance of the derivative of the sigmoidal function, this result can further be derived as

$$(Jg)_{i,i} = \underbrace{-1 + \beta \cdot (W)_{i,i} \cdot f\big(u_{0,i} + \xi_i(t)\big) \cdot \Big(1 - f\big(u_{0,i} + \xi_i(t)\big)\Big)}_{=\mu_{i,i}(\boldsymbol{u}_0, \boldsymbol{\xi})}\,. \tag{4.43}$$

Similarly, for $i \neq j$

$$\begin{aligned}(Jg)_{i,j} &= \frac{\partial g_i}{\partial \xi_j(t)} \\ &= \frac{\partial}{\partial \xi_j(t)} \sum_{k=1}^{l} (W)_{i,l} \cdot f\big(u_{0,l} + \xi_l(t)\big) \\ &= \underbrace{\beta \cdot (W)_{i,j} \cdot f\big(u_{0,j} + \xi_j(t)\big) \cdot \Big(1 - f\big(u_{0,j} + \xi_j(t)\big)\Big)}_{=\mu_{i,j}(\boldsymbol{u}_0, \boldsymbol{\xi})}\,. \end{aligned} \tag{4.44}$$

Thus, $\mu_{i,j}$ in Equation 4.43 and 4.44 is defined as

$$\mu_{i,j}(\boldsymbol{u}_0, \boldsymbol{\xi}(t)) = \beta\,(W)_{i,j}\, f\big(u_{0,j} + \xi_j(t)\big)\big(1 - f(u_{0,j} + \xi_j(t))\big) - \delta_{i,j}\,, \tag{4.45}$$

where $\delta_{i,j} = 1$ for $i = j$, and $\delta_{i,j} = 0$ otherwise, is the Kronecker delta (Bronstein et al., 2008). Because the Jacobian matrix is evaluated at $\boldsymbol{\xi}(t) = \mathbf{0}$, the same is applied to $\mu_{i,j}$, yielding

$$\mu_{i,j}(\boldsymbol{u}_0, \mathbf{0}) = \beta\,(W)_{i,j}\, f\big(u_{0,j}\big)\big(1 - f(u_{0,j})\big) - \delta_{i,j}\,. \tag{4.46}$$

Thus, the matrix $\boldsymbol{M}(\boldsymbol{u}_0) \in \mathbb{R}^{N\times N}$ is defined as

$$(M(\boldsymbol{u}_0))_{i,j} = \mu_{i,j}(\boldsymbol{u}_0, \mathbf{0})\,. \tag{4.47}$$

Dropping all terms of quadratic or higher order and inserting Equation 4.40 into Equation 4.38, the Taylor expansion is approximated by

$$\begin{aligned} & g(\boldsymbol{\xi}(t)) = \boldsymbol{M}(\boldsymbol{u}_0) \cdot \boldsymbol{\xi}(t) \\ \Rightarrow \quad & \boldsymbol{\tau} \circ \frac{\partial}{\partial t}\boldsymbol{\xi}(t) = \boldsymbol{M}(\boldsymbol{u}_0) \cdot \boldsymbol{\xi}(t) \\ \Leftrightarrow \quad & \frac{\partial}{\partial t} \cdot \boldsymbol{\xi}(t) = \underbrace{\frac{\mathbf{1}}{\boldsymbol{\tau}} \circ \boldsymbol{M}(\boldsymbol{u}_0)}_{=\boldsymbol{A}(\boldsymbol{u}_0)} \cdot \boldsymbol{\xi}(t)\,. \end{aligned} \tag{4.48}$$

The entries in the matrix $\boldsymbol{A}(\boldsymbol{u}_0)$, hereafter also referred to as the stability matrix, are given by

$$(A(\boldsymbol{u}_0))_{i,j} = \frac{1}{\tau_i} \cdot (M(\boldsymbol{u}_0))_{i,j} = \frac{1}{\tau_i} \cdot \mu_{i,j}(\boldsymbol{u}_0, \mathbf{0}), \tag{4.49}$$

in accordance with the definition for the componentwise multiplication given in the Appendix. As shown in Jetschke (1989), the stability of the system now depends on the real parts of the N (not necessarily different) Eigenvalues $\lambda_1, \ldots, \lambda_N$ of the matrix ${}_c\boldsymbol{A}(\boldsymbol{u}_0)$, the complexification of the matrix $\boldsymbol{A}(\boldsymbol{u}_0)$. The conditions for a stable point of the system are

$$\begin{aligned} &\mathcal{R}\left(\lambda_i\right) \leq 0 \quad \forall 1 \leq i \leq N \\ \Rightarrow &\lim_{t\to\infty} \boldsymbol{\tau} \circ \frac{\partial \boldsymbol{\xi}(t)}{\partial t} = 0. \end{aligned} \tag{4.50}$$

Likewise, for an unstable point, they are

$$\begin{aligned} &\exists \lambda_i : \; \mathcal{R}\left(\lambda_i\right) > 0 \\ \Rightarrow &\lim_{t\to\infty} \left| \boldsymbol{\tau} \circ \frac{\partial \boldsymbol{\xi}(t)}{\partial t} \right| = \infty\,. \end{aligned} \tag{4.51}$$

7.4. A Subset of Fixed Points and Networks. General statements about stationary solutions of the basic fixed point equation seem difficult because they depend on the actual properties of a neural field graph (i.e., the weight matrix, input, resting levels and time scales must be known). However, some properties can be derived given certain restrictions. These properties, along with the restrictions they require, are introduced in the following sections.

7.4.1. *Strong Activity.* One of the restrictions stems from the fact that with the sigmoidal function as nonlinearity all synapses always have at least a small, remaining influence on the neurons they are connected to, even if the activations of the neurons they originate from are far below threshold. In the opposite case, the interaction is never fully realized because the sigmoidal function never becomes exactly one, but always stays slightly below it. Aiming at an approximation for the sigmoidal nonlinearity in these situations, the concept of strong activity is defined.

Definition 4.4 (Strongly active and inactive neuron states, strong activity)**.** A neuron j is considered *strongly active* if

$$f(u_j(t)) \geq 1 - \epsilon \tag{4.52}$$

for some small $\epsilon \in \mathbb{R}_+$. Conversely, a *strongly inactive* state is defined by

$$f(u_j(t)) \leq \epsilon\,. \tag{4.53}$$

When a neuron is either strongly active or strongly inactive, this is also referred to as *strong activity.*

Substituting $u_j(t) = u_{act}(t)$, Equation 4.52 yields that a neuron is strongly active if

$$\begin{aligned} & f(u_{act}(t)) \geq 1 - \epsilon \\ \Leftrightarrow \quad & \frac{1}{1 + e^{-\beta u_{act}(t)}} \geq 1 - \epsilon \\ \Leftrightarrow \quad & \frac{1}{1-\epsilon} - 1 \geq e^{-\beta u_{act}(t)} \\ \Leftrightarrow \quad & u_{act}(t) \geq -\frac{1}{\beta} \ln\left(\frac{1}{1-\epsilon} - 1\right) \\ \Leftrightarrow \quad & u_{act}(t) \geq -\frac{1}{\beta} \ln\left(\frac{\epsilon}{1-\epsilon}\right) \\ \Leftrightarrow \quad & u_{act}(t) \geq -\frac{1}{\beta}\big(\ln(\epsilon) - \ln(1-\epsilon)\big)\,. \end{aligned} \tag{4.54}$$

Conversely, for strong inactivity with $u_j = u_{inact}(t)$, Equation 4.53 yields the symmetric result

$$\begin{aligned} & f(u_{inact}(t)) \leq \epsilon \\ \Leftrightarrow \quad & \frac{1}{1+e^{-\beta u_{inact}(t)}} \leq \epsilon \\ \Leftrightarrow \quad & \frac{1}{\epsilon} - 1 \leq e^{-\beta u_{inact}(t)} \\ \Leftrightarrow \quad & u_{inact}(t) \leq -\frac{1}{\beta} \ln\left(\frac{1}{\epsilon} - 1\right) \\ \Leftrightarrow \quad & u_{inact}(t) \leq -\frac{1}{\beta} \ln\left(\frac{1-\epsilon}{\epsilon}\right) \\ \Leftrightarrow \quad & u_{inact}(t) \leq -\frac{1}{\beta}\big(\ln(1-\epsilon) - \ln(\epsilon)\big)\,. \end{aligned} \tag{4.55}$$

As a consequence, whenever a neuron j exhibits strong activity, the slope of the sigmoidal function at its state can be approximated by

$$\frac{df(u_j(t))}{du_j(t)} = \beta \cdot f(u_j(t)) \cdot (1 - f(u_j(t))) \approx 0\,, \tag{4.56}$$

because either $f(u_j(t)) \approx 0$, or $1 - f(u_j(t)) \approx 0$, if the value for ϵ is selected accordingly. The corresponding j-th column in the stability matrix is thus approximated by

$$(A)_{i,j} = \frac{1}{\tau_i}\left(\beta \cdot (W)_{i,j} \cdot \frac{df(u_j(t))}{du_j(t)} - \delta_{i,j}\right) \approx -\frac{1}{\tau_i}\delta_{i,j} \;\forall i \in \{1,\ldots,N\}\,. \tag{4.57}$$

If the above approximation holds for all neuron states in an activation vector $\boldsymbol{u}$ and the vector fulfills the basic fixed point equation (Equation 4.31), it follows that

$$\boldsymbol{A}(\boldsymbol{u}) \approx -\,diag\left(\frac{1}{\tau_1},\ldots,\frac{1}{\tau_N}\right). \tag{4.58}$$

In other words, $\boldsymbol{A}(\boldsymbol{u})$ can be approximated by a scaling matrix. Thus, its complexification has the trivial Eigenvalues $-\frac{1}{\tau_1}$, ..., $-\frac{1}{\tau_N}$. Therefore, any point that fulfills the basic fixed point equation and contains only strong activation values is also stable because, per definition, $\tau_i > 0$ and thus $\mathcal{R}\left(-\frac{1}{\tau_i} < 0\right)$ for all $i \in \{1,\ldots,N\}$.

7.4.2. *Single Peak Solutions.* The goal in this section is to extract the circumstances for the existence of a stationary point $\boldsymbol{u}$ for which only one entry is above the threshold, i.e., a stationary solution $\boldsymbol{u}$ restricted to

$$u_i \begin{cases} > 0 & : \quad i = k \\ < 0 & : \quad \text{otherwise} \end{cases} \tag{4.59}$$

for some $k \in \{1,\ldots,N\}$. Furthermore, it is assumed that all neurons described by $\boldsymbol{u}$ exhibit strong activity, specifically that u_k is strongly active and u_i is strongly inactive for $i \neq k$. Therefore, if $\boldsymbol{u}$ fulfills the basic fixed point equation, it is also a stable state due to Equation 4.58. From the constraints on the activity, if follows that

$$(f(\boldsymbol{u}))_i \approx \begin{cases} 1 & : \quad i = k \\ 0 & : \quad \text{otherwise.} \end{cases} \tag{4.60}$$

Inserting this into the basic fixed point equation yields

$$\begin{aligned} \boldsymbol{u} &= \boldsymbol{W} \cdot f(\boldsymbol{u}) + \boldsymbol{c} \\ &\approx \boldsymbol{W} \cdot (0,\ldots,0,1,0,\ldots,0)^T + \boldsymbol{c}\,. \\ &= (W)_{\cdot,k} + \boldsymbol{c}\,. \end{aligned} \tag{4.61}$$

Switching to index notation and expanding the definition of $\boldsymbol{c}$, the following relation can be derived for $\boldsymbol{u}$:

$$u_i \approx (W)_{i,k} + s_i + h_i \,. \tag{4.62}$$

From the conditions for the strong inactivity it follows that for $i \neq k$

$$(W)_{i,k} + s_i + h_i \approx u_i < -\frac{1}{\beta}\left(\ln(1-\epsilon) - \ln(\epsilon)\right) \tag{4.63}$$

and

$$(W)_{k,k} + s_k + h_k \approx u_k > -\frac{1}{\beta}\left(\ln(\epsilon) - \ln(1-\epsilon)\right) \,. \tag{4.64}$$

7.4.3. *Maximum-Slope Single Peak Solutions.* In Section 7.4.2, the influence of changes in the activation are negligible because of the assumption of strong activity, resulting in $\boldsymbol{A}(\boldsymbol{u})$ being a diagonal matrix as given in Equation 4.58. In the present case, solutions for which there exists a k such that the value of $f'(u_{0,k})$ becomes maximal are investigated. All other conditions are the same as in Section 7.4.2.

As an analysis of minima and maxima reveals, $f'(u)$ becomes maximal at $u = 0$. The value at this point is

$$f'(0) = \beta \cdot f(0) \cdot (1 - f(0)) = \beta \cdot \frac{1}{2} \cdot \left(1 - \frac{1}{2}\right) = \frac{\beta}{4} \,. \tag{4.65}$$

Therefore, we assume that $u_{0,k} = 0$. From the basic fixed point equation if follows that

$$\begin{aligned} \boldsymbol{u}_0 &= \boldsymbol{W} \cdot f(\boldsymbol{u}_0) + \boldsymbol{c} \\ &\approx \boldsymbol{W} \cdot \left(0, \cdots, f(u_k) = f(0) = \frac{1}{2}, \cdots, 0\right)^T + \boldsymbol{c} \\ \Rightarrow \qquad u_{0,i} &\approx \frac{1}{2}(W)_{i,k} + c_i \ \forall\ i \,. \end{aligned} \tag{4.66}$$

Inserting the condition that $u_{0,k} = 0$ into Equation 4.66 reveals that $(W)_{k,k} \approx -2c_k$. The stability matrix becomes

$$(A)_{i,j} = \begin{cases} -\frac{\delta_{i,j}}{\tau_i} & : \quad j \neq k \\ \frac{\beta}{\tau_i} \cdot (W)_{i,k} \cdot \frac{\beta}{4} - \delta_{i,k} = \frac{\beta^2}{4\tau_i} \cdot (W)_{i,k} - \frac{\delta_{i,j}}{\tau_i} & : \quad j = k \end{cases} \tag{4.67}$$

To calculate the eigenvalues of ${}_c\boldsymbol{A}(\boldsymbol{u}_0)$, the following system of equations needs to be solved:

$$(i \neq k) \qquad -\frac{v_i}{\tau_i} + \left(\frac{\beta^2}{4\tau_i}(W)_{i,k}\right) \cdot v_k = \lambda \cdot v_i \tag{4.68}$$

$$\text{and} \qquad -\frac{v_k}{\tau_i} + \left(\frac{\beta^2}{4\tau_k}(W)_{k,k} - \frac{1}{\tau_k}\right) \cdot v_k = \lambda \cdot v_k$$

$$\Leftrightarrow \qquad v_k \cdot \left(\frac{\beta^2}{4\tau_k}(W)_{k,k} - \frac{2}{\tau_k}\right) = \lambda \cdot v_k \,. \tag{4.69}$$

From Equation 4.69 it follows that there are two cases:

$v_k \neq 0$: In this case, Equation 4.69 means that the eigenvalue corresponding to the eigenvector is

$$\lambda = \frac{\beta^2}{4\tau_k}(W)_{k,k} - \frac{2}{\tau_k} \,. \tag{4.70}$$

The remaining v_i can then be calculated by inserting the eigenvalue and v_k into Equation 4.68.

$v_k = 0$: In this case, Equation 4.68 states[3] that

$$\lambda = -\frac{1}{\tau_i} \,. \tag{4.71}$$

The real part of the eigenvalue in Equation 4.71 is always negative. Therefore, because of Equation 4.70, the stability of the fixed point only depends on the value of $(W)_{k,k} \approx -2c_k$, β and τ_k. It follows that

$$\begin{aligned} & 0 < \lambda = \frac{\beta^2}{4\tau_k}(W)_{k,k} - \frac{2}{\tau_k} \approx -\frac{\beta^2}{2\tau_k}c_k - \frac{2}{\tau_k} \\ \Rightarrow \qquad & c_k < -\frac{4}{\beta^2} \,. \end{aligned} \tag{4.72}$$

Thus, $\boldsymbol{u}_0$ as described above is stable if Equation 4.72 is true.

8. An Efficient Algorithm for the Construction and Update of the Extended Neural Field Graph

Although no general statements can be made about the structure of the neural field graphs, graphs containing a large number of neurons and only a small number of edges likely result in sparse extended graphs when δ_{max} is relatively small. Therefore, the algorithms presented here are based on graphs stored as adjacency lists. Although disadvantageous for the construction of the field graph (where an adjacency matrix has to be constructed from the lists in order to calculate the shortest path lengths using the Floyd-Warshall algorithm), this type of data structure allows for iteration of all incident neurons — one of the tasks repeated frequently during the graph update (see Section 8.2) — in linear time. The construction on the other hand is only performed once. Therefore, the decrease in performance due to the adjacency list storage is considered less relevant.

8.1. Direct Weight Balancing. In order to calculate the balanced weights w' for synapses in the extended neural field graph without prior knowledge of the unbalanced weights w, we use that from the definition

$$w(u,v) = \hat{w}_{i,j}(u,v) = \mu \cdot w_{i,j}(\sigma \cdot \delta(v,u)) \,. \tag{4.73}$$

it follows that

$$w(u,v) = w(u',v') \Leftarrow \delta(v,u) = \delta(v',u') \tag{4.74}$$

for neurons u, v, u', v' that satisfy $i = layer(u) = layer(u')$ and $j = layer(v) = layer(v')$. Given the reverse mapping of the (possibly non-injective) weight function,

$$w_{i,j}^{-1}(\omega) = \left\{\delta \middle| w_{i,j}(\delta) = \omega\right\} \,, \tag{4.75}$$

Equation 4.14 can be rewritten to

$$indeg_{\omega,l}(v) = \left|\left\{(u,v) \in E^e \middle| \forall\delta(v,u) \in w_{i,j}^{-1}(\omega) \wedge layer(u) = l\right\}\right| \,. \tag{4.76}$$

Although Equation 4.76 requires inverting the weight function, this can be done more efficiently by sampling the weight at the relevant distance values given as $d \in \{0, \ldots, \delta_{max}\} \subseteq \mathbb{N}$ and calculating the reverse mapping of those values once, before the construction process. The number of neurons meeting the criterion can then be counted from the distance matrix $\boldsymbol{D}$, and λ_{ω,l_1,l_2} can be calculated accordingly.

[3]This holds for at least one v_i with $v_i \neq 0$. Such a v_i always exists because otherwise $\boldsymbol{v}$ would be $\boldsymbol{0}$, and thus by definition not an eigenvector.

Algorithm 4.1 *constructBalancedFieldGraph*$(F_{in} = (V_{in}, E_{in}), \delta_{max}, \mu, \sigma)$

The symbols used are $M = \{1, \dots, m\}$, the set of layers; $w_{i,j}$, the interaction kernel between layers i and j.

```
Require: $V_{in} = (v_1, \dots, v_n)$
1: $(D)_{k,l} \leftarrow \delta(k, l),\ \forall\ k, l \in V_{in}$
2: for $(k, l) \in M \times M = M^2$ do
3:     for $d = 0$ to $\delta_{max}$ do
4:         $\hat{w}_{k,l,d} \leftarrow \mu \cdot w_{k,l}(\sigma \cdot d)$
5: $r, \lambda \leftarrow preBalance(\hat{w}, M, \delta_{max}, \boldsymbol{D})$
6: $F \leftarrow (V = V_{in}, E = \emptyset)$
7: for all $\{v\} \in V_{in}$ do
8:     $connect(v, F_{in}, F)$
```

Algorithm 4.2 *preBalance*$(\hat{w}, M, \delta_{max}, \boldsymbol{D})$

```
1: for $(i, j) \in M^2$ do
2:     $W_{i,j} \leftarrow \emptyset$
3:     for $d = 0$ to $\delta_{max}$ do
4:         $w \leftarrow \hat{w}_{i,j,d}$
5:         $W_{i,j} \leftarrow W_{i,j} \cup w$
6:         $r_{i,j}(w) \leftarrow r_{i,j}(w) \cup d$
7: for $(i, j) \in M^2$ do
8:     for $w \in W_{i,j}$ do
9:         $\lambda_{w,i,j} \leftarrow \max \left| \left\{ (k, l) \in V^2 \middle| (D)_{k,l} \in r_{i,j}(w), layer(k) = i, layer(l) = j \right\} \right|$
10: return $r, \lambda$
```

Algorithm 4.3 *connect*$(v, F_{in} = (V, E_{in}), F = (V, E))$. This algorithm is a modified graph exploration algorithm similar to Breadth First Search.

```
Require: $v \in V$
1: $S \leftarrow \{v\}$, $H \leftarrow \emptyset$
2: repeat
3:     $u' \in S$, $S \leftarrow S \setminus u'$
4:     if $u' \in H$ then
5:         continue from line 3
6:     else
7:         $H \leftarrow H \cup \{u'\}$
8:     for all $u \in \Gamma_{F_{in}}(u')$ do
9:         $d \leftarrow (D)_{v,u}$
10:        if $d \leq \delta_{max}$ then
11:            $S \leftarrow S \cup u$
12:            $w \leftarrow \hat{w}_{l_v, l_u, d}$
13:            $l_v \leftarrow layer(v)$, $l_u \leftarrow layer(u)$
14:            $a \leftarrow a_{w,l_u}(v)$
15:            $w' \leftarrow \frac{w}{a}$
16:            $E \leftarrow E \cup (u, v)$ with edge weight $w'$
17: until $H = \emptyset$
```

8.2. Extended Neural Field Graph Construction and Simulation. The algorithms for the construction and simulation of an extended neural field graph resulting from the discussions above are given in Algorithm 4.1–4.4.

Algorithm 4.4 *eulerStep*$(\Delta t, F^e = \big(G^e, u, w\big) sh\tau\gamma, \hat{w})$

```
1: $a \leftarrow 0$
2: for all $v \in V$ do
3:     $l \leftarrow layer(v)$
4:     $d_u(v) \leftarrow -u(v) + s(v) + h_l$
5:     $f_u \leftarrow f(u(v))$
6:     $a \leftarrow a + \frac{\Delta t}{\tau_l} \cdot f_u$
7:     $d_u(v) \leftarrow d_u(v) + \hat{w}_{l,l,0} \cdot f_u$
8:     for $v' \in \Gamma_{in}(v)$ do
9:         $d_u(v) \leftarrow d_u(v) + w(v', v) \cdot f(u(v'))$
10:    $d_u(v) \leftarrow \frac{\Delta t}{\tau_l} \cdot d_u(v)$
11: $c \leftarrow 0$
12: for all $v \in V$ do
13:    $d_u(v) \leftarrow d_u(v) + \gamma_g \cdot a$
14:    $c \leftarrow c + |d_u(v)|$
15:    $u(v) \leftarrow u(v) + d_u(v)$
```

8.2.1. *Complexity Analysis.* The asymptotic complexity of the algorithms introduced above is investigated with regard to the size of the neural field graph. First, the auxiliary functions *connect* and *preBalance* are treated. Afterwards, we turn our attention to the complexity of the construction of the extended field graph and the neuron update during runtime.

The *preBalance* function is dominated by four loops, totaling $\mathcal{O}\left(|M|^2 \cdot \delta_{max}\right)$ iterations.[4] Given that δ_{max} does not depend on the size of the source graph, this can be reduced to $\mathcal{O}\left(|M|^2\right)$. The iterations in the first outer loop run in constant time. Because $|M| \leq N$, it runs in $\mathcal{O}\left(|N|^2\right)$ time. In each iteration of the second outer loop, the maximum is calculated across all pairs in a subset of neurons. Iterating over all pairs of layers and in each such pair over all pairs of neurons is equivalent to iterating over all pairs of neurons. Therefore, the second loop runs in $\mathcal{O}\left(N^2\right)$ time, and thus overall, *preBalance* terminates in $\mathcal{O}\left(N^2\right)$ steps.

The *connect* function is essentially a Breadth First Search, modified so that it only explores the graph until a depth of δ_{max} is reached. Because Breadth First Search is known to have a runtime of $\mathcal{O}\left(|V| + |E_{in}|\right)$ (Cormen et al., 2001), the same is assumed for *connect.*

The construction of the extended neural field graph (Algorithm 4.1) contains two loops. The first one runs in $\mathcal{O}\left(|M|^2 \cdot \delta_{max}\right) = \mathcal{O}\left(|M|^2\right) = \mathcal{O}\left(N^2\right)$ time. The second loop iterates over all neurons and calls the connect function for each neuron, thus resulting in a complexity of $\mathcal{O}\left(N \cdot (N + |E_{in}|)\right) = \mathcal{O}\left(N^2 + N|E_{in}|\right)$. In addition to the loops, *constructBalancedFieldGraph* also contains a call to *preBalance* and calculates the shortest paths between all neurons. The latter can be done using the Floyd-Warshall algorithm presented in Chapter 3, Section 3.2. Thus, considering that the number of edges in a graph cannot exceed the square of the number of nodes, the overall complexity of *constructBalancedFieldGraph* is given by

$$\mathcal{O}(\underbrace{N^3}_{\text{Floyd-Warshall}} + \underbrace{N^2}_{\text{First loop}} + \underbrace{N^2}_{\textit{preBalance}} + \underbrace{N^2 + N|E_{in}|}_{\textit{connect}\text{ calls}}) = \mathcal{O}\left(N^3\right) . \tag{4.77}$$

The update of the neural field graph contains two iterations over all neurons. The first one contains another iteration over all incident neighbors of the current

[4]The second inner loop undergoes $|W_{i,j}|$ iterations. Because $W_{i,j}$ is constructed in the first loop by inserting the synaptic weight corresponding to d into $W_{i,j}$, it follows that $|W_{i,j}| \leq \delta_{max}$.

neuron,[5] thus its complexity is given by

$$\mathcal{O}\left(\sum_{v\in V}|\Gamma_{in}(v)|\right) = \mathcal{O}\left(|E|\right) . \tag{4.78}$$

The second loop contains only instructions that run in constant time, thus the overall complexity of *eulerStep* is given as $\mathcal{O}\left(|V| + |E|\right)$. Given that the construction of the graph fully connects a node to all of its neighbors up to a certain distance, the resulting graph contains significantly more edges than the input graph. Therefore, a pessimistic but more accurate estimate for the number of edges is given by $|E| = O\left(|V|^2\right)$, and the overall complexity of *eulerStep* thus becomes $\mathcal{O}\left(|V|^2\right)$.

[5]Because graphs are stored using adjacency lists, the time for the retrieval of the neighbors is neglected here.

Part 4

Experimental Findings and Conclusion

CHAPTER 5

Neural Field Properties

Chapter 2 demonstrates a number of properties specific to the neural field model with exemplary instances of neural fields. These properties include the ability to form input peaks that can merge when they are close together in the input space. Additional examples show that a neural field is capable of selecting one of two (or more) peaks in the field's input. Furthermore, peaks may be self-sustained if the input disappears. Because dynamical systems are a central part of the neural field model, it also inherits the property of being dependent on previous states, and thus, the history of the system.

In the present chapter, the neural field graph model presented in Part 3 is examined with regard to these properties. Examples are constructed that show that these characteristics can be replicated in the model if its parameters are selected accordingly.

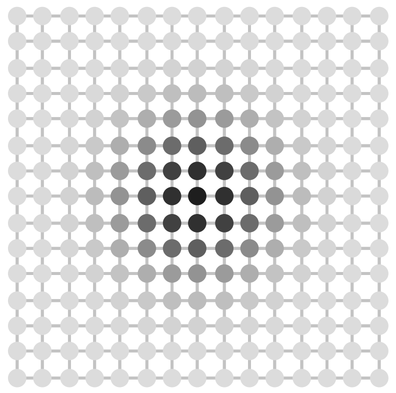
Certain parameters are kept constant during most of the experiments. Unless stated otherwise, the maximum extraction distance δ_{max} is set to 3 and the convergence threshold is selected as $\epsilon = 0.0001$. The default interaction sampling distance is set to $\sigma = 0.5$, and the sigmoidal function used as nonlinearity is calculated with $\beta = 4$. The activation values of all neurons in the neural field graphs are initialized to 0, and whenever a Gaussian interaction kernel is used to determine the synaptic weights between neurons, its standard deviation is set to $\sigma = 1$. Many of the experiments use random connected graphs as a basis for the neural field graphs. For those that do, the graphs are generated with the corresponding algorithm presented in Section 2 of the Appendix.

1. Peak Formation

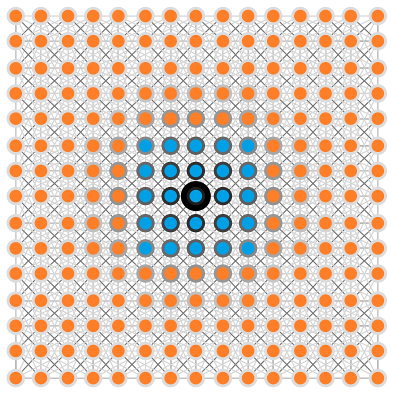
The first experiment is performed to find evidence of the successful formation of peaks. Due to the use of a 15×15 mesh structure for the neural field graph, this experiment is closely related to the simulation of a two-dimensional one-layer neural field. Neurons in the graph are therefore assigned a position in a two-dimensional input space based on the planar representation of the mesh graph. The strength of the input is based on the distance from the graph's center in the input space (see Figure 5.1a). The extended neural field graph is shown in Figure 5.1b. For the simulation, a global inhibition factor of $\gamma = 5$ is used, as well as a Gaussian interaction kernel, scaled by a factor of $\mu = 70$. Figure 5.1 indicates that a network with this setup is capable of forming a stable peak centered around the neuron with maximal input.

A second experiment is performed to show that a similar result can be achieved on a different, less regular graph that further deviates from the standard discretized representation of neural fields. In this case, a random connected graph is used instead of a mesh graph. The random graph is set to contain 250 nodes and 350 edges. The extended neural field graph constructed from it contains 7342 synapses (excluding connections from a neuron to itself). Given that the maximum number of edges (also excluding connections from nodes to themselves) for any directed graph is $2 \cdot (N \cdot (N - 1)) = 124500$, where N is the number of nodes, the expectation that

Figure 5.1 Evidence of peak-formation. Records are taken after the neural field graph converges with a threshold of $\epsilon = 0.0001$.



(a) The input of the neural field graph. The color of the neurons visualizes the value of their input with black representing the highest value, in this case 10, and the brightest shade of gray representing the lowest value of 0.



(b) The extended neural field graph after convergence. Neurons with an activation value that lies below threshold are colored orange, others are shown in blue. The ring around each neuron represents the its activation value, black being the highest activation and light gray the lowest. The bold black ring represents the peak center, selected based on the highest activation.

extended neural field graphs with a large number of neurons remain relatively sparse (see Section 8 of Chapter 4) seems to apply in this case. Additional parameters of the neural field graph are mostly the same as in the experiment that uses the mesh graph, only the interaction scale is modified to $\mu = 30$. Results of the experiment are shown in Figure 5.2. As before, a stable peak appears around one of the neurons receiving the highest amount of input.[1]

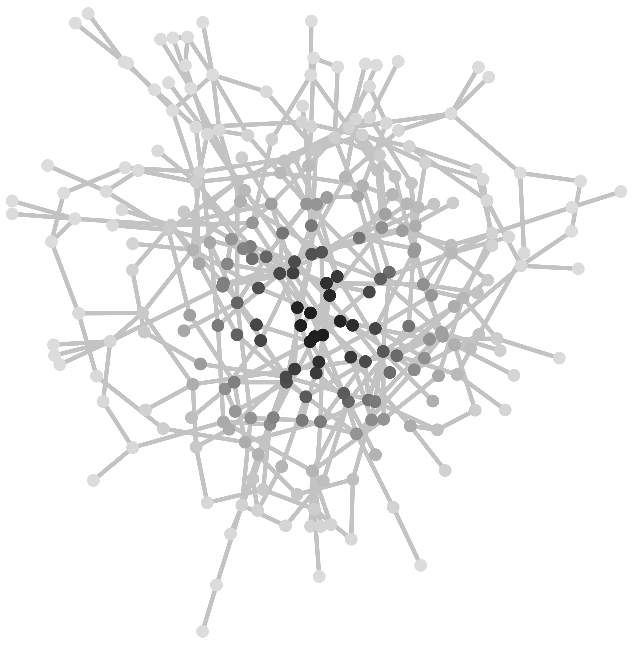
2. Merging Peaks

The next experiment studies the neural field graph model's capability for merging peaks that is also found in neural fields. The basis for the experiments is an extended neural field graph, created from a random connected graph. This graph contains 200 neurons and 210 synapses; the extended neural field graph contains 2826 synapses; and the interaction is given by a Gaussian kernel with a scaling factor of $\mu = 2$. No global inhibition is present (i.e., $\gamma = 0$). In order to facilitate the movement of the input peaks, the neurons are assigned a position in a two-dimensional feature space by the algorithm described in (Fruchterman et al., 1991), and the input strength for each neuron is determined by its distance from two points in the feature space. During each experiment, the graphs are created with this setup and simulated until convergence is achieved, and for each experiment, the points are moved closer together.

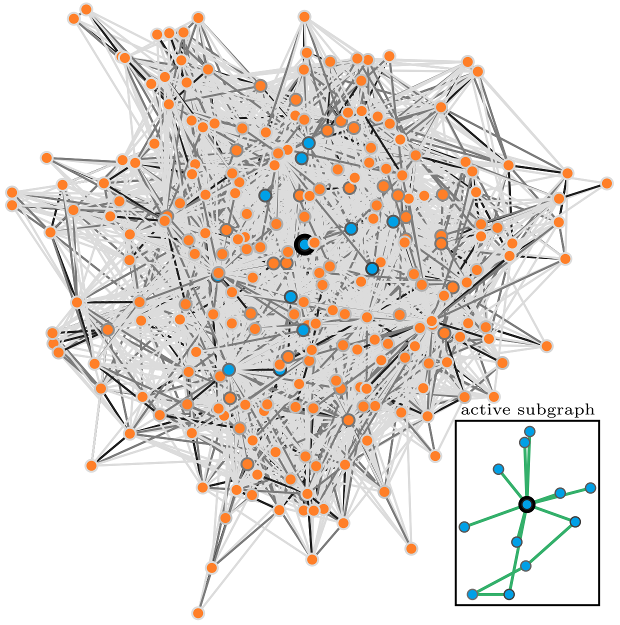
The results of these experiments are shown in Figure 5.3. Figure 5.3a indicates that two input peaks that are far apart from each other result in two distinct components in the active subgraph, corresponding to two peaks in the neural field graph. In Figure 5.3b, the peaks are closer together, and the two peaks are merged together because only one component is present in the active subgraph. However,

[1]In this case, more than one neuron receives the maximum amount of input; in the previously presented experiment that uses the mesh graph, only one such neuron exists.

Figure 5.2 Plots demonstrating the ability of the neural field graph model to form stable peaks on a connected random graph structure.

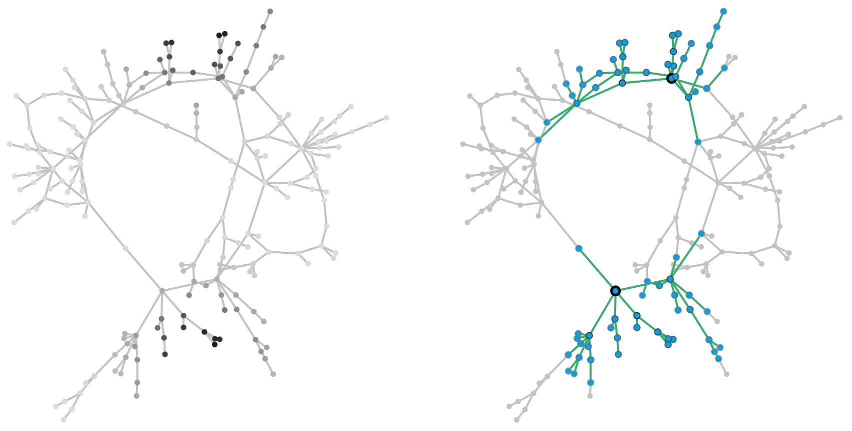


(a) Superimposition of the neural field graph, generated with the algorithm for random connected graphs described in Section 2 of the Appendix, and the input for each neuron. Each node in the graph represents a neuron. The darker the node, the higher the input, with black being an input of 10.0 and (almost) white being an input of 0.0.

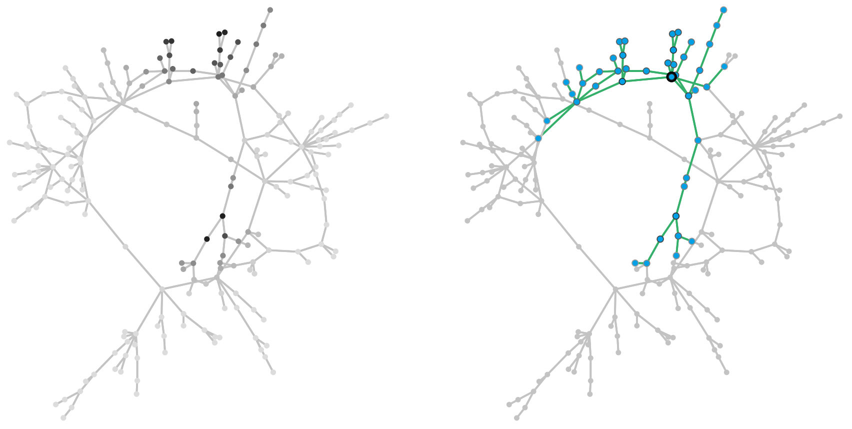


(b) The neural field graph after convergence. The active subgraph is shown in the lower right corner.

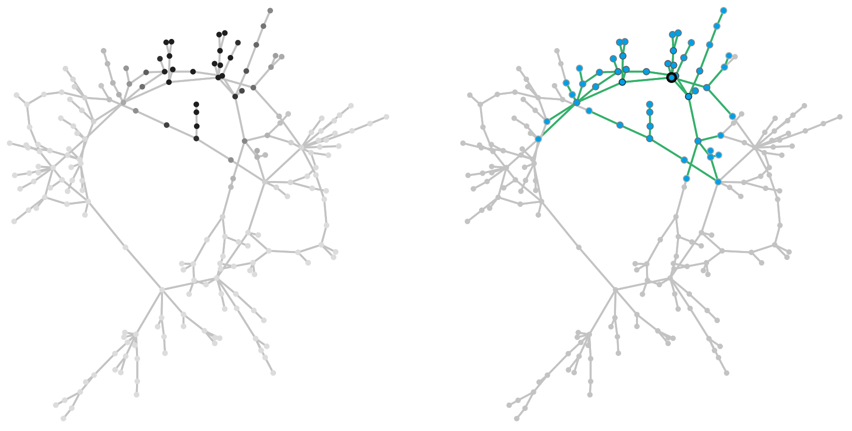
Figure 5.3 Merging of peaks in a neural field graph. The left column shows the graph structure and the input. As in Figure 5.2a, darker nodes represent stronger input. The right column highlights the active subgraphs of the converged neural field graph.



(a) The input peaks are far apart, one at the top of the image, one at the bottom. Two distinct components are present in the active subgraph.



(b) The bottom input moves upward and approaches the other peak. Both components of the active subgraph have now merged into one. However, the merged component is stretched out to include the lower input peak.



(c) Both peaks are now very close together. The active subgraph now looks similar to the upper component in Figure 5.3a.

the appearance of the remaining active subgraph is different from the active components exhibited in the case in which the input peaks are located further apart. In Figure 5.3c, the input peaks are close together, and the remaining component of the active subgraph is similar in structure to the one near the upper input peak in Figure 5.3a.

3. Self-Sustained Peaks

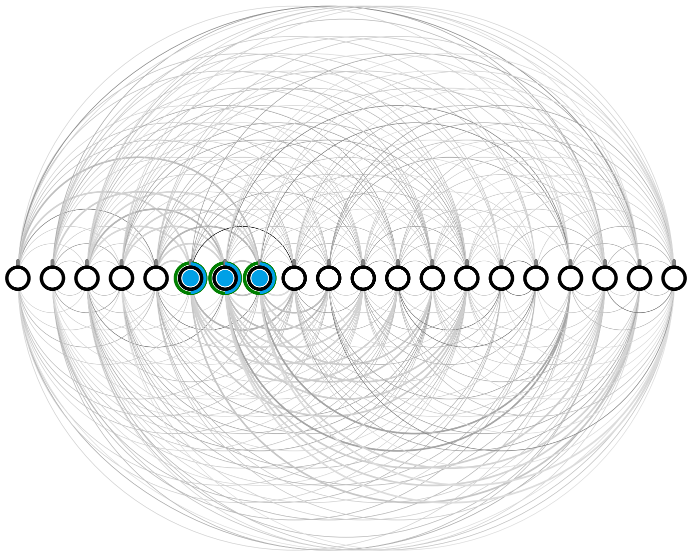
This experiment is conducted in order to determine whether the neural field graph model also has the capacity for self-sustained peaks. As described in Part 1, self-sustained peaks are those peaks in neural fields that remain active even when the input causing them disappears. This condition is simulated in the neural field graph by initializing a number of randomly selected neurons with a positive activation (specifically, with an activation value of one) and setting the input to zero for each neuron. Furthermore, the neural field graph is balanced and uses a Gaussian interaction kernel scaled by $\mu = 5$, with a global inhibition factor of $\gamma = 5$. The extended neural field graph is constructed from a random connected graph.

The results of this experiment are shown in Figure 5.4. As the converged neural field graph in Figure 5.4b suggests, some neurons retain above-threshold activation even in the absence of input. Because the simulation has converged (i.e., the final state of the simulation is probably a stable one), this result can be seen as evidence that the neural field graph model is capable of exhibiting self-sustained peaks.

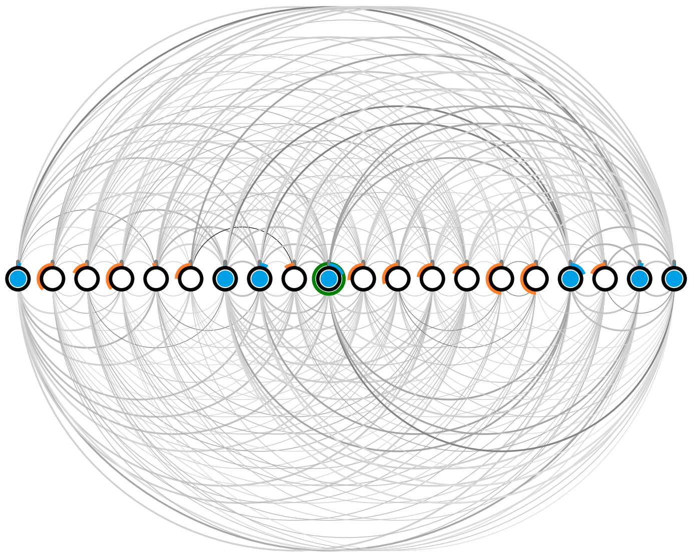
4. Selection

In neural fields, selection means that only one peak reaches above-threshold activation, even if more than one peak is present in the input. In order to replicate this property, a neural field graph based on a random connected graph is created. The input of the neurons is determined randomly, drawn uniformly from the interval $[0, 6]$. The interaction is based on a Gaussian kernel scaled by $\mu = 0.1$. Figure 5.5 allows for the hypothesis that, as with neural fields, global inhibition plays a central role in performing selection with neural field graphs. This is due to the fact that in the absence of global inhibition, multiple peaks can form in a neural field graph, evidenced by the number of active subgraphs (see Figure 5.5c). Once global inhibition is enabled, all but one peak are suppressed (see Figure 5.5d).

Figure 5.4 A representation of an extended neural field graph exhibiting a self-sustained peak. Neurons filled blue are active (above-threshold), the arc around each neuron represents its normalized activation. Pointing up is zero, meaning that a clockwise arc indicates positive activation (colored blue), while an counterclockwise arc indicates negative activation (colored orange). Connections between neurons are represented by arcs between them, with darker color meaning a stronger connection. The darkest connections have a strength of approximately 10.6, the lightest connections have one of approximately 0.65. The ones above the neurons connect the neuron on the right to the one on the left, and the ones below the neurons run in the opposite direction. A green circle around a neuron indicates that it is selected as a peak center.

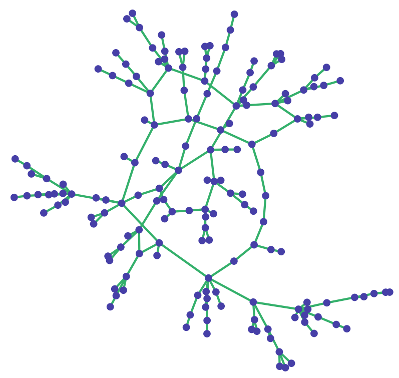


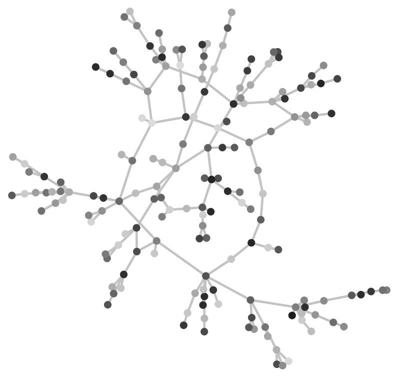
(a) The initial (instable) state of the neural field graph, before any simulation takes place. All active neurons are initialized with an activation value of one, the remainder is initialized with zero activation.

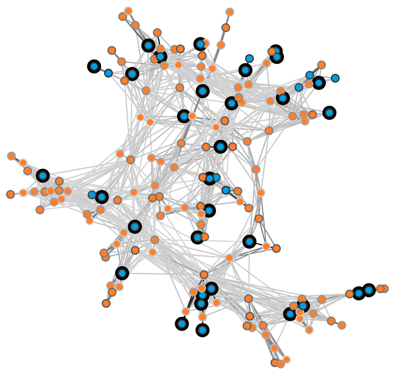


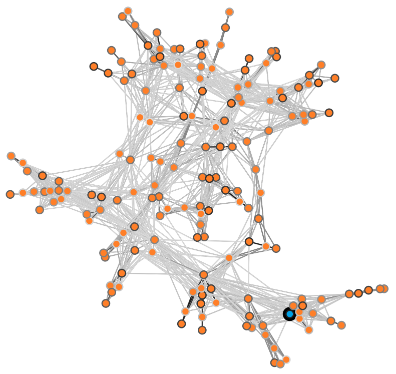
(b) The state of the neural field graph after convergence. The connections between the neurons are drawn thicker if the neuron from which the connection originates is active.

Figure 5.5 Plots demonstrating the ability of the neural field graph model to perform selection. Figure 5.5c and Figure 5.5d show an extended neural field graph after convergence is achieved. Lines between neurons represent the weight between them (darker means stronger weight). Orange neurons are inactive (i.e., below threshold) while blue ones are active. The gray circles around a neuron indicate the strength of its activation, darker circles once again mean stronger activity. A second, bold black circle around a neuron indicates that the neuron is detected as a peak center in its active subgraph. Selection appears to occur when the global inhibition factor is set to a positive value, thus suppressing any secondary peaks.


(a) Structure of the graph used as a basis for constructing the extended neural field graph.


(b) Input for each neuron. The darkness of a neuron indicates the strength of its input, with input being randomly selected from the range 0 to 6.


(c) The neural field graph fails to perform selection in the case where global inhibition is absent (i.e., $\gamma = 0$).


(d) This neural field graph has a global inhibition factor of $\gamma = 0.7$. Only one component exists in the active subgraph. Therefore, the neural field graph seems to successfully perform selection in this case.

CHAPTER 6

Fixed Point Search

Two methods are devised in order to determine how the parameters of the neural field graph model affect its behavior. Both find a large number of fixed points given an actual instance of an extended neural field graph. The first one uses an iterative method to find solutions to the model's basic fixed point equation (Equation 4.31); the second one simulates the model using a forward Euler approach. These methods have one thing in common: they need a starting point $\boldsymbol{u}_c$, henceforth referred to as candidate solution, from which to iterate or simulate towards a solution. For the iterative method, this starting point represents the initial guess for a solution $\boldsymbol{u}_0$ to Equation 4.31; for the simulation of the model, $\boldsymbol{u}_c$ represents the initial activation of the neurons.

1. Methods for Determining Stationary Solutions

Based on the observation that all stationary solutions of a neural field graph lie inside a linearly transformed hypercube (see Section 7.2), and the fact that the sigmoidal function maps most values to either approximately zero or one, the set of candidate solutions S is defined as the set of edges of the hypercube, i.e.,

$$S = \left\{ \boldsymbol{W} \cdot \boldsymbol{e} + \boldsymbol{c} \middle| \boldsymbol{e} \in \{0,1\}^N \right\}, \tag{6.1}$$

where $\boldsymbol{W}$ and $\boldsymbol{c}$ are the weight matrix and combined input and resting level as described for Equation 4.31. However, there is a drawback to this definition. The number of edges on a N-dimensional cube is given by $2^N = |S|$. As a consequence, using any of the methods below with this candidate solution set becomes computationally infeasible when the number of neurons in the neural field graph increases.

1.1. Iterative Determination of Solutions. To acquire solutions of the basic fixed point equation of the model, Newton's method for solving nonlinear equations is used (Bronstein et al., 2008). For that purpose, Equation 4.31 is restated to

$$F\left(\boldsymbol{u}_0^{(\mu+1)}\right) = \boldsymbol{W} \cdot f\left(\boldsymbol{u}_0^{(\mu+1)}\right) + \boldsymbol{c} - \boldsymbol{u}_0^{(\mu+1)} = \boldsymbol{0}\,. \tag{6.2}$$

Then, the Taylor expansion of the left hand side is calculated around $\boldsymbol{u}_0^{(\mu)}$, and all terms of quadratic or higher order are dropped:

$$F\left(\boldsymbol{u}_0^{(\mu)}\right) + \boldsymbol{J}F\left(\boldsymbol{u}_0^{(\mu)}\right) \cdot \underbrace{\left(\boldsymbol{u}_0^{(\mu+1)} - \boldsymbol{u}_0^{(\mu)}\right)}_{=\boldsymbol{d}^{(\mu)}} = \boldsymbol{0}\,, \tag{6.3}$$

where $\boldsymbol{J}F\left(\boldsymbol{u}_0^{(\mu)}\right)$ is the Jacobi matrix of F evaluated at $\boldsymbol{u}_0^{(\mu)}$. By restating the equation for $\boldsymbol{d}^{(\mu)}$, the value of $\boldsymbol{u}_0^{(\mu+1)}$ can be derived as $\boldsymbol{u}_0^{(\mu+1)} = \boldsymbol{u}_0^{(\mu)} + \boldsymbol{d}^{(\mu)}$ (i.e., it can be calculated iteratively if the values for $\boldsymbol{d}^{(\mu)}$ are known). Thus, Equation 6.3 is solved for $\boldsymbol{d}^{(\mu)}$:

$$\boldsymbol{d}^{(\mu)} = -\left(\boldsymbol{J}F\left(\boldsymbol{u}_0^{(\mu)}\right)\right)^{-1} \cdot F\left(\boldsymbol{u}_0^{(\mu)}\right)\,. \tag{6.4}$$

Substituting, for the sake of notational simplicity, $\boldsymbol{u}_0^{(\mu)}$ with $\boldsymbol{x}$, the entries of the Jacobi matrix can now be calculated as

$$\begin{aligned}(JF)_{i,j} &= \frac{\partial F_i(\boldsymbol{x})}{\partial \boldsymbol{x}_j} \\ &= \frac{\partial}{\partial \boldsymbol{x}_j}\left(\sum_{k=1}^{N} (W)_{i,k} \cdot f(x_k) + c_i - x_i\right) \\ &= (W)_{i,j} \cdot \frac{\partial}{\partial \boldsymbol{x}_j} f(x_j) - \delta_{i,j} \\ &= (W)_{i,j} \cdot \beta \cdot f(x_j) \cdot (1 - f(x_j)) - \delta_{i,j}\,, \end{aligned} \tag{6.5}$$

where $\delta_{i,j}$ is the Kronecker delta (Bronstein et al., 2008). In matrix-vector notation, the Jacobi matrix is given as

$$\boldsymbol{J}F(\boldsymbol{x}) = \boldsymbol{W} \cdot (f'(\boldsymbol{x}) \cdot \boldsymbol{1}^T) - \boldsymbol{I}, \tag{6.6}$$

where $f'(\boldsymbol{x})$ is the componentwise evaluation of the first order derivative of f with respect to x given by

$$(f'(\boldsymbol{x}))_i = f(x_i) \cdot \big(1 - f(x_i)\big)\,. \tag{6.7}$$

The iteration of this method is performed until the change of the candidate solution falls below a predetermined threshold ϵ_{Newton}, measured by

$$\left|\boldsymbol{d}^{(\mu)}\right| < \epsilon_{Newton} \tag{6.8}$$

In order to evaluate the quality of a solution $\boldsymbol{u}$ approximated in this manner, an error measure is defined. It is based on the fact that a solution must fulfill the basic fixed point equation, rewritten to the target equation

$$\boldsymbol{u} - (\boldsymbol{W} \cdot f(\boldsymbol{u}) + \boldsymbol{c}) = \boldsymbol{0}\,. \tag{6.9}$$

Thus, the solution error measures how much an approximated solution $\hat{\boldsymbol{u}}$ deviates from this target equation as

$$e(\hat{\boldsymbol{u}}) := (\hat{\boldsymbol{u}} - (\boldsymbol{W} \cdot f(\hat{\boldsymbol{u}}) + \boldsymbol{c}))^2\,. \tag{6.10}$$

1.2. Determination of Solutions via Simulation. When approximating fixed points with a simulation, a neural field graph is created with the desired parameters, and the activation of the neurons is set according to the values stored in a candidate solution. Thus, if $\boldsymbol{u}_c$ is the candidate solution and F the neural field graph with the neurons $V = \{1, \ldots, N\}$, then the activation for each neuron is set to

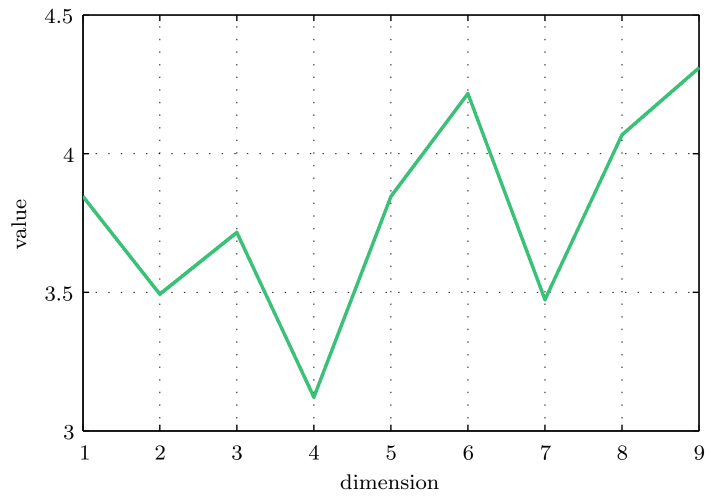
$$u(i, 0) = u_{c,i}\,. \tag{6.11}$$

The neural field graph is then simulated until the accumulated change falls below a certain threshold:

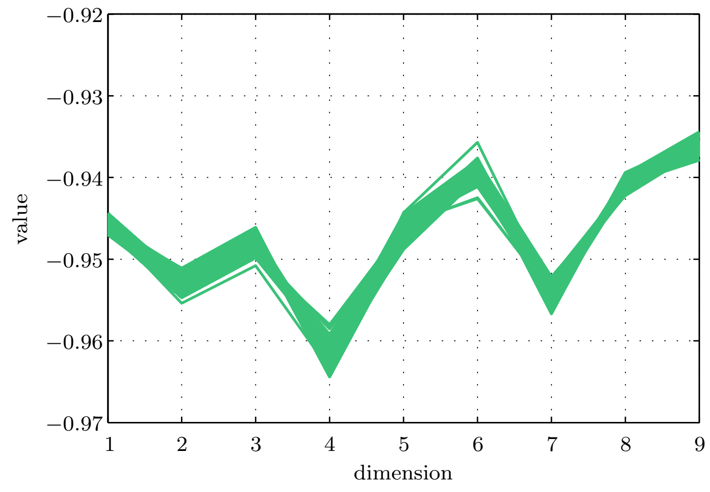
$$h \sum_{n=1}^{N} \frac{\partial}{\partial t} u(n, t) < \epsilon_{Simulation}, \tag{6.12}$$

where h is the time step of the forward Euler method used for the simulation of the neural field graph.

Figure 6.1 Representation of the points recorded from converged simulations with initial activation values for neurons selected from the set of candidate solutions, S. The value of each entry p_i in a point $\boldsymbol{p} = (p_1, p_2, \ldots, p_n)$ is plotted here; the "dimension" axis represents the index i, the "value" axis represents the activation value p_i of neuron i. Note that the order of the index is arbitrary, and the line is only used to help identify which values are connected across the dimensions. The neural field graph used for obtaining these points is constructed from a connected random graph containing nine neurons that all receive an input of seven. Both plots contain $2^N = 2^9 = 512$ points represented thusly.



(a) Plot for the neural field graph with a global inhibition factor of $\gamma = 0$.

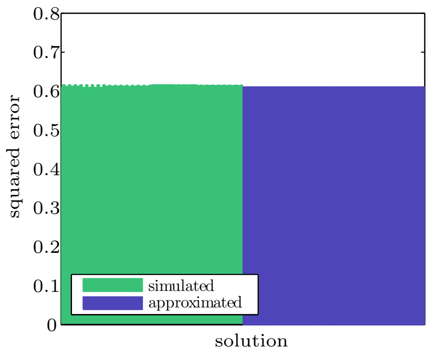


(b) Plot for a similar neural field graph with the global inhibition factor increased to $\gamma = 15$.

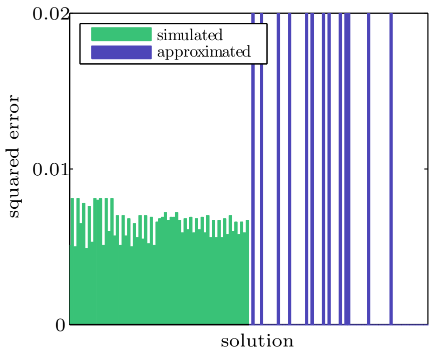
2. Evaluation & Comparison

2.1. Number of Fixed Points. When searching for fixed points with either of the methods described in Section 1, the resulting points are recorded. For the iterative determination, the resulting points comprise the points to which Newton's method converges for each point in the candidate solution set. For the determination via simulation, the resulting points are the activation values recorded from

Figure 6.2 Comparison of solution errors for simulated and approximated fixed points. The neural field graph is a 3×2 mesh graph with $\mu = 40$ and $\gamma = 15$. Three neurons receive an input of 5, the rest receives no input.



(a) Unbalanced neural field graph.



(b) Balanced neural field graph. The approximated points reach an error of up to approximately 31 where the bars exceed the scale, and 0 where no bars are present.

the neurons after convergence of the neural field graph is achieved. Figure 6.1 shows exemplary records of such points. It indicates that the resulting points differ slightly from each other. This could either suggest that the attractor for the neural field graph is not a single point, but rather a region, or may be a result of errors stemming from the numerical approximation. For further analysis, the latter is assumed due to the relatively small deviations of the result points (this is visible especially in Figure 6.1b), meaning that multiple points with only small differences are considered an indication of a single fixed point. In any case, compared to the size of the candidate solution set, the number of fixed points found seems low. Most experiments performed exhibited only one fixed point, while the maximum amount observed is three. Cases resulting in more than one fixed point usually involve a setup in which all neurons receive the same input, coupled with a relatively high global inhibition factor.

2.2. Solution Errors. During the examination of the fixed points of a neural field graph, the squared solution error (Equation 6.10) is recorded for the fixed points approximated with the iterative method introduced above. In addition, simulations are started with the same initial values that are used as the initial guesses for the iterative method. For the states of the converged neural field graph resulting from these simulations, the squared solution errors are calculated in the same manner. Results of two such experiments are shown in Figure 6.2. These results indicate that the iterative determination of solutions can sometimes outperform the simulation: in Figure 6.2a, the approximated solutions always have a slightly lower error than the simulated solutions. However, the difference is very small and may be due to noise from restrictions of the numerical precision. On the one hand, some of the approximated solutions shown in Figure 6.2b even reach an error measure of 0. On the other hand, Figure 6.2b also shows that in certain cases the approximation with Newton's method can fail, generating errors of approximately 31. The simulated solutions show a more steady error value that does not exceed 0.01. Similar observations can also be made in further experiments not detailed

Table 6.1 Measurements of the deviation of the candidate solution sets for different experimental setups. The input type column specifies the amount of input the neurons receive; "3 n.: 6, r. 0" means that three neurons receive an input of 6, while the remaining neurons receive no input. Values for $d(\boldsymbol{s}_i)$ are rounded to two decimal places. The neural field graphs presented here all use a Gaussian interaction kernel and $\delta_{max} = 3$. The last two rows correspond to the same setups presented in Figure 6.2.

Setup							$d(\boldsymbol{s}_i)$		
Graph Type	N	μ	γ	Balanced?	β	Input Type	Min.	Avg.	Max.
random	9	1	0	yes	4	all: 7	0	3.98	7.71
random	9	1	0	no	4	all: 7	0	2.8	5.47
random	10	1	15	yes	32	3 n.: 6, r. 0	3.53	229.59	462.68
random	10	1	15	no	32	3 n.: 6, r. 0	3.54	230.57	464.65
mesh (3×2)	6	10	15	yes	4	all: 40	10.29	27.27	47.4
mesh (3×2)	6	10	15	no	4	all: 40	11.92	33.5	70.57
mesh (3×2)	6	1	7	yes	4	all: 7	6.52	42.65	91.4
mesh (3×2)	6	1	7	no	4	all: 7	6.52	42.93	91.97
mesh (3×2)	6	40	15	yes	4	3 n.: 5, r. 0	1.51	36.02	59.28
mesh (3×2)	6	40	15	no	4	3 n.: 5, r. 0	1.56	43.31	82.34

here. In some runs, the iterative method failed to find any solutions with an acceptable error level. In these cases, the simulations usually converge to solutions with significantly lower error values.

The low performance of the iterative method may be due to quality of the initial guesses. It influences the convergence of Newton's method, and inadequate initial guesses may lead to non-converging behavior (Bronstein et al., 2008). Therefore, it is the focus of Section 2.3.

2.3. Deviation of the Candidate Solution Set. In the context of Newton's method for solving nonlinear equations that is used for the iterative method, the candidate solution set represents initial guesses. However, as stated before, the quality of these guesses greatly influences the quality of the solutions. Therefore, the following measure is proposed. Let $\boldsymbol{s}_i \in S$ be a candidate solution. Then the activation of the neurons in a neural field graph are initialized with the values in $\boldsymbol{s}_i$, and the graph is simulated. Once convergence is achieved, the activation values of all neurons, represented as a vector, are denoted by $\boldsymbol{t}_i$. With this, the *deviation of a candidate solution* is defined as

$$d(\boldsymbol{s}_i) = \|\boldsymbol{s}_i - \boldsymbol{t}_i\| \,. \tag{6.13}$$

While this deviation is not a measure of the quality of the solutions,[1] it does at least allow for some observations.

The deviations of the candidate solutions for different experimental setups presented in Table 6.1 suggest that the deviation of the solutions varies greatly. Therefore, the candidate solution set proposed may only be adequate for certain configurations of neural field graphs.

[1]Using the deviation as a quality measure would assume that both the iterative method and the simulation converge to the same state when initialized with the same candidate value. However, this cannot be true in general because the iterative method may also find solutions that represent an instable state of the neural field graph. The simulation, on the other hand, most likely only converges to such a state if it starts exactly in it.

CHAPTER 7

Further Results and Observations

1. Comparison of Peak Detection Methods

To investigate the differences between the distance-minimizing and activity maximum criteria for the selection of peak centers, two examples are shown in Figure 7.1. For the experiments, both graphs are constructed with balanced weights from a Gaussian interaction kernel and use a sigmoidal slope of $\beta = 4$.

The neural field graph shown in Figure 7.1a is based on a 15×15 mesh graph and constructed with an interaction scale of $\mu = 0.75$. It receives input in the range $[0, 10]$. The global inhibition is disabled by setting $\gamma = 0$. The second neural field graph, shown in Figure 7.1b, is based on a random connected graph with 150 nodes and 165 edges. Its interaction is scaled by $\mu = 5$, and a global inhibition of $\gamma = 1$ is present. The results in Figure 7.1a suggest that for this setup, there is one peak center for each peak in the neurons' input. However, only the neuron selected by the maximum activity criterion is located at the center of the input peak. Therefore, this criterion seems more fitting in this case, although the selection of the criterion likely depends on the actual application. The selected peak centers shown in Figure 7.1b both deviate from the actual center in the input space. However, this may be caused by inadequacies of the topology of the neural field graph in the region corresponding to the input peak.

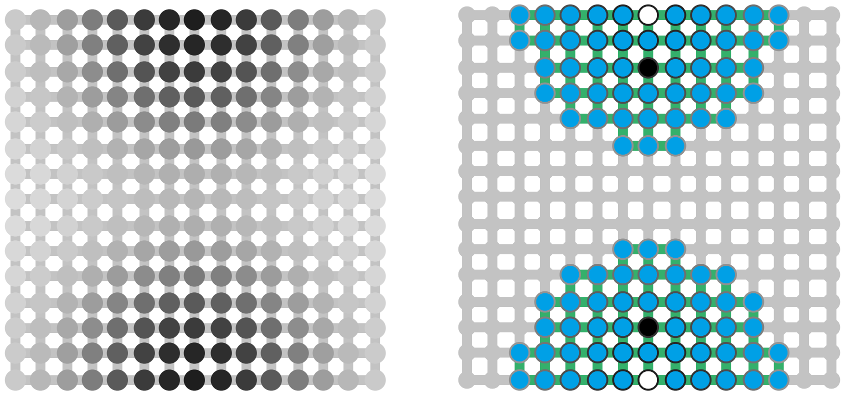
One of the results from Section 2 is replicated in Figure 7.2. Figure 7.2a highlights the two peaks present in the input. Due to the configuration of the neural field graph, it only contains one active subgraph after convergence (shown in Figure 7.2b). The activity maximum and distance-minimizing center peak criteria proposed in the present work only assign one peak center to each active subgraph. Therefore, the neural field graph model fails to select one neuron for each input peak in this case. This gives rise to the third peak criterion. The peaks selected by the local maximum criterion on the same graph are shown in Figure 7.2b. As desired, this criterion selects two peaks in the same active subgraph, and both centers are located near the center of the input peaks.

The experiments presented in Figure 7.1 are repeated using the local maximum peak criterion. In case of the mesh graph, the local maximum criterion selects the same peak centers as the activity maximum condition. However, for the random graph, the selection differs (see Figure 7.3). This result suggests that the local maximum criterion fails to select one neuron per input peak, but this may be due to the way the input is spread across the graph's topology in this case. Additionally, a different configuration of the neural field graph may lead to better results, as is suggested in Figure 7.3b.

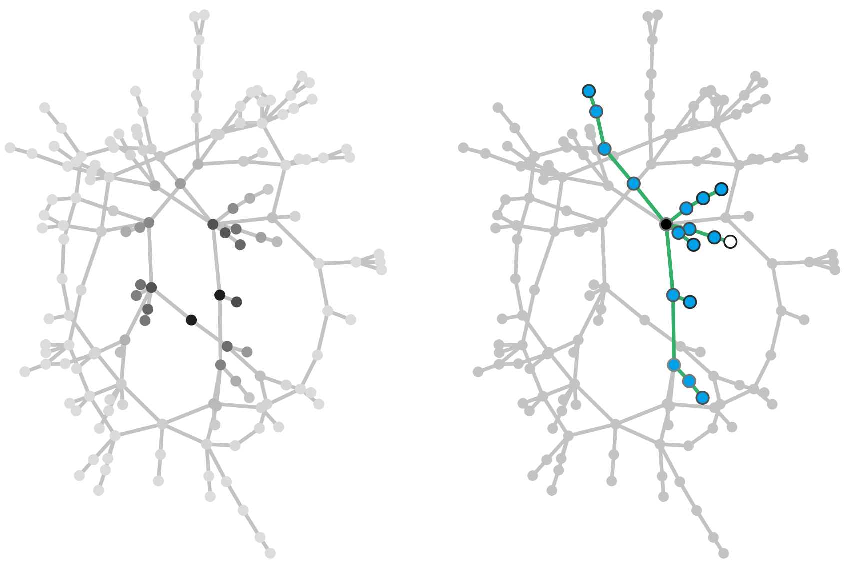
2. Weight Balancing

The goal in this section is to determine if the weight balancing method proposed in Section 5 fulfills its purpose. Figure 7.4 shows the state of the neurons of a random connected graph after convergence of the simulation. Due to the neural field graph's setup (i.e., equal input and relatively strong global inhibition), the neurons

Figure 7.1 Peak centers resulting from different criteria. The left column shows the input of the neural field graph and the underlying source graph. The right column shows the active subgraphs obtained after convergence, superimposed upon the graph structure. The neurons shown in white are the peak centers selected by the maximum-activity criterion, while the neurons shown in black are those minimizing the distance in the active subgraph.



(a) Peak criteria on a mesh graph with two input peaks.

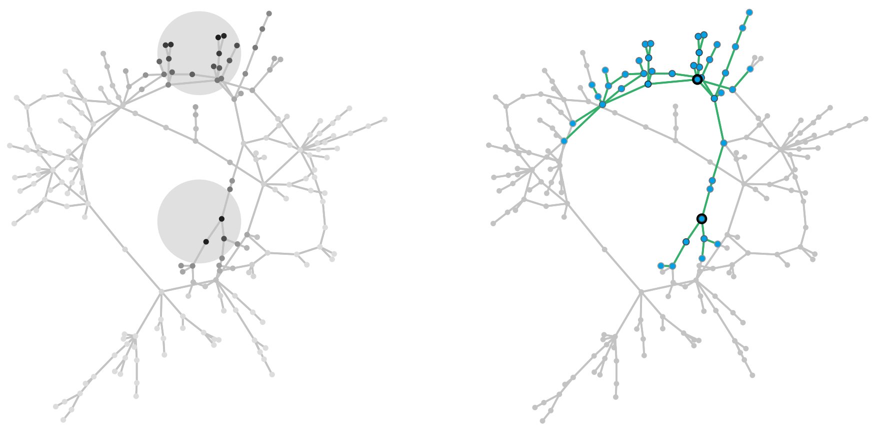


(b) Peak criteria on a random graph with one input peak.

are expected to converge to the same level of activation. However, as expected, the unbalanced graph shows different activation levels. As the plot shows, the balanced graph successfully converges to more balanced states. Some deviation remains that is likely a result of numerical errors stemming from the forward Euler method used for the simulation.

Figure 7.5 demonstrates a case in which the proposed weight balancing fails. The situation is the same as described in the successful case, except the neural field graph is created with a 3×1 mesh structure. Because the center neuron in the graph only has neighbors with a distance of one, while its neighbors both have connections to a neuron with a distance of two, the center neurons cannot be balanced properly. In practice, these cases seem to be unlikely, though, because with an increasing size

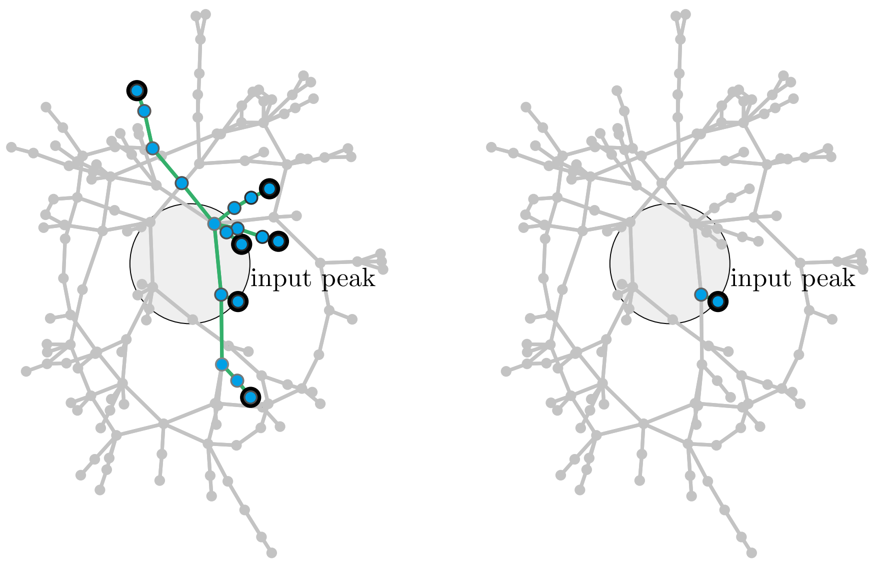
Figure 7.2 The situation described in Figure 5.3b is replicated here, but the local-maximum method for detecting peak centers is used instead, leading to a selection of centers that corresponds more to the actual peaks in the input.



(a) Two distinct peaks are present in the input. Their approximate position is highlighted by the light gray circles.

(b) Peaks selected by the local maximum criterion are highlighted here with bold, black circles around the peak centers. With the other methods, only one peak center would be detected.

Figure 7.3 The bold, black circles around the neurons mark the peak centers selected by the local maximum method. The neural field graph is the same as the one presented in Figure 7.1b.

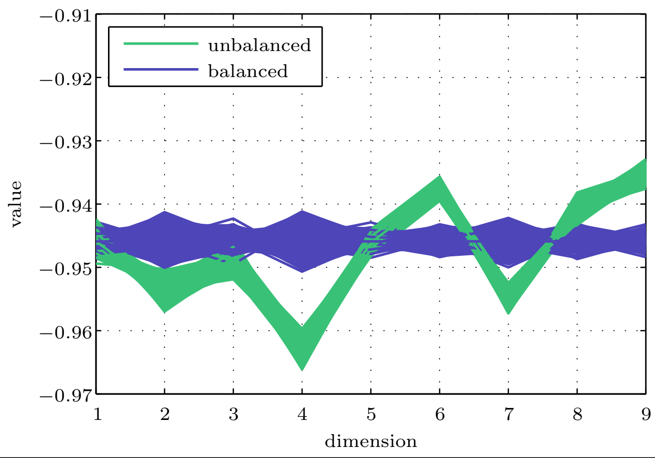


(a) Peak centers selected by the local maximum criterion on a neural field graph with the same configuration given in Figure 7.1b.

(b) Increasing the global inhibition factor to $\gamma = 3$ leads to a more accurate selection.

of the neural field graph, the likelihood of a neuron not having neighbors with a distance less than δ_{max} decreases.

Figure 7.4 Successful balancing of a neural field graph. The neural field graph is based on a connected random graph with nine neurons. All neurons receive the same amount of input, and the global inhibition factor is set to $\gamma = 15$. The plot shows the states of the neurons after convergence of a simulation starting from the candidate solution set S, each line represents one of the resulting $2^9 = 512$ states.



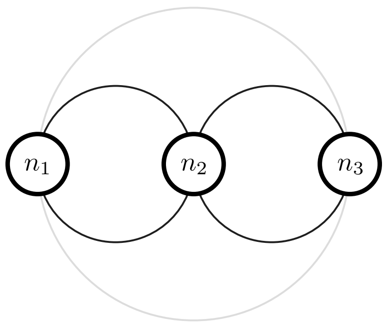
3. Sporadic Input

This section examines input in the form of isolated spikes. In this case, most neurons receive an input of zero, while only some neurons that are not directly connected to each other in the neural field graph receive a positive input. Assuming that at least some of the inputs are strong enough to cause above-threshold activation in the neurons receiving them, two situations can occur.

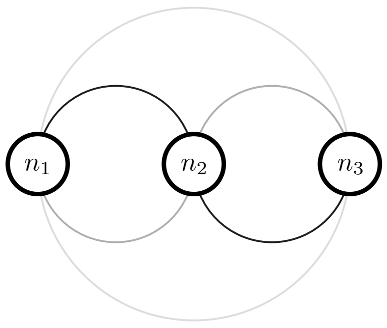
The first is that, for the most part, only neurons receiving the input spikes become active (i.e., the resulting active subgraphs contain only one neuron). This is the case if the weight of the synapse connecting an active neuron to an inactive neighbor is not strong enough to raise the neighbor's activation value above threshold. Notable exceptions from this are neurons that receive excitation from more than one active neighbor, but in general, sporadic input is expected to result in equally sporadic activity.

The second situation is the one in which the activity passed along from one neuron is sufficient to activate its neighbor. In the unbalanced case (and likely also in the balanced one), this means that the neighbor most likely also activates all of its neighbors and so on, leading to a fully activated neural field graph.

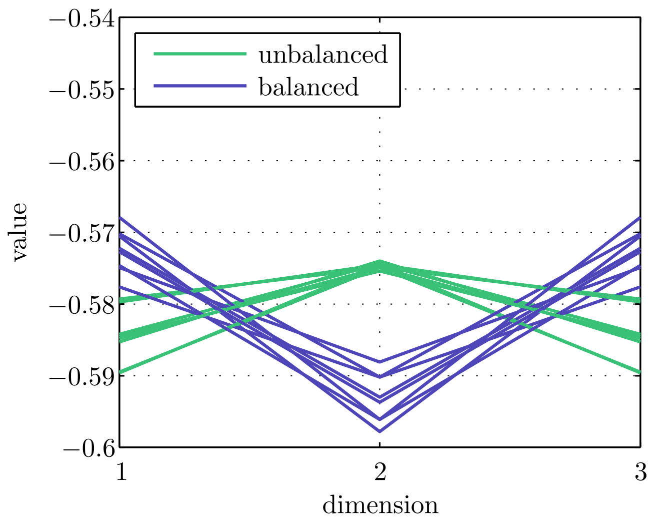
Figure 7.5 An example of a situation in which weight balancing fails. The neurons n_1 and n_3 both have incident synapses with two different weights: dark, w_1 and bright w_2 connections; n_2 only has connections of type w_1. Due to the lack of w_2 connections, n_2 is thus never balanced for that weight class by the balancing scheme proposed in the present work.



(a) Extended neural field graph.



(b) Balanced version of the extended neural field graph.



(c) Fixed points determined via simulation.

CHAPTER 8

Conclusion and Further Research

The present work presents an approach to modeling neural fields on undirected graphs. Conditions for its stability are analyzed, and experimental results indicate that it preserves characteristics of its continuous counterpart, the neural field model. However, properties not examined because they exceed the scope of the present work include the behavior of neural field graphs in situations equivalent to bifurcations in neural field graphs.

The method for balancing the synaptic weights proposed in Chapter 4 appears to successfully overcome the imbalance stemming from the sub-sampling of the input space for large neural field graphs. Although cases exist in which it fails, algorithms that may benefit from the neural field graph model likely use larger graphs, and therefore these cases may not occur in practice.

Another challenge is the peak detection method. Three alternative solutions are proposed in Section 6 of Chapter 4 that generally seem to yield acceptable results, but cases in which their selection of peak centers does not correspond to the actual peaks in the input are known for each one. This may be one of the major challenges in future work.

The last restriction is the static structure of the neural field graph model. It does not provide functions for modifying the neural field graph topology once the extended version is constructed. However, some applications may need such functionality during runtime. Currently, the only solution available would be reconstructing the extended neural field graph with the new structure, but this procedure has high computational requirements. A more efficient solution should be devised in a future work. Finding such a solution may be difficult for balanced neural field graphs because the insertion or removal of a neuron may mean that the whole neural field graph needs to be rebalanced.

The graph structure used for neural field graphs is currently restricted to undirected graphs. The model in its current form may also be extended to allow for directed graphs, but future research should determine the feasibility of this extension.

Additionally, the construction algorithm for the extended neural field graph assumes that an implicit cost function is associated with the graph structure. This function assigns costs of one to all synapses in the graph, except the reflexive connections which are assigned costs of zero. However, learning rules like Hebbian learning (Hebb, 1949) use different weights between neurons, and using these weights in the construction of the extended neural field graph is desirable. While the capacity for this should already be included in the neural field graph model, some of the assumptions made for the algorithms may not hold in this context. Specifically, the sampling of the weights at relevant distances (see Algorithm 4.1) and the calculations made in the *preBalance* function (Algorithm 4.2) may not be possible because the weight classes that occur in the neural field graph are not known beforehand. Furthermore, the weight balancing method may not be applicable in this case because too many different weights may be present in such graphs, leading to the same improper balancing discussed in Section 2 of Chapter 7. Thus, further research in this topic is needed.

The experiments in Part 4 indicate that the parameter selection of the neural field graph greatly influences its behavior. When aiming for a certain behavior, the corresponding parameters are currently determined empirically, but a more general approach is preferable. This may not be limited to the neural field graph model: to the knowledge of the author, this problem is generally unsolved in neural fields as well; if a solution is found for one of the models, it may be applicable to the other one as well.

Most of these challenges are currently only based on the theoretical treatment of the model given in the present work. Whether they are actual limitations can probably only be determined based on an application of the neural field graph model. Most suitable applications are those that use a graph structure for real-time processing of high-dimensional information. Another possibile application is given by artificial neural networks, realized with the dynamical model of neural field graphs. In contrast to their classical realizations, they may have the added benefit of being able to cope with feed forward architectures as well as recurrent setups.

Appendices

APPENDIX A

Miscellaneous Proofs, Theorems and Algorithms

1. Component Multiplication Operator

The component-wise multiplication $\boldsymbol{x} \circ \boldsymbol{y}$ of two vectors $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^n$ is given by

$$(\boldsymbol{x} \circ \boldsymbol{y})_i = x_i \cdot y_i \,. \tag{1}$$

For the componentwise multiplication of a matrix $\boldsymbol{A} \in \mathbb{R}^{n\times m}$ and a vector $\boldsymbol{v} \in \mathbb{R}^n$, $\boldsymbol{R} \in \mathbb{R}^{n\times m}$ is defined as

$$\boldsymbol{R} = \boldsymbol{v} \circ \boldsymbol{A} \tag{2}$$

with

$$(R)_{i,j} = v_i \cdot (A)_{i,j} \ . \tag{3}$$

1.1. Associativity. With the definitions given above, proof is given that

$$(\boldsymbol{x} \circ \boldsymbol{A}) \cdot \boldsymbol{y} = \boldsymbol{x} \circ (\boldsymbol{A} \cdot \boldsymbol{y}) \ . \tag{4}$$

The components of the left-hand side of Equation 4 expand to

$$\begin{aligned} ((\boldsymbol{x} \circ \boldsymbol{A}) \cdot \boldsymbol{y})_i &= \sum_{j=1}^{n} (\boldsymbol{x} \circ \boldsymbol{A})_{i,j} \cdot y_j \\ &= \sum_{j=1}^{n} x_i \cdot (A)_{i,j} \cdot y_j \,. \end{aligned} \tag{5}$$

The right hand side yields

$$(\boldsymbol{x} \circ (\boldsymbol{A} \cdot \boldsymbol{y}))_i = x_i \cdot \sum_{j=1}^{n} (A)_{i,j} \cdot y_j \quad \square \tag{6}$$

2. Algorithm for Random Connected Graphs

The following algorithm is used to create an undirected connected graph $G = (V, E)$ with a random structure and a predetermined number of nodes n and edges m.

```
$V \leftarrow \{v_1\}$
$E \leftarrow \emptyset$
$U \leftarrow \{v_2, \ldots, v_n\}$
while $U \neq \emptyset$ do
    randomly select $u \in U$, $v \in V$
    $E \leftarrow E \cup \{u, v\}$
    $U \leftarrow U \setminus \{u\}$
    $V \leftarrow V \cup \{u\}$
while $|E| < m$ do
    randomly select $u, v \in V$
    $E \leftarrow E \cup \{u, v\}$
```

After the algorithm finishes, the graph's nodes are positioned on a two-dimensional plane using an implementation of the force-directed placement algorithm presented in Fruchterman et al. (1991).

APPENDIX B

Bibliography

Shun-Ichi Amari. Dynamics of pattern formation in lateral-inhibition type neural fields. *Biological Cybernetics*, 27:77–87, 1977.

Uri M. Ascher and Linda R. Petzold. *Computer Methods for Ordinary Differential Equations and Differential-Algebraic Equations*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 1998. ISBN 0898714125.

J. Auslander, N.P. Bhatia, and P Seibert. Attractors in dynamical systems. *Boletin Sociedad Matemática Mexicana*, 9:55–66, 1964.

Estela Bicho, Pierre Mallet, and Gregor Schöner. Target representation on an autonomous vehicle with low level sensors. *International Journal of Robotics Research*, 19:424–447, 2000.

P. Blanchard, R.L. Devaney, and G.R. Hall. *Differential Equations*. Thompson, 2006.

J.A. Bondy and U. S. R. Murty. *Graph Theory with Applications*. North Holland, 1976. URL http://www.ecp6.jussieu.fr/pageperso/bondy/books/gtwa/gtwa.html.

I.M. Bronstein, K.A. Semendjajew, G. Musiol, and H. Mühling. *Taschenbuch der Mathematik*. Frankfurt, M. : Deutsch, 7 edition, 2008.

Gary Chartrand. *Introductory graph theory*. Dover Publications, 1985.

Stephen Coombes. Neural fields. *Scholarpedia*, 1(6):1373, 2006.

Stephen Coombes and Helmut Schmidt. Neural fields with sigmoidal firing rates: approximate solutions. Department of Mathematical Sciences, University of Nottingham, Nottingham, February 2010. URL http://eprints.nottingham.ac.uk/1233/1/DCDSsigmoidsPreprint.pdf.

Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. Introduction to algorithms, second edition, 2001.

Reinhard Diestel. *Graphentheorie*. Springer-Verlag, Heidelberg, 3rd edition, September 2006. URL http://www.math.uni-hamburg.de/home/diestel/books/graphentheorie/.

Wolfram Erlhagen and Estela Bicho. Dynamic field theory (dft): Applications in cognitive science and robotics. In *euCognition: The European Network for Advancement of Artificial Cognitive Systems*. Network Action 057-1, 2009.

Wolfram Erlhagen and Dirk Jancke. The role of action plans and other cognitive factors in motion exploration: A modeling study. *Visual Cognition*, 11:315–340, 2004. URL http://www.uiowa.edu/~icdls/dft-school/documents/DFT_Publications/Erlhagen%26jancke_viscog04.pdf.

Wolfram Erlhagen and Gregor Schöner. Dynamic field theory of movement preparation. *Psychological Review*, 109:545–572, 2002.

M. Fineman. *The Nature of Visual Illusion*. Dover Publications, 1996.

Robert W. Floyd. Algorithm 97: Shortest path. *Communications of the ACM*, 5 (6):345, 1962. ISSN 0001-0782. doi: http://doi.acm.org/10.1145/367766.368168.

Bernd Fritzke. A growing neural gas network learns topologies. In G. Tesauro, D. S. Touretzky, and T. K. Leen, editors, *Advances in Neural Information Processing Systems 7*, pages 625–632. MIT Press, Cambridge MA, 1995.

Thomas M. J. Fruchterman, Edward, and Edward M. Reingold. Graph drawing by force-directed placement, 1991.

L. A. Gilroy, H. S. Hock, and A. Ploeger. Differential activation solution to the motion correspondence problem. *Perception & psychophysics*, 63(5):847–861, July 2001. ISSN 0031-5117. URL http://view.ncbi.nlm.nih.gov/pubmed/11521851.

J. S. Griffith. A field theory of neural nets: I: Derivation of field equations. *Bulletin of Mathematical Biophysics*, 25:111–120, 1963.

J. S. Griffith. A field theory of neural nets: II: Properties of field equations. *Bulletin of Mathematical Biophysics*, 27:187–195, 1965.

J. Guckenheimer and P Holmes. *Nonlinear Oscillations, Dynamical Systems, and Bifurcations of Vector Fields.* Springer-Verlag, New York, 3rd edition, 1997.

Donald O. Hebb. *The Organization of Behavior: A Neuropsychological Theory.* Wiley, New York, new edition edition, June 1949. ISBN 0805843000. URL http://www.amazon.com/exec/obidos/redirect?tag=citeulike07-20&path=ASIN/0805843000.

P. B. Hibbard, M. F. Bradshaw, and R A Eagle. Cue combination in the motion correspondence problem. *Proceedings of the Royal Society B*, 267:1369–1374, 2000.

Howard S. Hock, J. A. S. Kelso, and Gregor Schöner. Bistability and hysteresis in the organization of apparent motion patterns. *Journal of experimental psychology. Human perception and performance*, 1(19):63–80, 1993.

Howard S. Hock, Martin Giese, and Gregor Schöner. The dynamical foundations of motion pattern formation: Stability, selective adaptation, and perceptual continuity. *Perception & Psychophysics*, 65:429–457, 2003.

Dirk Jancke, Wolfram Erlhagen, Hubert R. Dinse, Amir C. Akhavan, Martin Giese, Axel Steinhage, and Gregor Schöner. Parametric population representation of retinal location: Neuronal interaction dynamics in cat primary visual cortex. *The Journal of Neuroscience*, 19(20):9016–9028, 1999.

Gottfried Jetschke. *Mathematik der Selbstorganisation.* VEB Deutscher Verlag der Wissenschaften, Berlin, 1989.

Tyler McMillen. Taylor series, Spring 2008. URL http://math.fullerton.edu/tmcmillen/M310_S08/. Lecture Notes.

J. W. Milnor. Attractor. *Scholarpedia*, 1(11):1815, 2006.

Richard M. Murray, S. Shankar Sastry, and Li Zexiang. *A Mathematical Introduction to Robotic Manipulation.* CRC Press, Inc., Boca Raton, FL, USA, 1994. ISBN 0849379814.

Alan V. Oppenheim and R.W. Schafer. *Digital signal processing.* Prentice-Hall international editions. Prentice-Hall, 1975. ISBN 9780132146357. URL http://books.google.com/books?id=vfdSAAAAMAAJ.

E. Ott. Basin of attraction. *Scholarpedia*, 1(8):1701, 2006.

Frank Rosenblatt. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, 65(6):386–408, 1958.

Sebastian Schneegans and Gregor Schöner. *Handbook of Cognitive Science: An Embodied Approach*, chapter 13 – Dynamic Field Theory as a Framework for Understanding Embodied Cognition, pages 241–271. Perspectives on Cognitive Science. Elsevier Science & Technology Books, September 2008.

Gregor Schöner. *Cambridge Handbook of Computational Cognitive Modelling*, chapter Dynamical Systems Approaches to Cognition. Cambridge University Press, 2007.

Raimund Seidel. On the all-pairs-shortest-path problem. In *In ACM STOC*, pages 745–749, 1992.

Vanessa R. Simmering and John P. Spencer. Generality with specificity: the dynamic field theory generalizes across tasks and time scales. *Developmental Science*, 11(4):541–555, 2008.

John P. Spencer and Gregor Schöner. Bridging the representational gap in the dynamic systems approach to development. *Developmental Science*, 6:392–412, 2003.

John P. Spencer, Sammy Perone, and Jeffrey S. Johnson. *Toward a Unified Theory of Development: Connectionism and Dynamic Systems Theory Re-Considered*, chapter The dynamic field theory and embodied cognitive dynamics. New York: Oxford University Press, in press, 2010. URL http://www.uiowa.edu/~icdls/dft-school/documents/DFT_Publications/Spenceretal_CXDSChapter_Final.pdf.

S. Ullman. *The interpretation of visual motion / Shimon Ullman*. MIT Press, Cambridge, Mass. :, 1979. ISBN 026221007.

P. von Schiller. Stroboskopische alternativbewegungen. *Psychologische Forschung*, 17:179–214, 1933.

Eric W. Weisstein. Taylor series. From MathWorld – A Wolfram Web Resource, 2010a. URL http://mathworld.wolfram.com/TaylorSeries.html.

Eric W. Weisstein. Fixed point. From MathWorld – A Wolfram Web Resource, 2010b. URL http://mathworld.wolfram.com/FixedPoint.html.

Laurenz Wiskott. Slow feature analysis: A theoretical analysis of optimal free responses. *Neural Computation*, 15(9):2147–2177, 2003. doi: 10.1162/089976603322297331. URL http://www.mitpressjournals.org/doi/abs/10.1162/089976603322297331.

Laurenz Wiskott and Terrence Sejnowski. Slow feature analysis: Unsupervised learning of invariances. *Neural Computation*, 14(4):715–770, 2002.

Laurenz Wiskott, Jean-Marc Fellous, Norbert Krüger, and Christoph von der Malsburg. Face recognition by elastic bunch graph matching. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 19(7):775–779, 1997. URL http://www.cnl.salk.edu/~wiskott/Abstracts/WisFelKrue97a.html.

Laurenz Wiskott, Jean-Marc Fellous, Norbert Krüger, and Christoph von der Malsburg. Face recognition by elastic bunch graph matching. In L. C. Jain, U. Halici, I. Hayashi, and S. B. Lee, editors, *Intelligent Biometric Techniques in Fingerprint and Face Recognition*, chapter 11, pages 355–396. CRC Press, 1999. ISBN 0-8493-2055-0.

Erklärung

Ich erkläre, dass das Thema dieser Arbeit nicht identisch ist mit dem Thema einer von mir bereits für ein anderes Examen eingereichten Arbeit.

Ich erkläre weiterhin, dass ich die Arbeit nicht bereits an einer anderen Hochschule zur Erlangung eines akademischen Grades eingereicht habe.

Ich versichere, dass ich die Arbeit selbständig verfasst und keine anderen als die angegebenen Quellen benutzt habe. Die Stellen der Arbeit, die anderen Werken dem Wortlaut oder dem Sinn nach entnommen sind, habe ich unter Angabe der Quellen der Entlehnung kenntlich gemacht. Dies gilt sinngemäß auch für gelieferte Zeichnungen, Skizzen und bildliche Darstellungen und dergleichen.

Datum　　　　　　Unterschrift