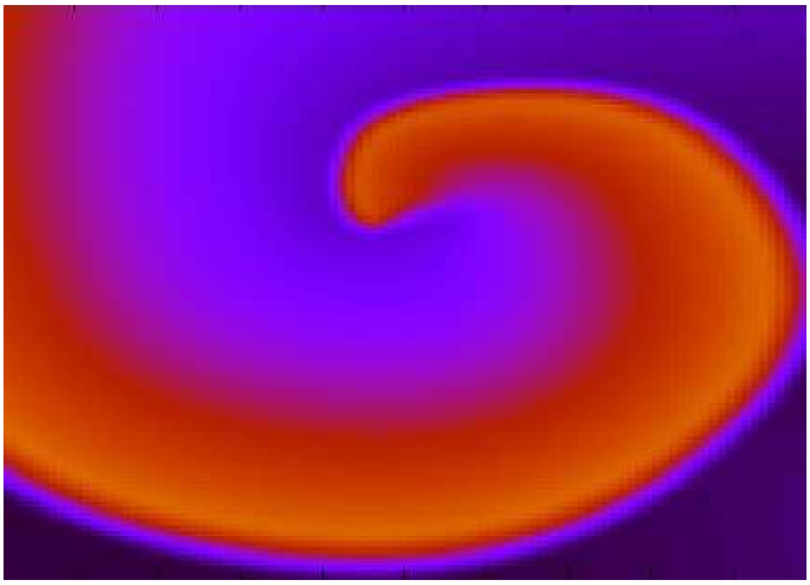
PantaRhei

A Simulation Framework

for Models of Biological Pattern Formation

École Polytechnique Fédéral de Lausanne (EPFL)
Biologically Inspired Robotics Group (BIRG)



Master Thesis
2005

Author: Martin Rumo
Supervisor: Jonas Buchli
Professors: Auke Ijspeert, Beat Hirsbrunner

"Panta Rhei"

"Everything flows"

Heraklit

Abstract

Dynamical systems have become an increasingly interesting tool to engineer biologically inspired control systems. The properties that are especially interesting to study, are the pattern generating capabilities of dynamical systems. There are two continuous space models, that have been studied extensively for their ability to generate static and dynamic pattern. Namely reaction diffusion systems and the neural field model. Both classes have been used so far, for different tasks in robotics such as perceptive tasks as well as embodied cognition and motor control. Both systems can be described in the unifying language of dynamical systems, and can be understood as models of activity pattern in continuous neural tissue. To our knowledge no attempt has been made so far to combine those important classes to a generalized class of dynamical systems for pattern formation. In this master thesis such a general class is introduced. Furthermore `PantaRhei` has been developed for the simulation and visualization of dynamical systems that are governed by this general class of integro-reaction-diffusion equations. It provides a simple scripting language to generate, run and plot dynamical systems of the class of integro-reaction-diffusion systems.

Contents

List of Figures

"11.15 Restate my assumptions.

1. *Mathematics is the language of nature.*
2. *Everything around us can be represented and understood though numbers.*
3. *If you graph these numbers, patterns emerge.*

Therefore: there are pattern everywhere in nature."

- Max Cohen in the film π

1 Introduction

When we see something, that repeats itself or a similar copy of itself, we conclude, that there is a template, a form or a set of rules that by its repetition generates a pattern. Evolutionary processes can, given a set of rules, exhibit pattern. A pattern is a very universal and abstract concept, that plays a vital role in many fields, such as psychology, linguistics and computer science. In the field of Artificial Intelligence (AI) there is a increased interest in pattern recognition as well as pattern formation. An "intelligent" agent must recognize patterns in order to create a simplified inner representation of its environment, on the other hand it must, according to the state of the environment and its inner state, generate an adapted behavioral pattern [Schöner, 2000] [Braitenberg, 1984].

We can find pattern in a variety of spatial and temporal scales. Those pattern are in a dynamical interplay and bring each other into being. Dynamical systems can be described and understood on different levels of abstraction. For example does the dynamical interplay of ionic channels and membrane potentials in nerve cells produce spiking patterns known as action potentials. On a more abstract level, when we're not observing the membrane potential, but rather an averaged spiking frequency, we can observe other patterns on a larger scale. Let's suppose we observe the averaged spiking frequency of motor neurons. In this case we can surely see some effects on the muscular level and therefore see limbs moving in a certain way, that we would also obviously also call a pattern. On each level the behavioral pattern can be described using a unified language, the language of dynamical systems. Pattern on different levels of abstraction can be described and modeled mathematically without knowledge of the details of an underlying dynamics that actually exhibits the pattern. Cognitive scientist, for example, have described a mathematical model that explains reaching pattern in infants, without referring in details to the neural system generating the pattern [Esther Thelen, 2000].

Pattern as we can see them animal fur or shells, seem static, but they were generated by a dynamic process. Other pattern that we observe, are inherently dynamic. The ocular

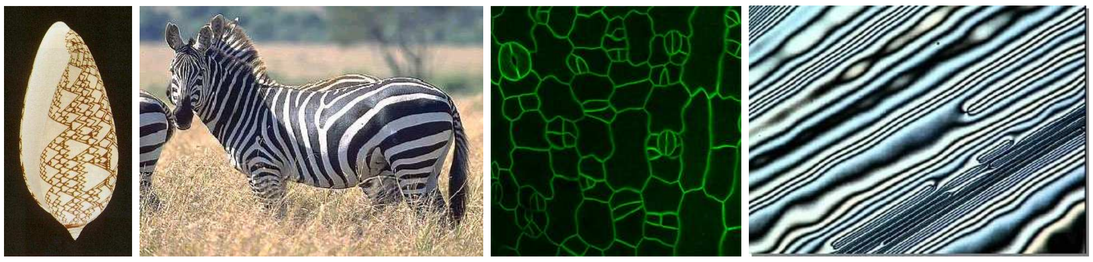


Figure 1: Pattern in different environments

dominance stripes for example reflect a certain pattern in the connectivity of neurons in the visual centers of the brain. Hallucination on the other hand have been described as activity pattern in the visual cortex. While the pattern in connectivity of the cortex, is more or less static, the processes on a smaller scale, that generate the pattern, reflect a dynamical system, which is mainly driven by the input to the system. So we have in addition to static pattern, dynamic pattern that express themselves in form of movement such as index oscillations.

1.1 Why Study Biological Pattern Formation?

Pattern emerge on a macroscopic level, because elements interact with each other on a microscopic level. This principle can be observed in ecology, physiology, neurobiology and many other disciplines. The rules which describe the interaction between microscopic elements define a dynamical system, able to produce certain pattern. Such dynamical systems, capable of producing spatial pattern are described in the framework of biological mathematics, mainly using partial differential equations.

There is an ongoing debate in the field of cognitive science, if dynamical systems can even be viewed as an alternative to the computational paradigm. [Gelder, 1997] [**?**] Given a set of attractors and repulsors in a dynamical systems, the temporal change of a system can be described in a geometrical way, in terms of trajectories and bifurcations. The change of variables in dynamical systems are continuous, that is a feature that is difficult to describe in a computation paradigm. Dynamical systems have other interesting advantages such as stability, flexibility, adaptivity and self-organization.

Another field in which dynamical system become increasingly important is the field of robotics. The reason why dynamical systems seem so attractive for robotics, is that a behavioral pattern can be continuously meshed with lower level movement pattern used in locomotion, because both can be described in the same language. Furthermore mechanisms of short term memory and long term memory can also be described in the same way.

1.2 Basic Goals of this Master Thesis

In the first part of this work, there will be given a broad view of what makes a bio-inspired approach so interesting in information processing or other control systems. Concepts named in the introduction will be clarified and explicitly defined. Especially the framework of dynamical systems will be formally defined.

As the `PantaRhei` interpreter is designed to simulate dynamical systems, that can be described in the framework of neural field and reaction-diffusion systems, those two models will be further introduced and their importance in the field of biological pattern formation will be shown.

In section 3 several contributions to the analysis and application of either reaction diffusion systems as well as neural fields in the broader context of AI and robotics will be listed.

In Section 4 the architecture of the resulting software is described in terms of its interface, its data structures and algorithms. A Manual describing how `PantaRhei` can be used will be found in Appendix B.
The results of example systems calculated with the interpreter are given and resulting pattern are discussed in Section 5.
In section 6 and 7 some concluding remarks are given as well as a brief note is made on how the software can be used and how it could be extended.
One can find two appendices at the end of the document. The first giving a complete grammar of the `PantaRhei` language and a reference manual.

2 Background

This section gives a general overview over the topic. In a first part, the properties of biological systems are discussed as well as its potency to inspire engineers in the development of performant and reliable systems. The main mathematical frameworks used to describe and analyze such systems is the framework of dynamical systems. The framework will be introduced and formally defined. A special kind of dynamical systems act upon a continuous space rather then of discrete variables. It will be shown, that given the vast dimensionality of neural systems, continuous field approaches are justified, in highly abstract on activity levels in neural tissue. The two main classes of biologically inspired models of pattern formation in continuous space are neural fields and reaction diffusion systems. Both will be introduced and defined. A look on the conditions in which dynamic pattern can occur in such systems is given in the end of this section.

2.1 Bio-Inspired Technology

The Center for Biomimetic and Natural Technologies, at University of Bath has created a database in which they gather mechanical tricks, that nature has developed in the course of time [uni, 2005]. Searching for "locomotion" in the data base yields several different ways that creatures locomote, each probably very robust, adaptive and energy-efficient in all different types of gaits. The performance of such systems of locomotion is due to an evolutionary process, that eliminates the solutions that could not adapt to changes either in the environment or in the physical situation of the agent. Neural mechanisms of information processing, are managed by vast amounts of small, simple and possibly unreliable that can self-organize and therefore are very stable and flexible. Therefore it is helpful to study biological systems that can adapt and self-organize.

2.1.1 Stability

In some technical systems, it is favorable to be able to stay in a steady state, when confronted with ever changing inputs from an perturbed environment. The pilot in a aircraft must try to keep is aircraft stable despite turbulences. The dynamic system that represents the myriads of neurons in the pilots brain are able to constantly adapt to outside forces in order to keep a dynamic equilibrium. In robotics there is a prominent search for such adaptive systems. Stability means also that even in presence of noise the behavioral variable must be kept stable. Extended systems that include filtering ability or mechanisms to diffuse incoming signals are of great use in such tasks. In the terminology of dynamical systems, such stable states are called attractors. It therefore is interesting, to find systems, that have at least one attractor, but it is important to note that multiple attractors can coexist.

2.1.2 Flexibility

Dynamical systems address the fundamental conflict, between stability and flexibility. Certain instabilities that are due to inputs to the system or an inherent random factor. A pattern is flexible, when it is able to change its state, by temporally losing its stability, but this property is only interesting, when a new fixpoint or cycle can be found, in which the system can maintain stable. That presupposes the existence of multiple attractors in the system. Such temporal instabilities that can occur in a non-linear system, where change of attractors becomes possible are very interesting because they differentiate elementary behavioral modules. Gait transition in locomotion is a good example where changes in the parameter of the system generates new oscillatory patterns.

2.1.3 Self-Organization

The very term "self-organization" seems to be introduced by the cybernetician W. Ross Ashby. He argued that a dynamical system always tends to evolve towards a equilibrium state. This process reduces the uncertainty about the state of the system, and therefor its statistical entropy. In this process however pattern can emerge, as pattern can be viewed as ordered structures. Self-organization from that point of view is the process of produce a globally coherent "order" out of mutual interactions between elements, that have the tendency to be unordered.

2.2 Dynamical Systems

It follows a more formal definition of the most important concepts of dynamical system theory.

2.3 Definition

A general dynamical system is a triple $D = (X, T, \phi)$. X is the state space T the temporal domain and ϕ is a state transition function for which must hold:

$$\phi(x, 0) = x$$

$$\text{and} \tag{1}$$

$$\phi(\phi(x, t)s) = \phi(x, s + t)$$

for all $x \in X$ and $s, t \in T$. Now if $T = \mathbb{N}$, the temporal domain equals the set of natural numbers, the dynamical system is a *discrete time* dynamical system. When $T = \mathbb{R}^+$, the temporal domain is the set of non-negative real numbers, the dynamical system is called a *continuous time* dynamical system. Often dynamical systems are defined by differential equations. A differential equation

$$dx(t)/dt = F(x(t))$$

$$\text{with} \tag{2}$$

$$x(0) = x_0$$

defines a dynamical system $D = (X, R^+, \phi)$ when ϕ is the solution of (2)

$$\phi(x, t) = \int_0^t F(x(s))ds$$

$$\text{with} \tag{3}$$

$$\phi(x, 0) = x$$

In the case of a discrete time system, a map $g : X \mapsto X$ a discrete time system $D = (X, N, \phi)$ where ϕ represents a iteration of g

$$\phi(x, t) = g^t(x) \tag{4}$$

2.4 Dynamics

To be able to talk about the behavior of a dynamical system one has to introduce some properties. By fixing either the instance in time or the initial state of the system one can more formally describe the behavior of a system.

trajectory We can define a map $\phi_x : T \mapsto X$ by $\phi_x(t) = \phi(x, t)$ for each possible state of x. this map defines all of the path that a particular x would take given the transition function ϕ. It defines how the system evolve given the initial state x.

flow The flow of a dynamic system is a map $\phi_t : X \mapsto X$ given by $\phi_t(x) = \phi(x, t)$. For each moment in T this function yields the state in which the system would be given the initial state x.

orbit For each initial state x one can compute all the states that the system goes through for all instances $t \in T$. This set is called the orbit of x. It is formally defined by:

$$\Gamma(x) = \{\phi_x(t) | t \in T\} \tag{5}$$

2.4.1 Attractors

We'll call a *limit point* p of a trajectory, a point that the trajectory reaches, when the sequence of time intervals reaches infinity: $p = \lim_{t \to \infty} \phi_x(t)$ as a trajectory can have several limit points, we'll call the set $\omega(x)$ the *limit set* of x. A limit set is called *invariant* if each point in $\omega(x)$ stays in $\omega(x)$ as time advances. Now we can define an attractor as being a closed invariant subset of the state space X.

stability If there is neighborhood U of the attractor A, so that for each point in U its limit set is in A, then the attractor is called *stable*. Stable attractor are interesting because even though the state of the dynamical system is perturbed by "outside forces", it has the tendency to move back into an equilibrium state, which is invariant without these "outside forces". If on the other hand the points in the neighborhood U have limit sets with empty intersections with the attractor A, then the attractor is called *unstable*. There are situation, when an attractor is neither stable nor unstable. In this situation the attractor A is a saddle. There are several different categories of attractors, that will be described below.

fixed point A fixed point is a trajectory which does not change. That is, for all $t \in T$ must hold: $\phi_x(t) = x$. We also say the the trajectory is stationary, or the dynamical system is in its equilibrium or steady state. The condition, for x to be a fixed point is $F(x) = 0$ for a continuous time system and $g(x) = x$ in the case of a discrete time system.

limit cycle A trajectory can always return to the same point in $\mathbb{R}^n$, more formally: $\phi_x(p) = x$ with $0 < p < \infty$. Then p is called the period of the trajectory of x. A limit cycle, then is a periodic trajectory or it can be the limit set of another trajectory. limit cycles are especially interesting, when the control system that an engineer wants to develop requires oscillatory pattern.

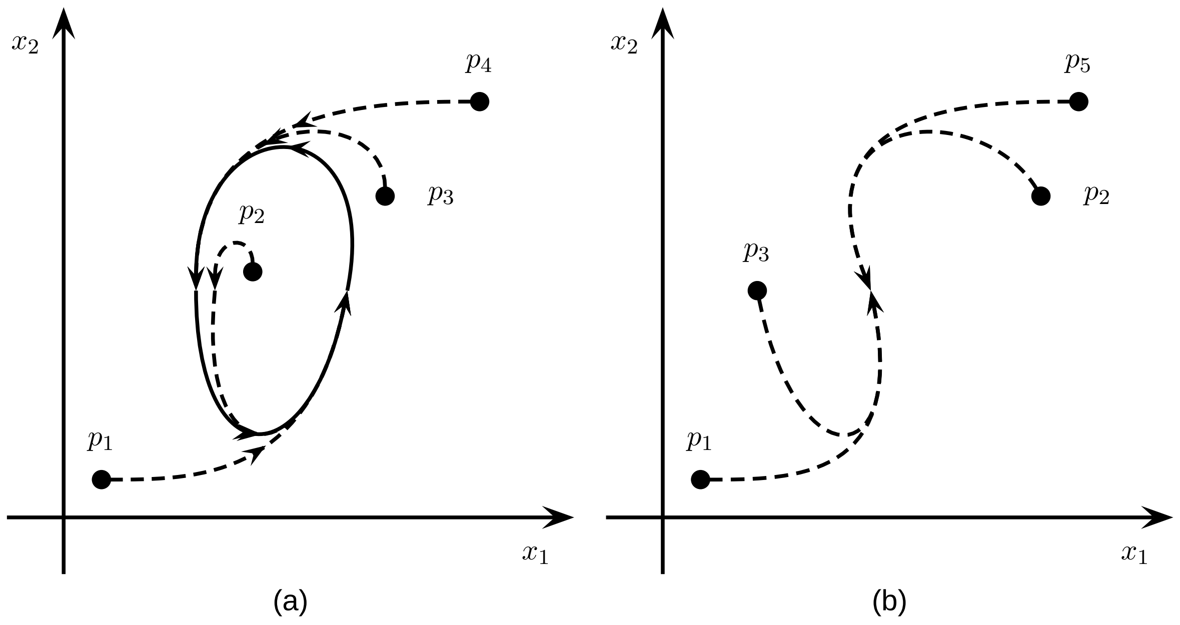


Figure 2: (a) the trajectory of several different initial states ($p_1 - p_4$) end up in a stable limit cycle. Whereas in (b) we have the trajectories of points p_1 to p_4 all ending up in a fixed point.

strange or chaotic attractors In several non-linear systems the distance of trajectory ϕ_x and ϕ_y of two close initial states x and y respectively, grow exponentially in time t. We then say that the system is sensitive to initial condition. These kind of system are hard to predict and problably therefore called chaotic or strange attractor.

2.5 Bifurcation

A bifurcation is a change in the dynamic structure (number, positition and quality of attractors). When a dynamical system depends on a set of parameters, then bifurcation can be studied. To formally define bifurcations one must define the *topological conjugancy*: Two dynamical Systems $D = (X, T, \phi)$ and $D' = (Y, T, \psi)$ are said to be topologically conjugate if there is a continuous map $h : X \mapsto Y$ with a one-to-one correspondence between the trajectories of D and D'. If in the parameter space there are to parameters μ_1 and m_1 in the neighborhood of m_0, for which the corresponding systems D_{μ_1} and D_{μ_2} are not topologically conjugate, then μ_0 is a bifurcation point. At a bifurcation point the number, position and/or the nature of attractors change.

2.6 Biological Justification for Continuous Space Models of neural systems

In this section the biological plausibility of the models proposed for biological pattern formation will be explained. Biological information systems such as a nervous systems, consists of huge amount of simple processing units called neurons. Neural network theory has provided us with tools to analyze the behavior of relatively small neural networks mainly to investigate their capability to learn. However, it is important to understand that in natural brain there is a high redundancy, what increases its reliability. [1] Some elements can disintegrate and their task must be compensated by other elements. This description is very much in line with the idea of building models based on functional units that represent cell

[1]human brain:7×10^{10} to 8×10^{10} with each having thousands of connections with other

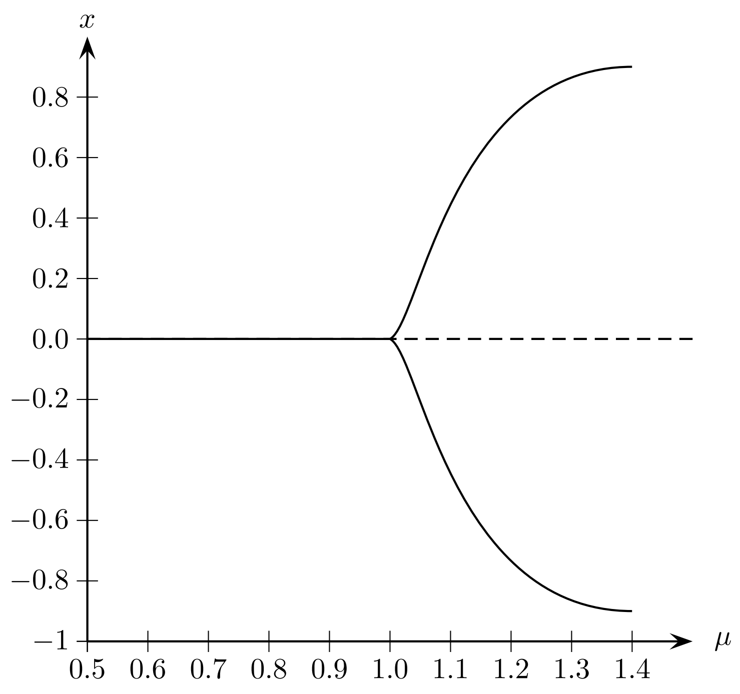


Figure 3: bifurcation point at $\mu = 1$

assemblies consisting of large numbers of individual neurons [Hans A. Mallot, 1996] . This idea, that significant concepts were actually coded not by neurons, but by groups of neurons, goes even back to Donald Hebb. Even though there are no precise ideas about the boundaries of such functions units, Edelmann speculates on the size of these populations, when he states that several thousands of neuron were necessary to encode significant stimulus categories in animals [Edelmann, 1987]. Averaging across many neurons allows the use of deterministic equations even if the behavior of individual neurons include a random component.

2.6.1 Randomness and Specificity in Neural Nets

In their work, Almud Schütz and Valentino Braitenberg [Valentino Braitenberg, 1990] propose three different categories of connectivity in neural systems, according to their degree of randomness.
In figure 4 we see the three categories depicted.

- (a) Each neuron is labeled (by a chemical marker for example). There are specific connections between individual neuron. How two layers or groups of neuron connect is totally defined through the labels.
- (b) The network is organized in layers or groups where inter-layer connection are statistically depended on the distance between nodes. There must be a notions of neighborhood in such models, but it is assumed that neuron connect with each other in a random manner on a microscopic level.
- (c) Here there is still distinction between different types of neuron, such as inhibitors and activators, but the neurons connect randomly with one another.

Amari [Amari, 1972] and others have investigated in the behavior of random neural net-

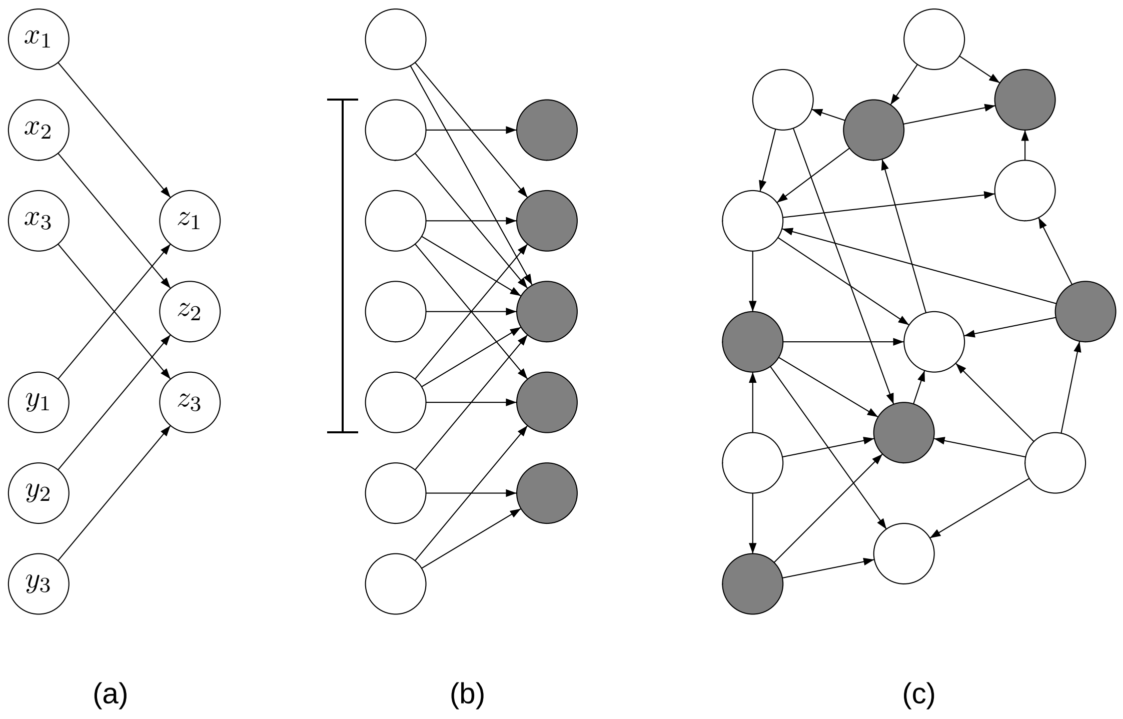


Figure 4: different categories of connectivity in the nervous system.

works. Networks that consist of one homogeneous class of neurons are proofed [2] to be mono-stable or bistable [3]. That is even the case, when the connections have positive and negative weights. It has been furthermore shown, that random nets consisting of excitatory and inhibitory classes of neurons, have not only mono-stable, bistable and tri-stable solutions, but can also generate stable oscillations.

2.6.2 Continuous Space Models

In figure 2 we have a plots of a dynamical systems consisting of two variables. However, there is no restriction to the dimensionality of the system. We have argued, that a nervous system is a dynamical system of such high dimensionality, that it seems appropriate, to look at this systems less as networks, but rather as homogeneous neural tissue and analyze them mathematically as fields. To do so we must give up the notion of individual neurons, but rather look at the averaged activity level at some point in continuous space. It should be clear, that learning on the level of individual neuron becomes impossible in such a scheme, because we cannot model a situation as described in figure 4(a). But we can average out over a situation depicted in figure 4(b) and 4(c) provided, we know something about the statistics in the connectivity of the neural net that is to be modeled. In the following section it will be shown, how reaction diffusion systems as well as the neural field model are appropriate models for continuous space modeling of certain aspects of neurodynamics.

[2]You can find a proof in [Amari, 1972]
[3]having one or two attractors

2.7 The Reaction-Diffusion Model

In situation 4(c) we have two classes of neurons, that are randomly connected with each other. There is no differentiation of layers or groups. In this situation the two classes are homogeneously spread in a given volume. If we look at these elements as morphogens just like Alan Turing described them in his landmark paper about pattern formation in chemical systems, we can adopt this model for our purposes [4] [Turing, 1952]. In this scheme different classes of neurons interact with each other. This interaction can be modeled by a local reaction between to virtual layers, each associated with a class of neurons present in the tissue. The activity of the neuron at some point in the domain take random walks through the tissue, due to the fact, that they are randomly connected with each other. Such random walks can be averaged out through a diffusion process. The general idea behind reaction diffusion systems is that two interacting class of elements can destabilize the system, whereas diffusion is seen as a more stabilizing smoothening process.

A reaction diffusion system takes the following general form:

$$\frac{\partial n}{\partial t} = f(n) + D\nabla^2 n \tag{6}$$

In equation (6) $n(x,t)$ is the value that underlies the dynamics at position x at time t. $f(n)$ sets the rule for a reaction and $D\nabla^2 n$ is the diffusion term parameterized by the diffusion rate D. Furthermore initial condition, the state of n at $t = 0$, and boundary condition[5] must be specified.
For pattern generating system two species systems are used. In such a case the hole system is governed by to interlinked partial differential equations, namely:

$$\begin{aligned} \tfrac{\partial u}{\partial t} &= f_u(u,v) + D_u\nabla^2 u \\ \tfrac{\partial v}{\partial t} &= f_v(u,v) + D_v\nabla^2 v \end{aligned} \tag{7}$$

In the last part of this section more details about special conditions under which pattern can emerge in such systems will be given.

2.8 The Neural Field Model

Neural fields model situations like the one depicted in figure 4(b). The connection weight of two neurons depend on a probability distribution parameterized by the distance between them. Here the notion of neighborhood comes into play.

Amari was one of the first, to apply a dynamical systems approach to neural networks theory. He introduced an important class of continuous networks as dynamic neural fields. For his mathematical analysis Amari studied artificial neural networks arranged in "fields" in the sense used in physics.

[4]

It is suggested, that a system of chemical substances, called morphogens reacting together and diffusing through a tissue, is adequate to account for the main phenomena of morphogenesis. Such a system, although it may originally be quite homogeneous, may later develop a pattern or structure due to an instability of the homogeneous equilibrium, which is triggered off by random disturbances[Turing, 1952]

[5]Because the specification of such a system include partial differential equations ($\nabla^2 = \frac{\partial^2 n}{\partial x^2}$)

Amari's model can be regarded as a spatially distributed population of model neurons which are connected in a random manner. Amari's equation describes the changes of the membrane potential depending both on time t and position x of a given neuron, and so is formulated as a differential equation:

$$\tau \frac{du(x,t)}{dt} = -u(x,t) + \int w(x,x')F(u(x',t))dx' + h + I(x,t) \tag{8}$$

Even if he treats the field like a continuous field, one can imagine the field as a neural network with a finite number of nodes that are fully connected and where each node is directly connected with itself. The following variables describe the system:

$u(x,t)$	The membrane potential u of the neuron x at time t.
h	The resting potential h is the same to each neuron in the field. this is only interesting in relation with a threshold, it can as well be set to 0.
$F(u(x,t))$	The activation function F is a threshold function. Normally a step function (see figure 9) a limited linear function (see figure 10) or a sigmoid function (see figure 11) is used.
$I(x,t)$	The sensory inputs at time t for neuron x
$w(x-x')$	A weighting function that determines the strength of the connection as a function of the relative positioning of the two neurons x and x'.
τ	This is the length of a single time step, so it determines also the number of time steps that pass until the system converges.

This equation describes the rule by which the potential of a neuron changes given a resting potential, an input and several lateral inputs from surrounding neurons. The next potential $u(x, t+\tau)$ can be calculated by adding up

- A resting potential which is the same value for all neurons.
- The input which is the output of a neuron in a previous layer or directly from the receptors.
- The lateral inputs that are calculated by the convolution term $\int w(x,x')F(u(x',t))dx'$.

The lateral inputs are of the main interest for a system, that also includes a reaction term, because the inputs and the resting potential can be specified in a reaction term described in section 2.7. As the reaction term acts only locally and a diffusion term only with it's nearest neighborhood, a spatial convolution introduces the possibility for long range lateral interactions in the system.

2.9 Conditions for Dynamic Stability

In order to grow pattern in a medium underlying a dynamical system, we have to have a mechanism, that searches an ordered equilibrium state. Such an equilibrium state can only be established, where there is a balanced interplay between activating and inhibiting forces.

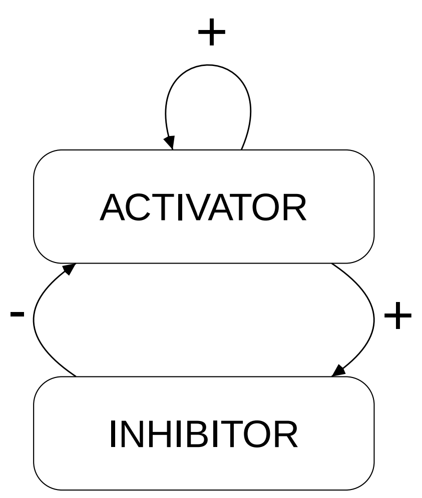


Figure 5: Dynamical equilibrium is found when activator and inhibitor balance each other out

It is interesting to note that in both of the systems proposed for biological pattern formation the self-activation is taking place more locally and inhibition takes place laterally on a longer range. In reaction diffusion systems, that are stable, the inhibitor diffuses faster. For neural fields interesting properties have been shown to occur in so-called lateral inhibition type neural fields [Amari, 1977]. In this type of neural fields we have positive Gaussian for the activator-activator connections and a negative Gaussian for the inhibitor-activator connections. For just one layered systems the two Gaussian form a Mexican hat function. This concept of lateral inhibition is found in real neural networks. Lateral inhibition leads to a competitive behavior of the nerve cells where noisy input vectors converge to contrasted output patterns. On the other hand, neurons in a certain neighborhood cooperate with each other and might together being able to activate other neurons and so expand the region of activation as a whole. Simple cooperation leads to smooth but uncontrasted output because the region of activity is expanding without being stopped by inhibited neurons, therefore cooperation must be combined with competition. Lateral inhibition stabilizes the region because it keeps the activation level of the neurons inside the region from always increasing, therefore it determines the edge of the region which is expanding. [Amari and Arbib, 1977]

3 State of the art

Both Models of biological pattern formation have been used extensively in robotics. Several approaches are cited in this section. Despite an extensive bibliographic search, no project has been found, that tried to combine the properties of dynamical systems, that rely on neural fields and reaction diffusion systems. Every project in this listing has either used neural fields or reaction diffusion systems. It is part of the goal of this master thesis to introduce a new class of spatio-temporal pattern generating system, that include properties of both classical approaches.

3.1 Analysis of Reaction Diffusion Systems

Even though Reaction diffusion systems were first described in the context of chemistry, they have been also analyzed in the context of developmental biology, which investigate in the formation of spatial pattern in the embryo and the differentiation of cells in that process [Philip K. Maini, 1997].

Hans Meinhardt, from the University of Tübingen, also put forward significant analysis of reaction diffusion system. He especially investigated in the conditions in which pattern in reaction diffusion systems can occur. Namely local self-activation and lateral inhibition [Hans Meinhardt, 2000].

Other authors where more interested in reaction diffusion system from a computational perspective. Thomas Henderson argues, that reaction diffusion systems can be analyzed from the point of view of their computational capabilities. He proposes that reactions diffusion models are a more general computational models than Turing machines, and in fact include them as much as they include models of recursive function theory [Henderson, 2001].

3.2 The Application of Reaction Diffusion Systems

Authors like Kirby and Conrad argue that the main drawback of conventionally programmable digital computers is their lack of efficiency, adaptability and evolvability. Therefore they investigated in structurally intelligent system that can behave "intelligent" without the explicit storing of symbols. Kirby and Conrad successfully applied their Turing-like reaction diffusion system on a robot for simple navigation tasks. They called their control systems "Turing's other Machine" [K.G. Kirby, 1986].

In some laboratories so-called smart chemicals are used to build completely new massively parallel bio-molecular computing engines, based on reaction diffusion principles. These methods are especially interesting for difficult computational problems in image processing, pattern recognition and path finding [Rees, 2003] [Rambidi, 2000].

As mentioned in section 1.1, the dynamics of reaction diffusion systems have become increasingly important in robotics. The fact, that the dynamics of such systems can be used to implement higher level tasks, such as navigation [Adamatzky *et al.*, 2004], but also central pattern generators, provided stable oscillating pattern emerge in the system. Central pattern generator have become the main concept in the study of locomotion in vertebrate [Ijspeert, 2002]. Therefore the paradigm of CGP's is omnipresent in the engineering of locomotion in biologically inspired robotics [Arena, 2000].

3.3 Analysis of Neural Field Models

The main effort in the theoretical analysis of neural fields dynamics have been made by Amari. He studied the behavior of one dimensional neural fields, and found, that three different kind of interesting pattern can emerge in neural fields, that use connections with laterally inhibiting weights. He showed the existence of mono-stable and bistable dynamics as well as the possibility of oscillating pattern when the system consists of an activating layer and a inhibitory layer [Amari, 1977].
Wellner and Schierwagen have shown in simulations, that similar pattern emerge in two dimensional neural fields. At the same time they showed that cellular automata-like simulations like they have been also used in reaction diffusion system [Weimar, 1997] are appropriate for the simulation of neural fields [Jörg Wellner, 1998].

3.4 The Application of Neural Field Models

It was Gregor Schöner that suggested to use a read out at some point in the neural field, to directly control the servos of some robot. The combination of Amari's neural field approach and Schöner's read out mechanism represents a elegant solution for robot navigation. It follows brief and certainly incomplete description of projects that have either used the neural field approach alone or in combination with Gregor Schöner read out mechanism.
Neural fields are often used in the context of perception, especially for visual systems. Because the neural field approach is a continuous model, the number of nodes is virtually unlimited. With a high resolution of input images in the input layer the number of nodes in the network should not be restricted. With its capability to simulate short term memory the neural field can be used for motion recognition or attention.

Motion recognition M. A. Giese of the MIT [6] has developed a neural field model for motion pattern recognition. In his approach one neural pathway calculates the form orientation from the input image, the other one calculates the optic flow of the motion orientation. In both pathways neural fields are used for the recognition.[Giese, 2000]

Attention Volker Stephan and Horst Michael Gross from the University of Ilmenau have investigated in modeling sequential visual attention with the help of columnar organized neural fields. Their neural fields are 2-layered and model pyramid cells, chandelier cells and globally acting inhibitory cells. Columnar organized means that the neurons at the same location in different layers are associated in columns.

Robot control In the research group around Schöner himself, Axel Steinhage and Ioannis Iossifidis have used dynamic neural field for robot control [Steinhage, 1997]. They used neural fields for a service and assistance robot named CORA [7]. The robot CORA is a manipulator with 8 degrees of Freedom (DOF) [Ioannis Iossifidis, 2001]. Schöners conception of the dynamical behavioral variable which is the solution that generates the robots behavior was extended in particular by Axel Steinhage to create more complex behavior such as behavioral sequences, behavioral organization and learning [Steinhage and Bergener, 1998] [Steinhage, 2000].

[6]Massachusetts Institute of Technology
[7]CORA is an abbreviation for COoperative Robot Assistant.

Learning by imitation Sorin Moga and Philippe Gaussier from the Groupe Neurocybernetique at the ENSEA [8] have used neural fields and the read out mechanism to treat visual information in order to model imitation in robots [Gaussier *et al.*, 1997].

[8]The ENSEA is the Ecole nationale superieure de l'electronique et de ses applications

4 Implementation

As mentioned in section 3 what concerns simulators for pattern generating systems, there are very few simulators that are optimized for either reaction diffusion systems or neural fields models. Since such systems include spatial derivates as well as spatial integration. Such systems are hard do implement on classical ODE simulating system. Furthermore there is to our knowledge no simulator that would include both models. Therefore such a tool is very useful for the exploration of Integro-Reaction-Diffusion systems. A simulation tool for spatially extended dynamical systems should first and foremost dispose of an interface through which a user can create and manipulate fields or layers on which the corresponding differential equation can act upon. It should be possible to set the temporal and spatial resolution with which the system should be simulated, and while the application is iterating through the simulation, it should be possible to visualize or/and save results in the form of raw data or in different plots and formats. In the following section some of the issues concerning the implementation of `PantaRhei` are addressed.
The dynamical systems to be simulated are restricted to a form with which it is possible to model neural field equations as well as reaction diffusion systems. We therefore need a system that provides local interlayer connections, kernel convolution methods and diffusion processes inside of each layer. Let n be the number of layers in such a system. Then a Integro-Reaction-Diffusion system could be defined through the following general form:

$$\frac{\partial u_i(x,t)}{\partial t} = R(u_1 \dots u_n, x, t) + D_i \frac{\partial^2 u_i}{\partial x^2} + \int_X K_i(u_i, x) F_i(u_i) dx \tag{9}$$

In equation (9) the sum of three terms define the evolution of the state u of our system.

1. The first term $R(u_1 \dots u_n, x, t)$ is a function or a an expression which computes a new value, given all the values of the layers 1 to n at the position x at time t.
2. The second term is the diffusion term. This term includes diffusion in each layer, parameterized by an individual diffusion rate D_i for each layer i.
3. The last term is a convolution term, that has a kernel K_i for each layer and a activity or threshold function F_i associated to layer i.

This form of a dynamical system is general enough to model dynamical neural fields, reaction diffusions systems, as well as combinations of the two.
`PantaRhei` is a language with which one can specify a Integro-Reaction-Diffusion system, setting parameters of the simulation, setting plotting and saving options and of course starting a simulation for the number of time steps specified. There are three different interface through which the user can send commands to the interpreter. In the following sections information will be given about the implementation of the parser as well as the grammar of `PantaRhei` .

4.1 The Interface

There are three different ways in which `PantaRhei` can read commands:

1. `PantaRhei` can be open in a interactive console.
2. A file can be given as command line option when invoking `PantaRhei` .
3. `PantaRhei` can be accessed through a pipe.

For convenience, the interactive console was implemented, using the `readline/history`-Library, that provide auto-completion and a history.

4.1.1 The Parser

The PantaRhei parser is implemented using the Boost Spirit Library (BSL) [spi, 2005]. After long and unsuccessful trials with YACC and LEX [yac, 2005] [9] Which concerns the BSL, the host language and the target language are both C++. By means of meta-template programming techniques [D. Abrahams, 2005] , a parser can be specified using a syntax which is very similar to the Backus-Naur-Form, but can be written as C++ source code. In figure the specification of a simple expression parser, using the BSL:

```
expr =   term >> '+' >> term
       | term >> '-' >> term
       | term ;

term =   factor >> '*' >> factor
       | factor >> '/' >> factor
       | factor ;

factor = integer
       | '(' >> expr >> ')' ;
```

Figure 6: A simple expression parser using the Boost Spirit Library

Attached to the different terminal symbols are semantic actions, that are executed, when the production parses. The whole grammar and the corresponding functions that serve as semantic actions, are specified in the file `Interpreter.h`. This file is the heart of the project, where everything is kept together. Because of the use of meta-templating techniques, the code is hard to understand at first. Once grasped the structure of file, one will surely appreciate the expressivity of the code.

4.1.2 Syntax

The entire grammar of PantaRhei is listed in Appendix B. In this section a quick overview is given on how the grammar is structured. The streams of token that PantaRhei reads in, is divided in statements. A statement can either be a command (with or without arguments), as well as a declaration or a definition. Any string, that is not a keyword can be used to assign an object to it. Commands have the general form of:

```
<command> <argument 1> ... <argument n>
```

or to set a global variable, such as temporal or spatial resolution as well as plotting range and style.

```
set <variable> = <value>
```

Possible commands are:

[9]LEX and YACC are hosts languages to specify parser. The can generate C and C++ Code. But unfortunately Flex++ and Bison++, the GNU derivates to generate C++ classes where not maintained for a long time.

- `help` takes no arguments and simply displays a short manual.
- `file` takes 1 argument, which is a previously declared layer.
- `plot` takes 2 argument, which is either a previously declared layer or a previously declared kernel and the format to which the plot should be saved. There are 3 different possible formats: `screen`, `jpeg` and `pstricks`.
- `run` takes no argument and starts the simulation
- `set` takes 2 arguments, where the first is the variable name and the second is the value to be set.
- `show` takes one argument which is either "`system`", to look at the system variables or "`symbols`" to have a look at the variables that are already stored in the symbol table.
- `quit` quits the application.

A declaration consists of the object type and an identifier:

```
<object type> <identifier>
```

Possible object types are:

- `layer |LAYER`
- `kernel|KERNEL`
- `reaction|REACTION`
- `diffusion|DIFFUSION`
- `matrix | MATRIX`
- `function|FUNCTION`
- `constant|CONSTANT`

An identifier can be any alpha-numerical string (including the underscore ("_")), that is not a keyword. One can declare and define objects in the same line.

Definitions are of the general form:

```
<identifier> = <initializer>
```

The `initializer` is different for each object type, depending which information are needed to define the particular object:

Constants Constants consists of a scalar double value.

```
<identifier> = <real>
```

Functions For a Function, one can specify up to four arguments. We will see, that there are no functions that require more. `PantaRhei` provides a set of basic functions that can be parameterized and linked to a new identifier.

```
<identifier> = <base function>(<argument 1>, ... <argument n>)
```

With n ≤ 4 of course.

Matrix Matrices are read from a file and utilized as initial conditions for layers. Therefore the initializer for a matrix is a file path, where directories a separated with a normal slash. The file the path leads to, is a normal ASCII-file with space-separated values. Such files can easily be exported from Matlab.

Diffusion A diffusion object just needs a float value (as diffusion rate) and therefore is defined just like a constant, as described earlier.

Reaction A reaction term is defined as a simple expression consisting of multiplications, divisions, addition and subtractions as well as the parenthesis to force priority of the different subexpressions. Its formal specification can be looked up in the appendix A.

4.2 Data Structure

4.2.1 Symbols

As depicted in figure 7 every type of data structure is directly derived from a symbol class. The whole dynamical system, that is to be simulated consists of an array of layers. Layers are the central structure in `PantaRhei` . A layer consists of a kernel, a diffusion, a reaction and a matrix, that will serve as initial condition. To specify a kernel, one must declare its size, two functions and a boundary condition. The kind of functions, that can be assigned are described in section **??**. A reaction term is an expression containing floats, constant variables and layer variables. A diffusion term is simply a float or constant variable. All the symbols are stored in a simple symbol table as described in section 4.2.2.

4.2.2 The Symbol Table

The symbol table is implemented with a map container of the Standard Template Library (STL) [stl, 2005]. The symbol table simply maps a string with a pointer to a Symbol. During the parsing of the symbol specification, the symbols are stored in the symbol table. While initializing and running the simulation, symbols are retrieved from the symbol table, using the associated identifier string. The symbol table must be visible in the kernel class as well as in the layer class, because they contain links to other symbols.

4.2.3 Expressions

The expressions in the reaction term are fairly simple. While parsing a expression in the input, a expression tree is build up, using a stack. Terminals in the tree, can either be float values or variables. Where variables either point to a scalar constant or a layer specified in the system. Operations link to two terminals with a basic operation such as multiplication, division, addition and substraction. Brackets are used to force the priority of operations

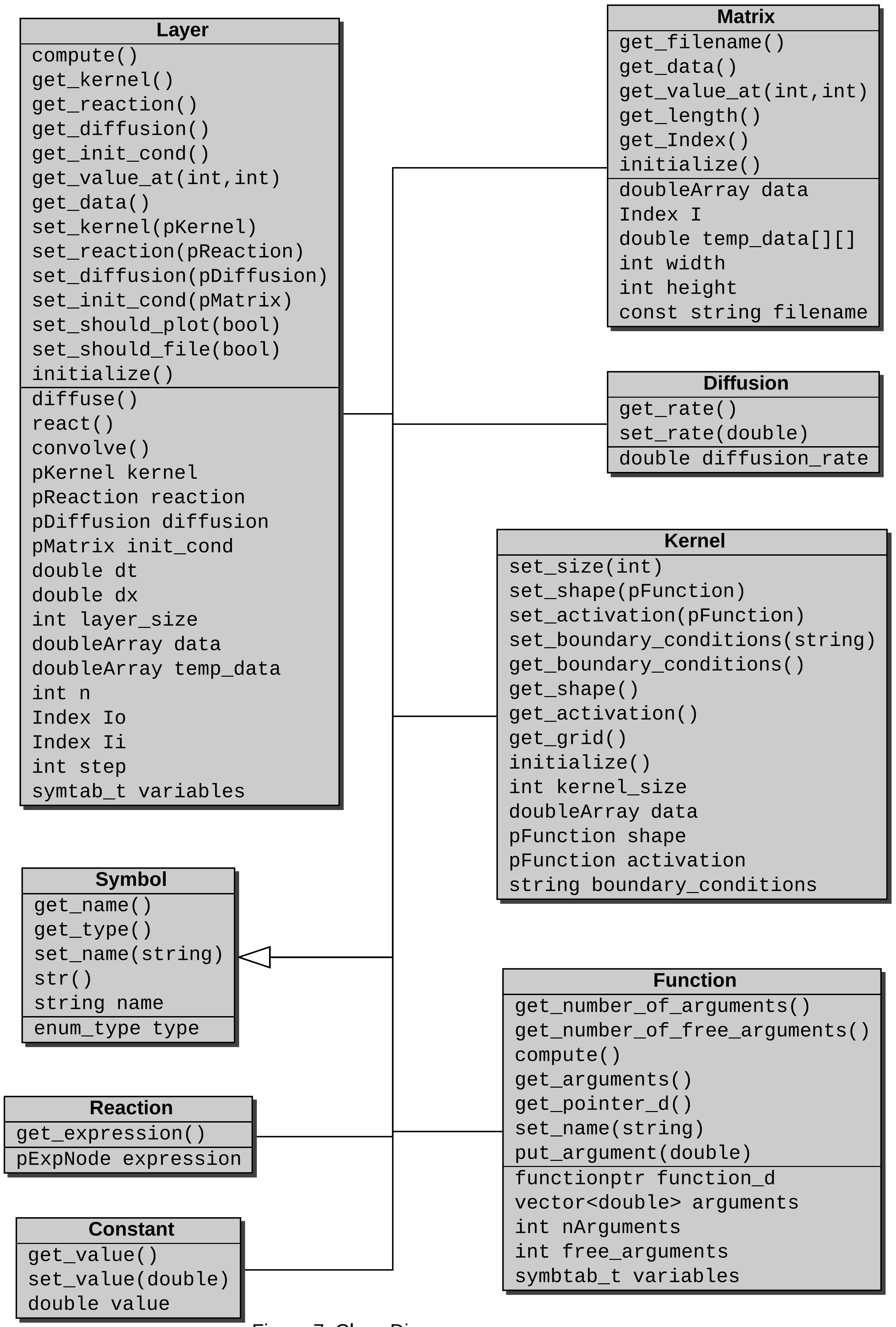


Figure 7: Class Diagramm

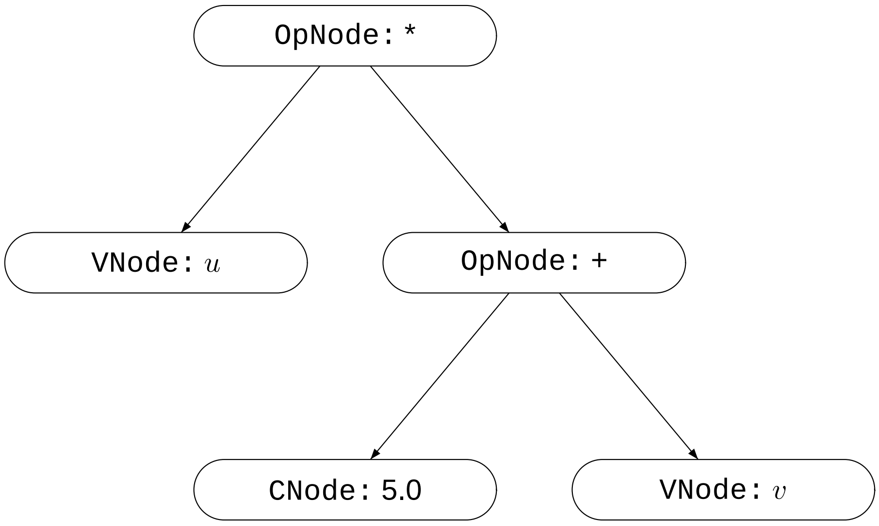


Figure 8: Storage of the expression $u(v + 5)$

and therefore alter the structure of the expression tree. The way the expression will be calculated at each point in the layer at each time step, is further described in section 4.3.1.

4.2.4 Kernels

The kernel consists of a matrix, which has a certain shape and a rule of activation. (See the convolution term in section 4) Two functions are used to specify the kernel instance. The function specifying the kernels shape is accessed, only in the initializing procedure. The kernel is then stored in an array, so that values can be accessed directly instead of being re-calculated at each step. The activation function on the other hand, must be calculated for each point in the domain and for each time step.

Weighting Function The function describing the shape of the kernel is some density function that describes the probability of an element to be connected with another element, depending on their distance from one another. We suppose, that the neighborhood [10] of an element is a circle, and is simply described by its diameter (the size of the kernel). We further assume, that the probability of two elements making a connection, obeys some form of normal distribution. That is why `PantaRhei` provides gaussians or mixture of gaussians as weighting functions. When the program is started several different basic functions are stored in the symbol table, that can be parameterized and utilized for the specification of a kernel.

- We have a normal Gaussian, where we can only choose the variance of the kernel.

```
my_kernel_shape = norm_gaussian(<variance>);
```

- To be able to scale a gaussian or to define all weights as negative, one must use

[10]The area in which the element can make connections

```
my_kernel_shape = scaled_gaussian(<scaling factor>, <variance>);
```

- To model a mixture out of negative and positive connections, there are two functions that will form a mexican hat type of kernel. We have the standard mexican hat (see figure 14)

  ```
  my_kernel_shape = norm_mexican(<positive variance>,<negative variance>);
  ```

  where the two variances for the negative and the positive gaussian are specified.
- For a scaled gaussian, the scaling factor for the positive and negative gaussian is to be added.

  ```
  my_kernel_shape = scaled_mexican( <positive scaling>,
                                    <positive variance>,
                                    <negative scaling>,
                                    <negative variance>);
  ```

In all of these density functions the mean value is in the middle of the kernel.

Activation Function The activation function models the frequency rate of the action potential given a certain membrane potential. The functions used, are typically threshold functions, that have also an upper bound. In real neurons, a refractory state in which the neuron is not capable to generate an action potential, creates an upper bound for the neurons firing rate. Three different functions are possible,

step The simplest function is a simple step function, as depicted in figure 9. The step function is defined by the function in equation (10)

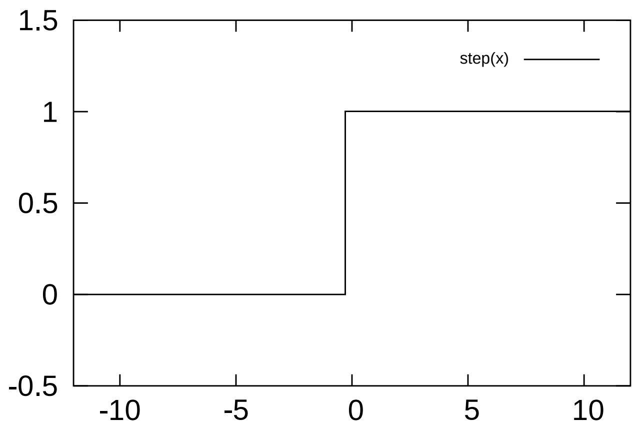


Figure 9: The Step Function

$$f(x) = \begin{cases} l\,,\ if\ \ x - \theta < 0 \\ u\,,\ else \end{cases} \tag{10}$$

Where u is the upper bound, l is the lower bound and θ is the threshold. A basic step function is provided by the symbol table. The function is accessed through:

```
my_step_function = step(<threshold>, <lower bound>, <upper bound>);
```

linear To take into account, the degree to which the membrane potential deviates from the resting potential, a limited linear function is provided. It is given by $f(x)$ in equation (11)

$$f(x) = \begin{cases} l\,,\ if\ \ x - \theta_l > 0 \\ u\,,\ if\ \ x - \theta_u > u \\ x\,,\ else \end{cases} \tag{11}$$

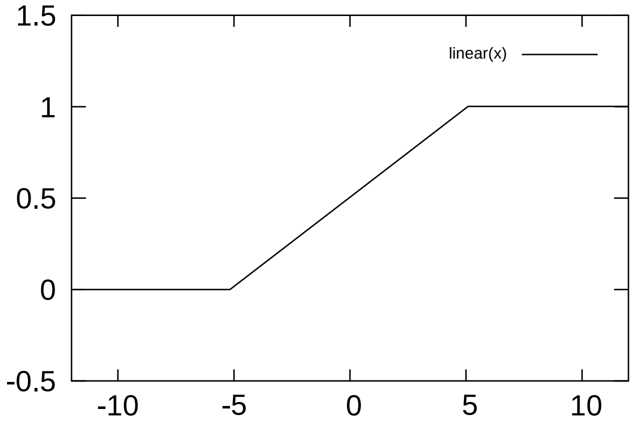


Figure 10: The Limited Linear Function

To make an instantiation, one must provide a lower threshold and a upper threshold as well as a lower bound and a upper bound.

```
my_step_function = limited_linear(<upper threshold>,
                                  <lower threshold>,
                                  <lower bound>,
                                  <upper bound>);
```

sigmoid Physiologically the most plausible activation function, is a sigmoid function, which values tend to either the upper or the lower bound. Also here, a basic function exists and can be parameterized, by specifying a shift in x and y direction and a scaling in x and y direction. This gives us the general equation:

$$y = s_1 \left(\frac{1}{1 + e^{-xs_2 + s_3}} \right) - s_4 \tag{12}$$

Where s_1 is the scaling factor on the y axis, s_2 is the scaling factor for the x axis, whereas s_3 and s_4 shift the output value on the x and y axis respectively. An example of different scaled and shifted sigmoid function is given in the figure 11:

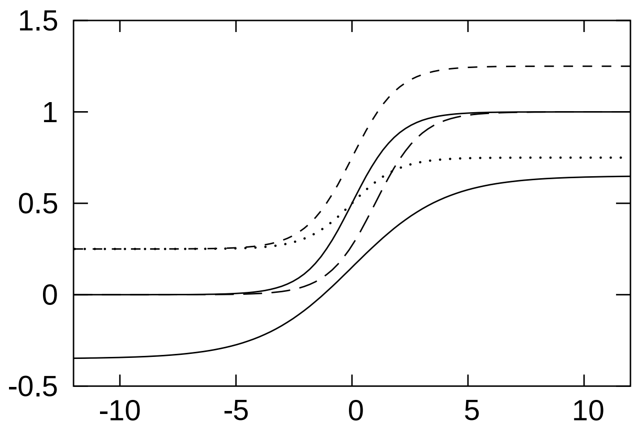


Figure 11: The Sigmoid Function

4.2.5 The Array Class

Because the simulations consist mainly of matrix calculations, it was appropriate to spend some time choosing a fast and flexible array class. It is also desirably to have a class that would support parallelization. The Library of choice was the A++/P++ array library [app, 2005] [Quinlan, 2000], that is use in the Overture framework [ove, 2005] for solving PDE's.

4.3 Algorithms

When the simulation begins, for each time step each layer in the system must be calculated. Each layer has all information needed to compute its next state. It does so in computing for each point in the plane: Its reaction given the states of the other layers, its diffusion given the state of its nearest neighbors and the associated diffusion rate and the convolution term given the kernel and the neighborhood specified in the kernel instance. The values are summed up and divided by the time step and added to the value at the previous time step. It follows a description of how the three terms, that are calculated.

4.3.1 Calculating the Reaction Term

In section 4.2.3 the dynamic data structure in which expressions are stored is described. An operation node links to two other nodes, that can be either a constant float value, a variable node pointing to either a layer or scalar value or an another operation node. All three node types inherit a virtual method `get_value` from its superclass `ExpNode`. When this method is invoked in a operation node, then `get_value` invokes `get_value` in the two nodes it links to. When a variable node or a float is reached the value can be read immediately. Now the result is passed upwards and the operations are computed upon them. This procedure is done for each point in the plane.

4.3.2 Calculating the Diffusion Term

The diffusion class contains just the diffusion rate as a float value. Diffusion is what makes the resulting dynamical system be governed by a partial differential equation. The stability of this calculation step highly depends on the resolution that is chosen for the layer. One has to calculate twice the spatial derivate $\frac{\delta^2}{\delta x^2}$. Given a spatial steps $\Delta x = \Delta y$ and a temporal step Δt we calculate the new value at point x in layer u as follows:

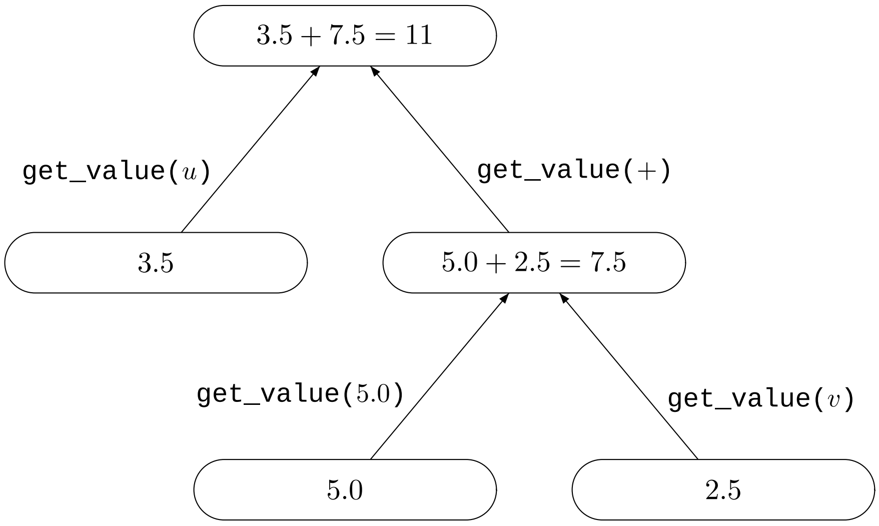


Figure 12: computing $3.5(2.5 + 5)$

$$u(x, y, t + \Delta t) = \begin{array}{l} D(u(x - \Delta x, y, t) - 2u(x, y, t) + u(x + \Delta x, y, t) \\ +u(x, y - \Delta y, t) - 2u(x, yt) + u(x, y + \Delta y, t)) \end{array} \tag{13}$$

Because we have to calculate the spatial derivate, we have to choose boundary condition. For reaction diffusion systems normally two different kind of boundary conditions: `PantaRhei` only supports the simple von Neumann zero flux boundary condition. That means, that at the borders of each layer the spatial derivate is set to zero.

4.3.3 Calculating the Convolution Term

The convolution term is the most complex of the terms to be calculated. A convolution on a 2 dimensional layer using a 2 dimensional kernel requires 4 nested `for`-loops. When we have a $m \times m$ Layer and a $n \times n$ kernel, the convolution over a layer has the complexity O(n^2m^2). Therefore the kernel should be kept small. Let $f(u)$ be a activation function as described in section 4.3.3 and $W = w_{ij}$ be the kernel. The discretized convolution at point $p = (x, y)$ for the next time step is then calculated as follows:

$$u(x, y, t + \Delta t) = \sum_{i=-r}^{i=+r} \sum_{j=-r}^{j=+r} w_{ij} f(u(x + i\Delta x, y + j\Delta y, t)) \tag{14}$$

Also for convolution boundary conditions must be specified. `PantaRhei` provides 4 different kind of boundary conditions for the convolution.

`repeat`:	Repeat the last valid value.
`avoid`:	Set all values outside the domain to zero.
`wrap`:	Wrap the domain to a torus.
`reflect`:	Let the index return inside the domain.

4.4 Plotting

As mentioned in the introduction, it is important for a simulation framework to have the ability do visualize the solutions obtained. In `PantaRhei` plotting is done, using a interface, to access the Gnuplot utility [gnu, 2005] through a pipe. The great advantage that Gnuplot has, is that it support several different terminals. Gnuplot also allows many options for the style of the plotting. Only a small set of options can be set through the `PantaRhei` interface.

In general, only layers and kernels can be plotted. When a kernel has been defined, the kernel shape can be plotted with the command: `plot <identifier> to <terminal>`. Where `<terminal>` is the format in which the plot should be outputted. Four values are possible for this option:

- `screen`: The plot is simply drawn to the screen. This is for observing the simulation for tests.
- `jpeg`: This option is chosen when one wants to save images of the simulation, to make movies and for HTML documents. For the generation of movies, `transcode` has been of great use. A bash script is provided with the package, that makes creating movies out of jpeg images very easy.
- `pstricks`: Even nicer is the LaTeX format using `pstricks`, because text and other labels or symbols can be nicely added within the LaTeX syntax, to render plots for printed documents.

It is absolutely necessary, to specify in advance, the range in which the values of the simulation are expected. Gnuplot must fix its range in order to not change the range with each plot that is generated. That is why before plotting one must specify:

```
set upperbound = <integer>
set lowerbound = <integer>
```

As already mentioned in section 4.1.2 it is possible to choose between two different styles of plotting: `map` and `landscape`. By choosing `map` the data will be plotted in a image, by choosing `landscape` the data will be plotted as a surface. There is an example of a `map`-plot in figure 15, and an example of a `landscape`-plot in figure 13.

4.5 Parallelization

Thanks to the A++/P++ array class it is possible to run the application in a network of heterogeneous platforms. This is a feature that can be interesting when one has to make simulations with high spatial resolution. For smaller simulations there is probably no gain in time, but rather loss, due to the overhead created by the network communication protocols. The interface that is used to run the application in parallel is the LAM/MPI [lam, 2005] interface, which creates a virtual machine using several computers in the network.

5 Results

In the following section the results of several simulations are demonstrated. The first example is a very simple one layer reaction diffusion system. Then the properties of a one layer neural field are demonstrated. Different properties of the fields dynamic, just like predicted by Amari [Amari, 1977] and showed for the two dimensional case by Wellner and Schierwagen [Jörg Wellner, 1998] can be reproduced using the `PantaRhei` environment. Furthermore, as diffusion is added, new properties arise. Another example drawn from [Jörg Wellner, 1998], namely the system consisting of two layers that supports traveling waves, we demonstrate how a slight change in the kernel shape can produce very different behavior. A classic example for a two species reaction diffusion system is the Belousov-Zhabotinskii reaction. A `PantaRhei` script is presented, simulating such a system. Each of the subsections have the same ordered structure. The model is first presented in a mathematical notation. Then the part of the `PantaRhei` code that defines the system is listed before a series of plots show the development of the system. And in the end the dynamics is discussed.

5.1 The Schloegl System

A Schloegl system is a very simple one species model of a reaction diffusion systems. It is defined by the following partial differential equation:

$$\frac{\partial n}{\partial t} = -k(x - p_0)(x - p_1)(x - p_2) + D\nabla^2 n \tag{15}$$

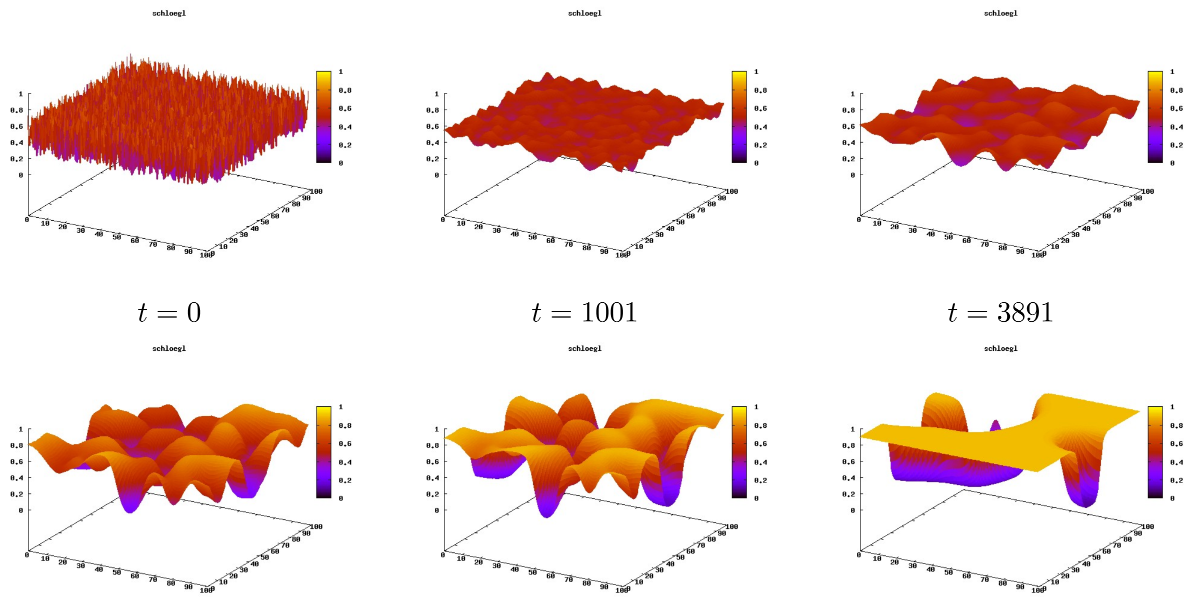
A complete specification of the system is coded in `PantaRhei` as follows:

```
matrix ic = init/random.init

constant h1 = 0.1
constant h2 = 0.5
constant h3 = 0.9
constant k  = 1.5

Layer schloegl = {NULL, -k*(schloegl-h1)*(schloegl-h2)*(schloegl-h3), 0.001, ic}
```

We can see the development of that system in figure 15. Starting from uniform random distribution, the diffusion term smoothes the distribution, before the reaction terms takes over and let each point of domain tend to one of the stable fixed points. We have chosen the following parameters for the system: $k = 1.5$, $p_0 = 0.1$, $p_1 = 0.5$ and $p_2 = 0.9$. This represents a symmetric case, where 0.5 is a unstable fixed point and 0.1 and 0.9 are stable fixed points. It can be easily seen, that the system whould not change when each point in the domain, was to be initialized with 0.5. but as soon as only one point is differs from 0.5 the neighborhood of that point whould change due to the diffusion process. Starting from that initial deviation the whole domain whould move to one of the stable fixed points. Because of is simplicity the schloegl system is a great system to demonstrate properties of dynamical systems such as attractors and fixed points.



$t = 0$ $t = 1001$ $t = 3891$

$t = 7521$ $t = 10001$ $t = 35381$

Figure 13: plots of the Schloegl system

5.2 One Layered Neural Field

The model of the one layered neural field is given through the following equation:

$$\tau \frac{du(x,t)}{dt} = -u(x,t) + \int w(x,x')F(u(x',t))dx' + h \tag{16}$$

As kernel w we have a mexican hat plotted in figure 14.
A complete specification of the system is coded in `PantaRhei` as follows:

```
// initial conditions

matrix s = init/single_layer100.init

// constants

constant h = 0.0

// kernel:

Kernel w = {24,
            scaled_mexican(0.1, 18, 0.1, 20),
            step(0.1, 0, 0.1),
            repeat}

// layers

Layer u = {w, -u+h, NULL, s}

set spatial resolution = 100
```

We can see the development of that system in figures 15 to , with slightly different parameters.

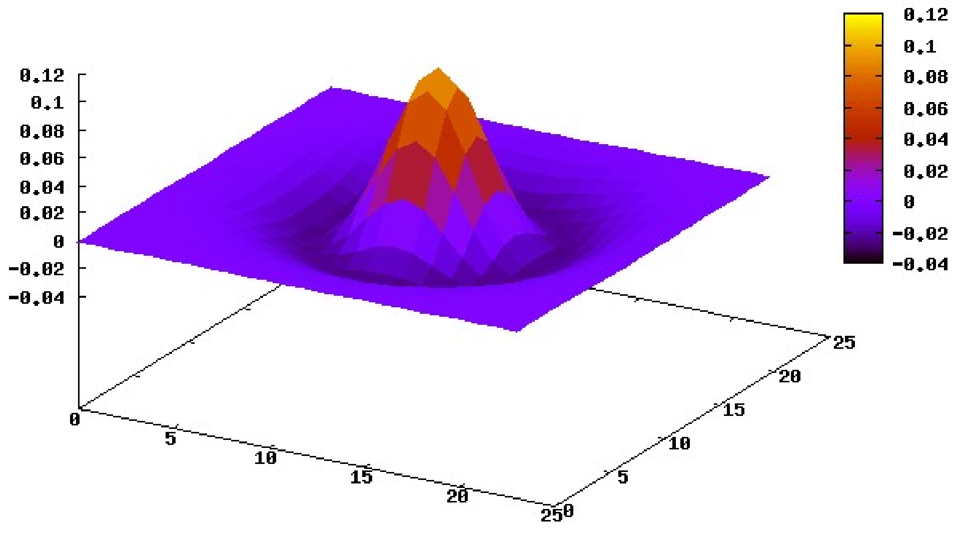


Figure 14: A mexican hat like kernel used for neural fields with lateral inhibition

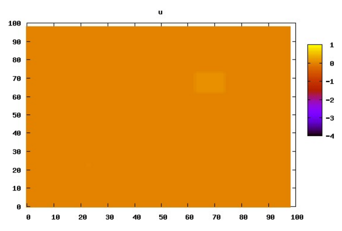
We can see that already such a simple system shows some interesting properties. In figure 15 a large region of excitation in the upper right corner of the domain is not further enhanced but has the tendency to even die out. Not so the the small point of excitation, barely visible at the bottom left of the plot. The region around this small point of excitation is strongly inhibited and therefore builds up a contrast to the excited region.

We have another situation in figure 16. The initial condition consists of several distributed points of excitation. Through cooperation of points that in a certain proximity to each other and competition between excited points, that are farther away, the field chooses some regions and enhances their contrast. Interesting to note here, that not all region become enhanced at the same time but some are kept suppressed for a while before they can build a island of excitation. So there is not only a spatial differentiation, but also a temporary one.

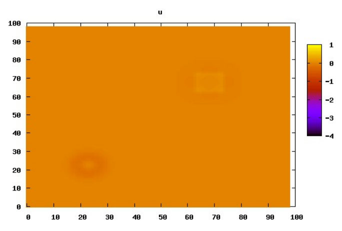
In the figures 17 and 18 diffusion is added to the system. In figure 17 we see, that diffusion somehow increases the effect of competition in the domain, this time only the regions can survive that can cooperate with each other. Isolated regions of excitation die out in such an environment. But still, the system is able to find a stable state in which three islands of activity remain. Not so in the situation plotted in figure 18. Here the diffusion rate is to high and the system finds itself in a stable state, where all the excitations have died out.

5.3 Two Layered Neural Field

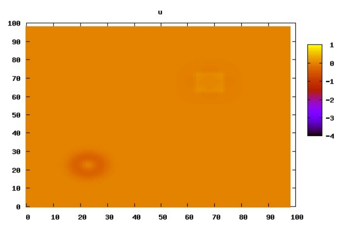
As Amari [Amari, 1977] and Wellner / Schierwagen [Jörg Wellner, 1998] have shown, neural fields support traveling waves, when the system consists of two layers. One is a



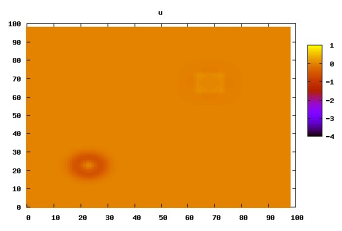
$t = 0$



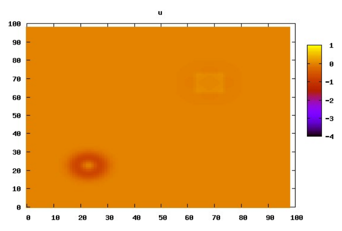
$t = 500$



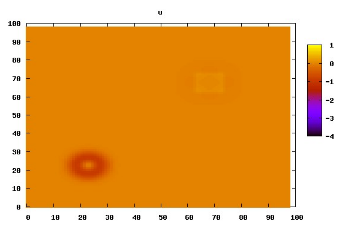
$t = 1000$



$t = 1500$



$t = 2000$



$t = 2500$

Figure 15: plots of the one layered neural field

excitatory layer, governed by:

$$\tau\frac{\delta u_1(x,t)}{\delta t} = -u_1(x) + \int w_1(x,x')f(u_1(x',t))dx' - \int w_2(x,x')f(u_2(x',t))dx' + h_1 \quad (17)$$

and a second inhibitory layer is governed by the differential equation:

$$\tau\frac{\delta u_2(x,t)}{\delta t} = -u_2(x) + w_3(x)f(u_1(x,t)) + h_2 \quad (18)$$

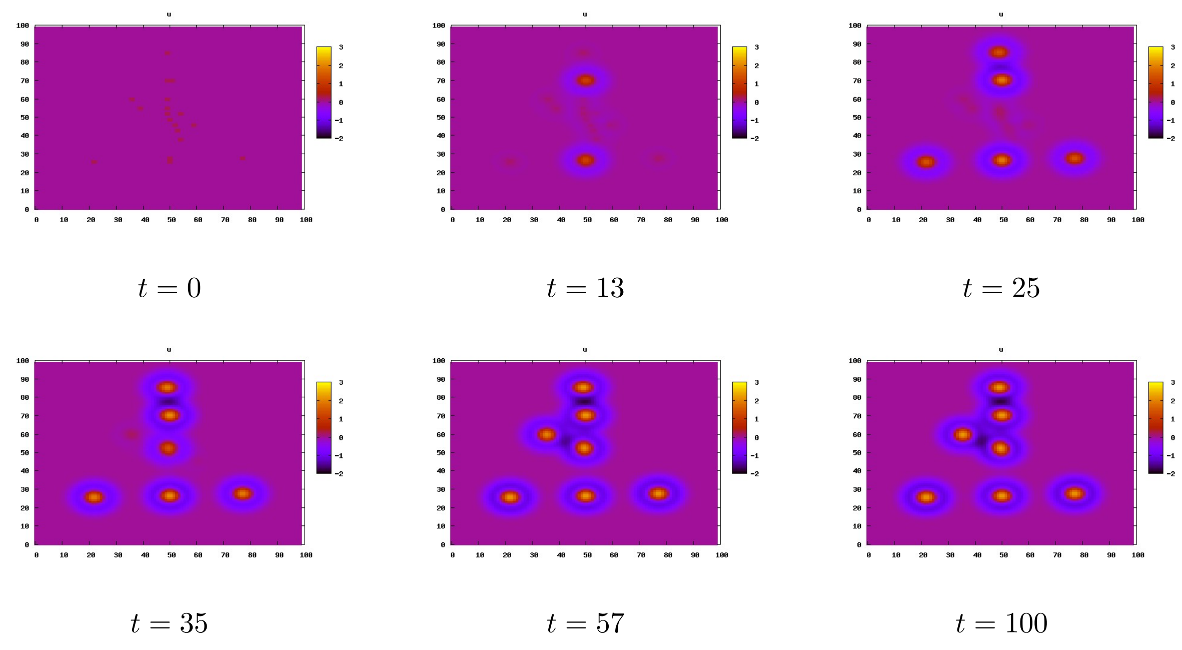
Two kernels are in use w_1 being only excitatory and w_2 containing only inhibitory connections. Such a equation can be specified in `PantaRhei` as follows:

```
// constants
constant h1 = 0.0
constant h2 = 0.0

// kernel:
Kernel w1 = {12, scaled_gaussian(0.1, 7), step(0.1,0, 0.1), repeat}
Kernel w2 = {12, scaled_gaussian(0.1, 7), step(0.1,0, 0.1), repeat}
Kernel w3 = {3, constant_kernel(0.3), step(0.1, 0, 0.1), repeat}

// layers
Layer u1 = {w1, -u1-tmp_u2+h1, NULL, s1}
Layer u2 = {NULL, -u2+h2+tmp_u1, NULL, s2}
Layer tmp_u1 = {w3, NULL, NULL,z}
Layer tmp_u2 = {w2, NULL, NULL,z}
```

We can see the development of that system in figure 19 and 20.
In the initial condition producing such traveling waves, we have a small line of excitation in both layers, but the regions are a little bit shifted. In the plots in figure 19 only one wave front is generated and moving out of the the domain. After 250 time steps the field is in its resting potential again. In figure 20 the kernel shape has slightly changed, but


$t = 0$ $t = 13$ $t = 25$

$t = 35$ $t = 57$ $t = 100$

Figure 16: One layer neural field: through cooperation and competition in the domain, the neural field chooses and contrasts the most significant excitations from the initial conditions.

the behavior is totally different. In this case, the field will generate a never ending cycle of traveling waves. Given the initial conditions described earlier, the waves are organized in spirals.

5.4 A Belousov-Zhabotinskii Reaction

The Belousov-Zhabotinski reaction is a very well known reaction of chemicals, discovered independently by Boris Belousov in the 50's and A. M. Zhabotinskii in the early 60's. The reaction results in the establishment of a chemical oscillator. A petri dish in which a BZ-reaction take place, is shown in figure 21. A oscillating waves has been simulated with `PantaRhei` .

The Belousov-Zhabotinskii reaction is governed by two differential equation, the activator being:

$$\frac{du}{dt} = D_u \nabla^2 u + \frac{u(1-u) - sv\frac{u-q}{u+q}}{\epsilon} \tag{19}$$

and the inhibitor is

$$\frac{dv}{dt} = D_v \nabla^2 + u - v \tag{20}$$

Such a equation can be specified in `PantaRhei` as follows:

```
matrix m1 = init/u.init
matrix m2 = init/v.init

Layer u = {NULL, (u*(1-u)- 3 * v * (u-0.002)/(u+0.002))/0.01, 0.5, m1}
Layer v = {NULL, u-v, 0.1, m2}
```

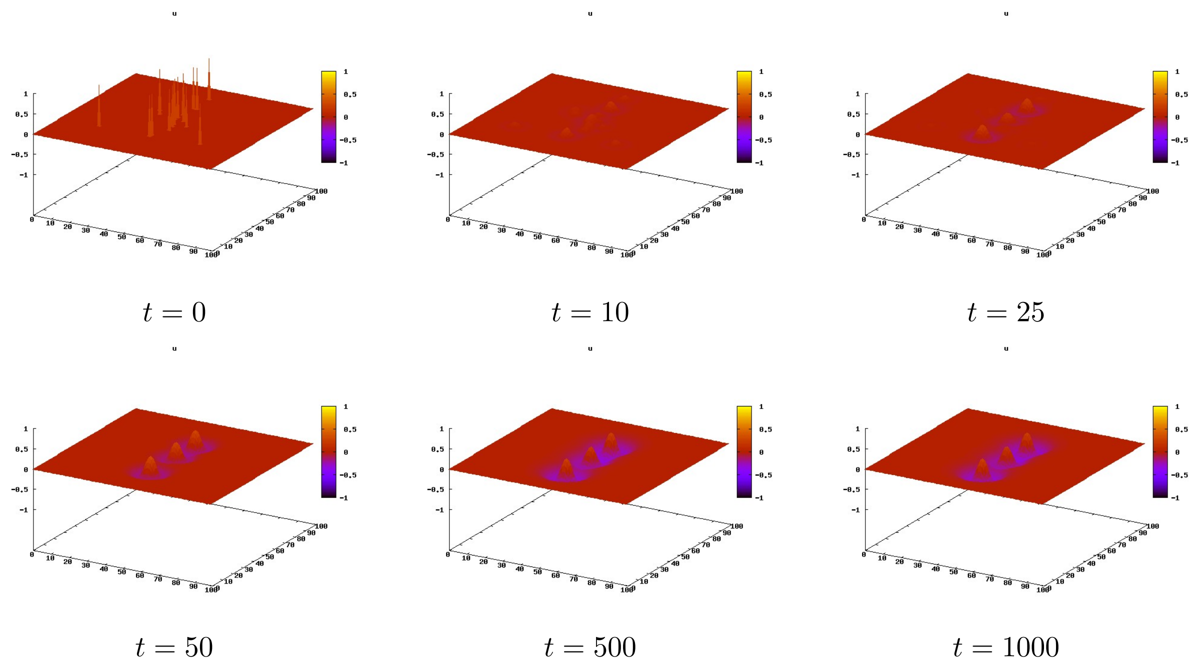


Figure 17: A simple Integro-Reaction-Diffusion system. A one layered neural field with low diffusion rate.

We can see the development of that system in figure 22. Starting from an initial condition, the system is going to self-organize to a traveling wave, which is going to split into a wave with a corner, which is moving aout of the domain. The other wave, keeps stable and forms a moving spiral.

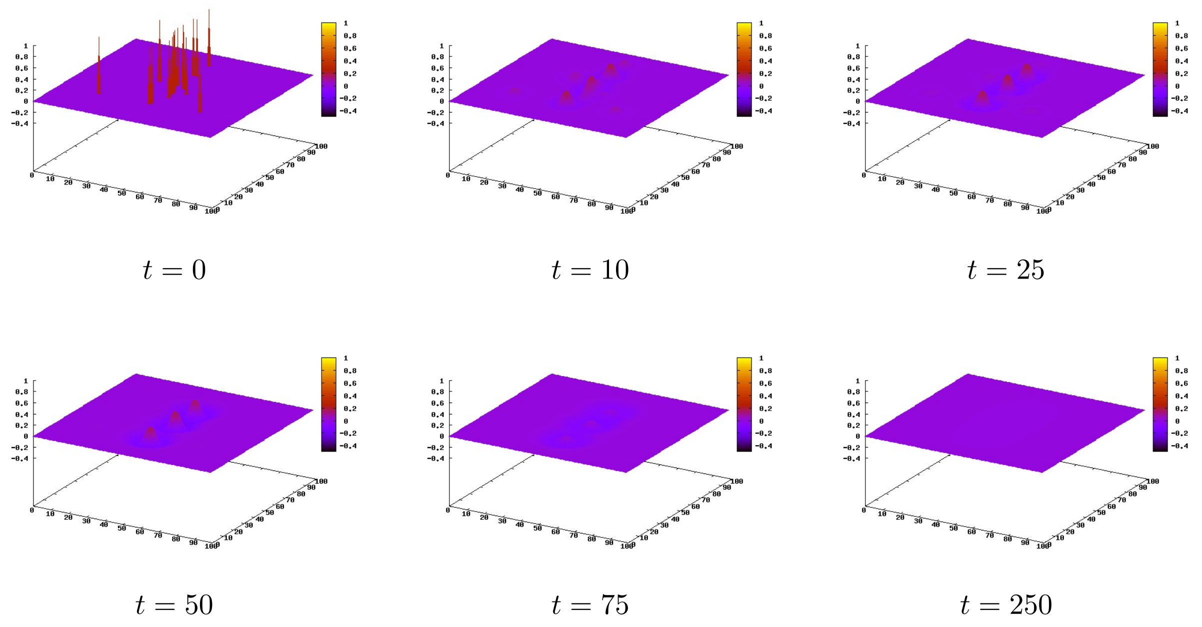


Figure 18: This neural field is very simular to the field plotted in figure 17 this field has got a higher diffusion rate.

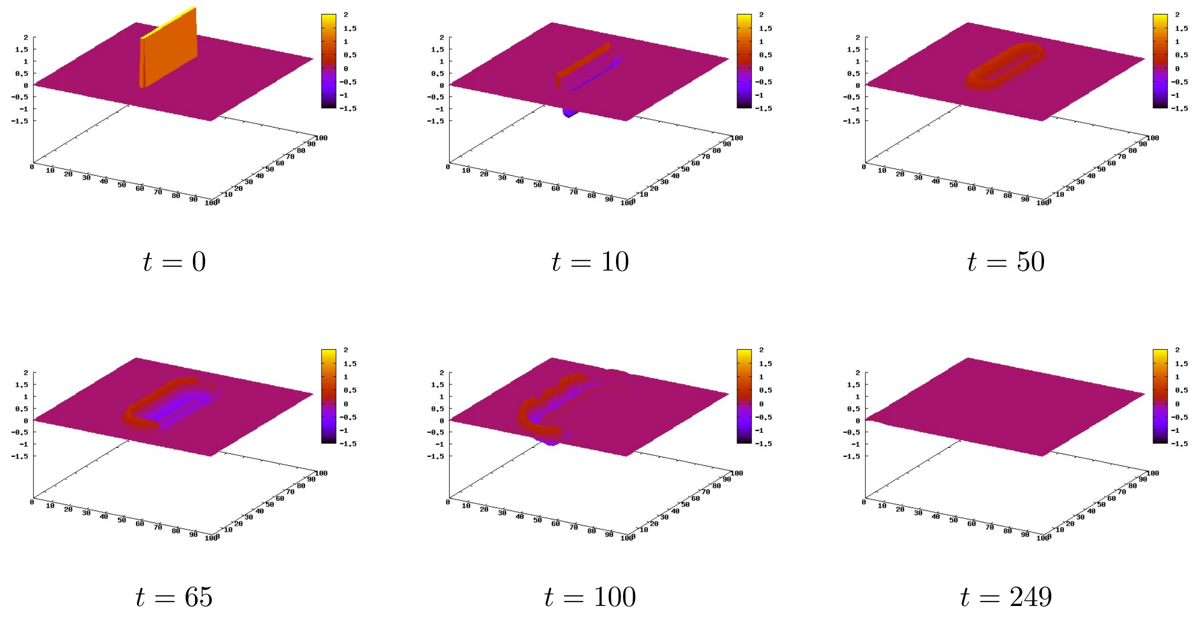


Figure 19: Plots of a two layered neural field, generating a single wave front, that is splitting.

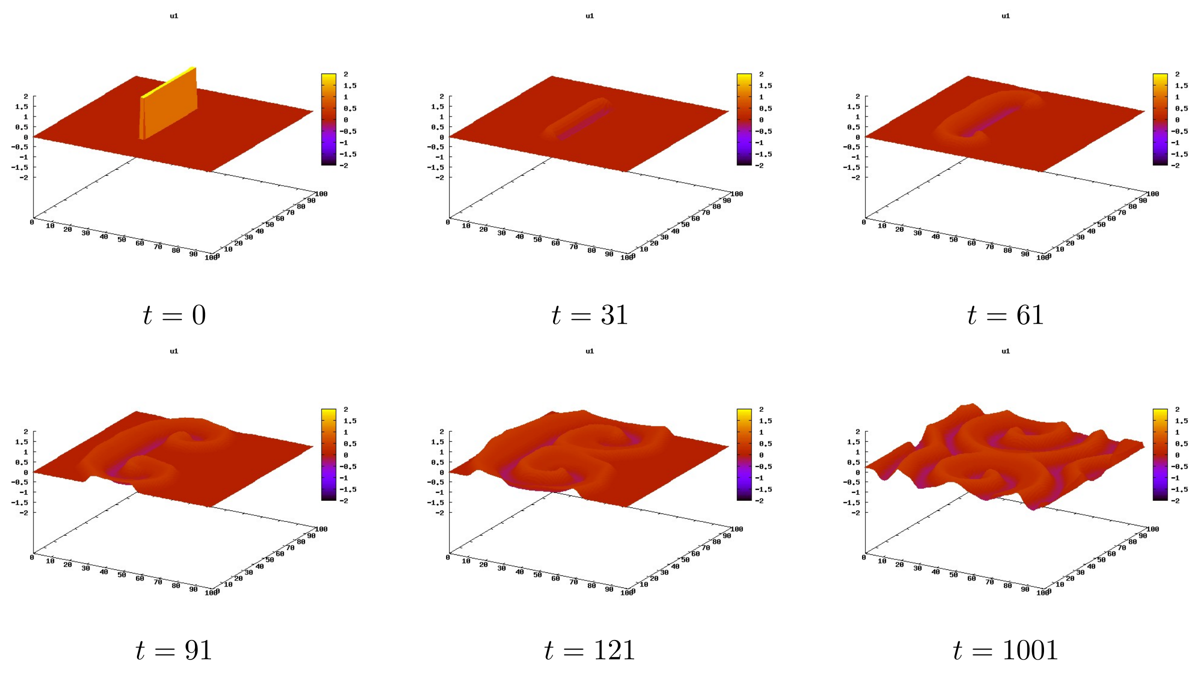


Figure 20: Plots of a two layered neural field, generating a stable pattern of oscillating traveling waves.

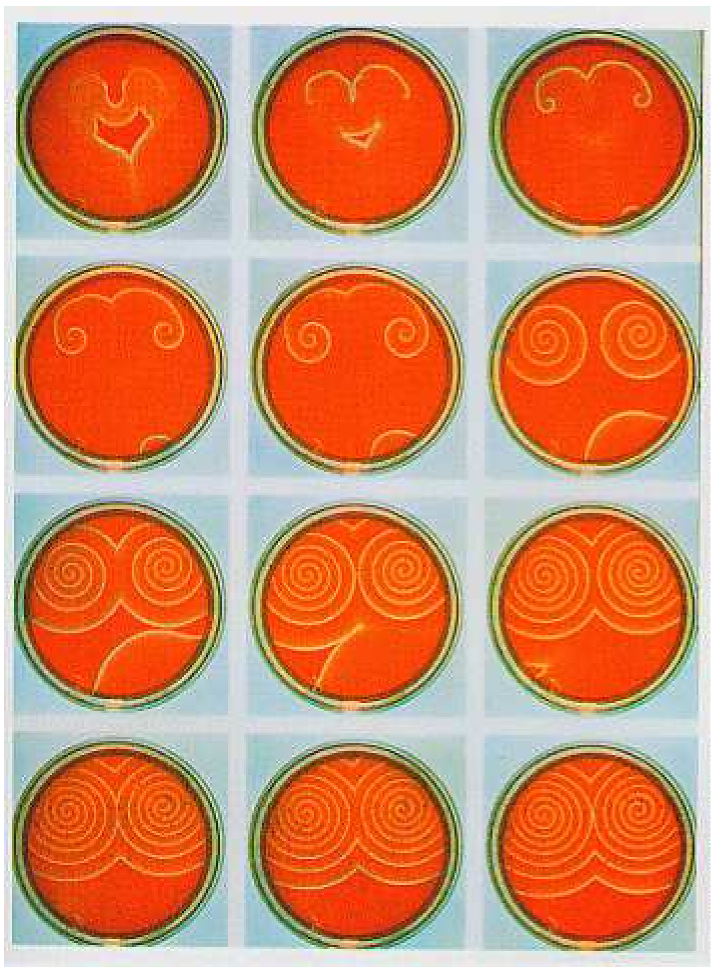


Figure 21: A petri dish with a Belousov-Zhabotinskii reaction.

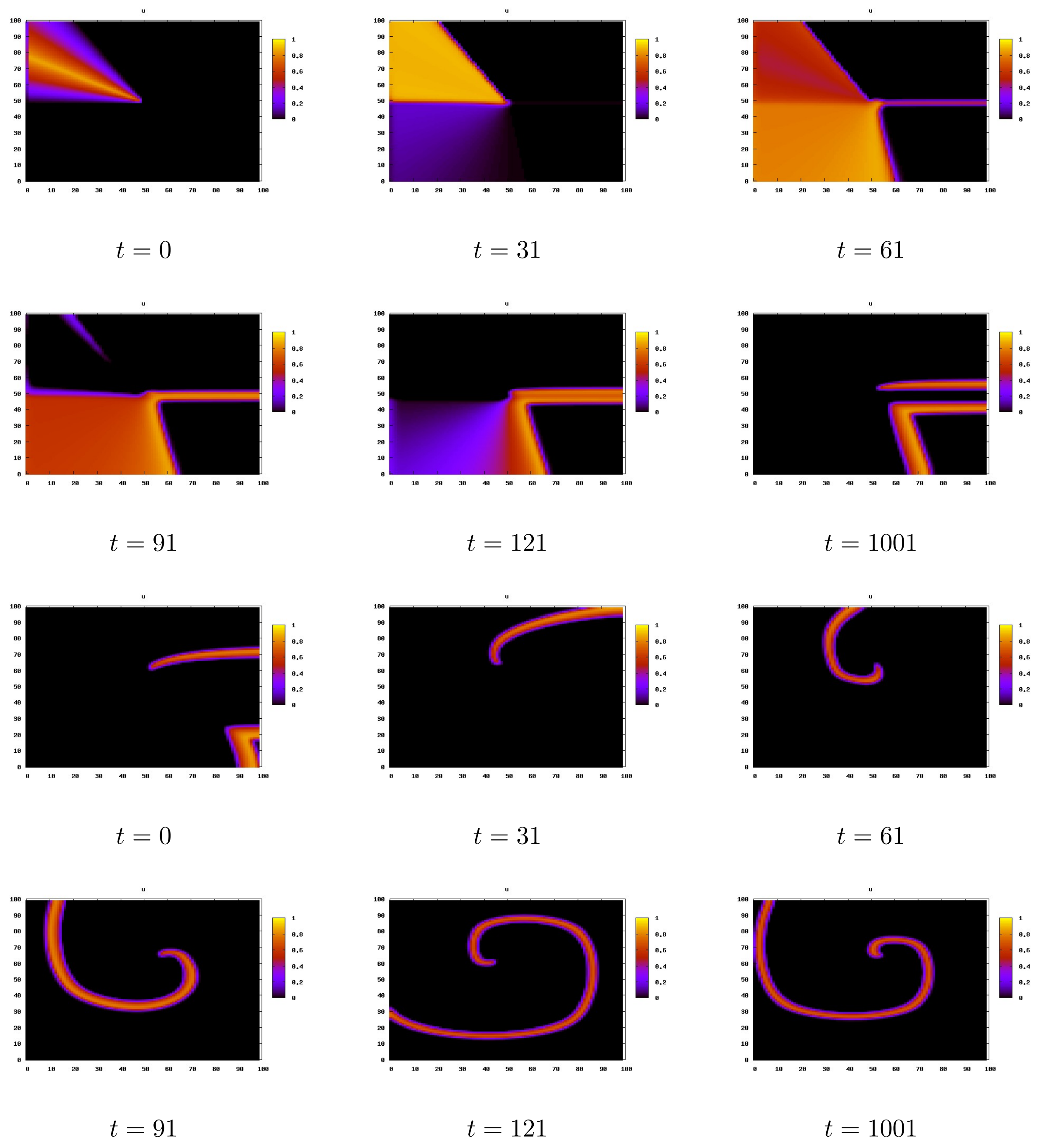


Figure 22: plots of the Belousov-Zhabotinskii reaction

6 Conclusions

It has been shown that the study of biological pattern formation is a important contribution to the design and development of adaptive and selforganizing control systems. Especially in the field of robotics, it may be interesting to unify flexible oscillater used in locomotion with self-organization and learning for higher order tasks that include action perception loops. Some models even suggest that embodied dynamics is a promising tool model an agents behavior [Esther Thelen, 2000].

The beauty and power of the dynamical approach in contrast to a merely computational approach based on a finite state automaton is that it has virtually an infinity of states and that transitions between states are continuous and smooth. Subsystems like described above could be coupled and continuously meshed. For example could the variable that determines the gait (running, walking) in the dynamical system that generates the oscillation, directly be read out in a dynamical system that controls a higher level behavioral variable in target aquisition for example.

There are to main classes of dynamical system for which the emergence of dynamical pattern has been shown and for which the conditions in which those system generate pattern are more or less understood. The two systems, neural fields and reaction diffusion systems have been described in section 2.8 and 2.7 respectively. In both systems, local self-excitation and lateral inhibition seems to play a crucial role in the emergence of stable pattern.

The theoretical analysis of dynamical system in extended media including spatial derivate are very hard. One must try to get a more intuitive feel of such systems. Search for stable pattern in simulations by showing that the simulated system does not explode to infinity or collapse to zero is hard and time consuming. Furthermore the existence of several stable attractors is to be shown in order to use the dynamics for interesting computational tasks.

`PantaRhei` provides a simple language for the specification of dynamical systems restricted to the class of reaction diffusion systems extended through a convolution term. `PantaRhei` makes it possible, to test the behavior of such classes of dynamical systems in a fast and simple manner. The user can instantly get a visualization of the dynamics and save them for documentations.

7 Outlook

As mentioned in section 6, `PantaRhei` is a tool, to be utilized for the analysis of pattern generating dynamical systems, such study is important for engineers that search for self-organizing control systems. Some authors even propose a new paradigm of amorphous computation which relies on a myriad of simple and eventually unreliable elements, that in forming pattern perform some kind of computations. For people interested in the engineering of such control systems the use and extension of `PantaRhei` can be interesting. In the following section some points of future extension of the software will be discussed.

7.1 Using `PantaRhei`

A complete reference manual is given in appendix B. Here the overall procedure for working with `PantaRhei` is given.
As systems that are simulated using `PantaRhei` are very sensitive to the choice of parameters such as initial conditions, resulutions and parameters of the model it is crucial in the begining to find a simple but working configuration. With "working" is meant, that the pattern generated are stable and interesting.
In a second step, one has to seek to change parameters,in order to alter the scale of the pattern or even the quality of the pattern, without loss of stability. This process can be automated by invoking `PantaRhei` through a pipe through another process or by calling it inside a shell script.
Once a interesting system is found and the range of parameters in which the system can bifurcate to another stable attractor, `PantaRhei` provides nice visualizing feature to prepare images for documents and demonstrations.

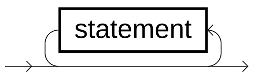
7.2 Extending `PantaRhei`

The simplest way to extend the possibilities of `PantaRhei` is to add new functions, that can be chosen either to form the shape of the kernel or the activation function for the convolution term. Functions can be declared and defined in the file `UserFunction.h` / `UserFunction.cc` respectively. Then, they must be added to the symbol table during the initialization of the symbol table.
Always when encountering a problem in the design and implementation of software, it is often unavoidable, for simplicity, to write a simpler and less general version. So `PantaRhei` could be extended in adding more generality to the dimensionality and size of kernels and layers. For some pattern to be generated it is sufficient to have just one dimension, but then where is it to be read in by the layer that has the it in its expression? It would be furthermore interesting to be able choose a different spatial (and maybe even temporal [11]) resolutions for each layer. One must be careful though, because more generality is likely to require more computations. On the other hand, the ability to specify smaller layers could remarkably reduce computational complexity.

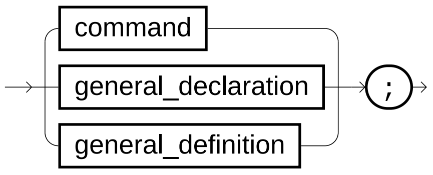
[11]see [J. Buchli,]

A The Grammar of Pantha Rei

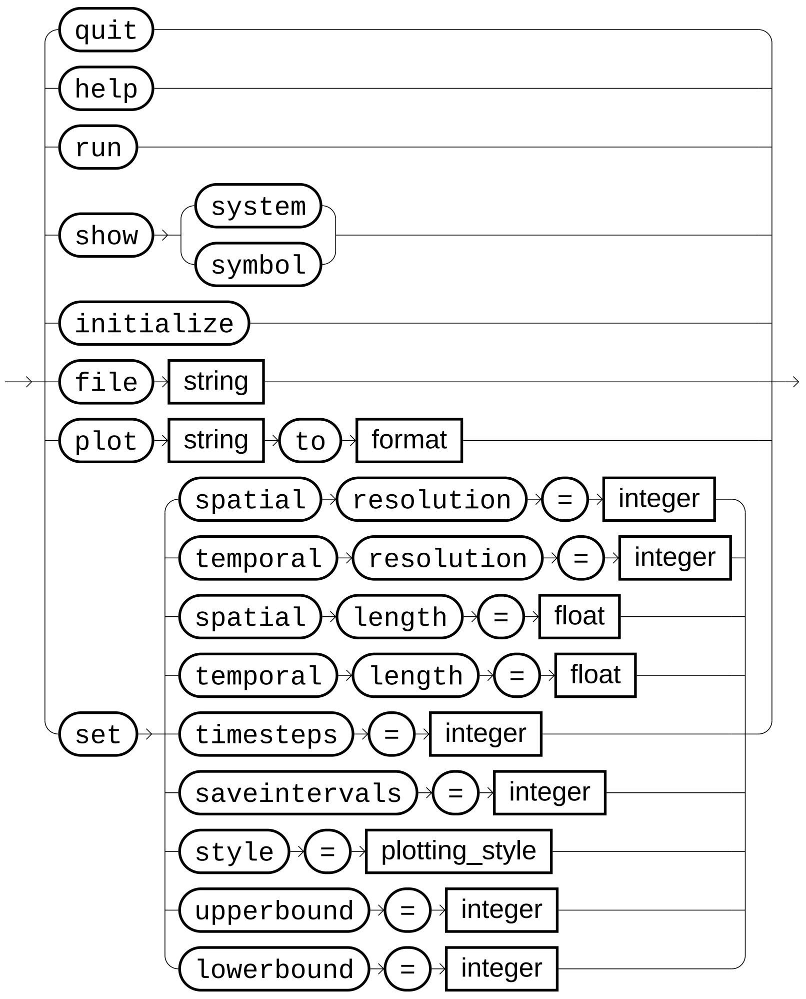
statements



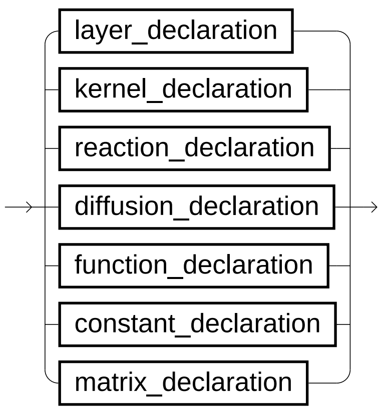
statement



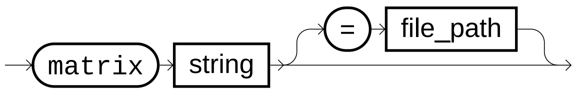
command



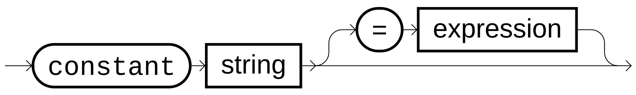
general_declaration



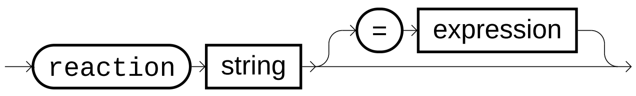
matrix_declaration



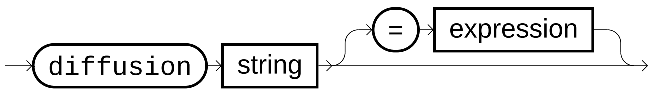
constant_declaration



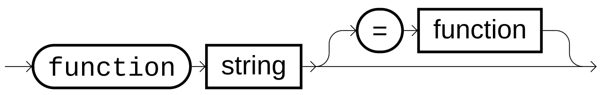
reaction_declaration



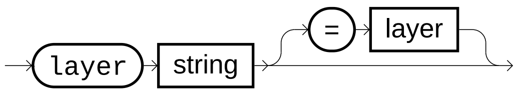
diffusion_declaration



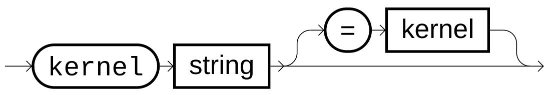
function_declaration



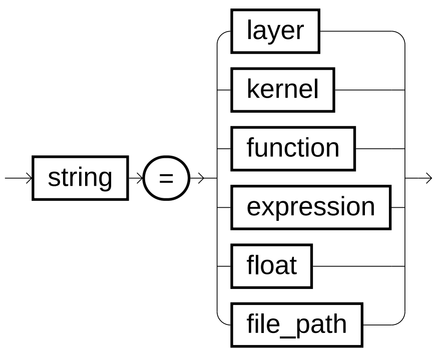
layer_declaration



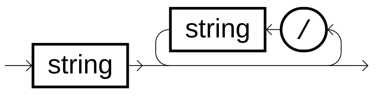
kernel_declaration



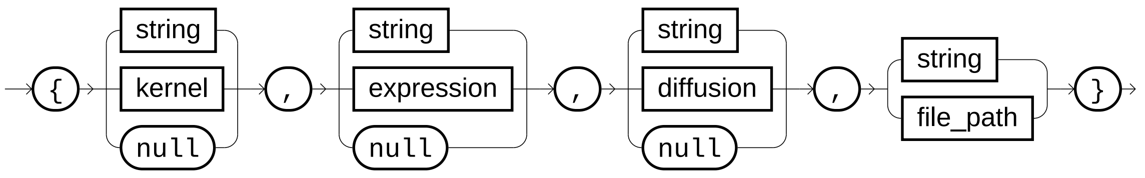
general_definition



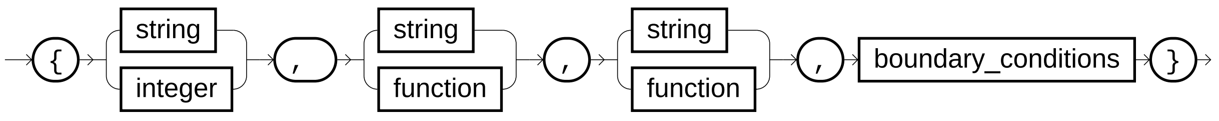
file_path



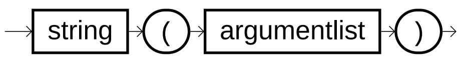
layer



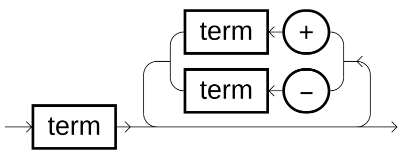
kernel



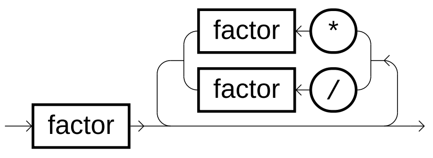
function



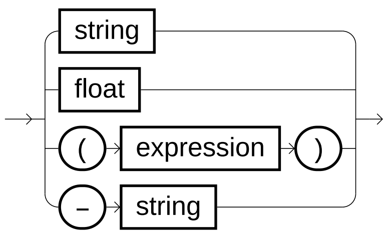
expression



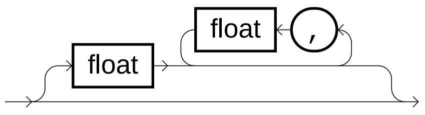
term



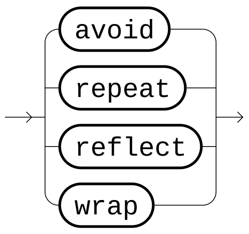
factor



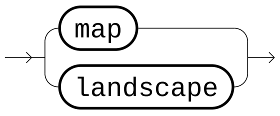
argumentlist



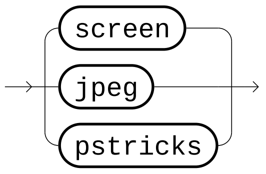
boundary_conditions



plotting_style



format



B Reference Manual

B.1 Requirements

`PantaRhei` was developed on Red Hat Linux distribution. To install and use `PantaRhei` the A++/P++ Array library [app, 2005] must be installed. Furthermore the Gnuplot plotting utility [gnu, 2005] must be installed with the pm3d extension [pm3, 2005].

B.2 Installing `PantaRhei`

A Makefile is provided with the package. Using the make-utility the complete source can be compiled. After compiling the source, move the executable one folder up the hierarchy. In your installation - folder you should then have a folder called data right beside the executable file. This is, where `PantaRhei` is storing the results and plots.

B.3 Starting `PantaRhei`

You can start `PantaRhei` with several options. You have to choose one of the following arguments, that determine the interface through which `PantaRhei` is accessed:

`-i`	Open `PantaRhei` in an interactive console.
`-f <filename>`	Open `PantaRhei` using a initialization file. If the file does not end with the command `quit`, `PantaRhei` will open the interactive console, after parsing the file.
`-p`	Access `PantaRhei` through a pipe.

Several optional arguments simply determine the level of verbosity with which `PantaRhei` should run.

`-d`	Run `PantaRhei` in debug mode. Every message is printed out.
`-v`	Run `PantaRhei` in verbose mode. All Warnings and other informations are printed.
`-l <filename>`	Use this option, to tell `PantaRhei` where to print the logger messages. If no file is given, a standard file is used.

B.4 Structure of a `PantaRhei` -file

Some examples files are provided with the package. They are quite intuitive, and can be used as boilerplate-code. Normally it is convenient, to organize a script using the following structure:

1. System properties:

```
set spatial length = <int>
set spatial resolution = <int>
set temporal length = <int>
set temporal resolution = <int>
set timesteps = <int>
set saveintervals = <int>
```

2. Plotting properties:

```
set style = <style>
set upperbound = <int>
set lowerbound = <int>
```

3. define and declare objects:

```
Constant <identifier> = <float>
Function <identifier> = <identifier> (<float> ...)
Reaction <identifier> = <expression>
Diffusion <identifier> = <float>
Kernel <identifier> = {<size>, <function>, <function>, <boundary condition>}
Layer <identifier>  = {<Kernel>,<Reaction>,<Diffusion>,<initial condition>}
```

4. Specify the structures you want to plot:

```
plot <identifier> to <format>
```

5. Often you want to display the system, initialize it and run the simulation:

```
initialize
show system
run
quit
```

References

[Adamatzky *et al.*, 2004] A. Adamatzky, P. Arena, A. Basile, R. Carmona-Galan, B. De lacy Costello, L. Fortuna, M. Frasca, and A. Rigriguez-Vazquez. Reaction-diffusion navigation robot control: From chemical to VLSI analogic processors. *IEEE Trans on Circuits and Systems*, 51(5):926–938, 2004.

[Amari and Arbib, 1977] S. Amari and M.A. Arbib. Competition and cooperation in neural nets. In J. Metzler, editor, *Systems Neuroscience*. New York: Academic Press, 1977.

[Amari, 1972] Sun-Ichi Amari. Characteristics of random nets of analog neuron-like elements. *IEEE Transactions on Systems, Man and Cybernetics*, (5):643 – 657, November 1972.

[Amari, 1977] Sun-Ichi Amari. Dynamics of pattern formation in lateral-inhibition type neural fields. *Biological Cybernetics*, (27):77 – 87, February 1977.

[app, 2005] A++ / p++. http://www.llnl.gov/casc/Overture/henshaw/documentation/App/manual/manual.html, 2005.

[Arena, 2000] P. Arena. The central pattern generator: a paradigm for artifical locomotion. *Soft Computing*, (4):251–265, 2000.

[Braitenberg, 1984] Valentino Braitenberg. *Vehicles*. MIT Press, Cambridge MA, 1984, 1984.

[D. Abrahams, 2005] A. Gurtovoy D. Abrahams. *C++ Template Metaprogramming*. Addison-Wesley, 2005.

[Edelmann, 1987] G. M. Edelmann. *Neural Darwinism*. New York: Basic Books, 1987.

[Esther Thelen, 2000] Gregor Schöner at al. Esther Thelen. The dynamic of embodiment: A field theory of infant perseverative reaching. *Behavioral and Brain Science*, 2000.

[Gaussier *et al.*, 1997] P. Gaussier, S. Moga, J. Banquet, and M. Quoy. From perception-action loops to imitation processes: A bottom-up approach of learning by imitation, 1997.

[Gelder, 1997] Tim Gelder. The dynamical hypothesis in cognitive science. *Behavioral and Brain Science*, 1997.

[Giese, 2000] M. A. Giese. Neural field model for the recognition of biological motion patterns. Paper, 2000.

[gnu, 2005] Gnuplot. http://www.gnuplot.info/, 2005.

[Hans A. Mallot, 1996] Fotios Giannakopolous Hans A. Mallot. Population networks: a large scale framework for modelling cortical neural networks. *Biological Cybernetics*, (75):441–452, 1996.

[Hans Meinhardt, 2000] Alfred Gierer Hans Meinhardt. Pattern formation by local self-activation and lateral inhibition. *BioEssays*, (22):753–760, 2000.

[Henderson, 2001] Thomas Henderson. Reaction-diffusion processes as a computational paradigm. 2001.

[Ijspeert, 2002] Auke Jan Ijspeert. Vertebrate locomotion. In M. Arbib, editor, *Handbook on Brain Theory and Neural Networks, 2nd Edition*. MIT Press, Cambridge, USA, 2002. to appear.

[Ioannis Iossifidis, 2001] Axel Steinhage Ioannis Iossifidis. Controlling an 8 dof manipulator by means of neural fields. Paper, 2001.

[J. Buchli,] A. Ijspeert J. Buchli. Implementation examples for multiscale dynamical systems. Poster.

[Jörg Wellner, 1998] Andreas Schierwagen Jörg Wellner. Cellular-automata-like simulations of dynamic neural fields. In M. Holcombe and R. Paton, editors, *Information Processing in Cells and Tissues*, pages 295–304. Plenum Press, New York, 1998.

[K.G. Kirby, 1986] M. Conrad K.G. Kirby. Intraneuronal dynamics as a substrate for evolutionary learning. *Physica D*, (22):205–215, 1986.

[lam, 2005] Lam / mpi. http://www.lam-mpi.org/, 2005.

[ove, 2005] Overture. http://www.llnl.gov/casc/Overture, 2005.

[Philip K. Maini, 1997] Helene N.P. Chau Philip K. Maini, Kevin J. Painter. Spatial pattern formation in chemical and biological systems. *J. Chem. Soc.*, 20(93):3601–3610, 1997.

[pm3, 2005] pm3d. http://www.sci.muni.cz/ mikulik/freeware.html#pm3d, 2005.

[Quinlan, 2000] Daniel Quinlan. *A++/P++ Manual (Version 0.7.5)*. Lawrence Livermore National Laboratory, August 2000.

[Rambidi, 2000] Nicholas G. Rambidi. Lure of effective neural net implementation: Roots and promises of information processing by reaction diffusion systems. Physics Departement, Moscow State University, 2000.

[Rees, 2003] David Rees. Turing's other machine: Adventures in reaction-diffusion engineering. January 2003.

[Schöner, 2000] Gregor Schöner. Dynamical systems approaches to neural systems and behavior. In *Int. Encycl. Social and Behavior Sciences*. January 2000.

[spi, 2005] Boost spirit library. http://spirit.sourceforge.net/, 2005.

[Steinhage and Bergener, 1998] A. Steinhage and T. Bergener. Dynamical systems for the behavioral organization of an anthropomorphic mobile robot, 1998.

[Steinhage, 1997] G Steinhage, A. Schöner. The dynamic approach to autonomous robot navigation. *Proceedings ISIE'97*, 1997.

[Steinhage, 2000] Axel Steinhage. Dynamic neural fields for robot control. Paper, October 2000.

[stl, 2005] Standard template library (stl). http://www.cppreference.com/cppstl.html, 2005.

[Turing, 1952] Alan Turing. The chemical basis of morphogenesis. *Philos. Trans. Roy. Soc. London Ser. B*, 327:37, 1952.

[uni, 2005] Center for biomimetics and natural technologies, university of bath. www.bath.ac.uk/ ensab/TRIZ, 2005.

[Valentino Braitenberg, 1990] Almut Schüz Valentino Braitenberg. *Anatomy of the Cortex: Statistics and Geometry (Studies of Brain Function)*. Springer Verlag, Hamburg, 1990.

[Weimar, 1997] J.R. Weimar. Cellular automata for reaction diffusion systems. *Parallel Computing*, 11(23):1699–1715, December 1997.

[yac, 2005] Yacc & lex. http://dinosaur.compilertools.net/, 2005.

Index