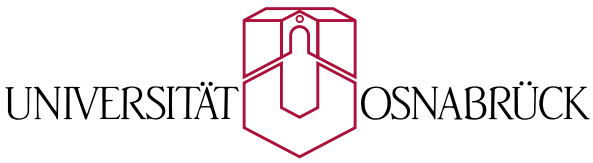


Institute for Computer Science
Neuroinformatics

PhD Thesis

Self-Regulating Neurons: A model for synaptic plasticity in artificial recurrent neural networks

Keyan Mahmoud Ghazi-Zahedi

January 2009

First supervisor: Prof. Dr. Frank Pasemann
Second supervisor: Prof. Dr. Martin Riedmiller

To Linus and Maxim

Abstract

Robustness and adaptivity are important behavioural properties observed in biological systems, which are still widely absent in artificial intelligence applications. Such static or non-plastic artificial systems are limited to their very specific problem domain. This work introduces a general model for synaptic plasticity in embedded artificial recurrent neural networks, which is related to short-term plasticity by synaptic scaling in biological systems. The model is general in the sense that is does not require trigger mechanisms or artificial limitations and it operates on recurrent neural networks of arbitrary structure. A Self-Regulation Neuron is defined as a homeostatic unit which regulates its activity against external disturbances towards a target value by modulation of its incoming and outgoing synapses. Embedded and situated in the sensori-motor loop, a network of these neurons is permanently driven by external stimuli and will generally not settle at its asymptotically stable state. The system's behaviour is determined by the local interactions of the Self-Regulating Neurons.

The neuron model is analysed as a dynamical system with respect to its attractor landscape and its transient dynamics. The latter is conducted based on different control structures for obstacle avoidance with increasing structural complexity derived from literature. The result is a controller that shows first traces of adaptivity. Next, two controllers for different tasks are evolved and their transient dynamics are fully analysed. The first is a controller solving the standard benchmark problem of pole balancing. The second is a controller performing light-seeking under varying ambient light conditions. In the second experiment, a light source cannot be distinguished from ambient light in the raw sensor data. The task is solved by the homeostatic property of the neuron model and the interaction of the robot with its environment.

The results of this work not only show that the proposed neuron model enhances the behavioural properties, but also points out the limitations of short-term plasticity which does not account for learning and memory.

Acknowledgements

It is my strong belief, that science and scientific achievement, although usually assigned to only one or a few individuals, are the result of good teamwork. This is why these acknowledgements are so important to me.

There is absolutely no doubt to whom I dedicate my first lines. Frank Pasemann, you have not only been my thesis advisor, supervisor and department head, but throughout the entire time, you have been to me a true *Doktorvater* (PhD-father) just as the German word expresses it. I will always remember, with great joy and no less gratitude our close and intense relationship with its ups and downs which have made it so valuable. I was very lucky to share the scientific, as well as the more personal, discussions with you, and also lucky for the opportunity to participate in the formation of the restructured INDY department. With your critical view on my work, you have constantly pushed me further. Thank you very much, it was *my* pleasure.

Next, I thank Martin Riedmiller, my second advisor, for accepting me as his PhD student, as well as for his patience and feedback.

For the majority of the time that I was working on this thesis, I was a research fellow at the former Fraunhofer Institute for Autonomous Intelligent Systems in Sankt Augustin, Germany, now the FhI for Intelligent Analysis and Informations Systems. I have to thank the leading team of Thomas Christaller, Stefan Wrobel and Marta Kreuzová for creating the work environment which makes the completion of this thesis so special to me.

My gratitude also belongs to Nihat Ay, of the Max-Planck Institute for Mathematics in the Sciences in Leipzig, Germany, who offered me the great opportunity to finish my thesis at the MPI MIS. From the MPI MIS, I also thank Jürgen Jost for inviting Frank Pasemann and myself in 2003, during which time the first formulation of the Self-Regulating Neuron was found.

As I mentioned at the beginning, science is teamwork. During my time at the FhI AIS, I was very lucky to be part of a great team. Martin Hülse and I were the first staff members when Frank Pasemann led the restructuring of the INDY department. I remember, with great pleasure, the first years during which we shared one office and wrote ISEE, a software package initially designed for our own experiments, which turned out to be the technical basis for half a decade of research in the INDY team. In no time, we grew from colleagues to great friends, sharing some rough moments but even more great ones, which make such a relationship lasting and memorable. The team of two soon grew to include Björn Mahn and Steffen Wischmann. The four of us gave INDY the face and good reputation it was respected and envied for throughout the institute. We worked together as friends, not just colleagues, to achieve something bigger. It was a perfect match and a great team, both on and off the field. A team, which was soon completed by Arndt von Twickel.

I have to thank many colleagues at the FhI AIS for creating a warm and familiar atmosphere. Three need special mention, Ralph Breithaupt, Karl-Heinz Sylla and Thomas Wisspeintner. Ralph and I, although in different departments, participated in many discussions on a wide variety of topics, both personal and professional — something which helped a lot and which I certainly miss. Karl-Heinz Sylla, Ralph and I, together worked and headed, as a team, some very intense projects. I do not want to miss these intensive times, although I would not want them back. To Karl-Heinz Sylla, I owe all my knowledge of software engineering. I will always remember his kindness, even in the most stressful times. There was always time for a chat, a cup

of tea, and a sympathetic ear. Thomas Wisspeinter and I also accompanied each other during my entire time at the FhI AIS. Unforgettable are our many inspiring and creative breaks. I am proud to call them my friends.

My other dearest colleagues, to whom I owe a lot, are Eva Sommer, Hedi Szameit, Peter Schöll, Stefan Kubina, Herman Streich, Joachim Hertzberg, Erich Rome, Paul-Gerhard Plöger, Fotios Giannakopolous, Herbert Jäger, Stefan Härtig, and many more.

This work would also not be possible without the help of people outside these institutes. First of all my family, my parents Mahmoud and Silvie Ghazi-Zahedi, my sister Natasha, my grandmother Edith Wills and my dear uncle Ahmad Ghazi-Zahedi. They have all been of great support and have shown endless patience with me. I have to mention my father again, who as an artist, gave me valuable suggestions on the first images that I created for this thesis.

I am lucky to have very close friends, who I have known for many years. Olaf, Boris and Ilona own their share of this thesis. I now know Olaf Peisker for over two decades. You are like a brother to me. I met Boris Diebold during my time as a student, about 10 years ago. Without you, I would not have pushed myself this far to reach what I have reached. Ilona Wienecke, a very close friend of more than 15 years, has been a very valuable support. Thank you very much for your openness. This makes a true friendship.

I thank Rainer Grossmann for his support on many evenings when I started my night shifts, as well as the many enjoyable and insightful discussions on topics outside of my scientific field. It has helped me to maintain a larger perspective. Julius Popp earns my gratefulness. He has been a great support since I moved to Leipzig. Not only did he give me the warmest welcome, but also a great opportunity to work in art projects, and to use his atelier to continue writing my thesis in the evenings.

From the MPI MIS I thank Thomas Kahle, Wolfgang Löhr, Bastian Steudel, Susanne Schindler, and Antje Vandenberg and my other colleagues at the MPI MIS, for their support in making it especially easy for me to feel at home here very quickly.

For revising previous versions of different chapters of this thesis, I thank Susanne Schindler, Ralph Breithaupt, Irene Markelić, and Mario Negrello.

My special thanks go to Iman Awaad for her never-ending encouragement, many nice discussions, the pleasure of working together in the XPERO project, and last but not least, for her endurance to read my entire thesis, and improve its readability with many very valuable comments.

This work was partly funded by DFG grants PA 480/4, CH 74/9, and a PhD scholarship of the independent research group ,,Information Theory of Cognitive Systems Group" of the Max Planck Institute for Mathematics in the Sciences.

Contents

Chapter 1

Introduction

> *I am only a relation among my parts that are perceived while they are in relation to each other. But these parts are in turn divisible into other relations (and so on), therefore every system of relations, being aware of itself, being indeed the awareness of self, is a thinking nucleus. I think me, my blood, my nerves; but every drop of my blood thinks itself.*
> *Does it think itself as I think me? Surely not* [...] *Every thing thinks, but according to its complexity.*
>
> *Roberto de La Grive, main character in Umberto Eco's book* The island of the day before (Eco, 1998)

In the quote given above, the main character thinks about himself and what makes him conscious. His conclusion is that his consciousness is the result of the interaction of all the parts of his body and cannot be assigned to any subset of it. The questions of what consciousness and intelligence are and what leads to them remain unsolved and are therefore still of great interest, in science as well as in literature. The quote taken from literature describes well, also if not scientifically, the context in which consciousness and intelligence are understood in this work. This context will be clarified throughout and summarised at the end of this introduction.

This work will not discuss the terminology and will not give a definition of consciousness or intelligence. In this work they are used as abstract concepts (Brooks, 1991b). The focus is on finding general principles of neural signal processing. This is in accordance to Förster (1993, 2003), who describes consciousness as a very basic process that transforms sensory data into motor commands using a form of internal (non-symbolic) representation.

There are many and diverse fields of science that try to find answers to the question of how the brain functions. A short overview of the history of sciences related to the field of neuro-informatics is given in the second chapter of this work. The field which is most closely related to this work is artificial intelligence, the science of modelling human intelligence or human problem-solving strategies commonly labelled as intelligent. The field dates back to the late 1950s. At that time, intelligence was understood as a rule-based process operating on symbols. The rules were designed by the scientist to model specific problem-solving strategies, e.g. playing chess. The symbols were representations of objects, their properties and their relations to each other. The objects could be real-world objects, such as a chair, but for simplicity most often simulated

worlds with very limited input and output space, so called micro-worlds (A. Clark, 1996) or block-worlds (Brooks, 1991c) were used. In the example of chess, the symbols encode the type of the chess piece, where it is located on the chess board, and how it can move from thereon. In current robot applications, the notion of objects and object properties in some applications is replaced by Gibson's concept of affordances (Gibson, 1977). Affordances are defined as all action possibilities latent in the environment.

The most prominent solution in the field of classical artificial intelligence in the early time was the General Problem Solver (GPS) (Newell & Simon, 1963). Early success of GPS in automated theorem proving and in solving problems, such as the Towers of Hanoi was reason enough to believe that human thinking has been fully understood and successfully modelled. GPS was a heuristic search algorithm on trees spanning a specific problem-space. It was soon clear that GPS had strong limitations. It could not handle uncertainties and ambiguities. In addition, scaling from the micro-worlds to larger problem-domains, such as e.g. chess, showed its limitations. It was the advances in computational power and the improvements to the unchanged algorithm in chess competitions that led to the belief, which is still popular today, that it is only a matter of time until artificial intelligence becomes viable as further advances in computational power fulfil the enormous computational requirements of GOFAI[†].

It was discussed early in this field, that learning was essential to improve the algorithms, but little or no effort was put into it. In the field of artificial neural networks, learning methods were used for automated pattern recognition and clustering of input data. The results were often misleading as wrong clusters were found. To avoid this, a lot of preparation and very careful selection of the training and evaluation data was required by the experimenter. Even when trained, the solutions were not good at generalising from the training and evaluation data sets to completely new data sets. In comparison, generalisation is performed very well by humans. Although many impressive engineering results were achieved in this field of research, not much contribution was made to the initial claim of understanding and modelling human intelligence. The statements given above apply not only to the beginning of artificial intelligence, but are as relevant to today's GOFAI approaches, which face the same problems.

In the 1980s, a new field of research emerged which was initially called Behaviour-Based Robotics (Brooks, 1986, 1991c, 1991b) and later reformulated as Embodied Artificial Intelligence (Pfeifer & Scheier, 1999; Pfeifer & Bongard, 2006). This field proposed a very different approach. Intelligence was now understood as a process which must be embodied and situated, hence it only occurs within the sensori-motor loop (Cliff, 1990). Many impressive results were produced in a comparably short time, e.g. the first autonomous robots acting not in artificial micro-worlds or laboratories, but in unstructured office environments (Brooks, 1986, 1989, 1990). Other experiments showed how an intelligent design of the body reduces the amount of required control (Maris & Boekhorst, 1996) or even does away with the need for any control at all (McGeer, 1990; Wisse, 2004). The first cited example by Brooks is an autonomous robot that operates in office environments, collecting empty soda-cans and an autonomous walking machine, both only using locally available, and hence comparably weak computational resources. The second example by Maris and Boekhorst is a group of simple two wheeled robots, with two infra-red distance

[†]GOFAI is an abbreviation for Good Old Fashioned Artificial Intelligence and was first used by Haugeland (1985)

sensors, operating in a bounded environment with moveable boxes. The motors are strong enough to push a single box, but too weak to push two boxes. Depending on the arrangement of the distance sensors, a group of such robots rearrange the cubes in heaps through the use of a simple obstacle avoidance behaviour and a form of collision detection measuring how many boxes are pushed. The grouping of the boxes into heaps was not encoded anywhere in the control program, but is the result of the simple behaviour control and the morphology of the robot. The third example by McGeer is the passive dynamic walker – a device that walks down the slope without the need for any controller and using instead the dynamics of the body and the force of gravity. Wisse rebuilt such passive dynamic walkers and also introduced very limited actuation to compensate for the loss of kinetic energy that occurs during walking, which was previously evened out by the force of gravity and the slope.

This new approach to artificial intelligence provided many insights about how embodiment and situatedness influence behaviour and how careful design can reduce the complexity of the behaviour control system. This was also demonstrated by Braitenberg (1984) with his *Gedanken-experiments* in which he showed with various vehicles how behaviour, which appears to be complex to an observer, is the result of very simple control structures. His vehicles are related to neuro-biological findings.

Despite these early and convincing successes, solutions in these fields were mostly biologically-inspired pre-programmed solutions. As a result, good and simple solutions for previously difficult problems (as the control of an autonomous walking machine in an unstructured environment) were found. Approaches, such as Brooks' subsumption architecture (Brooks, 1986) are however weakly related to biological nervous systems and therefore do not provide new insights about the fundamental principles of neural signal processing.

This work starts exactly at this point. It is not just the behaviour of an embodied and situated agent which is of interest, but, more specifically, how the behaviour results from generalisable principles of neural signal processing. As Malsburg (1981) puts it, there is every reason to believe in the existence of general principles governing the function of the brain. The assumption is that findings in the structure–function relationship of artificial recurrent neural networks controlling an autonomous robot in the sensori-motor loop will relate to biological findings and as a next step even provide new insights, which are novel in the science of biology. How well the results will relate to biology depends on the chosen neuron model (which is discussed below). Robotics is here practised not as an engineering discipline, but as a method to find and understand general principles of neural signal processing.

The appropriate neuron model depends on the chosen level of abstraction. For biological nervous systems such as the brain, many different description levels are available. This begins at the highest level of detail, modelling the bio-chemical processes involved in signal transmission over a synapse and in changes of the morphology of a neuron. The highest level of abstraction might be defined where brain areas are understood as interacting functional units. In this work, basic principles are of interest, not the bio-chemical mechanisms which lead to them. This is in correspondence to the field of cybernetics (Wiener, 1948), which is the science of describing systems at an abstract level. Animals and machines are considered and treated equally as systems. It is the laws that determine their behaviour that are of interest, and not how they are related to detailed internal biological or technical mechanisms.

To determine an appropriate level of abstraction, the point of interest of the brain must be

clarified. The brain consists of a vast number of neurons and synaptic connections between them. Recurrent connections and the non-linearity of the input-output response of biological neurons are the basis for the assumption that brains are well described as non-linear dynamical system. Findings in humans justify this assumption as dynamical features such as chaos (Sarbadhikari & Chakrabarty, 2001; Faure & Korn, 2001) and hysteresis (Eckmiller, 1974; Kleinschmidt et al., 2002) are observed. These types of dynamics are understood as general principles of neural signal processing, if they produce behaviours in the sensori-motor loop. Hence, a minimal model must be chosen that is able to expose these dynamical effects. The standard additive neuron model with a non-linear transfer-function shows the desired dynamical features already in single neurons with a recurrent connection and small neuro-modules (Pasemann, 1993, 2002). Hence, it is the chosen level of abstraction in this work.

To avoid the shortcomings of classical artificial intelligence, neural systems are embedded and situated, acting in the closed sensori-motor loop. Autonomous systems of this kind operate in uncertain or even partially unknown and dynamic environments. In this context, it is assumed that no teacher, teaching signal or training and evaluation data sets are available during the lifetime of an autonomous agent. Hence, it must have the ability to adapt in a self-sustaining manner to changing properties in the external world and its own body's properties.

Therefore, this work focusses on self-organising properties of recurrent neural networks in the sensori-motor loop. Learning and adaptation are the result of locally interacting Self-Regulating Neurons coupled in a (recurrent) neural network. A neuron is now a three dimensional system with two new intrinsic properties, which are referred to as the transmitter and receptor strength. Each neuron modulates its input and output as a result of the neuron-intrinsic properties. This is the concept of homeostasis, first introduced by Cannon (1932) and later proposed by Ashby (1954) as a general principle for adaptive systems, which Ashby demonstrated in his machine called the Homeostat. Currently, the target value is of arbitrary choice, but as long as one is interested in non-linear dynamical effects, there is a canonical choice for it.

This work follows the idea of Behaviour-Based Robotics that fully self-sustaining robots and behaviours of lower complexity must be built and understood first, before higher levels of complexity are taken into account (Brooks, 1991c). Thus, the most basic form of adaptivity discussed in biology is modelled. It is short-term plasticity (STP) by synaptic scaling. STP is defined by the duration of changes of synaptic strength as a result of sensory input. The length of time of the STP effect after the triggering stimulus is presented is of the same order of magnitude of time as the activating stimulus, and its duration is in the range of seconds. Synaptic scaling refers to a biological mechanism where all incoming synapses are modulated in their strength by the post-synaptic neuron.

The Self-Regulating Neuron (SRN) model is analysed with respect to its dynamical properties, decoupled from the sensori-motor loop. This leads to an understanding of the attractor landscape and reveals that the SRN model exposes the desired property of homeostasis. Embedded, a network is constantly driven by external stimuli. The neurons will therefore, in general, not converge to their asymptotically stable state. Consequently, the next step is to analyse the transient dynamics of the model while a SRN network controls a robot in an environment performing a simple negative tropism task of obstacle avoidance. The result of the analysis leads to an understanding of the relationship between the plasticity parameters of the SRN model and the behaviour relevant transients of the systems. From the results of the analysis, an obstacle

avoidance controller is constructed that performs comparably well to the best-known minimal static neuro-controller, but requires less structural complexity. It solves the task by adaptation of the synaptic connections to different encountered environmental conditions.

To compare the SRN model with other static structures, a solution for a standard benchmark problem for control theory is generated by artificial evolution. The controller performs equally well compared to static neuro-controllers for the pole-balancing task known from literature. Next, the model is evaluated for adaptivity and robustness. In a light-seeking experiment (positive and negative tropism), a robot has to find a light source for varying ambient light conditions. It is only equipped with two light intensity sensors, and cannot distinguish between a light source and an ambient light in the raw sensor data. The result of the experiment will show a pure feed-forward structure of only one layer that solves the task, utilising the homeostatic property of the SRN model and the environment.

This thesis is organised as follows: The following chapter (chap. 2: Background) presents a short overview of the history of neuro-informatics and details the scientific background of this work. This thesis is at the juncture of cybernetics, robotics, and biology, and the contribution of each field is discussed.

The third chapter (chap. 3: Methods) introduces the methods used in the remainder of this work. Dynamical systems theory gives the mathematical framework to describe, model and analyse the basic principles of neural signal processing. Artificial neural networks with the standard additive neuron model are motivated based on the behavioural properties of biological neurons, and finally, artificial evolution is proposed as an algorithmic method to construct recurrent neural networks of arbitrary structure.

The fourth chapter (chap. 4: Self-Regulating Neuron Model) introduces the Self-Regulating Neuron model and presents analytical and numerical analyses of single neurons and small neuro-modules. It concludes with a comparison of related learning mechanisms for artificial neural networks.

The fifth chapter (chap. 5: Experiments on Plasticity Parameters) discusses the transient dynamics of neuro-controllers in the sensori-motor loop on the basis of different obstacle avoidance controllers of increasing structural complexity with respect to variations of the plasticity parameters. The result is a controller that already shows first traces of adaptivity.

In the sixth chapter (chap. 6: Artificial Evolution of SRN-Controllers) two controllers which have been evolved for different tasks are fully analysed with respect to their behaviour-relevant dynamics. The result is a controller that demonstrates the adaptivity property of the SRN model.

The experiments show that adaptivity increases the behavioural properties of robots in uncertain environments, but that there are also limitations, determined by the chosen level of plasticity. STP only allows adaptivity. For learning and memory, different mechanisms, which are mainly related to structural changes, are required. These limitations and possible next steps are discussed in the last chapter (chap. 7: Discussion).

The appendices presents the software tools which were developed during this thesis (app. A: ISEE) and how they can be extended for other experiments (app. B: Howto's). With the exception of the mathematical solution of the stability analysis of the SRN model (app. C: SRN Model Stability Analysis), all results presented in this thesis were obtained with ISEE.

To close the introduction, the quote given at the beginning is rephrased with the terminology

of the fields of research discussed above. The quote pointed out that consciousness is the result of the interaction of every part of the body. In the terms given in this introduction, this means that from the sensors, which perceive an environment, the nervous system that collects and processes the sensor signals, to the actuators which drive the body in an environment, and the body itself (the morphology, denoting the shape of the body and the arrangement of the sensors and actuators), everything takes part in the process, and consciousness can not be understood if one of them is removed. Following the approach of first fully understanding lower levels of complexity before approaching higher levels, this introduction concludes with the last sentence of the quote, that each thing thinks, but with respect to its own complexity.

Chapter 2

Background

The question (and particularly the answer to it) of how the brain functions and how its ability to learn and to adapt is achieved depends highly on the chosen perspective. For an impression of the diversity, consider the following arbitrarily chosen examples taken from Malaka and Spitzer (2006) which are not meant to be comprehensive. Developmental psychology analyses the stages in which the brain develops certain learning abilities such as categorisation (Mills et al., 2005). In the field of neuro-physiology, researchers are interested in how the somatosensory mapping in the brain adapts to disturbances (Dinse & Merzenich, 2002). Neurology uses methods of functional magnetic resonance imaging to determine the location of different kinds of knowledge and mental processes (Wolbers & Büchel, 2005). In the field of artificial intelligence there is still a strong belief that brain functions are well modelled with symbolic rule based systems (Steels, 2007).

This work follows an approach which is at the junction of three disciplines, namely cybernetics, behaviour-based robotics, and neuro-biology. This chapter discusses how these fields contribute to this work by answering the following questions:

1. What is the main concept on which this work is based on?
   (see sec. 2.2: Cybernetics)

2. Why are robots essential in this approach?
   (see sec. 2.3: Behaviour-Based Robotics & Embodied Artificial Intelligence)

3. How does the proposed model of synaptic plasticity relate to biological mechanisms?
   (see sec. 2.4: Biological Systems)

This chapter begins with a brief overview of the history of brain related research which will explain how the disciplines mentioned disciplines contribute to this work. Each of the following sections is based on one main source, as it covers most of the topic. Additional sources are used to supplement the sections, when necessary.

For the field of cybernetics the main source of inspiration is the book "Design for a Brain", by William Ross Ashby (Ashby, 1954). His book presents the concept of an ultra- and multistable system as a model for the brain and its ability to adapt. The second section discusses his concept, its physical implementation; the Homeostat, and their implications for this work.

The Self-Regulating Neuron model is proposed as a method for adaptation of recurrent neural networks of arbitrary structure in the sensori-motor loop. As proof of concept, different recurrent neural networks are implemented to control an autonomous robot. This follows the approach first described by Rodney Brooks in his publications "Intelligence without Representation" (Brooks, 1991c) and "Intelligence without Reason" (Brooks, 1991b). He claims that in order to understand the principle of intelligence, one needs to build fully self-sufficient robots, starting at a very low level of complexity. Once lower levels are understood, the complexity of the system can be gradually increased until higher order intelligence, such as human intelligence, can be understood. Brooks proposes real autonomous robots as a platform for experimentation because they are verified against our natural environment, and not against artificial mirco-worlds. This concept will be discussed in the third section.

The Self-Regulating Neuron model presented in this work is biologically plausible as it relates to short-term plasticity by synaptic scaling. The fourth section is based on the book "Memory: From Mind to Molecules" by Larry R. Squire and Eric R. Kandel (Squire & Kandel, 1998) and discusses basic biological mechanisms for synaptic plasticity and their relation to the presented model.

2.1 A short overview of the history of neuro-informatics

This section gives a short history of neuro-informatics, with a strong focus on the disciplines which are related to this work (see above).

The first known description of the brain is the Edwin Smith Papyrus, which dates back to about 3000 BC. The papyrus is comparable to a medical handbook. It presents diagnosable symptoms of head injuries, possible treatments and a prognosis of the development if a treatment is applied. It does not discuss how the brain functions.

The first functional descriptions of the brain are found in ancient Greece. Different philosophers had varying opinions on the brain's main function. Hippocrates (460 BC) was the first to talk about it. He believed that the brain is responsible for sensitises and intelligence. Plato (387 BC) also believed that mental processes are located in the brain. His student, Aristoteles (335 BC) did not agree and assumed that mental processes were located in the heart, and that the brain's main function was to cool the blood. Although the first functional descriptions were available, the role of the brain was unclear and open to discussion.

The first anatomical studies are found in the book "Cerebri anatomi" written by Thomas Willis in 1664. Thomas Willis (1621-1673) was an English physician with important contributions to the science of anatomy. His book included descriptions of brain regions and brain nerves, and his numbering of the eleven cranial nerves holds until the present day. Examples for these cranial nerves are the olfactory nerve, which transmits the sense of smell, and the optic nerve, which transmits visual information to the brain. He was the first to use the term *reflex action* to describe elemental actions of the nervous system, relating a stimulus to a specific response. Because of his contributions, he is considered to be the father of neurology. Although his work provided anatomical descriptions of the brain, there were still no functional descriptions of the brain and its basic elements available at that time.

The modern form of neuroscience dates back to the end of the 19th century, when Santiago

Ranmón y Cajal formulated the *neuron doctrine* (1889). He discovered that the brain consists of discrete cells, each delimited by an external membrane. These cells are the elementary signalling units of the brain and are called *nerve cells* or *neurons*. The term *synapse*, describing a point-to-point connection between neurons originates from Charles Scott Sherrington (1897). Cajal and Sherrington are considered to be the founders of modern neuroscience. It was the first description of the basic elements and their contribution to the abilities of the brain.

Up to this point the brain has been described at different levels, functional, modular, and cellular level. However, it was still unclear how learning or adaptation is achieved. Donald O. Hebb was the first to formulate a general rule for synaptic plasticity in biological systems. In his book "The Organization of Behavior: A Neuropsychological Theory" (Hebb, 1949) he stated that a synapse is strengthened if the two corresponding neurons are correlated in their activity. Although this is a very simple rule, it is the origin and inspiration for the formulation of many learning rules and principles for artificial systems until the present day.

It took about 30 years for Hebb's hypothesis to be proven in an animal. It is possible, with modern techniques, to record the activity of single neurons. This allowed not only the verification of Hebb's learning rule but to also showed its limitations in biological systems. Bliss and Lømo (1973) first proved, in rats, that synaptic connections are strengthened depending on the duration and frequency of a presented stimulus. Recent progress in the field of genetics allows researchers to knock out specific genes and therefore investigate the role of biochemical processes in learning and memory. Today some of the basic biological mechanisms responsible for learning and memory are known. Two of them, namely short and long term potentiation are described in the fourth section of this chapter.

Besides research on animals and humans, other disciplines, which do not deal with biological systems, contributed to the research on how the ability of the brain to learn and to adapt may be achieved. In 1943 Warren S. McCulloch and Walter Pitts showed through theoretical experiments that a collection of neurons can perform logical operations. Inspired by the recordings of an action potential by Julius Bernstein (1868) their neurons were binary. A neuron can take one of two states, either firing or not-firing, represented by the digits 1 and 0, respectively. Their publication "A logical calculus of the ideas immanent in nervous activity" (McCulloch & Pitts, 1943) initiated two new fields of research. One is the theory of finite-state machines as a model of computation. The other is the field of artificial neural networks which is considered with respect to learning in the remainder of this section.

The first learning rule for artificial neural networks was published in "The Perceptron: A Probabilistic Model for Information Storage and Organization in the Brain" by Frank Rosenblatt (1958). A Perceptron is a strictly layered feed-forward network consisting of two layers[†] which receives a binary input vector and delivers a binary output as a response. Rosenblatt described an algorithm to alter the weights in order to minimise the error of the output with respect to a training set. This enabled the network to perform classification tasks in which sets of input vectors are mapped onto a set of output vectors.

It was shown by Marvin Minsky and Seymour A. Papert in their book "Perceptrons" (Minsky & Papert, 1969) that the Perceptron model was restricted. It could not solve a simple non-linear

[†]The layers are counted with respect to the number of synapse layers. A different numbering counts the number of neuron-layers. In this case the network would be considered as a three layered network.

classification task such as the XOR-problem. This proved that the Perceptron was insufficient to describe the brain and stopped the research in this field for about a decade.

In the 80's artificial neural networks with non-linear neuron models and the back-propagation algorithm were introduced. Back-propagation was initially developed by Paul J Werbos in 1974, and independently rediscovered in the early 1980s by David Rumelhart and David Parker. This algorithm enabled multilayer feed-forward neural networks with the ability to perform non-linear classification operations beyond those known for Perceptrons. After this breakthrough, a lot of research was contributed to the improvement of the back-propagation algorithm, as well as the formulation of other learning rules for feed-forward neural networks. Biological inspired learning algorithms for artificial systems are discussed later in this work (see sec. 4.3: Related work). However, this research in the field of of artificial neural networks concentrated on feed-forward structures, with only few exceptions such as the highly structured Hopfield network (Hopfield, 1982) for pattern recognition or Elman networks (Elman, 1990), which are an extension of Jordan networks (Jordan, 1986), to model time sequences, initially targeting natural language processing. A different method named *back-propagation through time* converts a recurrent neural network into a feed-forward network by adding layers of identical copies mapping the recurrences as feed-forward connections between the layers (Minsky & Papert, 1969; D. E. Rumelhart, McClelland, & the PDP Research Group, 1986).

Although the brain, with its large number of neurons and synaptic connections, can be considered as a highly recurrent structure with dynamical properties such as oscillations and hysteresis (Pasemann, 1996), considerably little effort was contributed to the research of recurrent neural networks. The first full mathematical analysis was published in "Dynamics of a single model neuron" by Frank Pasemann (1993). He showed that even with a simple non-linear neuron model, a single neuron with a recurrent connection can show dynamical properties such as oscillation and hysteresis. In the following publications it was shown that even small networks consisting of two, three neurons and ring structures have very large dynamical reservoirs (Pasemann, 2002, 1995) and that a single neuron with a damping term can show chaotic behaviour (Pasemann, 1997b).

Besides the approaches in biological and artificial neural networks, the 50's saw another discipline approach the understanding of the brain. The field of cybernetics, as defined by Norbert Wiener in his book "Cybernetics; or the Control and Communication in the Animal and the Machine" (Wiener, 1948) started to describe animals and machines equally on a system level. Important is not the full understanding of the internals, but a complete description of the behaviour of a system. William Ross Ashby states in his book "Design for a Brain" (Ashby, 1956) that a good model must hold an objective demonstration, preferably in a technical machine. Inspired by Cannon's formulation of homeostasis (1932), he built the Homeostat, a machine that demonstrates his idea of an ultrastable system, which he proposes as a model for the brain and its ability to adapt. The Homeostat and its implications for this work are discussed in the second section of this chapter.

Since the early 90's a new perspective has been contributed by the field of robotics. In the publication "Intelligence without Representation" (Brooks, 1991c) Rodney Brooks described the dilemma of current research in the field of artificial intelligence from his point of view. He claims that the research is concerned with different aspects of intelligence, such as linguistics, computer vision, knowledge representation, etc., but that the research is not concerned with the principle

of intelligence. To find such a principle, small systems with considerably low complexity must be built and fully understood first, before systems of higher complexity such as those found in humans or other biological systems can be understood. This field of research is referred to as Behaviour-Based Robotics. In addition to the work by Brooks, Rolf Pfeifer describes in his books "Understanding Intelligence" and "How the body shapes the way we think" (Pfeifer & Scheier, 1999; Pfeifer & Bongard, 2006) that intelligence is strongly coupled to the body and environment in which it acts, and cannot be understood or modelled separately. He shows in experiments with robots, that a carefully designed body reduces the need of a complex controlling unit. This corresponds to the *Gedankenexperiment* of Valentino Braintenberg, who showed in his book "Vehicles; Experiments in Synthetic Psychology" (Braitenberg, 1984) that a behaviour classified as complex by an observer, can be related to very simple internal control structures. One of his vehicles is discussed and used for experimentation in this work (see chap. 5).

The overview presented in this section emphasised cybernetics, robotics and neuro-biology. These are detailed in the following sections.

2.2 Cybernetics

The model for synaptic plasticity presented in this work is inspired by the book "Design for a brain" written by the British cyberneticist and psychologist William Ross Ashby (Ashby, 1954). Ashby was interested in finding a model to describe adaptive systems such as the brain. He formulated three questions to guide his effort in modelling the brain. First, what cerebral changes occur during the learning process. Second, why does behaviour usually change for the better, and third, what type of process shows the same property? To answer these questions, he developed the concept of an ultrastable system and constructed a physical machine, the Homeostat, as proof of concept. Both are presented and discussed here with respect to their implications for this work.

Compared to other fields which also research the same questions, such as e.g. biology and artificial neural networks, Ashby as a cyberneticist had a different perspective and different terminology to describe the brain. Therefore, the perspective and the terminology are introduced first before the concept is discussed.

Cybernetics was first defined by Wiener as

> "[...] *the science of control and communication, in the animal and the machine.*"
>
> (Wiener, 1948)

and later described by Ashby as

> "[...] *a 'theory of things', but it treats, not things but ways of behaving.*"
>
> (Ashby, 1956)

Cybernetics is the science that studies the abstract principles of organisation and behaviour in complex systems and is less concerned with their exact implementation:

> *"In essence, cybernetics is concerned with those properties of system that are independent of their concrete material or components. This allows it to describe physically very different systems, such as electronic circuits, brains, and organisations, with the same concepts and to look for isomorphisms between them."*
>
> (Heylighen & Joslyn, 2001)

To clarify this aspect, consider a car as the system and its locomotion as the behaviour of interest to the observer. To fully describe and predict the behaviour of the car at any instant, only the current motion (velocity vector), the mass, the friction, and the current acceleration are required. This set of parameters sufficiently describes the behaviour of interest. From the point of cybernetics it is of no matter how they are related to the interaction of the mechanical parts of the car's engine, for example.

In his book, Ashby (1954) uses terminology closely related to the scientific field of dynamical systems theory. This field was well established in the western world about twenty years later by the work of Andronov et al. (1973). Therefore, the notions of Ashby are similar but not compatible with today's understanding of dynamical systems. The discrepancies are preliminarily discussed in the following section before the concept of the ultrastable system and its physical implementation, the Homeostat are presented.

2.2.1 Terminology

In this section, it is assumed that the concepts of dynamical systems theory are known to the reader. A short introduction to the field is given in the next chapter (see chap. 3.1.1).

This section discusses Ashby's notions of a system, a field, stability and adaptivity and rephrases them in the context of dynamical systems theory.

System

A system must obey axiomatic demands (Ashby, 1954). It must be applicable to any form of material system which may be animate or inanimate. It must be precisely defined, and information about it must be available only by objective observation of the system itself, and no other source may be required. Deterministic systems are assumed.

A system is described by a set of variables which cover the aspects of interest of the system (reconsider the car example given above) and a set of rules which describe the system's behaviour over time. The system's state is given by the numerical values of the variables at a certain instant in time, and all possible states are captured in the system's phase-space. The evolution of a system over time is represented by a trajectory.

Ashby requires two main concepts to formulate the ultrastable system, the field and the step-function variable. Both are closely related and will be covered in the next section.

Field

A field is defined as a phase-space and the set of all trajectories in the phase-space. This is similar but not equivalent to the definition of a flow in a phase-space.

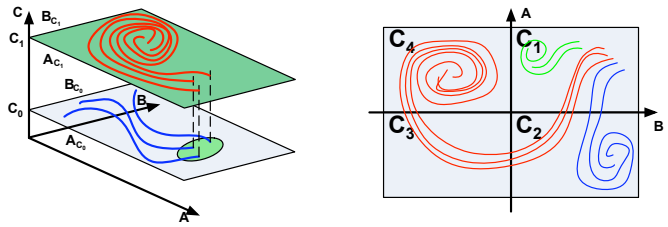


Figure 2.1: FIELD. Left: System with two full-function variables A, B and a step function variable C. If C is constant, the field is determined by the values of the variables A and B. If C changes, the resulting field of A and B changes. The ellipsoid marks the region of critical states that lead to the chance of the step-function variable C from C_0 to C_1 (bifurcation points). See the text below for a discussion and interpretation of this figure (see sec. 2.2.2). Right: Plot of the partitioning of the phase-space by the concept of fields and step-function variables. For a discussion, see text below.

In general, a field is a subset of the phase-space together with the related subset of trajectories. A field is labelled with respect to properties of its flow. This is elaborated on the basis of an example of a three-dimensional system (see fig. 2.1).

The example is a system described by three variables A, B, and C. The variables A and B are continuous, i.e. A,B $\in \mathbb{R}$ and C is of discrete nature, e.g. C $\in \mathbb{N}$. The variables A and B are called full-function or main variables, while C is called a step-function variable. The plot in the right hand side of figure 2.1 shows a projection of the three-dimensional system onto the plane given by the variables A and B. This plane is then divided into subsets for which the value of C is constant. In the example, four such subsets are shown, indicated by C_1 to C_4. These subsets are the fields of the system, as defined by Ashby.

In this formulation, a trajectory, which has its origin in one field, may leave it and enter another field (see fig. 2.1 [right-hand side, red and blue trajectories]). A field is now labelled with respect to its flow. If a trajectory enters and remains within a field, the field is said to be a terminal. It is clear from the given example (see fig. 2.1 [right hand side]), that this is no general field property, but a field-trajectory property, as the same field may be a terminal for some trajectories and not a terminal for others (see fig. 2.1 Right: C_1). The latter case is called an unstable field. The set of points which separate fields are called critical points, as they lead to a switch of the step-function variable, and hence to a new field (see fig. 2.1).

In the context of dynamical systems, this sort of behaviour is understood differently. A step-function variable corresponds to a system's parameter (Pasemann, 1996), but is not equivalent.

In dynamical systems, the behaviour of a system is characterised by attractors, such as a fixed point, periodic, quasi-periodic or chaotic attractors. The system's parameter does not directly contribute to the system's behaviour, but encodes regions of topologically equivalent attractors (Arrowsmith & Place, 1990), which are also called modes (Pasemann, 1996). In the example of the logistic map (see next chapter, figure 3.1), the output of the system varies for $r \in [2.4, 3[$, but it is topologically equivalent as it shows a fixed point attractor over the given interval. It is called topologically equivalent as the structure of the attractors for different values of r are equivalent despite the actual output of the system. In this context, a field is most closely related and hence, redefined as the basin of topologically equivalent attractors. This is a more narrow

definition, compared to the original definition by Ashby, in which a field can include attractors of very different type. In this new definition, a critical point corresponds to a bifurcation point.

Next, two system properties are defined by Ashby, namely stability and adaptivity.

Stability

The importance of stability is pointed out with an example of a biological system with its body temperature as a main variable. If the temperature drops too low or rises too high for a period of time, the animal dies. In order to survive, the body temperature must be kept within physiologically plausible boundaries, i.e. stable.

In terms of Ashby this relates to a terminal. Precisely, a trajectory is defined to be stable, when it remains within a region, once it has entered it.

In terms of dynamical systems, this concept of stability lies in-between Liapunov stability and asymptotic stability. A point x^* in the phase-space is called Lyapunov stable, if all trajectories that start sufficiently close to x^* remain close, that is if for every neighbourhood of U of x^* in the phase space, there exists a smaller neighbourhood $U' \subset U$ of x^*, such that every solution starting in U' will remain in U for all $t > 0$ (Strogatz, 1994). If x^* also is an attracting fixed point, i.e. all trajectories that start near to x^* approach it ($\lim_{t\to\infty} x(t) \to x^*$), it is called an asymptotically stable fixed point (Strogatz, 1994).

Ashby's notion of stability is replaced with asymptotic stability, which is plausible, when adaptivity (which follows next) is taken into account.

Adaptivity

A system can now be defined as *adaptive*, if it maintains its essential variables within physiological limits (Ashby, 1954). Adaptive behaviour ensures the survival of a system against external disturbances. A special form of adaptivity is called *homeostasis* (Cannon, 1932). It is the process regulating a variable towards a target value, with the important property that it fails, if the variable exceeds specific boundaries. Revisiting the body temperature example, this means, that if the temperature rises too high or falls too low, the system fails, and the animal dies.

This definition is well suited with the concept of asymptotic stability, as the presented form of adaptivity regulates towards a target value when a system is in the neighbourhood of the target value.

2.2.2 The Ultrastable System

With the terminology given above, the ultrastable system is defined as follows. Consider a system with variables $\vec{x} = \{x_i\}$, $i \in \mathbb{N}, x_i \in \mathbb{R}$ and parameters $\vec{p} = \{p_j\}$, $j \in \mathbb{N}, p_j \in \mathbb{R}$. Without loss of generality, it is assumed for simplicity of argumentation, that the phase-space only contains asymptotically stable fixed points and unstable fixed points, and that at least one of each exists.

To continue, first the notion of a parameter with respect to a variable must be clarified. Parameters in general are slow varying or static with respect to the internal dynamics. In a neural network, controlling a robot in the sensori-motor loop, the parameters on the one hand are the configuration parameters of the network (biases and synaptic weights) and on the other hand capture sensory input. Both types are considered in the parameters vector. In a static

network, where the biases and weights are constant, these would obviously not be taken into account, but in a dynamic network, as proposed in this work, the varying synaptic weights do modulate the behaviour of the system, and are therefore considered to be part of the parameter set. For a more detailed discussion on system parameter and mode switching in neural networks, the reader is referred to Pasemann (1996). The variable vector $\vec{x}$ of the system, in this example, refers to the activations or outputs of the neurons in the network.

A system, as described above, has the following behaviour. For a given set of parameters $\vec{p}$ and an initial condition $\vec{x}(t_0)$ the behaviour system either is stable or unstable. In the latter case, bifurcations will occur if the system has access to its parameters. Then, consecutive modulations of a subset of the parameters $\vec{q}(t) \subseteq \vec{p}(t)$, will lead to bifurcations until a stabilisation of the system occurs.

This process is the *principle of ultrastability* and is described by Ashby as:

> *"An ultrastable system acts selectively towards the fields of the main variables, rejecting those that lead the representative point [current state] to a critical state but retaining those that do not."*
>
> (Ashby, 1954)

The important point here is that a feedback mechanism is required, which allows the system to modulate its parameters. It will be argued later in this section, that it is this double feedback loop, i.e. modulation of the system's parameters by the system in addition to the external (sensori-motor) feedback loop, which is required for adaptivity.

In order to account for a large variety of disturbances, the system must have a rich reservoir of co-existing attractors. In the terminology of Ashby, this means that a system must have a large number of fields.

Ashby states that any model must hold the concept of objective demonstration, preferable in a physical implementation. Therefore the principle of ultrastability and its properties are demonstrated by him in a machine which he calls the *Homeostat*.

2.2.3 The Homeostat

The Homeostat is an impressive demonstrator for the abstract concept of ultrastability, not only in that it was built so early (1952), but also in its ability to clearly demonstrate that adaptive behaviour, which seems intelligent to an observer, does not require sophisticated regulatory mechanisms. This, and further implications will be discussed after the presentation of the Homeostat.

The Homeostat (see fig. 2.2) consists of four identical self-regulating (homeostatic) units, each carrying a rotatable magnet (see fig. 2.2 [left-hand side]). Each unit provides a DC current, proportional to the deviation of the magnet from its centre orientation, which is presented as input to the other three units and as a feedback signal to the unit it originated from. The algebraic sum of the four currents is passed through a device called the *communicator*. It determines the polarity and the strength of the input and applies the result as torque to the coil.

The polarity and strength of the inputs are two parameters of the system, denoted by X and P. Depending on the setting of the parameters the magnet shows a diverging or converging

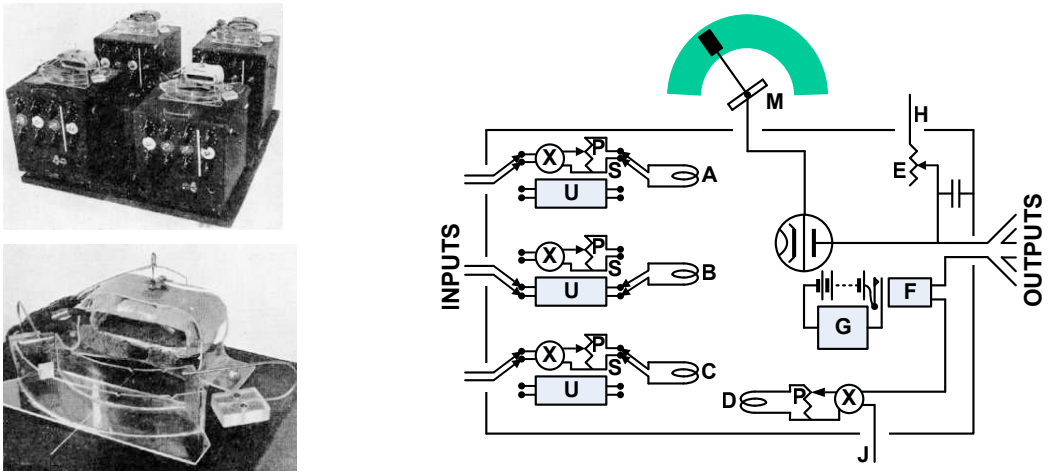


Figure 2.2: The Homeostat. By William Ross Ashby. The two figures on the left-hand side (taken from (Ashby, 1954)) show the machine constructed by Ashby. The upper of the two figures shows the entire system, while the figure below shows the top view on one of the four homeostatic units with its magnet. The figure on the right-hand side shows the schematic of the Homeostat (Ashby, 1954), where A-D are the coils through which the currents of the other homeostatic units and the self-coupling act on the unit's magnet M, X is the communicator which determines the polarity of the input reaching the coil, and P is a potentiometer that determines the fraction of the input which reaches the coil. The coil G of each uniselector is energised, by closing the relay F, when the magnet M reaches an extreme position. For a detailed description, see text.

behaviour as a reaction to its displacement from its centre. A device called the *uniselector* can automatically change the values of the parameters X and P. If triggered, the uniselector steps through 25 randomised settings. For the Homeostat, consisting of four units, this allows $25^4 = 390.625$ different configurations. Some might be equivalent, but this is of no matter. If the magnet reaches an extreme position, the uniselector is triggered and steps to the next setting of X and P.

The Homeostat can be understood as follows. A state of the system is defined by four main variables, the deviation of the four magnets, and the eight step-function variables which are given by the two parameters (X, P) for each unit. Every reconfiguration of the step-function variables X and P leads to a new field, and therefore to different trajectories of the four main variables.

The behaviour of the Homeostat can be described in the following way. If the system is disturbed by dislocating one or more magnets, one of two possibilities occurs. First, a coordinated behaviour of the four units may stabilise the magnets at their centres. Second, one or more magnets may reach an extreme position. In this case, the uniselector is triggered. This leads to a randomised reconfiguration of the corresponding units. The input currents that are presented to the unit are altered, leading to a change of the position of the magnet. The new configuration either leads to a stabilisation of the four units or to a diverging behaviour of at least one of them. This process is repeated until the system finally stabilises.

Different experiments were performed with the Homeostat. First, the behaviour of only

one unit was observed. The position of the magnet is stable, if the parameters provided by the uniselector define a negative feedback loop. If the magnet is displaced manually, it simply returns to the central position. But if the polarity of the input-output junction is reversed, the unit destabilises. The magnet diverges from the central position until it reaches an extreme position. As a result, the uniselector is triggered until a selected parameter setting stabilises the system. In further experiments, the same behaviour was observable for any combination of the four coupled units. Even when units were mechanically coupled, so that the magnets were forced to move together, the system would randomly reconfigure itself until it found a stable configuration. This demonstrates that the Homeostat is able to adapt to different situations, even to those it was not originally designed for (mechanically coupled units).

Discussion

The Homeostat is a physical machine constructed of coupled homeostatic units, each regulating its main variable (the rotatable magnet) towards a target value (the centre). The available mechanisms to achieve this goal is to modulate the input it receives by the step-function variables X and P, which alter the sign and strength of the sum of input currents. It must be noted that only information that is locally available to the homeostatic unit is used for this process. Explicitly, this means that only the deviation of the magnet as the result of the sum of input currents is used to change the parameters. No external information, such as the deviation of any of the other magnets is used directly.

An experimenter introduces a disturbance to the system by dislocating an arbitrary set of magnets from their centre. This leads to an ongoing reconfiguration of the Homeostat until a stable solution is found.

Although during Ashby's time, some reviewers assumed that *Design for a brain* would not be of lasting interest[†] (Milholland et al., 1954) there are still important implications for current research (Di Paolo, 2003).

First, the Homeostat demonstrates, that intelligent behaviour does not necessarily depend on intelligent mechanisms:

> *"The choice of random step-functions is conceptually interesting* [...] *as a proof that dumb mechanisms can yield adaptive responses which from the point of view of an external observer may look quite clever."*
>
> (Di Paolo, 2003)

Braitenberg demonstrates a similar aspect in his *Gedankenexperiments*, in which he showed how very simple control structures lead to behaviours which can be labelled with emotions by an external behaviour. The description of his vehicles two and three are endowed with fear and love (Braitenberg, 1984), although the behaviours result from very simple couplings between the sensors and actuators (see chap. 5). However, a random process for intelligent behaviour was not considered by Braitenberg.

Second, Ashby offers a solution to the fundamental problem that Dreyfus saw in classical artificial intelligence research (Dreyfus, 1992). Although autonomous robots are situated and

[†]It must be noted that Wiener described the homeostat as 'one of the great philosophical contributes of the present day' (Heylighen & Joslyn, 2001; Wiener, 1954).

embodied, and therefore already account for some of Dreyfus's criticism, they are not *intentional*. Di Paolo describes intentions as habit formation (Di Paolo, 2003), a property that distinguishes real animals as cognitive systems from autonomous robots. This was already shown in Ashby's work, who demonstrates that

> *"[...] a closed sensorimotor loop is not enough for adaptation, but that at least a double feedback structure is needed."*
>
> (Di Paolo, 2003)

and himself states that

> *"[...] the behaviour of a stable system may be described as 'goal-seeking'."*
>
> (Ashby, 1954)

In the quote given above, Di Paolo refers to the interaction of the (external) sensori-motor loop and the (internal) dynamics of the system which alters system's parameters. How this relates to habit formation and the Homeostat is elaborated in the following.

But first, the concept of the ultrastable system is briefly restated and extended with the notion of a multistable system. An ultrastable system is one, that rejects unstable fields and converges towards terminals, i.e. stabilises against external disturbances by means of self-reconfiguration. For large systems, the search space grows exponentially with the number of parameters. To reduce the number of reconfigurations of a complex system with a large set of parameters, such systems are divided into loosely-coupled, ultrastable systems. Changes in a subset of the variables $\vec{x}$ and parameters $\vec{p}$ no longer influence the entire system, which significantly reduces the required time to stabilise (Ashby, 1954). This type of system is referred to as a multistable system. Such a system of loosely coupled ultrastable systems will initially require many reconfiguration steps until it stabilises. If a similar disturbance is encountered, the previously configured sub-system may already account for it. In terms of Ashby, this means that after some time, the ultrastable systems will configure in such a way, that their reconfiguration will influence as few other ultrastable systems as possible, in order to decrease the necessary time to stabilise after some already encountered external event has occurred.

Returning to Di Paolo's quote and his notion of habit formation, the configuration of the multistable systems at first can be considered as (randomly) testing different behaviours as possible reactions to external disturbances. After a while, behaviours which were proven to be suitable in a situation will be chosen again, if the same or a similar situation occurs. In terms of Di Paolo, this is habit formation, which he states was shown by Ashby to require at least a double feedback structure. The importance of habit formation in biological systems is obvious as the time required to react to an external disturbance can be crucial for the survival of the system.

In addition to the arguments of Di Paolo, current research gives strong experimental evidence that homeostasis is a general principle in neural systems. A review of homeostatic control of neural activity on different scales is given by Davis (2006). The biological relevance of homeostasis in learning and synaptic plasticity is discussed later in this chapter (see sec. 2.4).

The remainder of this chapter discusses robots as an experimentation platform used to understand intelligence (see sec. 2.3). Additionally, the understanding of the presented model as a biologically plausible model for synaptic plasticity and therefore also for learning is explained (see sec. 2.4).

2.3 Behaviour-Based Robotics & Embodied Artificial Intelligence

Classical artificial intelligence research focusses on replicating human intelligence by building isolated functional units. Among others, examples for such units are visual processing, symbolic representation, and reasoning. In a second step, these functional subunits are supposed to produce intelligent behaviour, when combined appropriately. This approach has been discussed and criticised early in the literature (Dreyfus, 1972). Dreyfus analysed, in detail, the most prominent approaches in artificial intelligence[†] and clearly points out the differences of the symbol-based systems to human intelligence. Probably the most debated criticisms of artificial intelligence is the Chinese Room Experiment (Searle, 1980). In his *Gedankenexperiment*, Searle argues that any symbolic and rule-based system does nothing else but meaningless symbol processing. The analogy to a rule-based artificial intelligence system is a room, in which a person, only capable of understanding and speaking English, receives questions in Chinese through a slot in the wall. The person can only discriminate the symbols by their shape, and has no further understanding of what is written. Given an English rule book with instructions how to process the Chinese symbols, the person is now able to produce answers from the questions, which are not distinguishable to the answers she or he would be able to give to questions asked in English. The point made here is that in the case of English, the person does *understand* the questions, whereas in the case of Chinese there is no such understating, but meaningless symbol processing. This was later rephrased as the symbol-grounding problem (Harnad, 1991).

The Chinese Room Argument initialised a long-lasting discussion with over 100 articles having been published on it (Pinker, 1999) and even led Hayes to the statement that

> *"[...] cognitive science is the ongoing program of showing Searle's Chinese Room Argument to be false."*
>
> (Lucas & Hazes, 1982) cited from (Harand, 2001)

It is beyond the scope of this work to discuss the Chinese Room Argument and its answers in full extent. For detailed discussions about the argument, the reader is referred to the literature (Churchland & Churchland, 1990; Harand, 2001; Anderson & Copeland, 2002; Cole, 2004; Harnad, 2005; Steels, 2007). However, the *Gedankenexperiment* demonstrates that this form of classical artificial intelligence does not contribute to the understanding of the mechanisms underlying intelligent behaviour.

A new approach to artificial intelligence was proposed by Brooks (1986, 1986), which Rolf Pfeifer describes as the

> *"[...] probably the biggest change in the history of artificial intelligence [...] "*
>
> (Pfeifer, 2007)

This section is based on two of his following publications (Brooks, 1991b, 1991c) in which he describes his approach as *Behaviour-Based Robotics* and argues against the sense-model-plan-act paradigm in robotics. It was Rolf Pfeifer who later emphasised the importance of the body,

[†]In the third edition Dreyfus (1992) analysed the years of GOFAI research after the first and second edition of his book in which he analysed the first two decades of GOFAI (1957–1967, 1967–1977).

and started a new research field called *Embodied Artificial Intelligence* (Pfeifer & Scheier, 1999; Pfeifer & Bongard, 2006).

Behaviour-Based Artificial Intelligence is based on four key ideas[†]:

Situatedness: The robot uses the world as its model. Instead of building an internal model of the world, it continuously relies on its sensors. This way the world directly influences the behaviour of the robot. Specifically, no abstract model of the world is presented to the robot.

The world is its own best model.

Embodiment: Embodied systems have two advantages. The first advantage is that they are situated and fully validated against the real world.

For the second advantage, first consider a totally isolated system. Such a system cannot sense changes in the environment as a feedback to its own actions. Hence, the internal process is reduced to the processing of symbols that are meaningless (in the sense of Searle, see above) to the system. Placing a system in an environment grounds this process. It gives the internal process a meaning, such as the survival of the system. The second advantage therefore is that the real world naturally limits the amount of possible regression of the systems internal process to pure symbol handling.

The world grounds regress.

Intelligence: The intelligence of the robot can not be assigned to the computational unit. Intelligence appears only with respect to the environment in which it acts.

Intelligence is determined by the dynamics of interaction with the world.

Emergence: An intelligent behaviour emerges from the interaction of the components of the system and from the interaction of the complete system with the environment. It can not be assigned to a single component or to the system isolated from environment.

Intelligence is in the eye of the observer.

Note that the idea of embodiment is an argument against the Chinese Room Experiment, and that the idea of emergence is very closely related to the quote taken from Umberto Eco's novel, with which the introduction of the thesis began.

Behaviour-based artificial intelligence does not decompose intelligence into functional sub-units. It is assumed that there is a principle to intelligence, and that it is strongly coupled to the morphology and the environment of the system. Brooks proposes an incremental approach to finding this principle. Starting from a very simple, but fully autonomous system, the complexity of the system is gradually increased. On every level of complexity, the system is validated

[†]Statements written in italic are cited from Brooks (1991b).

against the natural environment. In his subsumption architecture, new behaviour modules are added incrementally, in co-existence to the previous ones, in order to gradually increase the overall complexity of the system. The important difference to classical artificial intelligence is that each behaviour module has full access to all sensors and all actuators. The final behaviour is the result of the interaction of the modules, where inputs and outputs of lower level modules can be suppressed or inhibited by higher level layers (Brooks, 1986).

Several years after Brooks, Cliff concludes that models in computational neuroscience are meaningless unless embodied within a sensori-motor system, which means that an external feedback loop is closed from the motor output to the sensor input (Cliff, 1990). He is considered as the origin of the notion of a sensori-motor loop or sensori-motor system, and calls his approach *Computational Neuroethology*. The advantage is

> *"[...] that the semantics of the network are well grounded, and thus results are generated by observation rather than interpretation."*
>
> (Cliff, 1990)

The important statement here is that the reaction of a system to inputs are not interpreted, as in classical rule-based artificial intelligent systems. The results are observed, while the agent interacts with its environment. The quality of e.g. an agent's obstacle avoidance behaviour is not determined by presenting different symbols of various situations and interpreting the outputs as reactions to the symbols, but rather through observations of collisions or avoidance of obstacles, while the agent acts in the environment.

That Ashby's understanding of the brain as a dynamical system is consonant with the concept that (intelligent) behaviour requires the sensori-motor loop is discussed by Chiel and Beer (1997).

This work follows the Behaviour-Based and Embodied Artificial Intelligence approach. But, in contrast to Brooks, this work does not use the subsumption architecture consisting of distinctive behaviour modules, as this type of architecture has the limitation that it requires preprogrammed and manually designed behaviours. Instead, a single recurrent neural network controls a robot, which is then considered and analysed as a dynamical system. This is discussed in detail in the next chapter.

To summarise the previous sections, first the concept of an ultrastable system consisting of homeostatic units was motivated as a model for the brain and its ability to adapt. It was then proposed to use an autonomous mobile robot as a research platform to understand the principles of intelligence. The next section of this chapter introduces basic biological learning and plasticity mechanisms.

2.4 Biological Systems

The importance of learning and memory is described by Squire and Kandel by the following quote:

> *"[...] every thought we have, every word we speak, every action we engage in — indeed, our very sense of self and our sense of connectedness to others — we owe to our memory, to the ability of our brains to record and store our experiences."*
>
> (Squire & Kandel, 1998)

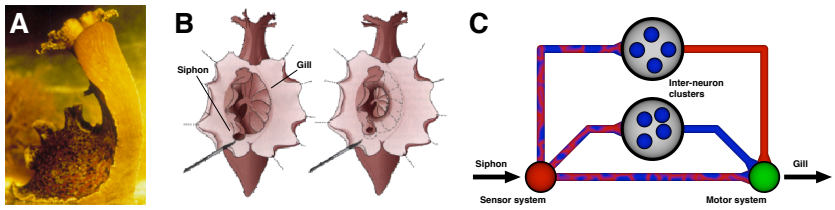


Figure 2.3: Aplysia. A) Picture of the marine snail Aplysia. B) Figure of the Aplysia, showing the gill withdrawal reflex triggered by a mild stimulation of the siphon by a paint brush. C) Highly reduced schematic drawing of the gill withdrawal reflex circuit. The gill withdrawal reflex of the Aplysia is well-suited for experimentation on non-declarative memory as only 100 uniquely identifiable neurons, which are identical in every animal, contribute to the reflex. The small number of neurons and their size ($\approx$ 1mm) allow easy handling and observations of structural and non-structural changes in the gill withdrawal signalling pathway as a result of different trainings. Of this 100 neurons, seven are motor neurons for the gill, six are motor neurons for the siphon, and two disjunct clusters each of 40 neurons are connected monosynaptically (either inhibitory or excitatory) to the motor system (C). The sensory system is connected to the motor system and the inter-neuron modules. In the case of habituation (see text below), the synaptic pathway from the sensor to the motor system, and the pathway from the sensor system to the excitatory cluster is weakened. The figures are taken from (Squire & Kandel, 1998), while the schematics is redrawn by the author of this work.

It is this ability of our brain to record and store experiences that is understood as learning. In order to identify a general principle of learning, which is the aim of this work, a more precise description is required. Inspired by neuro-biology this section develops such a description. It will be discussed that learning can be classified into two major categories, declarative and non-declarative. These two types are distinguished by the duration of the behavioural change and the underlying bio-chemical mechanisms.

2.4.1 Learning and Memory

What follows in this section is a means of categorising the various forms of learning an memory as opposed to precise definitions of the forms. As the transition between the underlying processes are smooth, different classifications are possible and also found in literature. But first recall a part of the quote given above:

> "[...] *to the ability of our brains to record and store our* experiences."

If we think about our experiences, the first items that might occur to us are events, facts, or people we have met. This form of memory is called explicit, conscious, or *declarative* memory.

Declarative memory was first described by William James in 1890 as

> "[...] *the knowledge of a former state of mind after it has already once dropped from the consciousness; or rather it is the knowledge of an event, or fact, of which we have not been thinking, with the additional consciousness that we have thought or experienced it before.*"

Summarising this quote, declarative memory is described by W. James as the ability to willingly recall formerly stored experiences, such as remembering events from the childhood.

Research in the field of declarative memory focuses on how and where such experience is encoded, stored, retrieved and also how and why it is forgotten. This approach is understood as the cognitive perspective of memory. In the field of human research, experiments were performed with patients suffering from lesions in specific brain regions. A well know patient is HM. In an attempt to stop HM's epileptic seizures, his medial temporal lobe, a certain region in the brain, was completely removed. After this operation HM was cured from the seizures but could not store any new experiences.

Nevertheless, he was able to fully recall experiences he had made years before the operation. For example, HM was able to describe in detail the street he grew up in as a child, just as a healthy person can. But the nurse, taking care of HM over decades, had to reintroduce herself every morning, as HM was unable to recognise her. Simple memory tasks, like remembering a certain number, were not possible unless HM would steadily repeat the number to himself. As soon as his attention was drawn away, he could no longer remember the number, nor that he had been asked to do so.

Although this indicated that HM had lost his ability to learn with the removal of the medial temporal lobe, he did not lose it completely. He was able to perform comparably well in skill learning tasks, such as mirror-drawing. In this task, a proband tries to follow a figure on a piece of paper with a pencil. The person cannot observe his hand directly, as the visual feedback is only available through a mirror. Thus, moving the hand to the left appears in the mirror as if the hand is moving to the right. Initially probands can hardly follow the figure and therefore commit errors, which are reduced through continuous training. HM was able to increase his performance comparably well, although he could not remember doing the task before. This indicated that there is a different form of memory, which is not concerned with storing experiences like facts, events, and their like. This other form of memory is named non-declarative memory.

Non-declarative memory is the memory of processes and is expressed as a change in behaviour. In contrast to declarative memory, which is characterised by the conscious recollection of stored experiences, non-declarative memory is unconscious. A typical example of this form of memory is how we learn to drive a car. In the beginning of this learning process, we use declarative memory about facts, such as how to accelerate and stop the car. We concentrate on which foot to use, so that we are sure to break or to accelerate. By continuously driving a car, our behaviour starts to change. Soon we do not have to recall which foot to use to accelerate, we use the correct foot automatically. This change in behaviour is done unconsciously, and described as non-declarative learning or memory.

One of the best known probands in this field of research is the *Aplysia* (see fig. 2.3). The Aplysia is a marine snail which has approximately 20.000 neurons. A group of about 100 cells contributes to a simple behavioural task – the gill withdrawal reflex (Squire & Kandel, 1998). A stimulus to the siphon or the tail of the Aplysia results in a withdrawal of the gill (see fig. 2.3 B). This reflex can be modified by learning. The cells involved in this task are distinctive, unique and identifiable in every animal (see fig. 2.3). This allows researchers to construct a neural wiring scheme for this special task. Another advantage is that the cells are easy to handle as they are about 1mm in size. They can be seen with the bare eye without the need of a

microscope. This simplified handling enables scientists to perform experiments and to identify what a specific neuron or synaptic connection contributes to the change of the behaviour of the Aplysia. Experiments focus on how the gill withdrawal reflex changes when different training stimuli are presented. This allows researchers to distinguish between three different forms of non-declarative memory; habituation, sensitisation, and conditioning.

Habituation is the ability to ignore a stimulus if the stimulus does not carry any information. In the case of the Aplysia, a repeated mild stimulus of the siphon with a fine paintbrush results in a decrease of the withdrawal reflex. Analysing the neuron activity of the contributing cells leads to the conclusion that the strength of the synaptic pathway between the sensor and motor neuron of the cells included in this task is weakened.

Sensitisation is the ability to change the behaviour if the stimulus is aversive. If the Aplysia receives a shock at its tail, it withdrawals its gill more completely. After training the animal, the reaction to a mild siphon stimulus is significantly increased. Analysis reveals that the strength of the synaptic pathway between sensor and motor neuron is strengthened.

Classical Conditioning was first described by Pavlov around 1895 and is the ability to learn to associate two stimuli or a stimulus and a response. Pavlov, who studied the digestive reflexes of dogs, noticed, that a dog would salivate as soon as the attendant, who had fed the dog in the past, would enter the room. The salivation was triggered by a formerly neutral stimulus – the attendant.

The withdrawal reflex of the Aplysia can be classically conditioned in the following way: A very weak electric shock is applied to the siphon as conditioned stimulus (CS), followed by a stronger electric shock applied to the tail as an unconditioned stimulus (US). After a training period, a mild stimulation of the siphon results in a strong withdrawal of the gill. The reflex after the training is much stronger than it is when the two stimuli (CS and US) are presented uncorrelated.

Up to this point, we have discussed two major types of learning and memory, declarative and non-declarative. From the patient HM we have seen that declarative memory acts on brain regions, as at least the medial temporal lobe is involved. Analysing and understanding the function and the interplay of certain brain regions is not the focus of this work. This work is concerned with a model for Self-Regulating Neurons and synaptic plasticity. Therefore, in the following sections, only the basic mechanisms involved in non-declarative memory are discussed.

2.4.2 Biological Mechanisms

In the previous section, the Aplysia was presented as a well-suited proband for experiments focusing on the different forms of non-declarative memory; habituation, sensitisation and classical conditioning. It was briefly described, that the change of behaviour results from a change of the strength of the synaptic pathway from the sensor to the motor system. Experiments show, that not only the type of non-declarative learning or memory can be distinguished, but also the duration of the changes of behaviour and therefore also of the potentiation of the synaptic pathway. If the training lasts for seconds or minutes, the change of behaviour is observable for seconds or minutes afterwards. The time for training and the resulting synaptic potentiation are of the same magnitude. But if the animal is trained for days, the change in behaviour is observable for weeks after the training (Squire & Kandel, 1998). The observable changes are

present at least a magnitude larger in time compared to the training. This allows researchers to distinguish two forms of non-declarative memory; short-term potentiation (STP) and long-term potentiation (LTP). The understanding is that STP is present on a time scale from seconds to minutes, and LTP is present on a time scale above minutes to years.

In the following, the biological mechanisms for STP and LTP are discussed as far as required for the remainder of this work. The interested reader will find a well-written and more detailed description, also suitable for non-biologists, by Squire and Kandel (1998). Detailed bio-chemical processes of plastic neurons are discussed by Smythies (2002) and the mechanisms of signal transmission and synaptogenesis are presented by Cowan, Südhof, and Stevens (2001).

Short-Term Potentiation is the change of the strength of a synaptic connection present for a time range from seconds to minutes. It is triggered comparably to the bio-chemical propagation of action-potentials by the release of neuro-transmitter of the pre-synaptic terminal. The propagation of an action-potential and STP differ in the kind of reaction induced in the post-synaptic neuron by the released transmitter. Therefore, we will briefly recall how an action-potential is transported over the synaptic cleft and then point out the similarities, as well as the differences between both processes. A more detailed description of the properties of a biological neuron is presented in the next chapter.

When an action potential arrives at the pre-synaptic neuron terminal, it triggers the release of a chemical neuro-transmitter into the synaptic cleft (see fig. 2.4). The released transmitters bind to receptors of the post-synaptic cell. These receptors are called *ionotropic receptors*, and the ionic channels controlled by these receptors are called *transmitter-gated ion channels*. Depending on the type of ions permitted to diffuse through the ion channel, the electric potential in the post-synaptic neuron increases or decreases. If the potential increases, the connection between the neurons is referred to as an *excitatory* synapse. If the potential decreases, it is called *inhibitory* synapse. If the potential of the post-synaptic neuron rises above a threshold, a chemical reaction leads to the generation of a new action-potential, which is propagated over the axon to the next neuron (see sec. 3.2). The duration of the potential change of the post-synaptic neuron, triggered by an action-potential of the pre-synaptic neuron, is in the range of milliseconds. Short-term potentiation lasts much longer then milliseconds indicating a different kind of process which will be explained next.

In the case of STP a second type of receptor is involved. This type of receptor is called *metabotropic receptor*, as it does not control an ion channel, but influences the metabolism of the neuron. A transmitter binding to a metabotropic receptor does not influence the neuron potential by the diffusion of ions, but through the activation of an enzyme, which in this case is adenosine cyclase. The activation of this enzyme changes the concentration of a intracellular signalling molecule called *second* or *intracellular messenger*. The first known second messenger is cAMP (cyclic adenosine monophosphate) which is synthesised from ATP (adenosine-triphosphate) by the adenosine cyclase. Second messengers have at least three different functions:

1. They transport the extracellular signals (action-potentials) into the cell.

2. They amplify the signal.

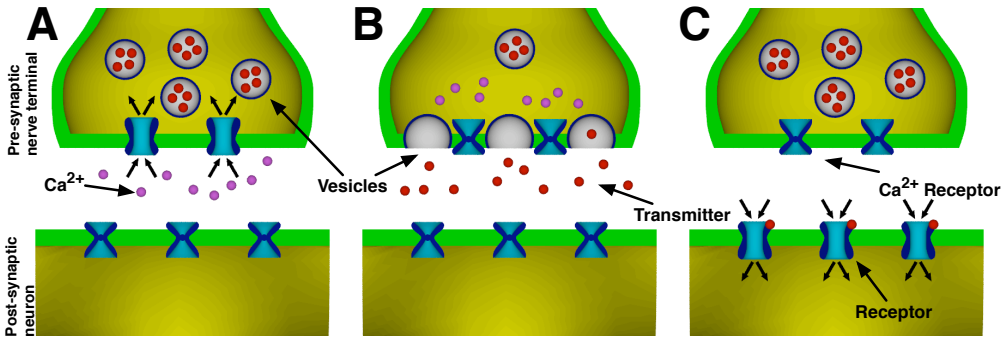


Figure 2.4: Propagation of an action-potential over the synaptic cleft. This figure shows the propagation of an action-potential over the synaptic cleft in three steps. In the first step (A) an action potential arrives at the pre-synaptic nerve terminal. This opens the Ca^{2+}-channels of the terminal. The diffusing Ca^{2+} ions (B) trigger the release of a transmitter. The transmitters are kept in vesicles, which can release none or all of their transmitters at the same time. The transmitters diffuse over the synaptic cleft and dock onto ionotropic receptors (C). The ionotropic receptors open their ion channel and permit the diffusion of ions. Depending on the diffusing ions, this leads to an increase or decrease of the post-synaptic neuron potential. The change of the potential in the neuron is the propagated signal. The image was redrawn by the author of this work, but is taken from (Squire & Kandel, 1998).

3. They regulate a variety of cellular functions in response to the signal, resulting in a state change of the cell.

Experiments showed that cAMP is important for the increased release of transmitters in the pre-synaptic neuron terminal. The production of cAMP here is triggered by a serotonin binding metabotropic receptor. The increase of cAMP leads to the closing of some ion channels (K^+ ion channel) which results in a broadening of the action potential. The broadening of the action potential increases the amount of transmitters released, when an action potential arrives at the pre-synaptic terminal. The increased release of transmitters through the serotonin binding receptors is the underlying mechanism for classical conditioning. Here an interneuron releases serotonin to increase the signal propagated from the sensory system to the motor system of the gill withdrawal reflex (see fig. 2.3 C).

The described process, triggered by the metabotropic receptors, lasts in the range of seconds to minutes and is the biochemical mechanism for STP.

Long-Term Potentiation is the change of the strength of a synaptic pathway present for a time range above minutes and was first described for a biological system by Bliss and Lømo (1973). This type of memory needs a different type of cellular mechanism. As we will see, this is due to changes of the cell anatomy.

An Aplysia trained for sensitisation that lasts over weeks shows a significant increase in the average number of synaptic terminals of the sensor neurons involved in the reflex. The average number is doubled from initially 1300 to 2600 per neuron (Squire & Kandel, 1998). This doubling of terminals is present as long as the change in behaviour is observable. After a few weeks the increased number of terminals is gradually decreased to the original number.

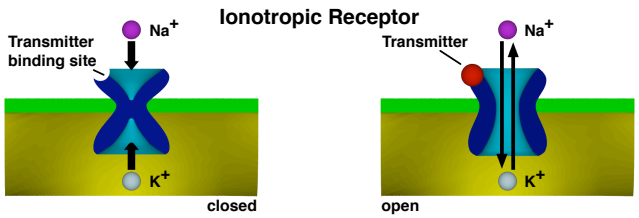


Figure 2.5: IONOTROPIC RECEPTOR. The figures show a part of a neuron membrane. The ionotropic receptor (blue) is initially closed (left). When a transmitter docks onto the transmitter binding site (right), the ion channel is opened. Then K^+-ions can diffuse through the opened channel from the inner (yellow) to the outer (white) of the cell, and Na^+-ions can diffuse from the outer into the inner. The change of concentrations of the ions changes the inner neuron potential and triggers a new action-potential if a threshold is reached. The image was redrawn by the author of this work, but is taken from (Squire & Kandel, 1998).

With the decrease of synaptic terminals, the observable change of behaviour is finally lost. Similar findings are also described for synapse formation in the mammalian central nervous system during childhood (McAllister, 2007) and in adult animals (Alvarez & Sabatini, 2007). The preservation of differences in the growth of dendrites and axons by evolution are discussed by Ye et al. (2007).

An increase in synaptic terminals leads to an increased release of neurotransmitter from the pre- to the post-synaptic neuron. Therefore, the effect of the sensory system on the motor system controlling the gill withdrawal is increased by the altered number of terminals. This explains how a mild stimulus of the siphon results in an increased withdrawal reflex weeks after the training stimulus was last presented.

The following paragraph will discuss the biological mechanism which causes a change of the cell anatomy in the case of LTP. It is beyond the scope of this work to present a complete overview of the known biological mechanisms, however, it is important to show the differences in the underlying mechanisms of STP and LTP in order to classify the synaptic plasticity model proposed in this work. For a more detailed description, the reader is referred to Squire and Kandel (1998) and Cowan et al. (2001). The intention here is to show that LTP and STP are triggered similarly but are the results of different biological mechanisms.

The metabolism of a cell is controlled by its nucleus. It contains the DNA from which the proteins are synthesised. To change the morphology of the cell, new proteins must be synthesised, and hence to alter the process, incoming action potentials must reach into the nucleus.

This is realised by the following process. When cAMP is synthesised, smaller proteins, called *subunits* are released as a result of a biochemical process. A certain group of these subunits are called *catalytic subunits*. It was shown that catalytic subunits diffuse into the nucleus and influence the synthesis of proteins (see fig. 2.6). To alter the process of generating proteins, the concentration of diffusing subunits and therefore cAMP must be higher then the amount released by STP. The necessary concentration is reached when the cell is stimulated with high frequency stimulus over a long period of time. Only when the concentration of the catalytic subunits in the nucleus is high enough to change the protein synthesis are new synaptic terminals

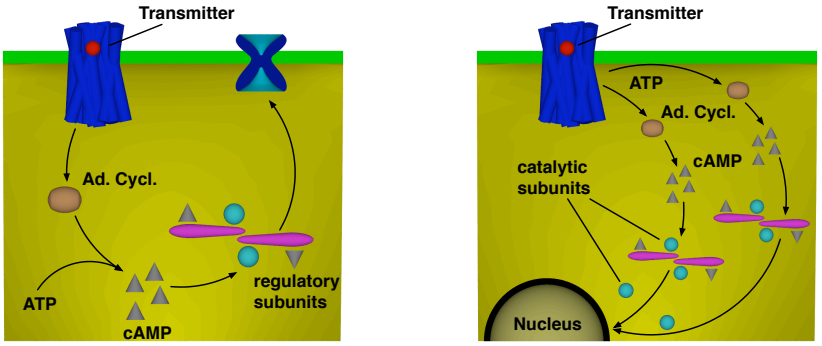


Figure 2.6: STP & LTP. Short-term potentiation (STP, left) and long-term potentiation (LTP, right) are triggered by a neuro-transmitter binding to a metabotropic receptor. The receptor activates an enzyme (adenosine cyclase) which syntheses cAMP from ATP. The cAMP binds to a protein containing catalytic and regulatory subunits. In the case of STP (left) the regulatory subunits influence the ionotropic receptors, to broaden an incoming signal. In the case of LTP (right), the catalytic subunits diffuse into the nucleus and alter the protein generation process. For LTP, compared to STP, a significantly higher concentration of cAMP is needed to free enough catalytic subunits, so that the protein generation process is altered. The image was redrawn by the author of this work, but is taken from (Squire & Kandel, 1998).

grown. The newly grown synaptic terminals are kept for a duration that is much longer than the initiating process. The process of growing new synaptic terminals, triggered by the significant larger concentration of cAMP, is the biological mechanism for LTP.

There is also an intermediate mechanism responsible for STP and LTP. A certain receptor, called NMDA receptor is activated in both cases. Depending an on the post-synaptic concentration of Ca^{2+} either STP or LTP is realised (Malenka & Siegelbaum, 2001).

To conclude the section of the biological mechanisms, we can summarise, that short-term plasticity is a sub-cellular process, in which the increased concentration of second messengers leads to an amplification of a signal.

In contrast, long-term plasticity is the result of a structural change of the neuron due to a significantly higher concentration of second messengers.

An overview of synaptogenesis including formation, maturation, elimination is given by Craig and Shatz (2001). The environmental conditions of dendrite growth is discussed in (Camel et al., 1986). Evidence is given in rat experiments that the environmental conditions determine the amount of dendritic branching (Smythies, 2002). Rats reared in the dark show growth of new dendrites in the visual cortex when exposed to light conditions within two days (Valverde, 1971). Other experiments show that this holds, in general, for dendritic growth when rearing rats first in simple environments and later exposing them to complex environments (Volkmar & Greenough, 1972; Uylings et al., 1978; Fiala et al., 1978; Juraska et al., 1980; Camel et al., 1986; Wallace et al., 1992). That dendritic growth is a mechanism observed in adult animals as a result of sensory input is discussed by Alvarez and Sabatini (2007). Currently, proper synapse formation is discussed as the substrate for cognition, and improper formation as a reason for neurodevelopmental disorders (McAllister, 2007).

This work is concerned with a model of synaptic plasticity and not with a model of structural changes within a neural network. Therefore, the model is considered as STP and hence, as a model for non-declarative learning or memory. Later in this work (see chap. 5 and chap. 6), the transient dynamics of embedded and situated Self-Regulation Neurons are analysed while they control a robot in an environment performing a task. It will be shown that the changes of the synaptic weights as a response to a chaining stimulus are in the range of seconds[†]. Therefore in the presented context, the proposed Self-Regulating Neuron model is understood as a model for non-declarative, short-term potentiation or *short-term memory* for short.

Up to this point, it was discussed how synapses can be modified. Important questions are: when do synaptic modifications occur, when is a synapse potentiated, and when is it depressed? These questions are discussed in the following section.

2.4.3 Synaptic plasticity

Synaptic plasticity is the change of synaptic strength as a response to an event. This section discusses the events which led to a potentiation or a depression of a synapse.

In 1949 Donald Hebb formulated a rule for synaptic plasticity:

> *"When an axon of cell A is near enough to excite a cell B and repeatedly or persistently takes part in firing it, some growth process or metabolic change takes place in one or both cells such that A's efficiency, as one of the cells firing B, is increased."*
>
> (Hebb, 1949)

This formulation is known as the Hebb'ian Learning Rule and states that the synaptic connection between two neurons is strengthened if their activity is correlated. It was summarised later in the popular statement

> *"cells that fire together, wire together"*
>
> (Zigmond et al., 1999)

Synaptic plasticity of this type, where a synapse is modified in response to a correlated pre- and post-synaptic neuron activity is referred to as *homosynaptic* plasticity (see fig. 2.7).

Yet, only potentiating synapses, as described by Hebb, would lead to a destabilisation of synaptic strength. An increased synaptic connection leads to an increased correlation of the pre- and post-synaptic neuron activity, which in return leads to a potentiation of synaptic strength. The connected neurons lose their ability to differentiate as they constantly fire synchronously. Therefore, a mechanism for bi-directional synaptic plasticity must exist.

Experiments have shown that there is a temporal component to the original Hebb'ian learning rule in which the sign of the homosynaptic change depends on the precise timing of the pre- and post-synaptic activity (W. B. Levy & Steward, 1983; Bi, 2002). Stimulating spike pairs with different time delays were introduced to a pre- and post-synaptic neuron. As a result, the synapse was potentiated, if the pre-synaptic spike was introduced before the post-synaptic spike.

[†]Seconds refers to the simulated real-time. The simulations used in this work are faster than real-time, but according to the update frequency of the simulation and the step size of the time used in the simulator, the real-time can be calculated.

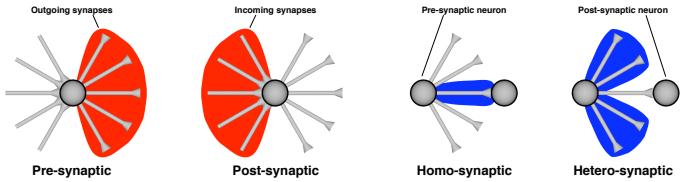


Figure 2.7: Forms of Synaptic plasticity. Different forms of synaptic plasticity, from left to right: *Pre-Synaptic* modifications affect all outgoing synapses, *Post-Synaptic* modification affect all incoming synapses, *Homo-Synaptic* modifications affect the synapse connecting correlated neurons, and *Hetero-Synaptic* modification affect neighbouring synapses of the synapse affected by correlated neurons, here shown for the outgoing synapses of a neuron.

Similarity, the synapse was depressed, if the pre-synaptic spike was introduced after the post-synaptic spike. The degree of potentiation and depression decreased if the time delay between the spikes was increased. The size of the time delay window for potentiation and depression depends on the dendritic location (Froemke et al., 2005). Synaptic plasticity depending on the temporal relationship of pre- and post-synaptic neuron activity is called *spike-timing-dependent synaptic plasticity* (STDP). It was shown that STDP is sufficient to produce competitive Hebb'ian learning in which an increase of one synapse leads to the decrease of neighbouring synapses, without the need of any global regulation methods (Song et al., 2000). This modification of a neighbouring synapse is a spatial relationship, which is referred to as *heterosynaptic modification* (see fig. 2.7) and is defined as the change of a synapse without a correlated pre- and post-synaptic activity, but as a result of a homosynaptic change in the neighbourhood (Turrigiano, 1999; Turrigiano et al., 1998; Royer & Paré, 2003; Castellani et al., 2001).

Spatial modifications occur in two different forms, post-synaptic modifications and pre-synaptic modification (see fig. 2.7). Post-synaptic modifications affect the incoming synapses of a neuron, pre-synaptic modifications respectively affect the outgoing synapses. For post-synaptic homosynaptic potentiation both heterosynaptic depression and potentiation have been observed. Post-synaptic modification is also referred to as *synaptic normalisation* or *competition* and pre-synaptic modification is also referred to as *synaptic scaling* (Turrigiano, 1999; Turrigiano et al., 1998). Possible reasons for synaptic normalisation could be the conservation of the total synaptic strength to prevent synapses from saturation or complete depression (Miller, 1996). Turrigiano (1999) describes synaptic scaling as the result of post-synaptic homeostatic regulation, which is proven to be multiplicative in the form that all incoming synapses are modified proportionally. The reason may be that the neuron remains responsive to inputs from different synapses, and that not one single synapse dominates. This is important during the development of the neural circuit, when new synaptic connections may grow rapidly, as well as at the beginning or end of the circuit formation, when the number of synaptic connections is small or large (Turrigiano et al., 1998).

To conclude this section, there is evidence that synaptic plasticity does not only occur as a local (temporal, homosynaptic) effect, but that local changes affect neighbouring synapses (spatial, heterosynaptic). The spatial effect occurs on outgoing as well as incoming synapses.

The combination of both, homosynaptic and heterosynaptic plasticity have at least two novel

properties compared with the isolated mechanisms (Bailey et al., 2002). First, the duration of the modifications are enhanced in a non-additive way. Second, there is a higher selectivity of synaptic potentiation. As an example, a homosynaptically potentiated synapse with heterosynaptically depressed neighbouring synapses is emphasised in relation to its neighbours.

This section gave an overview of synaptic plasticity as observed in biological systems. They are based on the biological mechanisms discussed in the previous section.

2.5 Summary

This chapter presented the background for this thesis. The concept of Ashby provides the fundamental understanding of adaptive processes. The importance of embodiment in understanding cognitive processes was emphasised in the second section, and the biological relevance was discussed in the last section. In this context, the proposed neuron model is understood as a model for short-term plasticity as defined by Squire and Kandel (1998) with synaptic scaling as described by Turrigiano (1999).

The following chapter presents the methods used in this work. These are related to the presented background given in this chapter.

Chapter 3

Methods

The previous chapter introduced the scientific background of this work. It was discussed that Ashby's concept of ultrastability and his Homeostat are the motivation for the formalisation of the SRN model. The requirement of the sensori-motor loop in understanding basic principles of neural signal processing was motivated by the fields of Behaviour-Based Robotics and Embodied Artificial Intelligence. The relationship between the type of synaptic plasticity, which is modelled by the SRN model, and short-term plasticity by synaptic scaling was discussed in the previous chapter.

This chapter presents the methods and approaches that are used in the remainder of this work. These are dynamical systems theory, recurrent neural networks, including the neuron model of choice on which the SRN model is based, and evolutionary robotics.

3.1 Dynamical Systems Theory

This section motivates dynamical systems theory as a mathematical framework used to model and understand cognitive processes. Other frameworks are available, of which the digital computer model is the most common in the field of artificial intelligence. Both approaches are compared and arguments are given for the use of dynamical systems theory in the following section. This approach of modelling is not new, as already Ashby used similar terminology for his model for the brain (see previous chapter). His notion of a *field* is related to a *phase-space* and a *flow*. Dynamical systems theory is motivated first by means of a comparison with the digital computer metaphor before the terminology which is used in the remainder of this work is defined.

Historically, the brain was compared and described in terms of common technological achievements, or as Rodney Brooks states it:

> *"In my own lifetime I have seen popular 'complexity' metaphors for the brain evolve. When I was a young child, the brain was likened to an electromagnetic telephone switching network. Then it became an electronic digital computer. Then a massively parallel digital computer. And delightfully, in April 2002, someone in a lecture audience asked me whether the brain could be 'just like the world wide web'."*
>
> *Rodney Brooks, foreword to* (Pfeifer & Bongard, 2006)

This quote shows, how the current state of technology has influenced and still influences the way the brain is understood. Common technological achievements such as digital computers define the terminology and the understanding of the brain. The result is that the description of both share the same main characteristics (Elman, 1998; Gelder, 1998):

1. Processing / Cognition is carried out by discrete operations executed in serial order.
2. Memory is distinct from the processor / cognition.
3. Processor operations are / Cognition is described in terms of rules (derived) from programming languages.

Before the two approaches can be compared, the notion of a dynamical system is briefly reviewed. A dynamical system is a set of variables, which numerically capture different system properties. The evolution of a system over time is described by a set of equations. A definition will follow in the next section.

It is known that within the brain, there exists a high level of recurrence on different levels. This has been shown by Felleman and Van Essen (1991) locally between neurons and between cortical areas for the macaque monkey. Concerning the recurrent structure in the brain and the non-linearity of the input-output response of neurons (see sec. 3.2), dynamical effects, such as hysteresis, oscillation, synchronisation and chaos, are very likely to occur. This is supported by studies that have proven the existence of chaos (Skarda & Freeman, 1987) and hysteresis (Kelso, 1995) in the brain. It is, therefore, plausible to assume that dynamical systems theory is a good method of describing cognitive processes (Pasemann, 1996).

It is Gelder (1998)[†] who discusses and compares in detail, the two approaches of dynamical system theory and the digital computer metaphor, and finally concludes why dynamical systems theory is better suited to model the brain. He proposes *The Dynamical Hypothesis*, and discusses four main considerations which favour it over the digital computer metaphor:

1. Natural cognition happens in real time.
2. Dynamical systems theory is the pre-eminent mathematical framework for the description of temporal phenomena.
3. Embeddedness of cognition.
4. Emergence and stability.

These four statements must be elaborated.

Natural cognition happens in real-time: Cognition is an ongoing interactive process with the environment. In many cases, exact timing is essential for the survival of the individual. In predator-prey conditions this can be clearly seen. This is an objection to the discrete operations in serial order of the digital computer paradigm. Planning followed by sequential processing of the actions does not account for fast reactive responses to highly dynamic environments and are not well suited for tasks that require exact timing.

[†]See also Port and Van Gelder (1998).

Dynamical systems theory is the pre-eminent mathematical framework for the description of temporal phenomena: Dynamical system theory is the theory of rates of change of systems (see above and next section). Each system is described by a set of differential or difference equations, which determine the change of the system over time. This argument follows the previous one. When cognition is viewed as a real-time process, and not as a sequence of instructions, then dynamical system theory is the canonical choice to describe it.

Embeddedness of cognition: Every natural cognitive system is embedded at least three times, first in the nervous system, second in the body, and third in the natural environment (Brooks, 1991c; A. Clark, 1996; Cliff, 1990; Pfeifer & Bongard, 2006). Gelder (1998) states that

> *"Dynamical cognition sits comfortably in a dynamical world."*
>
> (Gelder, 1998)

This means that the world is well-described in dynamical terms, since Newton first invented the field in the mid-1600s, when he described the law of motion and gravitation (Strogatz, 1994; R. H. Abraham et al., 1997). Describing embeddedness is a non-trivial task, which increases in difficulty, when fundamentally different systems must be combined.

Emergence and stability: Emergence and stability are the result of self-organising systems. Self-organisation involves the mutual dependence and interaction of a large number of components. Again, dynamical systems theory is the dominant mathematical framework for the description of such systems. In an example of ordinary differential equations, consider each variable, or subset of variables, as a component. Van Gelder states that if the real world is best modelled using dynamical systems theory, then systems embedded in the real world should follow. This relates to Brooks and Clarks objection to micro- and block-worlds (see previous chapter), which are designed to fit the requirements of the model of choice. Scaling from the micro-worlds to the real world application therefore fails (Dreyfus, 1972).

Another advantage is that behaviour can now be understood geometrically, in terms of attractors, transients, stability, bifurcations, chaos, etc.; features which are not available in the classical computational understanding of the brain (Gelder, 1999). These terms are defined in the next section.

Summary

In addition to the argument of the embodied artificial intelligence approach versus that of classical artificial intelligence, this section introduced an alternative to the of latter approach which follows from the digital computer metaphor. An argument for the plausibility of using dynamical systems as a mathematical framework was given and the manner in which it differs from the classical paradigm was presented. In the next section, previously used terminology is technically defined and supplemented with the terminology used in the remainder of this work.

3.1.1 Terminology

The previous section discussed dynamical systems theory as a well suited mathematical framework to understand and describe cognitive processes. This section presents the terminology used

in the remainder of this work. It may be omitted if the concepts of dynamical systems theory are already known.

A *dynamical system* is described as a set of states, and a rule that determines its evolution over time (Stewart, 1999; Ott, 1993; Strogatz, 1994; Thomson & Stewart, 2002; Abraham & Shaw, 1992; Alligood et al., 1996; Champneys et al., 2007; Izhikevich, 2007; Arrowsmith & Place, 1990), i.e. how it progresses from one state to the next. This section will give a more precise definition of a dynamical system, including its state and deterministic rule, but first a few basic concepts, required for the definition, are presented.

A *system's variable* or *variable* for short, is a measurable quantity which has a definite numerical value (Ashby, 1954). It is denoted with $x_i, i \in \mathbb{N}$, where i is the index of the variable. The variable itself is typically a real number ($x_i \in \mathbb{R}$), but other domains such as e.g. binary ($x_i \in \{0, 1\}$) and integer ($x_i \in \mathbb{Z}$) are possible. For generality $x_i \in \mathbb{R}$ is assumed here. With $\vec{x} = \{x_i\}_{i=0,1,\ldots,n-1}$ we denote the set of variables, and $|\vec{x}| = n$ is its dimension. Every variable is a function of time, and the notation $x_i(t)$ is used to emphasis this. It refers to the numerical value of the variable x_i at time $t \in \mathcal{T}$. In the case of $\mathcal{T} = \mathbb{N}$ we speak of discrete-time systems, and in the case of $\mathcal{T} = \mathbb{R}$ of continuous-time systems. The numerical values of the system's variables at a certain instant in time $\vec{x}(t)$ is called the *system's state*. The set of all possible states of a system, is its *phase space*.

If the system $\vec{x}$ is released from an initial state $\vec{x}(t_0)$, and its progress over time is captured, then the evolution of the system leads to an infinite set of states depending on the initial condition. This set is referred to as a *trajectory* of the system.

A *vector-field* in a phase-space is a function f that assigns to each point $\vec{x}$ in the phase-space a tangent vector $f(\vec{x})$ (Bamon & Roussarie, 1996; Thomson & Stewart, 2002). How f is described is presented later in this section. At this point, it suffices to think of it as a function in the vector space $f\colon \mathbb{R}^n \mapsto \mathbb{R}^n$, which describes the progress of the system depending on a state.

A trajectory in a phase-space is now formed by a piecewise addition of infinitesimal steps in the direction indicated by the vector-field (Thomson & Stewart, 2002). Hence, the vectors defined by the vector-field are the tangents of the trajectory. This leads to the important property, that no two trajectories cross, as a crossing would mean that the function $f(\vec{x}(t_i))$ has two solutions for the state $\vec{x}(t_i)$. As deterministic systems are assumed here (in contrast to e.g. stochastic systems), this is not the case.

As the vector-field determines non-crossing trajectories that cover the entire phase-space, it is said to define a *flow* in phase-space (Thomson & Stewart, 2002), which is the set of all trajectories.

A *dynamical system* is now defined by its phase-space and its vector-field. The phase-space covers all possible states of the system, and the vector-field describes its dynamics.

Up to this point, the process of obtaining a vector-field has not been discussed. This process relates to the function f which was previously introduced. Continuous-time dynamical systems are defined by a set of first-order *differential equations*, and discrete-time dynamical systems are defined by an *iterated map* which is given by a set of *difference equations*:

Continuous-time			Discrete-time		
$\dot{x}_1$	$=$	$f_1(x_1, \ldots, x_n, \vec{p})$	$x_1(t+1)$	$=$	$g_1(x_1(t), \ldots, x_n(t), \vec{p})$
$\vdots$			$\vdots$		
$\dot{x}_n$	$=$	$f_n(x_1, \ldots, x_n, \vec{p})$	$x_n(t+1)$	$=$	$g_n(x_1(t), \ldots, x_n(t), \vec{p})$

The vector $\vec{p}$ is a set of parameters. Different interpretations of $\vec{p}$ are possible. Socolar (2006) describes the set of parameters as externally imposed functions of position, which occur if a system operates in an inhomogeneous environment. Pasemann (1996) describes parameters in the case of neuro-modules. The parameters are not used directly for calculation, but determine the mode in which a neuro-module processes incoming signals to compute an output. He demonstrates this with the XOR-example, which can switch between OR, XOR and NAND, depending on a system parameter p.

In this context the parameters are understood as discussed in the previous chapter. They combine sensor input as well as system inherent properties, which in this case are the synaptic weights.

It was noted earlier in this section, that no two trajectories cross, but two trajectories may asymptotically approach the same point in the phase-space and one trajectory may self-intersect. The first case refers to an attractor and will be discussed later in this section. The latter case corresponds to a periodic motion. The minimum set of points of a trajectory that satisfies the properties of a trajectory is called the *orbit* of the system (R. H. Abraham et al., 1997). When finite, an orbit is called *cyclic* or *periodic* and the number of points is its *order*. A special case of an orbit is a period-1 orbit, called a *fixed point* (R. H. Abraham et al., 1997). It is defined as a point in the phase-space, that maps to itself:

$$\vec{x} = f(\vec{x}).$$

If a fixed point attracts trajectories that start nearby, it is called an *attracting fixed point*. This relates to the previously mentioned case of trajectories which asymptotically approach the same point, but do not cross.

An asymptotically stable fixed point is called a *fixed point attractor*. The generalisation of this concept is an attracting subset in the phase-space, which is called a *period-n attractor*, where n corresponds to the cardinality of the subset.

The *basin* of an attractor is the set of all points in the phase-space, to which trajectories are attracted by the attractor (R. H. Abraham et al., 1997).

The boundaries of basins are called *separatrices* or *frontiers*. We can now return to the initial definition of the fixed point, and classify different types of fixed points, namely, stable (see above), unstable, saddles and limit cycles. Stable and unstable fixed points are also called sinks and sources, due to their geometric interpretation of flows originating and ending in them.

An *unstable fixed point* repels trajectories within its neighbourhood, and then moving outwards. A *saddle* has at least two separated basins. There are two types of saddles, those which attract trajectories which start in the separatrices and repel those which start in the in the basin[†], and those which repel trajectories which start in the separatrices and attract those

[†]R. H. Abraham et al. (1997) define basins which points diverge as *basin of infinity*.

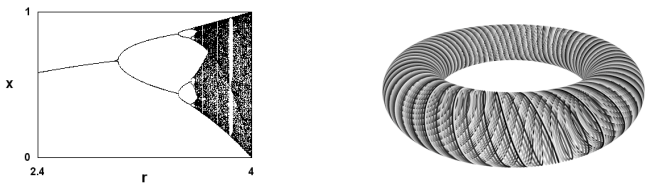


Figure 3.1: Logistic map / Quasi-periodic attractor. Left: A simple example for a discrete dynamical system with one control parameter. Depending on the value of the parameter, the behaviour of the system changes qualitatively. From left to right, the system evolves from a fixed point, over a period doubling route to chaos. For a discussion see the text below. Right: An example of a quasi-periodic attractor. The trajectory colour varies from black to light grey. The colouring shows that the trajectory is close to itself, but does not self-intersect. For a periodic motion with e.g. two periods, this occurs, if the quotient of the two periods is irrational (Thomson & Stewart, 2002).

which start in the basins. A *limit cycle* is a *periodic orbit* of period larger than one, which can be either repelling or attracting. An *asymptotically stable limit cycle* is called a *periodic attractor*. A trajectory that leads towards an attractor is called a *transient*.

We have now defined fixed points and periodic attractors. There are two more types of attractors, namely chaotic and quasi-periodic attractors.

For the chaotic attractor we first need to define chaos. Chaos is a loose generic term for complex, seemingly irregular motions of deterministic dynamical systems, characterised by a sensitive dependence on initial conditions (Thomson & Stewart, 2002). In other words, two trajectories starting near one another will not be correlated any more after some short period of time. This can be specified by the *Liapunov exponent*. The idea of the Liapunov exponent is as follows: let x_0 denote some initial condition and δ_0 be an extremely small initial separation from the initial condition: $x_0 + \delta_0$. Let δ_n be the separation after n iterations. If $|\delta_n| = |\delta_0|e^{n\lambda}$, then λ is called the Liapunov exponent (Strogatz, 1994). A positive exponent means that neighbouring trajectories separate extremely fast, so that there is a time horizon beyond which predictions break down (Strogatz, 1994). A negative exponent is evidence of a periodic attractor, a positive one of a chaotic attractor, and an exponent equal to zero of a quasi-periodic attractor. Quasi-periodic refers to an orbit that seems periodic, because it is asymptotically close to itself, but does not self-intersect as described for the periodic attractors. An example is a trajectory on a torus that does not self-intersect (see fig. 3.1 [right hand side]).

After presenting the terminology, a standard example for a one-dimensional discrete-time dynamical system is given; the logistic map (see fig. 3.1 [left hand side]). It is defined as

$$x(t+1) \quad = \quad rx(t)(1-x(t)), \qquad x, r \in \mathbb{R}, t = 0, 1, \ldots$$

In this example r is called the *system's parameter* or *control parameter*. For different values of r, the behaviour changes significantly. From a fixed point attractor, to a period-2 attractor, and finally over a period doubling route to chaos (see fig. 3.1 left, from left ($r = 2.4$) to right ($r = 4$)). The points, i.e. the values of r, for which the qualitative change of behaviour occurs are called a *bifurcation points*. The diagram (see fig. 3.1) is, therefore, referred to as a *bifurcation*

diagram.

For every continuous dynamical system, a discrete dynamical system can be derived by means of a stroboscopic imaging of the continuous dynamic, which is also known as Poincaré-mapping (Thomson & Stewart, 2002). On the other hand, there is no single continuous solution to a discrete dynamical system.

Only discrete dynamics are considered in the following work. This is consistent with previously made statements, as this work is concerned with the qualitative behaviour and the understanding of the underlying principles. Therefore, choosing a discrete representation is of no disadvantage. It is rather a canonical choice when numerical simulations are required. In such a case, a continuous-time system must be discretised and approximated with methods such as Euler integration or Runge-Kutta (Strogatz, 1994). The use of discrete-time systems avoids this form of approximation.

This section presented dynamical systems theory as a well suited mathematical framework used to model cognitive systems. The method used to construct a dynamical system, which is embedded in the sensori-motor loop using the morphological and environmental properties by the means of minimal cognitive systems (Beer, 1996), is not clear. Furthermore, the combination of structurally coupled non-linear systems is mathematically difficult (Strogatz, 1994). Therefore, this work uses artificial recurrent neural networks understood as dynamical systems, and artificial evolution as an algorithmic construction method. The following two sections of this chapter will present these two approaches.

3.2 Artificial Recurrent Neural Network

Artificial neural networks were first introduced by Rosenblatt (1958) with a simplified neuron model. Motivated by the action potential of biological neurons, a neuron in a perceptron network is a binary switch, where the value one relates to a firing of the neuron and zero to a silent neuron. The input of each neuron is the weighted sum of the neurons of the previous layer. Each neuron has a threshold value. If the input is larger than the bias, the neuron is activated (output of one), otherwise it remains inactive (output of zero). A perceptron network is a strictly layered feed-forward network of two layers and was initially used for pattern recognition. An iterative learning method was introduced, which allowed the modification of the weights of the network with respect to a training set. It was shown by Rosenblatt that the learning algorithm converges in finite time.

In 1969 Minsky and Papert showed that a perceptron network is limited in possible applications, as it could only serve as a linear separator of the input space (Minsky & Papert, 1969). A perceptron network is not able to solve the XOR problem. This halted research in the field of artificial recurrent neural networks until the early 80's, when Rumelhart introduced a different feed-forward neural network architecture together with a different neuron model and the back-propagation learning algorithm (D. E. Rumelhart, Hinton, & Williams, 1986; D. Rumelhart et al., G1986; D. E. Rumelhart, McClelland, & the PDP Research Group, 1986). In this new structure, neurons had a non-linear but differentiable transfer-function, allowing a gradient-descent-based training of the networks.

This section introduces the neuron model on which the Self-Regulating Neuron model is

based. First, an overview of the properties of biological neurons is provided.

3.2.1 Biological Neurons

It was Ramón y Cajal who first discovered that the brain was not a syncytium, a continuous mass of fused cells sharing a common cytoplasm[†], but a large network of individual, signalling nerve cells, which connect via synapses with each other (Cajal, 1892) [‡] and (Cajal, 1906). Helmholtz and DeBois-Raymond discovered that the electrical activity of the nerve cells provide the mechanism for information transmission between neurons (Helmholtz, 1850; Bois-Reymond, 1848)[‡].

With modern brain imaging techniques, the brain structure, at the more global scale, is better understood. Not only is it layered, but also modularly organised in cortical areas for specific brain functions (Kandel et al., 2000). This work is not concerned with brain models but with a model of synaptic plasticity at the cellular level. Therefore, only a model for an individual biological neuron is of interest, and not the modular structure within in a biological brain, of which it is a part. The following section covers the basic properties of a single biological neuron, and mathematical neuron models.

Properties of biological neurons

A neuron is a single cell, which can be divided into three subsections (see fig. 3.2 C), the dendrites, the cell body (soma), and the axon (Arbib, 1995). The dendrites can be considered as input devices of the neuron. The number of connections to a single neuron over the dendrites is neuron cell-specific but can reach up to 100,000 (Dayan & Abbott, 2001). The neurons connected to the dendrites are referred to as pre-synaptic neurons. The electrical signals over the dendrites together change the membrane potential. When it is raised above a certain threshold value, the neuron releases an electric signal over its axon to up to 180 other nerve cells (Dayan & Abbott, 2001). Note that a single pre-synaptic neuron has several connections to a post-synaptic neuron. The resting membrane potential is -70mV and is kept constant through ion channels, also called receptors (see fig. 2.5), which maintain different concentrations of predominantly sodium (Na^+), potassium (K^+), calcium (Ca^{2+}) and chloride (Cl^-) inside and outside of the nerve cell. When an electric signal reaches the nerve cell body, the membrane potential is either increased (depolarisation) or further decreased (hyperpolarisation). When the depolarisation reaches approximately -55mV to -50mV, the neuron generates an electric signal called an *action potential*, which is a 100mV spike of about 1ms duration. The action potential is carried by an electro-chemical process over the axon to the other connected, so called post-synaptic cells.

A few milliseconds after the release of an action potential, no new one may be released. This time is called the refactory period (Kandel et al., 2000). When the action potential moves along the axon, it finally reaches the synaptic cleft, the connection point between the pre- and post-synaptic neuron, where it releases synaptic vesicles. Depending on the ions held by the vesicles, the membrane potential of the post-synaptic neuron is either increased or decreased. In the case

[†]The cytoplasm, briefly stated, is the fluid filling a cell.
[‡]Cited from Kandel, Schwartz, and Jessell (2000).

of depolarisation, the synapse is referred to as an excitatory synapse; in the case of hyperpolarisation, as an inhibitory synapse. The amount of released vesicles and the number of connection points (synaptic clefts) determine the strength of the depolarisation or hyperpolarisation.

This section provided an overview of the biological properties of a neuron. For a more detailed description the reader is referred to *Biological Systems* (see sec. 2.4) and Kandel et al. (2000). The following section introduces the mathematical neuron model.

3.2.2 Mathematical neuron model

There are different types of neuron models of which the most dominant are, single-compartment versus multi-compartment and spiking versus firing-rate neuron models. The first two, single- and multi-compartment, differ in the modelling of the membrane potential.

In the single-compartment model, only one membrane potential is taken into account for the entire neuron. In the multi-compartment case, the neuron structure, and especially the dendrites are considered as a sequence of discrete compartments, each with its own membrane potential. The membrane potential is then calculated along the branches towards the cell body as a function of neighbouring compartments (Dayan & Abbott, 2001).

Firing-rate neuron models only account for the spike frequency, which is modelled by a single scalar value, e.g. in the leaky integrator models (Arbib, 1995). Seung (2007) draws an analogy to light in order to understand the assumption underlying the model. Light comes quantised in photons, equivalent to the action potentials or spikes arriving at the post-synaptic neuron. But light intensity is well specified by the rate of photons, when the number of photons is large. It is similar with the rate encoding of spikes. If the number of incoming spikes is assumed to be large, the rate is a sufficient description.

For spiking neurons, both the spike frequency and the timing of the spike reaching a post-synaptic neuron are essential to the model. It is assumed that both the frequency and the timing are an important part of the information processing (Gerstner & Kistler, 2002b). An overview of models and tools for spiking neurons is given by Brette et al. (2007).

For each of the four possible combinations, very different levels of detail are available. The preferability of a model depends on the properties of the biological neuron which are of interest (Rabinovich et al., 2006).

This works is not interested in the dynamics within a single neuron, nor with the information encoding in spike trains, but in principles of neural signal processing. Therefore, the simplest neuron model, incorporating the dynamical features discussed in the previous section, is preferred. It is the standard additive neuron model, which fully satisfies the requirements and which is discussed in the following section. It is a single-compartment standard additive neuron model with a sigmoidal transfer-function, which can be interpreted as a rate encoding model.

Standard additive neuron model

A single neuron is considered to have inputs from other neurons, connected via incoming synapses, an internal activation and an output which is propagated over outgoing synapses to other neurons (see fig. 3.2 B). The input synapses are assumed to be current sources which either exhibit (depolarise) or inhibit (hyperpolarise) the cell of interest. The internal activation

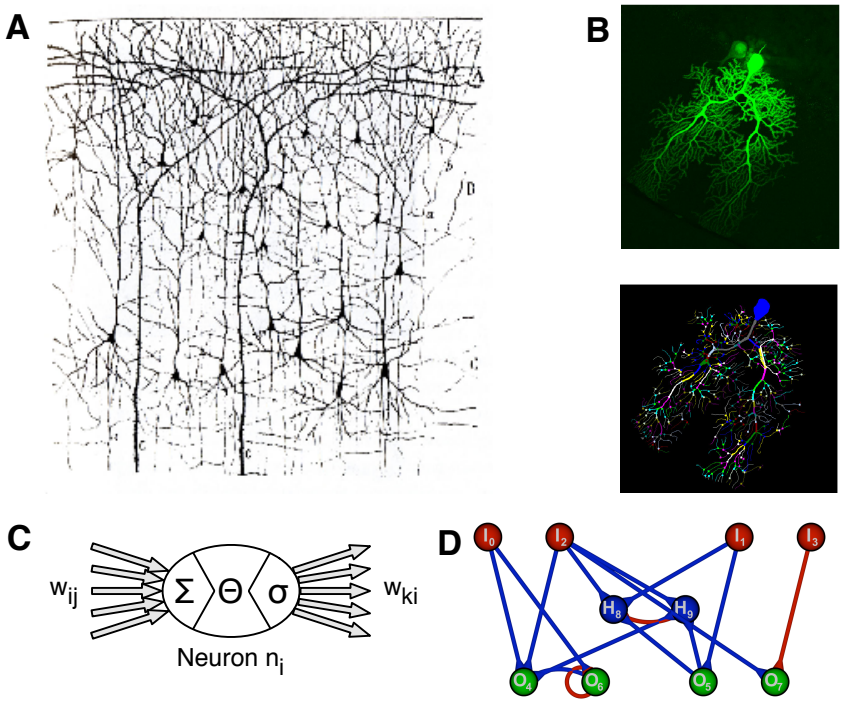


Figure 3.2: Biological neuron and mathematical modelling. The upper half (denoted with A and B) of this figure shows images of biological neurons. A shows an image of layers of connected pyramidal cells of the human visual cortex created by Cajal. The image is taken from Braitenberg (1984). The two figures on the right-hand side (B) show a purkinje neuron injected with Lucifer Yellow and the schematics of the dendrites extracted from the image. Both images are taken from Cell Centered Database provided by the University of California (Martone et al., 2007). The two lower figures (C and D) show, on the left-hand side (C), schematics of the artificial neuron, and on the right-hand side (D), colour coding of a neural network, used in the remainder of this work. Input neurons are coloured in red, output neurons in green, and hidden neurons in blue. Inhibitory synapses are visualised in red, excitatory in blue, respectively. If a bias is used, it is shown by a bias neuron and a grey synapse (not shown).

of the neuron is related to the membrane potential of a biological cell. It is calculated as the weighted sum over the incoming synapses and an internal bias value, which is related to the threshold membrane potential for the release of an action potential. The discrete time equation for a single neuron is then given by:

$$a_i(t+1) \quad = \quad \Theta_i + \sum_j w_{ij} o_j(t) \qquad \text{where}$$

$a_i(t) \in \mathbb{R}$	is the activation of the i-th neuron at time t
$\Theta_i \in \mathbb{R}$	is the constant bias value of the i-th neuron
$w_{ij} \in \mathbb{R}$	is the strength of the synaptic connection from neuron j to neuron i
$o_j(t) \in \mathbb{R}$	is the output of the j-th neuron at time t
$i, j \in \mathbb{N}$	are the neuron indices

$t = 0, 1, \ldots$ is the time index

The synaptic strength reflects the number of synaptic vesicles released for every action potential, and the output of a neuron reflects its firing-rate. It is calculated from the activation by the transfer-function, denoted with $f(x)$. There are several common transfer-functions in literature, the binary, linear and sigmoidal. Historically the first proposed function is the binary transfer-function used in the perceptron model (McCulloch & Pitts, 1943):

$$\begin{aligned} o_i(t) &= f(a_i(t)) \\ f\colon \mathbb{R} \mapsto \{0,1\}, \quad f(x) &:= \begin{cases} 1 & x > \vartheta \\ 0 & x \leqslant \vartheta \end{cases} \end{aligned}$$

where ϑ is the threshold value. An output of one symbolises one released action potential, and an output of zero a silent neuron (no action potential relaesed). The limitations of this model were discussed by Minsky and Papert (1969) and in the previous chapter (see sec. 2.1).

The second model is the bounded linear model:

$$f\colon \mathbb{R} \mapsto [0,1], \quad f(x) := \begin{cases} 0 & x \leqslant 0 \\ 1 & x \geqslant 1 \\ \mathrm{id}(x) & \text{else} \end{cases}$$

This model already related to a normalised firing-rate.

The third model is a non-linear transfer-function, the standard sigmoid:

$$f\colon \mathbb{R} \mapsto \mathbb{R}, \quad f(x) := \frac{1}{1+e^x}$$

This function has two main advantages. First, it is a non-linear function. Second, if the output of a neuron is considered as the firing-rate of action potentials, then this transfer-function is a biologically plausible choice. Stein (1967) describes the frequency-current curve of the mean frequency of action potentials of neurons excited by an external current to be markedly non-linear. More recent research shows that the relation of injected current into a cell, and the number of spikes per second, are in fact sigmoidal. Although the publication by Wilson et al. (2004) examines the effect of different blockades of specific ion channels, the plots of the control neuron clearly shows a sigmoidal frequency-current curve. Biologically, the upper boundary is the result of the refractory period, while the lower boundary is given by the spontaneous activity which occurs when there is no input signal and the membrane potential is at its resting potential (Tsodyks et al., 1999).

In robotics applications it is more common to use the hyperbolic tangent as a transfer-function because it also delivers negative output values, which can be used to directly drive the motors of a robot. It is given by the following equation:

$$f\colon \mathbb{R} \mapsto \mathbb{R}, \quad f(x) := \frac{e^x - e^{-x}}{e^x + e^{-x}}$$

Pasemann (1993) showed that a single neuron with the hyperbolic tangent transfer-function has the same dynamical properties as the single neuron with the standard sigmoid transfer-function.

i_i, I_i	—	The i-th input neuron, where i is a global index over all neurons.
o_i, O_i	—	The i-th output neuron, where i is a global index over all neurons.
h_i, H_i	—	The i-th hidden neuron, where i is a global index over all neurons.
n_i	—	The i-th neuron, where i is a global index over all neurons.
$\mathcal{I}$	—	The set of all input neuron indices of the network.
$\mathcal{O}$	—	The set of all output neuron indices of the network.
$\mathcal{H}$	—	The set of all hidden neuron indices of the network.
$\mathcal{N}$	—	The set of all neuron indices: $\mathcal{N} = \mathcal{I} \cup \mathcal{O} \cup \mathcal{H}$.
$\mathcal{C}$	—	Connectivity matrix $\mathcal{C} = (c_{ij})_{i,j \in \mathcal{N}}, c_{ij} \in \{-1, 0, 1\}$.
$\mathcal{W}$	—	Weight matrix $\mathcal{W} = (w_{ij})_{i,j \in \mathcal{N}}, w_{ij} \in \mathbb{R}$.
$\mathcal{T}$	—	The ordered set of time indices: $t \in \mathcal{T} = \{0, 1, \ldots, T-1\}, \lvert\mathcal{T}\rvert = T$.
$\tau(x)$	—	The hyperbolic tangent transfer-function: $\tau(x) = \tanh(x), x \in \mathbb{R}$.
r	—	A recurrent neural network of arbitrary structure, $r = \{n_i, w_{ij}\}$, $i, j \in \mathcal{N}$, where $w_{ij} \in r$ if and only if a connection from neuron n_j to n_i exists, whereas for $\mathcal{C}$ and $\mathcal{W}$ the corresponding cell is set to zero if $w_{ij} \notin r$.
$N(r)$	—	The number of neurons in the neural network r, $N(r) = \lvert\{n_i \lvert n_i \in r\}\rvert$, $i \in \mathcal{N}$.
$W(r)$	—	The number of synapses in the neural network r, $W(r) = \lvert\{w_{ij} \lvert w_{ij} \in r\}\rvert$, $i, j \in \mathcal{N}$.
$W(n_i)$	—	The number of synapses connected to neuron n_i, $W(n_i) = \lvert\{w_{ij}, w_{ji} \lvert w_{ij}, w_{ji} \in r\}\rvert$, $i, j \in \mathcal{N}$.

Table 3.1: Neural Network Conventions. The conventions given in the table above are used in the remainder of this work to describe neural networks.

For any network, there exists a homeomorphism to transfer one to a dynamical equivalent of the other (Pasemann, 2002). Of course the pre- and post-processing must be adapted due to the different domains of the functions. Pre-processing refers to the mapping of the sensor values to the working domain of the transfer-function, while post-processing refers to the mapping of the output neurons to the motor system.

A neural network is a layered structure of neurons. In this work three different types of neurons are distinguished, input, output, and hidden neurons (see fig. 3.2 D). Input neurons receive signals from the sensory system of a robot. They are implemented as buffers, hence their transfer-function is the identity and only outgoing connections of the input neurons are allowed into the network. Output neurons are motor driving neurons. Their transfer-function is the hyperbolic tangent. Both, incoming and outgoing connections are allowed. Hidden neuron are equivalent to output neurons, but their values are not fed into the motor system of the robot.

This section closes with conventions that describe a neural network, which are used in the remainder of this work (see tab. 3.1). The connectivity and weight matrix $\mathcal{C}, \mathcal{W}$ are given in the

following form. Neurons are listed in the order input, output, and hidden neurons. Let $i_r, \ldots, i_s$ be the set of input neurons, then the matrices are (exemplary for $\mathcal{W}$) given by:

$$\mathcal{W} = \begin{pmatrix} w_{ir} & \cdots & w_{is} & w_{ii} & \cdots & w_{ij} & w_{ik} & \cdots & w_{il} \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ w_{jr} & \cdots & w_{js} & w_{ji} & \cdots & w_{jj} & w_{jk} & \cdots & w_{jl} \\ w_{kr} & \cdots & w_{ks} & w_{ki} & \cdots & w_{kj} & w_{kk} & \cdots & w_{kl} \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ w_{lr} & \cdots & w_{ls} & w_{li} & \cdots & w_{lj} & w_{lk} & \cdots & w_{ll} \end{pmatrix}$$

The next section introduces and discusses artificial evolution in the context of evolutionary robotics as an algorithmic method to generate recurrent neural networks of arbitrary structure to solve a task, given the morphology of the agent and environment within which it is situated.

3.3 Artificial Life and Evolutionary Robotics

The previous section discussed dynamical systems theory as a mathematical framework to understand cognitive processes, and the standard additive neuron model as the abstraction of choice for a biological neuron. Formulating a dynamical system, such as a recurrent neural network, which shows a desired behaviour in the sensori-motor loop is a very difficult task, as non-linear elements have to combined (Strogatz, 1994). Therefore, evolutionary robotics, which is a stochastic search, is chosen as an algorithmic method to find solutions. An advantage of artificial evolution is that it enables the discovery of new solutions and principles if the search space is kept open.

Evolutionary robotics emerged from the field of Artificial Life. John von Neumann is considered the father for Artificial Life, or ALife for short. He believed in, and first described, self-replicating machines (Neumann, 1966, 1951; S. Levy, 1992; J. F. Walker & Oliver, 1997).

ALife is the science devoted to understanding life, not through its biology or chemistry, but through the fundamental dynamical principles underlying biological phenomena (Langton et al., 1992). The goal is to recreate life in other physical media, opening them to new kinds of experimental methods and testing. Farmer and Belin (1992) even claim that it will be possible to create a new species within the coming 100 years. The consequences of such prognoses and their likelihood to fail, is discussed by Dreyfus (1992) in the case of classical artificial intelligence.

ALife covers a large range of sub-disciplines. A few examples are self-replication, the search for the origin of life, evolution, and colony dynamics. The list can easily be extended. Of interest here is artificial evolution in the context of robotics, which is referred to as evolutionary robotics (Nolfi & Floreano, 2000).

Artificial evolution, in general, is a population-based stochastic, search method. A generation is given by a population of different potential solutions. An evaluation criteria, the fitness-function, determines the quality of the solution with respect to the given problem. A rank or fitness-based selection method determines the parents for the next generation. A variation operator then varies the offspring, resulting in a new generation, and hence a new population of individuals is evaluated.

Evolutionary algorithms is dated back by Fogel (2008) to the work of Friedman (1956) on evolutionary robots, Box (1957) on simulating evolution for industrial plant productivity, Friedberg (1958) on evolving computer programs, Rechenberg (1965) and Schwefel (1965) on evolving physical devices, Kaufman (1967) on evolving mathematical systems, and Conrad (1969) on simulating ecosystems.

The most dominant approach in the field of evolutionary robotics are genetic algorithms, which was first introduced by Holland (1974). In his approach bitstrings of fixed length encode a set of parameters. A bitstring is referred to as a chromosome, and the subsections of the chromosome, typically also of fixed length, relate to genes. The bit sequence of each gene is either converted to an integer or real value number, which codes one parameter of the system that is evolved. For neural networks, the structure is kept fixed, and the genes encode the synaptic weights and biases. A chromosome is evaluated against a fitness-function, which determines the quality of the solution parametrised by the genes. From a selected parent population, offspring are created through the crossover of two parents, determined by their fitness values (Holland, 1974). Crossover is performed by identical copies of the parents, which are then modified by exchanging bit sequences of the same length and positions among the copies. A mutation operator switches the value of one bit. The probability of mutation is kept small. Typically, for every individual, only one bit is mutated.

Koza (1992), a former student of Holland's, later extended the approach to genetic programming, a method which directly generates computer programs through artificial evolution. For an overview of evolutionary algorithms, the reader is referred to Bäck (1996) and Goldberg (1989).

In this work, the goal is to generate neural networks of arbitrary structure, which enable the full analysis of the underlying dynamics. A solution should not be (partially) predefined by a fixed neural network structure. Therefore, a different approach is required. The ENS^3 (evolution of neural systems by stochastic synthesis), introduced by Dieckmann (1995), is a method used to simultaneously evolve the structure and parameters of a recurrent neural network. The method is explained in detail in the following section.

3.3.1 Evolution of Neural Systems by Stochastic Synthesis – ENS^3

Genetic Algorithms (see previous section) are applied to neural networks only in the parameter space (weights and biases). The structure of the network is predefined by the experimenter and evolution is performed in a batch process without any interaction. An inappropriately chosen structure limits the quality of the generated solutions to the task of interest and the quality of the solutions can only be evaluated by the experimenter after the batch process has terminated. This leads to experiments in which recurrent neural networks with a small number of neurons are evolved first. Typically, the neural networks are fully connected, i.e. every neuron is connected to every other neuron and itself. If the solutions are not satisfactory, the number of neurons is slowly increased until a good solution is found.

The evolutionary strategy ENS^3, first described by Dieckmann (1995), supports the simultaneous variation of the structure and the parameters of arbitrary recurrent neural networks. The method is similar to the GNARL algorithm (Angeline et al., 1994). First, a brief overview of ENS^3 is given, followed by the formal description.

Overview

Like other evolutionary strategies, ENS^3 operates on populations of solutions which undergo a variation-evaluation-selection loop. A population $p(t)$ is a set of (recurrent) neural networks r of arbitrary structure

$$\begin{aligned} p(t) &= \{r_0(t), r_1(t), \ldots, r_{P-1}(t)\} \\ |p| &= P \\ t &= 0, 1, \ldots \end{aligned}$$

The time index t denotes the generation of the population, hence, $p(0)$ is the initial population. The elements of $p(t)$ are also referred to as individuals. A network is denoted by $r_k(t)$, and a neuron of this network is denoted by $n_i \in r_k(t)$. The conventions set out in the previous section (see tab. 3.1) are used here.

A variation operator V modifies the structure of a network by insertion and deletion of neurons and synapses, and modifies the parameters of a network by changes of the bias and weights. Insertion and deletion of neurons occurs only on the set of hidden neurons. The number of input and output neurons is predefined in accordance with the given morphology.

An evaluation operator E assigns a single scalar to each individual of the population. The scalar, named fitness value, reflects the quality of the solutions with respect to a given task. Here, a fitness-function calculates the fitness value while the robot acts in an environment. This is elaborated later in this chapter (see sec. 3.3.2) and in the appendix (see app. A).

A selection operator S determines a subset of the current population which will survive and form the parents of the next generation.

A new generation is then given by the mapping:

$$p(t) \mapsto SEVp(t)$$

Formal description

The following formalism is a summary of the publications of Dieckmann (1995), Hülse et al. (2004), Pasemann et al. (2001), complemented by the author of this work.

Before the operators are discussed, a description of the parameters is required. The parameters control the operators and are open to user interaction. They are listed together with some additional notation found in table 3.2.

Variation: The variation operator V is a selection of operators, which act on the k-th individual of the population $p(t)$. The operators are denoted by N_k^+, N_k^- (insertion and deletion of neurons), C_k^+, C_k^- (insertion and deletion of synapses), N_k^*, C_k^* (variation of biases and weights). The variation operator is then given by:

$$V : p(t) \mapsto \prod_{k=0}^{P-1} C_k^* C_k^+ C_k^- N_k^* N_k^+ N_k^- p(t)$$

The parameter and structure variation operators $N_k^{*,+,-}$ and $C_k^{*,+,-}$ are of stochastic character. Each operator is explained in the following.

Used internally as part of the algorithm.	
X_k	Operator on the k-th individual of a population.
$u(x)$	Uniform distributed random variable, $u(x) \in [0, x]$.
$g(x, \sigma)$	Gaussian distributed random variable, with mean x and deviation σ.
Evolution control parameters, open to user interaction.	
$p(X)$	Probability function of the operator X, $p(X) \in [0, 1]$.
p_C	Percentage of neurons connected to an inserted neuron, $p_C \in [0, 1]$.
w^*	Initial weight for an inserted synapses.
Δw^*	Maximal variance of the initial weight, $w_{ij} \in [w^* - \Delta w^*, w^* + \Delta w^*]$.
Θ^*	Initial bias for an inserted neuron.
$\Delta\Theta^*$	Maximal variance of the initial bias $\Theta_i \in [\Theta^* - \Delta\Theta^*, \Theta^* + \Delta\Theta^*]$.
Δw	Maximal variance of existing weights $w_{ij} \mapsto w_{ij} \in [w_{ij} - \Delta w^*, w_{ij} + \Delta w^*]$.
$\Delta\Theta$	Maximal variance of existing biases $\Theta_i \mapsto \Theta_i \in [\Theta_i - \Delta\Theta^*, \Theta_i + \Delta\Theta^*]$.

Table 3.2: Evolution parameters and variables. The table shows the symbolic representation of parameters and their explanation. The parameters are either used as part of the algorithm or open to user interaction in order to control the evolution. The probabilities $p(X)$ are given without the index k for the operator X as the probabilities are equal for every neural net $r_k(t) \in p(t)$. The functions $u(x)$ and $g(x, \sigma)$ are given with an index in the text below, where (if not otherwise noted) the index k of the function refers to an evaluation of the function for every individual $r_k(t) \in p(t)$, and the index i and ij to every neuron $n_i \in r_k(t)$ and synapse $w_{ij} \in r_k(t)$ of every individual $r_k(t) \in p(t)$, respectively.

The neuron deletion operator N_k^-: acts only on the set of hidden neurons. A uniformly distributed random variable determines, for each hidden neuron $h_i \in r_k(t)$, if it is deleted. For a deleted hidden neuron all synapses incident to and from it will be removed with the neuron. For the i-th neuron the operator is given by:

$$N_k^- : r(t) \mapsto r(t) \backslash \{h_i, w_{ji}, w_{ij} | \forall j \in \mathcal{N}, \forall i \in \mathcal{H} : u_i(0, 1) < p(N^-)\}$$

where $u_i(x)$ is value of the random variable for the i-th hidden neuron $h_i \in n(t)$.

The neuron insertion operator N_k^+ inserts a number of hidden neurons which is proportional to the number of already existing neurons. For each new neuron, a random set of synapses is inserted. Each new synapse is initialised with a Gaussian-distributed random variable around the defined initial weight value w^*. Let h_i^+ be an inserted hidden neuron, and w_i^+ be an inserted synapse (either incident to or from neuron h_i^+), then the operator N_k^+ is given by the following set of maps:

1. $N(r_k(t)) \mapsto N(r_k(t))(1 + u_k(p(N^+)))$
2. $N(h_i) \mapsto N(r_k(t))u_i(p_C)$
3. $\Theta_i \mapsto \pm g_i(\Theta^*, \Delta\Theta^*)$

4. $w_i^+ \mapsto \pm g_i(w^*, \Delta w^*)$

and reads as follows:

1. The number of inserted neurons is proportional to the number of existing neurons, multiplied by a random variable depending on the probability $p(N^+)$.
2. Each new hidden neuron h_i^+ is connected to a subset of all other neurons. The number of new connections depends on the probability p_C.
3. For each new hidden neuron h_i^+, the bias Θ_i is set within the interval $[s\Theta^* - \Delta\Theta^*, s\Theta^* + \Delta\Theta^*]$ with a Gaussian distribution, where s is the sign, $s \in [-1, 1]$. The probability of the sign s is equally distributed.
4. For each new hidden neuron h_i^+, the synaptic weights of the new connections w_i^+ are set within the interval $[sw^* - \Delta w^*, sw^* + \Delta w^*]$ with a Gaussian distribution, where s is the sign, $s \in [-1, 1]$. The probability of the sign s is equally distributed. The probability of an insertion of incoming or outgoing synapses of the hidden neuron h_i^+ is also equally distributed.

The parameter variation operators N_k^*, C_k^* are very similar. Each operator acts on the set of existing neurons or synapses, respectively.

$$N_k^* \colon \Theta_i \mapsto \Theta_i + g_i(0, \Delta\Theta), \forall i \in \mathcal{H} \cup \mathcal{O} : u_i(0,1) < p(N^*)$$
$$C_k^* \colon w_{ij} \mapsto w_{ij} + g_{ij}(0, \Delta w), \forall i, j \in \mathcal{N} : u_{ij}(0,1) < p(C^*)$$

The synapse insertion operator C_k^+ inserts a number of synapses, which is proportional to the difference between the number of maximal possible synapses and the number of currently present synapses in the network:

$$E := N(r_k(t))^2 - |\mathcal{I}| \quad \text{maximal number of possible synapses}$$
$$D := E - W(r_k(t)) \quad \text{maximal number of insertable synapses}$$
$$C_k^+ \colon W(r_k(t)) \mapsto W(r_k(t)) + Du_k(p(C^+))$$

The new synapses are initialised with the same map that is used to initialise the new synapses when a neuron is inserted (see neuron insertion operator N_k^+ above).

The synapse deletion operator C_k^- works on the set of the existing synapses. For each synapse, the probability of being deleted is calculated. If a hidden neuron h_i is isolated after the synapse operator was applied to a neural network, i.e. it has no synapse incident to or from it, it is deleted as well. The operator C_k^- is then given by the following set of maps:

$$r_k(t) \mapsto r_k(t) \backslash \{w_{ji}\}, \forall w_{ij} \in r_k(t) \colon u_{ij}(0,1) < p(C^-)$$
$$r_k(t) \mapsto r_k(t) \backslash \{h_i\}, \forall h_i \in r_k(t) \colon W(h_i) = 0$$

This concludes the description of the variation operator.

Evaluation: The evaluation operator is problem-specific (Dieckmann, 1995) and is given by:

$$E\colon p(t) \mapsto (p(t), e(t)),$$

where $p(t)$ is the set of individuals, and $e(t)$ is the set of fitness values for each individual in $p(t)$.

The fitness values in $e(t)$ are scalars calculated using a fitness-function, which is a quality measurement of the potential solution with respect to the given task. In the context of evolutionary robotics, the fitness-function, together with the environment and the morphology of the robot, determines the fitness of an individual. This is elaborated in the next section (see sec. 3.3.2). The fitness-function F is a function of the individual and its life span $T_k(t)$ and is described by:

$$\begin{aligned} e_k(t) &= F(r_k(t), T_k(t)) - K(r_k(t)) \\ F(r_k(t), T_k(t)) &= \sum_{s=0}^{T_k(t)-1} f(s, r_k(t)) \end{aligned}$$

The function $f(s, r_k(t))$ is defined by the experimenter and, in general, only uses information which is locally available to the controller (sensor data, motor commands, network state). Functions, which access global parameters, such as the position of the robot in the global coordinate system, can also be applied.

Examples for fitness-functions are given in the chapter (see chap. 6). Due to the fact that structurally smaller networks are open to analysis with the methods of the dynamical systems theory, a cost function K is introduced to favour them. It is given by:

$$K(n_k(t)) = k_n|\mathcal{H}| + k_s|\mathcal{W}|, \quad k_n, k_s \in \mathbb{R}^+$$

where k_n, k_s are small numbers, indicating the cost for each neuron and synapse.

Selection: The selection operator determines the survival and reproduction of the individuals of the population in one generation, and is given by:

$$S\colon (p(t), e(t)) \mapsto \prod_{k=0}^{P-1} R_k^{\nu(e_k(t))} p(t),$$

where R is the reproduction operator. The function $\nu(x), \nu\colon \mathbb{R} \mapsto \mathbb{N}$ calculates the number of copies of each network for the next generation, hence, $\nu(e_k(t))$ is a stochastic integer variable. Only individuals which produce at least one offspring are propagated to the next generation:

$$\nu(e_k(t)) \in \mathbb{N}\backslash\{1\} = \{0, 2, \ldots\},$$

where 0 means that the individual is removed from the population, and 2 relates to a propagation to the next generation together with one offspring. The calculation of $\nu(e_k(t))$ is rate-based and is computed from a Poisson distribution[†] (see alg. 1) with parameter λ_i , which is given by:

$$E := \max\{e(t)\}$$

[†]The algorithm was developed by Uli Steinmetz. No publication available.

Algorithm 1 Poisson distributed offspring calculation.

Require: λ_k {see equation (3.1)}
Require: u {uniformly distributed random variable in $[0, 1]$, re-evaluated at every occurrence.}
1: $a = e^{-|\lambda_k|}, \quad b = 1, \quad c = 0$
2: **while** $b > a$ **do**
3: $\quad b = bu$
4: $\quad c = c + 1$
5: **end while**
6: **return** c

$$\begin{aligned} \sigma^2 &- \text{standard deviation of e(t)} \\ h(x) &:= e^{-\frac{\gamma}{\sigma^2}(E-x)} \\ \lambda_k &:= \frac{z}{\sum_{j=0}^{n} h(e_k(t))} h(e_i(t)) \end{aligned} \tag{3.1}$$

$$\begin{aligned} \gamma &\in \mathbb{R}^+ \quad [\text{user defined, interactive}] \\ z &\in \mathbb{N}\backslash\{0\} \quad [\text{user defined, interactive}] \end{aligned}$$

The first multiplicand of λ_k is for normalisation, while z can be interpreted as the average population size and is open to user interaction. With this parameter, the experimenter determines the average number of individuals in a population. The parameter γ represents the selection pressure. The higher γ, the less parents form the new generation through a higher number of copies (elite building).

Evolution: The evolution is then given by the repeated mapping

$$p(t) \mapsto SEVp(t)$$

It must be noted that only the offspring, i.e. the copies of the parents from the previous generation are open to variation. The parents are re-evaluated unchanged. This supports, but does not guarantee, that the average fitness of the population does not decrease but monotonically increases. The average can fluctuate, if the variance of the initial conditions of two consecutive generations is too large.

After formalising the evolution algorithm ENS^3, the process of evolution and analysis of recurrent neural networks is discussed.

3.3.2 The approach to artificial evolution

This section discusses how the evolutionary algorithm introduced in the previous section is embedded in the evolutionary robotics approach followed in this work.

The approach differs from other evolutionary robotics approaches (Harvey et al., 1997; Nolfi & Floreano, 2000; Miglino et al., 1995; Bongard & Pfeifer, 2001; Pollack et al., 1999; Lipson & Pollack, 2000; Lipson, 2005). The main difference is that the evolution is not conducted in

a batch process, but that all control parameters of the evolution are open to interaction. This will be discussed at the end of this section.

In the approach discussed here, generating a solution is the first part, which is stringently followed by complete analysis of the underlying dynamical properties of the neural network in relation to the behaviour induced by it. The main scientific goals are:

1. Find general principles of neural signal processing.

2. Extracting those principles and generalising them to other problem domains.

The first goal relates to an understanding of the behaviour-relevant dynamical features in the sensori-motor loop, or otherwise stated, the structure–function relationship of embodied and situated neuro-controllers. Examples are hysteresis phenomena related to co-existing fixed point attractors for obstacle avoidance or quasi-periodic oscillators for a walking behaviour. The second goal relates to general principles extracted from the first, which can be generalised to other morphologies, environments and problem domains. An example is the MRC (Hülse & Pasemann, 2002), which is a minimal recurrent neural network for obstacle avoidance of a wheeled robot (see chap. 5). Its underlying general principle was identified as the interaction of three hysteresis phenomena. The understanding of their interaction was then used to design a sensor filter for a reactive walking machine (Manoonpong et al., 2004). Another example is the SO(2)-network (Pasemann, Hild, & Zahedi, 2003), a configurable two-neuron network with quasi-periodic outputs, which is used as a central pattern generator for walking machines (Manoonpong, 2007; Markelić & Zahedi, 2007) and as a building block in modular robots (Klaassen et al., 2004).

Next, the approach is discussed with respect to its three phases, the design of the experiment, the evolution, and the analysis. The approach is visualised in the figure 3.3.

Design of the experiment

Every experiment begins with the definition of a specific question. Examples for such questions are:

- How can a minimal neural controller be achieved for a goalkeeping behaviour in the RoboCup domain (Zahedi et al., 2004)?

- How can locomotion be achieved for stick insects (Twickel & Pasemann, 2006)?

- How can a reactive behaviour be achieved for walking machines with artificial whiskers (Manoonpong, 2007)?

- How is fast locomotion achieved in a four legged robot (Markelić & Zahedi, 2007)?

- How can communication be achieved in cooperating and competing robot swarms (Wischmann, 2007)?

When the question is specified, an appropriate hardware platform must either be built (Twickel & Pasemann, 2006; Manoonpong, 2007) or chosen (Markelić & Zahedi, 2007). The hardware platform is necessary as simulations never capture all physical properties of the morphology or

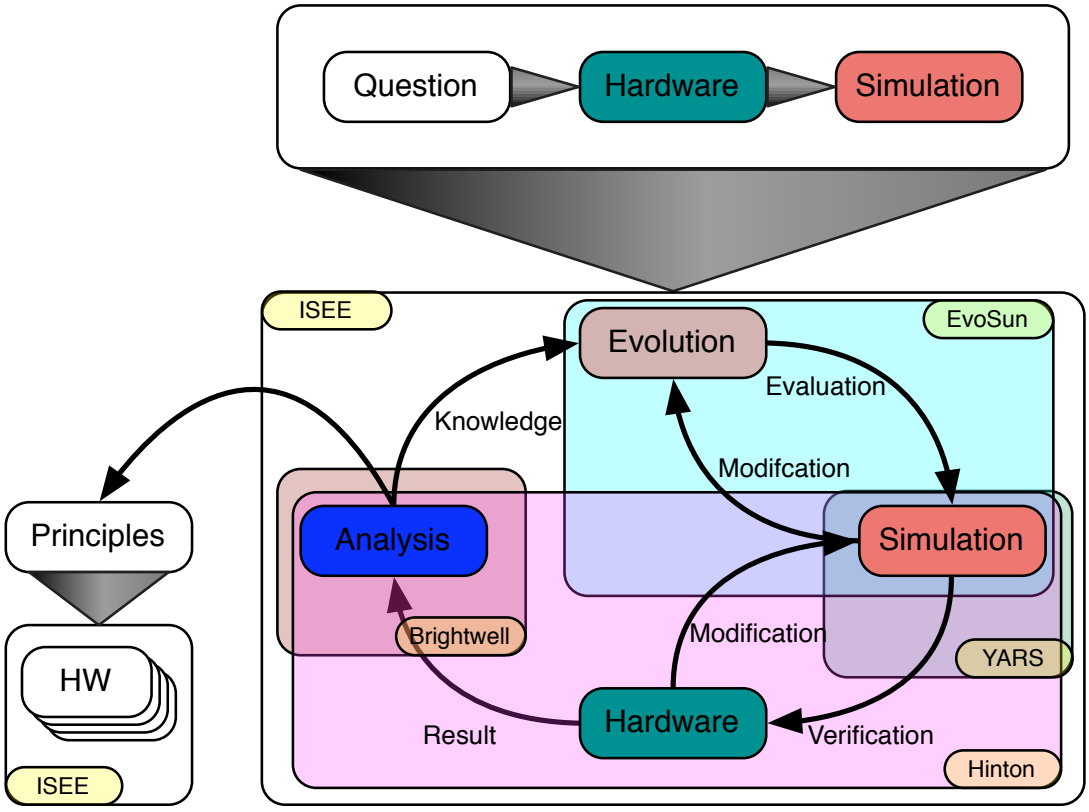


Figure 3.3: The approach of evolutionary robotics. A detailed explanation is given in the text. This image is used with the permission of Martin Hülse.

environment (for a discussion, see the appendix A.3.5). Convincing experiments can also be executed solely in simulation (Sims, 1994), but undoubtful proof of the concepts requires a real robot. This is in correspondence to Brooks statement that the world is its own best model (see previous chapter). In addition to Brooks, Tani (2007) argues that proof of concept in a physical robot platform is superior to simulation, as setting up a demonstration with a physical robot is very time consuming, and the entire set of parameters (robot and environment) is not known by the experimenter. Therefore only robust behaviours hold a physical demonstration.

The next step is to choose an appropriate simulation, which not only captures the main characteristics of the morphology and environment, but also provides a good trade-off between precision and simulation speed. A simulation is chosen for two main reasons. First, if it is faster than real-time, the time consuming evolutionary process is sped up significantly. Second, hardware damaging behaviours are very likely to occur in an unrestricted evolution. These aspects are discussed in detail in the appendix (see app. A.3.5). Once these three parts, the question, hardware, and simulator, are defined, the next step is to evolve neural networks which answer the question of interest.

Evolution

The evolution begins with a choice of the initial population. In general, this is the empty network (see above, sec. 3.3.1) only consisting of input and output neurons, corresponding to the morphology of the robot. For the evolution of a behaviour for walking machines, predefined initial neural networks (e.g. CPG) can be chosen for the initial population, and co-evolution of disjunct populations of neuro-modules enables the parallel evolution of functional sub-units which are then connected to one neural network before the evaluation (Markelić & Zahedi, 2007).

The next steps within the process involve choosing and parametrising the fitness-function, and set the initial evolution parameters (see sec. 3.3.1). As this is not a batch process, the fitness-function and evolution parameters are not fixed but may be changed on-line by the experimenter in accordance to the continuous observation of the behaviours generated by the artificial evolution. Intermediate solutions are tested on the physical platform and the observations are used to modify the simulator in order to close the simulator-reality gap discussed in the literature (Jacobi et al., 1995; J. Walker et al., 2003).

The experimenter decides when a solution is found, and when it is terminated.

Analysis

The resulting neural network is analysed in two aspects. First, it is analysed while it controls the robot in the environment as part of the sensori-motor loop. This is the analysis of the transient dynamics. But the network is also analysed separate from the sensori-motor loop, as an isolated dynamical system. This is the analysis of the attractor structure. Results of both methods lead to an understanding of the structure–function relationship of a network, which can then be extracted and generalised to other problem domains.

In conclusion, this chapter presented the mathematical context in which a neural network is discussed and analysed, the mathematical neuron model, on which the Self-Regulating Neuron is based, and provided an introduction to the evolutionary robotics approach used in this work. A detailed description of the methodology and the software tools for the methods presented in this chapter is given in the appendix (see app. A).

In the remainder of this work, the SRN model is introduced and analysed with respect to its dynamical properties (see chap. 4). This is followed by an analysis of its transient dynamics in the sensori-motor loop (see chap. 5). Finally an adaptive behaviour is evolved (see chap. 6).

Chapter 4

Self-Regulating Neuron Model

The previous chapters discussed the background of, the motivation for and the methods utilised in this work. This chapter introduces the Self-Regulating Neuron (SRN) model, and presents a mathematical and numerical analysis of its dynamical properties.

The neuron model proposed here was first introduced in (Pasemann et al., 2004; Zahedi & Pasemann, 2007) and is an extension of the standard additive neuron model discussed in the previous chapter (see sec. 3.2). The standard neuron model is defined by the following equation:

$$a_i(t+1) \quad = \quad \Theta_i + \sum_{j=0}^{N-1} w_{ij}\tau(a_j(t)), \tag{4.1}$$

where $a_i(t) \in \mathbb{R}$ denotes the activation of the neuron i at time $t = 0, 1, \ldots$, $\Theta_i \in \mathbb{R}$ the bias value, $N \in \mathbb{N}$ the total number of neurons in the network, $w_{ij} \in \mathbb{R}$ the synaptic weight from neuron j to neuron i, and $\tau(x), \tau\colon \mathbb{R} \mapsto \mathbb{R}$ is the hyperbolic tangent transfer-function.

The standard additive neuron model is extended by substituting the synaptic weight w_{ij} with an activity-dependent term, which is the product of a pre- and post-synaptic neuron property. These properties will be referred to as the *transmitter strength* and *receptor strength* of the pre- and post-synaptic neuron.

The neuron model is motivated by Ashby's Homeostat (see sec. 2.2). A neuron is understood as a homeostatic unit coupled in a network which regulates its activation towards a target value.

The first section discusses the SRN model in detail, followed in the second section by an analysis of its attractor landscape.

4.1 Self-Regulation Neuron Model

The Self-Regulating Neuron model extends the standard additive neuron model (see eq. 4.1) by an activity-dependent product of a pre-synaptic and post-synaptic neuron property, which redefines the previously static synaptic weight w_{ij}.

In analogy to biological neurons, the pre-synaptic neuron property is referred to as the transmitter strength, of the neuron and is denoted by η. The post-synaptic neuron property is the receptor strength of the neuron and is denoted with ξ. The synaptic strength w_{ij} is then

defined as:

$$\begin{aligned} w_{ij}(t) &:= c_{ij}\xi_i(t)\eta_j(t) \\ \forall t \in \mathcal{T} : \xi_i(t), \eta_j(t) &\in \mathbb{R}^+ \end{aligned} \tag{4.2}$$

where $\eta_j(t)$ denotes the transmitter strength of the pre-synaptic neuron n_j at time t, $\xi_i(t)$ the receptor strength of the post-synaptic neuron n_i at a time t, and c_{ij} the sign of the synapse. The sign of the synapse is static and given by the corresponding entry of the connectivity matrix:

$$\begin{aligned} \mathcal{C} &= (c_{ij})_{i,j\in\mathcal{N}} \\ c_{ij} &\in \{-1, 0, 1\} \end{aligned}$$

where $c_{ij} \in \{-1, 0, 1\}$ corresponds to an inhibitory synapse, no connection, or an excitatory synapse from neuron n_j to n_i, respectively.

The receptor and transmitter strengths are defined to be strictly positive (see eq. 4.2), so that the type of the synapse, i.e. excitatory or inhibitory, is only determined by c_{ij}.

The equation for the standard additive neuron model (see eq. 4.1) is now rewritten as

$$a_i(t+1) = \Theta_i + \sum_{j=0}^{N-1} \underbrace{c_{ij}\xi_i(t)\eta_j(t)}_{w_{ij}(t)} \tau(a_j(t)) \tag{4.3}$$

$$= \Theta_i + \xi_i(t) \sum_{j=0}^{N-1} c_{ij}\eta_j(t)\ \tau(a_j(t)) \tag{4.4}$$

The remainder of this section discusses the desired target value, the equations for the transmitter and receptor strengths, and the synaptic weight. Equation (4.4) shows that the SRN model is related to synaptic scaling (see sec. 2.4) as the post-synaptic receptor strength $\xi_i(t)$ scales all incoming signals.

Target value a^*

There are several feasible choices for the target value. The most interesting dynamical effects arise when the neuron operates within the non-linear domain of the transfer-function. Hence, for the SRN model the point of maximal non-linearity ($\tau'''(x) = 0$) is chosen for the target value. The transfer-function here is the hyperbolic tangent. A variation of the Self-Regulating Neuron model for the standard sigmoid transfer-function is discussed in (Zahedi & Pasemann, 2007). Let the hyperbolic tangent be denoted by τ, then

$$\begin{aligned} \tau(x) &:= \tanh(x) \\ \tau'(x) &= 1 - \tau(x)^2 \\ \tau''(x) &= 2\tau(x)(\tau(x)^2 - 1) \\ \tau'''(x) &= -6\tau(x)^4 + 8\tau(x)^2 - 2 \\ \tau'''(a^*) = 0 &\Leftrightarrow \pm a^* = \pm\text{arctanh}\left(\sqrt{\frac{1}{3}}\right) \approx \pm 0.65848 \end{aligned}$$

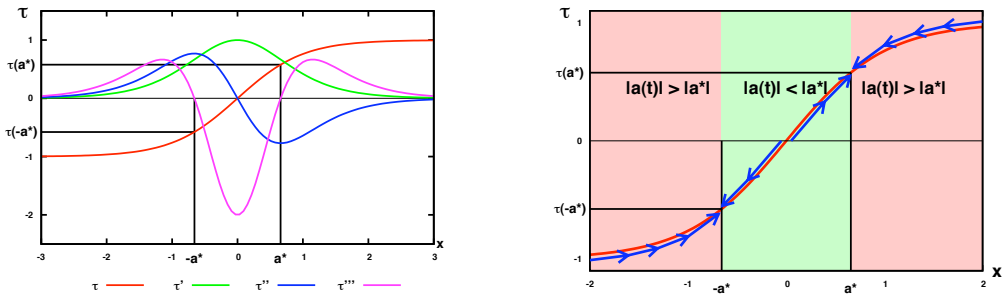


Figure 4.1: TRANSFER-FUNCTION, ITS DERIVATIVES, AND RECEPTOR BEHAVIOUR. Left: Hyperbolic Tangent, its derivatives and the defined target value a^*. Right: The property of the receptor equation in relation to the target value and the transfer-function.

As the transfer-function $\tau(x) = \tanh(x)$ is point-symmetric around the origin (see fig. 4.1), there are two target values, $\pm a^*$, which are referred to as the upper and lower target value of the system. For the regulated output of the neuron, it follows that for the desired target output of the neuron:

$$\tau(\pm a^*) \quad = \quad \pm 0.57735 \tag{4.5}$$

Receptor strength ξ

The receptor is designed to control the amount of input into the neuron depending on the deviation of the current activation from the target value. If the absolute current value of the activity is below the desired target value $|a(t)| < |a^*|$, the receptor $\xi(t)$ is increased. The effect of an increased receptor $\xi(t)$ results in of two cases (see fig. 4.1).

In the case of a positive input, the activity is raised towards the positive target value a^*. In case of a negative input, the current activity is further decreased towards the negative target value $-a^*$. Consequently, the receptor $\xi(t)$ must be decreased if the current activity is above the desired value $|a(t)| > |a^*|$.

The equation for the receptor $\xi(t)$ is defined by

$$\xi_i(t+1) \quad := \quad \epsilon + \xi_i(t)(1 + \beta g(a_i(t))), \quad 0 < \beta < 1 \tag{4.6}$$

where $g(a(t))$ defines the activity-dependent change of the receptor, and β controls the rate of change. Due to limited computational precision, a small technical value $0 < \epsilon \ll 1$ is introduced, so that the receptor cannot decrease to zero. It is of the following order of magnitude:

$$\epsilon := 10^{-4}. \tag{4.7}$$

As such, it may be neglected in the analysis of the system as it does not contribute to its dynamics. The function $g(x)$ is defined as

$$g\colon \mathbb{R} \mapsto \mathbb{R}, \quad g(x) \quad := \quad \tau(a^*)^2 - \tau(x)^2 \Rightarrow \begin{cases} g(x) > 0 & \text{if} \quad |x| < |a^*| \\ g(x) < 0 & \text{if} \quad |x| > |a^*| \\ g(x) = 0 & \text{if} \quad |x| = |a^*| \end{cases} \tag{4.8}$$

From the equations (4.5), (4.6) and

$$\forall x \in \mathbb{R} : \tau(x) \in]-1, 1[\tag{4.9}$$

it follows that

$$\forall x \in \mathbb{R} \colon g(x) > -1$$

which results in a strictly positive receptor strength:

$$\forall t > t_0 \in \mathcal{T}, \xi(t_0) > 0 \Rightarrow \xi(t) > 0. \tag{4.10}$$

Transmitter strength η

The transmitter imparts the internal activation of the neuron to the neural network. If the internal neuron activation is large, the excitation or inhibition of the neuron, over the synaptic connections into the neural network, is defined to be larger compared to a smaller internal neural activity. The transmitter level is defined by

$$\eta_i(t+1) \quad := \quad \epsilon + (1-\gamma)\eta_i(t) + \delta h(a_i(t)), \quad 0 < \gamma, \delta < 1 \tag{4.11}$$

where $(1-\gamma)$ is a decay term, δ the growth term, and ϵ a negligible value (see eq. 4.7). The parameters γ and δ control the decay and growth of the transmitter, respectively. The growth term $h_i(t)$ reads as follows:

$$h \colon \mathbb{R} \mapsto \mathbb{R}, \quad h(a(t)) \quad := \quad 1 + \tau(a_i(t)). \tag{4.12}$$

It follows from equation (4.9) and equation (4.12) that the transmitter strength is strictly positive:

$$\forall t > t_0 \in \mathcal{T}, \eta(t_0) > 0 \Rightarrow \eta(t) > 0. \tag{4.13}$$

Synaptic weight

The synaptic weight $w_{ij}(t)$ can be rewritten in the following form:

$$\begin{aligned}
w_{ij}(t) &= c_{ij}\xi_i(t)\eta_j(t) \\
&= c_{ij}\Big[\Big(\xi_i(t-1)\big(1+\beta g\big(a_i(t-1)\big)\big)\Big)\cdot\Big((1-\gamma)\eta_j(t-1)+\delta h\big(a_j(t-1)\big)\Big)\Big] \\
&= c_{ij}\Big[\Big(\xi_i(t-1)+\xi_i(t-1)\beta g\big(a_i(t-1)\big)\Big)\cdot\Big((1-\gamma)\eta_j(t-1)+\delta h\big(a_j(t-1)\big)\Big)\Big] \\
&= c_{ij}\xi_i(t-1)\Big[(1-\gamma)\eta_j(t-1)+\delta h\big(a_j(t-1)\big)+\beta(1-\gamma)\eta_j(t-1)g\big(a_i(t-1)\big) \\
&\quad +\beta\delta g\big(a_i(t-1)\big)h\big(a_j(t-1)\big)\Big].
\end{aligned} \tag{4.14}$$

The term (4.14) of the synaptic weight is comparable to Hebb'ian Learning Rule (see sec. 4.3), which defines the weight change as the product of the pre- and post-synaptic activity scaled by a factor (see eq. 4.39). The term (4.14) is also found if the weight change of the SRN model is calculated (see eq. 4.16):

$$\begin{aligned}
\Delta w_{ij}(t) &= w_{ij}(t+1) - w_{ij}(t) && (4.15)\\
&= c_{ij}\Big(\xi_i(t+1)\eta_j(t+1) - \xi_i(t)\eta_j(t)\Big)\\
&= c_{ij}\left(\Big(\xi_i(t)(1+\beta g(a_i(t)))\Big)\Big((1-\gamma)\eta_j(t) + \delta h(a_j(t))\Big) - \xi_i(t)\eta_j(t)\right)\\
&= c_{ij}\xi_i(t)\Big(\delta h\big(a_j(t)\big) - \gamma\eta_j(t) + \beta g\big(a_i(t)\big)\eta_j(t)\\
&\quad +\beta\delta g\big(a_i(t)\big)h\big(a_j(t)\big) && (4.16)\\
&\quad -\beta\gamma g\big(a_i(t)\big)\eta_j(t)\Big).
\end{aligned}$$

The equation (4.15) is referred to as the SRN plasticity rule.

Summary

The Self-Regulation Neuron model is a discrete-time, three-dimensional dynamical system defined by the following set of difference equations:

$$\begin{aligned}
\rho_i(t) &:= (a_i(t), \xi_i(t), \eta_i(t)) && (4.17)\\
a_i(t+1) &= \Theta_i + \xi_i(t)\sum_{j=0}^{N-1} c_{ij}\eta_j(t)\tau(a_j(t)) && (4.18)\\
\xi_i(t+1) &= \epsilon + \xi_i(t)(1+\beta(\tau(a^*)^2 - \tau(a_i(t))^2) && (4.19)\\
\eta_i(t+1) &= \epsilon + (1-\gamma)\eta_i(t) + \delta(1+\tau(a_i(t))) && (4.20)
\end{aligned}$$

The system's behaviour is controlled by three parameters β, γ and δ, which are referred to as the plasticity parameters. They are defined to be strictly positive:

$$0 < \beta, \gamma, \delta < 1 \tag{4.21}$$

For simplicity, they are set alike for each neuron in a network, but could also be chosen individually. As input neurons are buffers (see sec. 3.2), their transmitter strength is set constantly to one:

$$\eta_i(t) := 1, \quad \forall t \in \mathcal{T} \vee \forall i \in \mathcal{I} \tag{4.22}$$

The SRN model is fully defined by the equations (4.18) to (4.21). The next section analyses its dynamical properties by means of single neurons and small neuro-modules. The technical term ϵ is neglected in the analysis hereafter, as it does not contribute to the dynamics of the system.

4.2 Dynamical properties

The previous section introduced and specified the Self-Regulating Neuron model. This section presents numerical and mathematical analyses of the dynamical properties by means of single neurons and small neuro-modules. The bifurcation diagram (see app. A.3) is chosen to visualise the asymptotic dynamics of the system, in which a slow-varying input is simulated by the modulation of a bias Θ.

4.2.1 Single neuron with excitatory synapse

The following set of equations fully describes the behaviour of the excitatory single neuron:

$$\begin{aligned} a(t+1) &= \Theta + \xi(t)\eta(t)\tau(a(t)) \qquad [excitatory : c = 1] \\ \xi(t+1) &= \xi(t)(1 + \beta g(a(t))) \\ \eta(t+1) &= (1-\gamma)\eta(t) + \delta h(a(t)). \end{aligned}$$

Figure 4.2 shows the bifurcation diagrams for the four system properties of the single neuron with excitatory synapse depending on Θ: activation (see fig. 4.2 A), receptor (see fig. 4.2 B), transmitter (see fig. 4.2 C), and synaptic (see fig. 4.2 D) strength ($\tau(\Theta)$, $\xi(\Theta)$, $\eta(\Theta)$, $w(\Theta)$). The bifurcation diagram of the neuron's output $\tau(\Theta)$ (see fig. 4.2 A) indicates two distinguishable regions.

For a bias within the interval $\Theta \in [-a^*, a^*]$, there are co-existing stable fixed point attractors, but in contrast to the excitatory single neuron with static synapse (see fig. 4.2 E), the neuron's output is constant at $\tau(a^*)$ and $-\tau(a^*)$ for varying values of the bias Θ. This region is referred to as the *homeostatic region*. For bias values outside of the interval $\Theta \notin]-a^*, a^*[$, the output of the neuron is largely determined by the sigmoidal transfer-function. This region is referred to as the *sigmoidal region*. Next, the behaviour of the neuron in both regions is analysed mathematically.

Sigmoidal region

The sigmoid behaviour follows from the equation (4.6):

$$\begin{aligned} \Delta\xi(t) &= \xi(t+1) - \xi(t) \\ &= \beta\xi(t)g(a(t)) \\ \Theta \notin]-a^*, a^*[\;&\Rightarrow\; \lim_{t\to\infty} \xi(t) = 0 \\ &\Rightarrow\; \lim_{t\to\infty} w(t) = 0 \\ &\Rightarrow\; \lim_{t\to\infty} a(t) = \Theta \\ &\Rightarrow\; \lim_{t\to\infty} o(t) = \tau(\Theta). \end{aligned}$$

Homeostatic region

To analyse the behaviour of the system in the homeostatic region, the fixed point equations are required. A fixed point $\rho^* = (a^*, \xi^*, \eta^*)$ of the neuron is given by

$$a^* = \Theta + \xi^*\eta^*\tau(a^*) \tag{4.23}$$

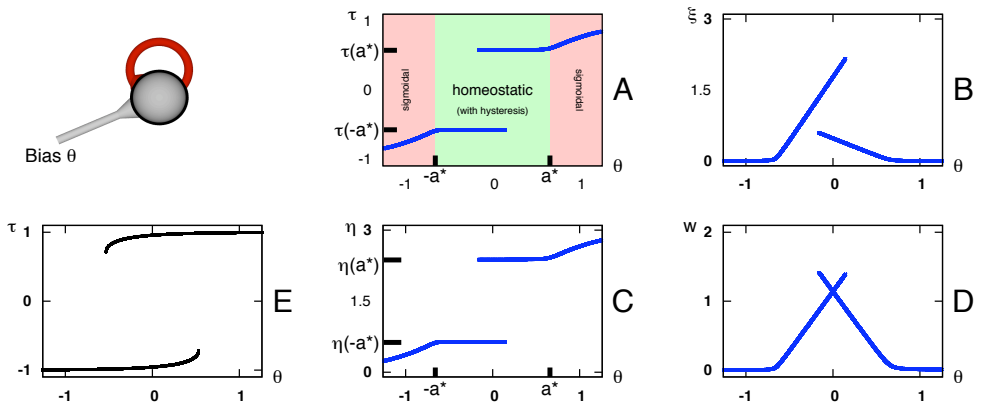


Figure 4.2: SINGLE EXCITATORY NEURON. Bifurcation Diagram for a single neuron with excitatory synapse. Plasticity parameterss are set to $\beta = 0.1, \gamma = 0.01, \delta = 0.015$. The values are chosen empirically. For each input value, 5000 convergence iterations are calculated before the attractor is plotted. The bifurcation diagrams A-D show the attractors of the four neuron properties of the Self-Regulating Neuron model. These are output (A), receptor (B), transmitter (C), and strength of the recurrent connection (D). As a comparison, figure E shows the attractor for the output of the standard additive neuron model. Diagram A depicts a homeostatic and two sigmoidal regions.

$$\begin{aligned}
\xi^* &= \xi^* \left(1 + \beta g^*\right) \\
&= \xi^* \\
\eta^* &= \left(1 - \gamma\right) \eta^* + \delta h^* \\
&= \frac{\delta}{\gamma} h^* \\
\Rightarrow w^* &= \xi^* \eta^* \\
&= \frac{a^* - \Theta}{\tau\left(a^*\right)} \\
h^* &= 1 + \tau\left(a^*\right) \\
g^* &= \tau\left(a^*\right)^2 - \tau\left(a^*\right)^2 \\
&= 0
\end{aligned} \tag{4.24}$$

A non-trivial solution for the transmitter coordinate ξ^* can be derived from the activation coordinate a^* (see eq. 4.23):

$$\begin{aligned}
\xi^* &= \frac{a^* - \Theta}{\eta^* \tau(a^*)} \\
&= \frac{a^* - \Theta}{\frac{\delta}{\gamma}(1 + \tau(a^*))\tau(a^*)}.
\end{aligned}$$

If the weight w of the recurrent connection of single neuron is zero, the output of the neuron is given by the transfer-function, i.e. the hyperbolic tangent. Consequently, a non-trivial value for

the weight ($w > 0$) must exist within the homeostatic region. The equation (4.24) defines the consistency condition:

$$w^* > 0 \quad \Leftrightarrow \quad \frac{a^* - \Theta}{\tau(a^*)} > 0. \tag{4.25}$$

For the two defined target values $\pm a^*$, and from equation (4.25), it follows that:

I. For upper target value $+a^*$:

$$\begin{aligned} a^* - \Theta &> 0 \qquad [\tau(a^*) > 0] \\ \Rightarrow a^* &> \Theta \end{aligned} \tag{4.26}$$

II. For lower target value $-a^*$:

$$\begin{aligned} -a^* - \Theta &< 0 \qquad [-\tau(a^*) < 0] \\ \Rightarrow -a^* &< \Theta \end{aligned} \tag{4.27}$$

It follows from equation (4.26) and equation (4.27) that a stable non-trivial value for the excitatory synapse can only exist within the interval given by the upper and lower target value:

$$w^* > 0 \quad \Leftrightarrow \quad \Theta \in]-a^*, a^*[\tag{4.28}$$

Hence, an asymptotically stable fixed point of the self regulating neuron can only exist in the interval given above (see eq. 4.28). Assuming it exists (for proof see next section), the values for the asymptotic synaptic weight, the transmitter and receptor strength may be calculated as follows:

$$\eta^* = \frac{\delta}{\gamma}(1 + \tau(a^*)) \tag{4.29}$$

$$= \frac{\delta}{\gamma}\left(1 \pm \sqrt{\frac{1}{3}}\right) \tag{4.30}$$

$$\approx \frac{\delta}{\gamma}0.42 \quad \vee \quad \frac{\delta}{\gamma}1.58 \tag{4.31}$$

$$\Rightarrow \eta^* \in \left\{\frac{\delta}{\gamma}0.42, \frac{\delta}{\gamma}1.58\right\} \tag{4.32}$$

$$w^* = \frac{a^* - \Theta}{\tau(a^*)} = 1.1405 - \sqrt{3}\Theta \tag{4.33}$$

$$\xi^* = \frac{a^* - \Theta}{\eta^* \tau(a^*)} \in \left\{\frac{\gamma}{\delta}\left(2.716 - 4.124\Theta\right), \frac{\gamma}{\delta}\left(0.722 - 1.096\Theta\right)\right\} \tag{4.34}$$

The next section proves the stability of the fixed point and presents the calculations for the boundaries of the homeostatic region exemplarily for different sets of the plasticity parameters.

Stability analysis of the fixed point

In the previous section, the condition under which an asymptotically stable fixed point for the system occurs was discussed. In the following section, a stability analysis is conducted to prove

	α	β	γ	δ	lower target value $-a^*$		upper target value $+a^*$	
Set 1	± 1	0.1	0.01	0.015	-0.6585	0.1958	-0.1958	0.6585
Set 2	± 1	0.01	0.01	0.02	-0.6585	0.1559	-0.1559	0.6585
Set 3	± 0.7	0.01	0.01	0.02	-0.2675	0.04232	-0.04232	0.2675

Table 4.1: PARAMETER DEPENDENCE OF THE HOMEOSTATIC REGION. Parameter sets and the results for the stability calculations of discrete-time dynamical systems based on the eigenvalues of the Jacobian-matrix of the SRN model with positive recurrent connection. For discrete-time dynamical systems, a fixed point is asymptotically stable if the absolute values of the eigenvalues are smaller than one. The values in the table are the outer boundaries, hence, the values which follow from the condition $\|\lambda_i\| = 1$. For the stability analysis, three different sets are presented in order to determine the parameter subsets which affect the various boundaries of the homeostatic region. The comparison between set 1 and set 2 shows that the boundaries of the upper and lower target values are affected by the parameter β, γ, δ. The comparison between set 2 and set 3 shows that both, the outer and inner boundaries are affected by the parameter α. For a discussion of the values and their dependence on the plasticity parameters, see text below.

that the previously extracted condition leads to an asymptotically stable fixed point. This is done by calculating the eigenvalues of the Jacobian matrix as a function of Θ for a specific set of the plasticity parameters β, γ, δ of the system for the fixed point $\rho^* = (a^*, \eta^*, \xi^*)$, where α is introduced as a factor for the target value a^*.

The plasticity parameter α is introduced as a scaling factor of the target value a^*, which also enables the calculation of the stability for different target values. In an application for a mobile robot, the parameter can reflect an internal drive, increasing (the parameter α and therefore the target value and) the internal drive to force a behaviour switch, and otherwise remain low. The parameter α would then implement a fading mechanism between different behaviours similar to other methods, such as in the Dual-Dynamics approach (Jäger & Christaller, 1997).

The calculation steps are given for an experimentally-determined parameter set (Set 2, see tab. 4.1) which is taken from the following chapter. The results of the stability analysis are discussed for three different parameter sets (see tab. 4.1).

The stability of a dynamical system at a fixed point can be determined by the linearisation of the map near the fixed point (Strogatz, 1994). In the case of a one-dimensional map, the linearisation is given by the derivative at the fixed point, which is replaced by the Jacobian matrix in the case of non-linear maps of higher dimensions (Alligood et al., 1996). Let

$$
\begin{aligned}
f\colon \mathbb{R}^3 \mapsto \mathbb{R}^3, \quad f &= (f_a, f_\xi, f_\eta) \\
\rho &= (a, \xi, \eta) \\
\rho^* &= (a^*, \xi^*, \eta^*) \\
f_a\colon \mathbb{R} \mapsto \mathbb{R}, \quad f_a(\rho) &= \Theta + \xi\eta\tau(a) \\
f_\xi\colon \mathbb{R} \mapsto \mathbb{R}, \quad f_\xi(\rho) &= \xi(1 + \beta(\tau(a^*)^2 - \tau(a)^2)) \\
f_\eta\colon \mathbb{R} \mapsto \mathbb{R}, \quad f_\eta(\rho) &= (1-\gamma)\eta + \delta(1 + \tau(a))
\end{aligned}
$$

then the Jacobian matrix at the coordinate ρ is given by

$$\begin{aligned} Df(\rho) &= \begin{pmatrix} \frac{\delta f_a}{\delta a}(\rho) & \frac{\delta f_a}{\delta \xi}(\rho) & \frac{\delta f_a}{\delta \eta}(\rho) \\ \frac{\delta f_\xi}{\delta a}(\rho) & \frac{\delta f_\xi}{\delta \xi}(\rho) & \frac{\delta f_\xi}{\delta \eta}(\rho) \\ \frac{\delta f_\eta}{\delta a}(\rho) & \frac{\delta f_\eta}{\delta \xi}(\rho) & \frac{\delta f_\eta}{\delta \eta}(\rho) \end{pmatrix} \\ &= \begin{pmatrix} \xi\eta\left(1-\tau(a)^2\right) & -2\beta\xi\tau(a)\left(1-\tau(a)^2\right) & \delta\left(1-\tau(a)^2\right) \\ \eta\tau(a) & 1+\beta\left(\tau(a^*)^2-\tau(a)^2\right) & 0 \\ \frac{\gamma}{\delta}\frac{a^*-\Theta}{1+\tau(a)} & 0 & 1-\gamma \end{pmatrix}. \end{aligned}$$

The next steps are to first apply the values of the coordinates of the fixed point ρ^*, second, replace them by the fixed point equations (see eq. (4.33) to (4.34)) and finally, to substitute the plasticity parameter by numerical values in order to receive a Jacobian matrix depending on the bias Θ. The substitution of the plasticity parameters by numerical values is required as the analytic solution is not computable. The detailed calculation is given in the appendix (see app. C). The resulting matrix for the parameter set

$$\begin{aligned} a^* &:= \alpha \cdot \operatorname{arctanh}\left(\tfrac{1}{\sqrt{3}}\right) \\ \alpha &= 1 \qquad \beta = 0.01 \\ \gamma &= 0.01 \quad \delta = 0.02 \end{aligned}$$

is

$$Df(\rho^*) = \begin{pmatrix} -\frac{2\sqrt{3}\left(\Theta-\operatorname{arctanh}\left(\frac{\sqrt{3}}{3}\right)\right)}{3} & \frac{\frac{2}{300}\left(\Theta-\operatorname{arctanh}\left(\frac{\sqrt{3}}{3}\right)\right)}{\frac{\sqrt{3}}{3}+1} & \frac{1}{75} \\ \frac{2}{3}\sqrt{3}\left(\frac{\sqrt{3}}{3}+1\right) & 1 & 0 \\ -\frac{0.5\left(\Theta-\operatorname{arctanh}\left(\frac{\sqrt{3}}{3}\right)\right)}{\frac{\sqrt{3}}{3}+1} & 0 & 0.99 \end{pmatrix}.$$

The eigenvalues are then given by[†]:

$$\begin{aligned} \lambda_1 &= \left(-0.03543\Theta^2 - 0.05702\Theta^3 - 0.00672\Theta\right)^{\frac{1}{3}} - 0.3849\Theta \\ &\quad + \frac{0.1481\Theta^2 + 0.06137\Theta + 0.005364}{\left(-0.03543\Theta^2 - 0.05702\Theta^3 - 0.00672\Theta\right)^{\frac{1}{3}}} + 0.9168 \\ \lambda_2 &= 0.9168 - (0.5+0.866\mathrm{i})\left(-0.03543\Theta^2 - 0.05702\Theta^3 - 0.00672\Theta\right)^{\frac{1}{3}} \\ &\quad - \frac{(0.5-0.866\mathrm{i})\left(0.1481\Theta^2 + 0.06137\Theta + 0.005364\right)}{\left(-0.03543\Theta^2 - 0.05702\Theta^3 - 0.00672\Theta\right)^{\frac{1}{3}}} - 0.3849\Theta \\ \lambda_3 &= 0.9168 - (0.5-0.866\mathrm{i})\left(-0.03543\Theta^2 - 0.05702\Theta^3 - 0.00672\Theta\right)^{\frac{1}{3}} \end{aligned}$$

[†]For presentation, in order to increase the readability, every additive term with $a\Theta^b, a < 10^{-4}, b > 3$ is neglected, because stability only occurs for $|\Theta| < a^* < 1$, so that they are approximated by zero. This is not valid if the simplified eigenvalues are used for further calculations (see app. C).

$$-\frac{(0.5+0.866\mathrm{i})\left(0.1481\Theta^2+0.06137\Theta+0.005364\right)}{(-0.03543\Theta^2-0.05702\Theta^3-0.00672\Theta)^{\frac{1}{3}}}-0.3849\Theta$$

The fixed point ρ^* is stable, if the modulus of each of the eigenvalues is smaller than and not equal to one:

$$\forall i \in \{1,2,3\}: \quad \|\lambda_i\| \;<\; 1. \tag{4.35}$$

The solutions for $\|\lambda_i\| = 1$ (outer boundaries for the asymptotically stable fixed point) are given and discussed for three different sets of plasticity parameters (see tab. 4.1).

Discussion

The results of the stability analysis for the three parameter sets (see tab. 4.1) show that the outer and inner boundaries are dependent on different parameters. The outer boundary of the homeostatic region is only dependent on the parameter α. This can be expected, as α varies the target value, and therefore, the consistency condition for the weight (see eq. 4.25). However, different values of α also affect the inner boundary condition (see tab. 4.1 Set 2 vs. Set 3). This is the result of the same effect which is observed for the other plasticity parameters, discussed next.

The parameter β, γ, δ only affect the inner boundaries of the homeostatic region, and have no effect on the outer boundaries (see tab. 4.1 Set 1 vs. Set 2). This can be expected, as the parameter β, γ, δ do not affect the consistency condition, but rather the asymptotic synaptic weight, and therefore, the width of the hysteresis domain (Hülse & Pasemann, 2002). This also holds for variations of the parameter α.

4.2.2 Single neuron with inhibitory synapse

The bifurcation diagram of the single neuron with an inhibitory recurrent connection (see fig. 4.3) indicates that there are three distinguishable regions, a period-2 attractor for the interval $\Theta \in]-x^*, x^*[$, a sigmoid region for $\Theta \in [-a^*, a^*]$, and regions of co-existing attractors, where the two previous regions overlap. The behaviour of the SRN with inhibitory synapse is similar to the static model (see fig. 4.3 E), but the co-existing attractors are a qualitative difference. For the mathematical analysis of the system's behaviour and the proof of the stability of the period-2 attractor, the stability of the fixed point of the second iterative must be proven:

$$\begin{aligned} \rho(t+2) &= \rho(t) \\ \rho(t+3) &= \rho(t+1). \end{aligned}$$

The equations read as follows:

$$\begin{aligned} a(t+2) &= \Theta - \xi(t+1)\eta(t+1)\cdot\tau(a(t+1)) \\ \xi(t+2) &= \xi(t+1)(1+\beta g(t+1)) \\ \eta(t+2) &= (1-\gamma)\eta(t+1)+\delta h(t+1) \\ g(t+1) &= \tau(a^*)^2 - \tau(a(t+1))^2 \end{aligned}$$

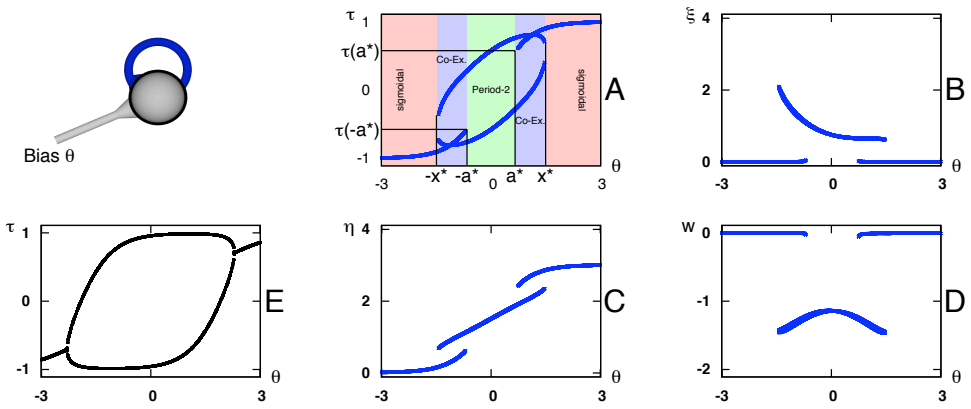


Figure 4.3: SINGLE INHIBITORY NEURON. Bifurcation diagram for a single neuron with inhibitory synapse. Plasticity parameters are set to $\beta = 0.1, \gamma = 0.01, \delta = 0.015$. The values are chosen empirically. For each input value, 5000 convergence iteration steps are calculated before the attractor is plotted. The bifurcation diagrams A-D show the attractors of the four neuron properties of the Self-Regulating Neuron model, which are output (A), receptor (B), transmitter (C), and strength of the recurrent connection (D). The figure E shows the attractor for the output of the standard additive neuron model. The bifurcation diagram of the output of the Self-Regulating Neuron model shows a switchable oscillator, similar to the static neuron model. However, in contrast to the static model, the Self-Regulating Neuron model shows a constant amplitude over a large range, and co-existing attractors where the oscillatory regions overlaps with the sigmoidal regions.

$$h(t+1) \quad = \quad 1 + \tau(a(t+1))$$

For this set of equations, the Jacobian matrix and the eigenvalues must be computed, in order to verify the observations indicated by the bifurcation diagram. The characteristic polynomial grows too large in complexity and its solution is beyond the scope of this thesis.

Summary

The previous sections discussed the asymptotic properties of the single neuron dynamics by means of bifurcation diagrams. For the excitatory single neuron, the indications given by the diagrams were supplemented by a mathematical and numerical analysis.

In both cases, the neuron with excitatory and inhibitory recurrent connection, a slow varying input was simulated through a shift of the bias value. Embedded in the sensori-motor loop, a Self-Regulating Neuron will receive input via input neurons. This is a qualitative difference to a variation of the bias value, as the former case, the input is open to regulation by the receptor strength.

The following sections discuss the observable behaviour of small neuro-modules, consisting of one input and one output neuron. The attractors of the system are discussed for inhibitory and excitatory connections between the two neurons, with and without inhibitory/excitatory recurrent connection of the output neuron. The analysis will lead to the next chapter, in which

the presented neuro-modules are used to implement an obstacle avoidance behaviour in order to analyse the transient dynamics with respect to the plasticity parameters.

There are six possible configurations for the input-output neuro-modules (see fig. 4.4) which have been described. They are discussed separately in the following two sections.

4.2.3 Input-output neuro-module without recurrent connection

The neuron modules presented in this section consist of an input and output neuron connected by a synapse (see fig. 4.4 A/B). The behaviour of both systems is described by

$$\begin{aligned}
\forall t \in \mathcal{T} : \eta_I(t) &:= 1 \\
\Theta_O &:= 0 \\
I(\Theta_I) &= \mathrm{id}(\Theta_I) = \Theta_I \\
a_O(t+1) &= c_I \xi_O(t) I(\Theta_I) \\
&= c_I \xi_O(t) \Theta_I \\
w(t) &= c_I \xi_O(t) \\
\xi_O(t+1) &= \xi_O(t)(1 + \beta g(t)) \\
g(t) &= \tau(a^*)^2 - \tau(a_O(t))^2
\end{aligned}$$

where $I(\Theta_I)$ is the output of the input neuron with respect to its bias and is given by the identity, $c_I = \{-1, 1\}$ is the sign of the synapse, $\xi_O(t)$ is the receptor of the output neuron, and the output neuron's bias is constant and set to zero $\Theta_O = 0$. The input neuron's transmitter is set to one (see eq. 4.22). The equations for the fixed point of the systems are given by

$$a_O^* = c_I \xi_O^* \Theta_I \tag{4.36}$$

$$\xi_O^* = c_I \frac{a_O^*}{\Theta_I} \tag{4.37}$$

$$\Rightarrow w_I^* = c_I \frac{a_O^*}{\Theta_I}$$

The equations (4.37) and (4.36) explain the attractor dynamics seen in the bifurcation diagram (see fig. 4.4 A/B). For input values which are not equal to zero, the output neuron regulates its activity towards the target value, and, therefore, shows a homeostatic behaviour (see the previous section). If the system is started with a zero input value, the activity and therefore the output remains zero. This is indicated by a single dot in the bifurcation diagram (see fig. 4.4 A/B), which follows from

$$\begin{aligned}
I(\Theta_I(t_0)) &= I(t_0) = 0 \\
\Rightarrow a_O(t+1) &= c_I \xi_O 0 \\
\Rightarrow \tau(a_O(t)) &= 0
\end{aligned}$$

When the system is started with an input value not equal to, but approaching zero it diverges (see eq. 4.37):

$$I(t_0) \neq 0$$

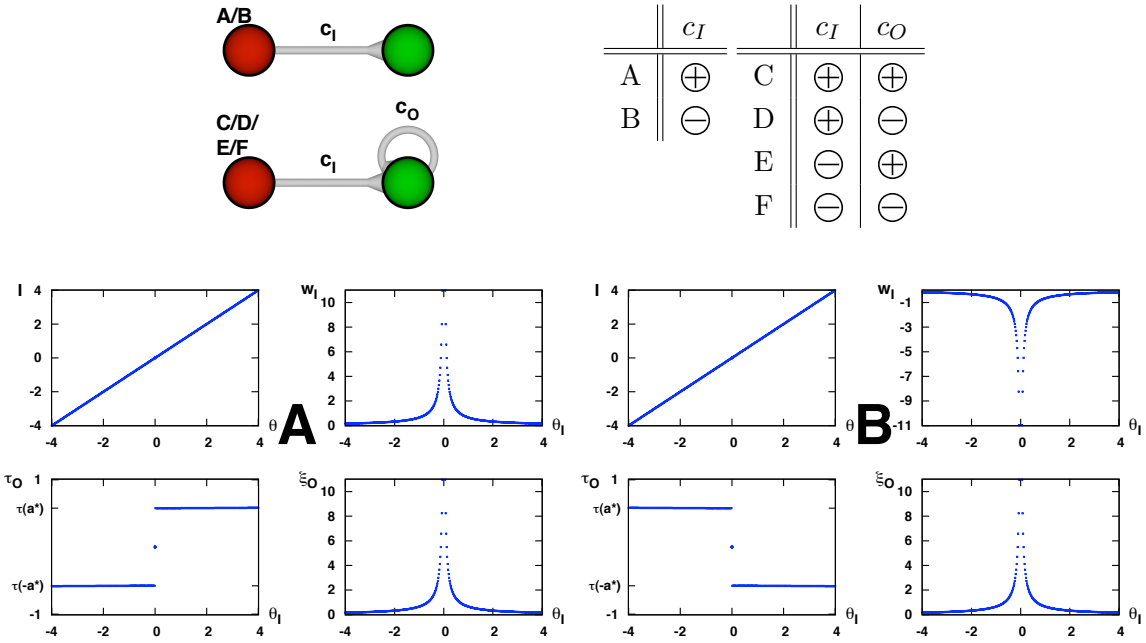


Figure 4.4: Numerical analysis of input-output neuro-modules. The upper half of the figure shows the six different combinations for the input-output neuro-modules (A/B/C/D/E/F). The lower half of the figure shows the bifurcation diagrams for the input-output neuro-modules without recurrent connection. The left hand side (A) shows the bifurcation diagrams for the module with excitatory, the right hand side (B) with inhibitory connection. Both plots show very similar behaviour. The synaptic weight is inverted due to the changed value of c. Note that the synaptic weight diverges, when the input approaches zero, and that the system remains zero, if started with an input of zero. Both effects are discussed in the text.

$$\Rightarrow \lim_{I(\Theta_I(t))\to 0} w^* \quad = \quad c_I \infty.$$

Discussion

The bifurcation diagram shows that the output neuron is, in both cases, able to regulate its activation for any presented value at the input neuron. For the neuro-module B, the outputs of the output neurons are inverted due to the inverted sign of the connecting synapse.

A diverging weight for input values close to zero can be considered as an undesired property of the Self-Regulating Neuron model for two reasons.

First, consider the cart-pole or pole-balancer problem (Barto et al., 1983; Geva & Sitte, 1993; Pasemann, 1998, 1997a; Pasemann & Dieckmann, 1997b, 1997a). In this standard benchmark problem, a controller has to regulate its input and output towards zero in order to balance the pole with minimal energy expenditure. Second, in technical applications there is a computational limitation which results in exceptions in the software if variables diverge.

As this work is concerned with a homeostatic neuron model within the sensori-motor loop, this is not considered as an undesired property, but as *one* property of the model. The reason is that a diverging synaptic weight due to an input approaching zero will have an effect on the actuators, as a result of the sensori-motor loop. The amplified change of the actuator will have an effect on the input causing the diverging behaviour of the neuron. Hence, it is very unlikely that an input close to zero will remain long enough to lead to computational limitations of the system. If desired, this could easily be caught technically by artificially limiting the values (see sec. 4.3).

The next chapters present different experiments which show how this property is utilised for producing desired behaviour when applied to a robot controller, and how an output of zero can be achieved using two output neurons for a single actuator. This is inspired by the concept of agonist/antagonists found in biology (Klinke & Silbernagl, 2005). The Braitenberg vehicle (see sec. 5.2.1) uses the diverging property in order to avoid an obstacle. The pole-balancer with four input and two output neurons is able to establish an output of zero for input values close to zero (see sec. 6.1).

4.2.4 Input-output neuro-module with recurrent connection

This section discusses the attractor dynamics of an input-output neuro-module with a recurrent connection of the output neuron (see fig. 4.4 C/D/E/F). The dynamics of these systems are given by the following set of equations:

$$\begin{aligned}
\Theta_O &:= 0 \\
a_O(t+1) &= c_O\xi_O(t)\eta_O(t)\tau(a_O(t)) + c_I\xi_O(t)I(\Theta_I) \\
w_I(t) &= c_I\xi_O(t) \\
w_O(t) &= c_O\xi_O(t)\eta_O(t) \\
\xi_O(t+1) &= \xi_O(t)(1+\beta g(t)) \\
\eta_O(t+1) &= (1-\gamma)\eta_O(t) + \delta h(t) \\
g(t) &= \tau(a^*)^2 - \tau(a_O(t))^2 \\
h(t) &= 1 + \tau(a_O(t))
\end{aligned}$$

The equations for the fixed point of the system with positive recurrent connection is given by

$$\begin{aligned}
a_O^* &= c_O\xi_O^*\eta_O^*\tau(a_O^*) + c_I\xi_O^*I(\Theta_I) \\
&= \xi_O^*(c_O\eta_O^* + c_II(\Theta_I)) \\
\xi_O^* &= \xi_O^* \\
&= \frac{a_O^*}{c_O\eta_O^*\tau(a_O^*) + c_II(\Theta_I)} \\
\eta_O^* &= (1-\gamma)\eta_O^* + \delta(1+\tau(a_O^*)) \\
&= \frac{\delta}{\gamma}(1+\tau(a_O^*)) \\
w_I^* &= c_I\eta_O^*
\end{aligned}$$

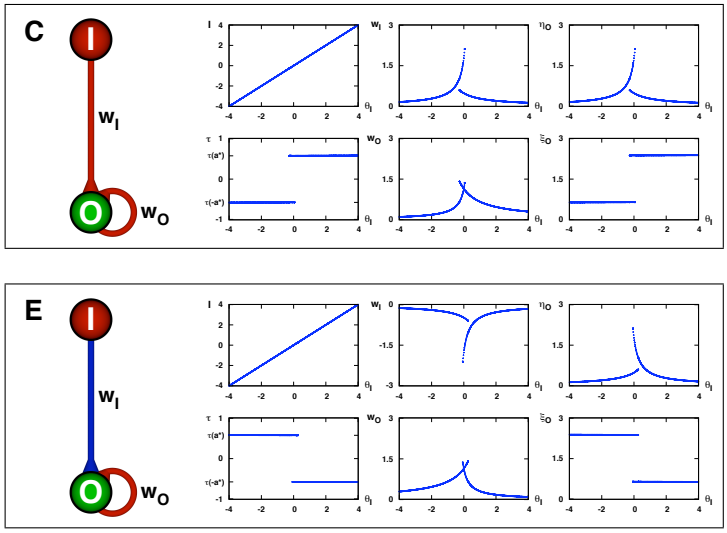


Figure 4.5: Input-output neuro-modules with excitatory recurrent connection. Both neuro-modules show a comparable behaviour. For both diagrams, the bias Θ_I of the input neuron was modified to produce the plots. The bifurcation diagrams of neuro-module E are axially inverted around the y-axis compared to those of module C, except for the synaptic weight w_I which is additionally axially inverted around the x-axis. This behaviour can be expected since inverting the synaptic connection w_I corresponds to changing the direction of the variation of Θ_I. The behaviour of both modules is comparable to the input-output neuro-modules without recurrent connections. The additional excitatory recurrent connection supports the homeostasis of the neuro-module, which leads to the observed behaviour.

$$w_O^* = \frac{a_O^* - c_I \eta_O^* I(\Theta_I)}{c_O \tau(a_O^*)}$$

Discussion

For neuro-modules with an excitatory self-connection (see fig. 4.5), the attractor of the output neuron is equivalent to the observed attractor of the output neuron without recurrent connection (see fig. 4.4 A/B). This can be expected, as the input-output neuro-module without recurrent connection already showed the homeostatic behaviour. From the analysis of the single neuron with excitatory recurrent connection, it is known that the excitatory connection is responsible for the homeostatic behaviour. The homeostatic behaviour of the excitatory recurrent connection together with the homeostatic behaviour of the input-output neuron leads to the observed behaviour. It must be noted that the recurrent connection does not establish a hysteresis effect as in the single neuron set-up.

For the input-output module with an inhibitory recurrent connection (see fig. 4.6), the behaviour is different. It shows both, the homeostatic regions known for the excitatory single neuron, as well as regions with oscillatory behaviour known for the inhibitory single neuron. There is an offset of the oscillatory region, which is axially symmetric for the inhibitory and excitatory connection between the input and output neuron. A detailed mathematical analysis

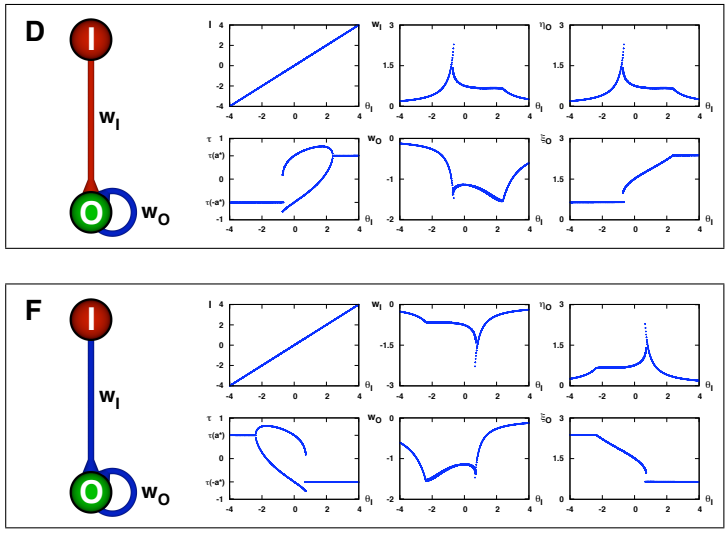


Figure 4.6: Input-output neuro-modules with inhibitory recurrent connection. Both neuro-modules show a comparable behaviour. For both diagrams, the bias Θ_I of the input neuron was modified to produce the plots. The bifurcation diagrams of neuro-module F are axially inverted around the y-axis compared to those of module D, except for the synaptic weight w_I which is additionally axially inverted around the x-axis. This behaviour can be expected since inverting the synaptic connection w_I corresponds to changing the direction of the variation of Θ_I. The behaviour of both modules is a superposition of the input-output neuro-module without recurrent connection, and the single neuron with inhibitory recurrent connection. Both plotted neuro-modules (D/F) show homeostatic regions and regions with a period-2 attractor. In contrast to the single neuron with inhibitory recurrent connection, there are no coexisting attractors.

of the neuro-modules is beyond the scope of this thesis.

4.3 Related work

The previous sections introduced the SRN model and analysed it with respect to its dynamical properties. As the SRN model is a form of synaptic plasticity or weight dynamics, it is comparable to other local learning rules for artificial recurrent neural networks. This section gives an overview of related local learning mechanisms and before discussing them with respect to the SRN model.

In the following sections, a variation of the previously defined notation (see sec. 3.2) is used. This notation is first introduced before the methods are discussed. The firing-rate of the pre- and post-synaptic neuron is denoted by u and v, respectively. The vector $\vec{u}$ consists of the firing-rate of all pre-synaptic neurons of the neuron v, and $\vec{w}$ is the corresponding weight vector. If not otherwise stated, a linear neuron model is assumed for simplicity. The firing-rate of the post-synaptic neuron v is then given by:

$$v = \vec{w} \cdot \vec{u}$$

and the change of the activation is denoted by:

$$\tau_t \frac{dv}{dt} = -v + \vec{w} \cdot \vec{u} = -v + \sum_{b=1}^{N_u} w_b u_b,$$

where N_u is the number of pre-synaptic neurons (length of the vectors $\vec{w}, \vec{u}$), and τ_t is a time constant that controls the firing-rate response dynamics (Dayan & Abbott, 2001).

In the following sections, seven related methods are presented and discussed with respect to the SRN model. The methods are the Hebb'ian Learning Rule, the BCM Rule, an approach to evolution and learning, homeostatic learning by Di Paolo and H. Williams, homeokinese, and finally, temporal sequence learning.

The basic Hebb'ian Learning Rule

Donald Hebb formulated this rule which states that the synaptic connection between two neurons is strengthened if the pre- and post-synaptic activities are correlated (see sec. 2.4.3). Formally, this reads:

$$\tau_w \frac{d\vec{w}}{dt} = v\vec{w}, \tag{4.38}$$

where the time constant, τ_w, controls the rate at which the weights change (Dayan & Abbott, 2001).

The equivalent general discrete time formalisation reads:

$$\Delta w_{ij}(t) = \mu f(a_i(t)) g(a_j(t)), \tag{4.39}$$

which means that the synaptic weight, w_{ij}, is varied with respect to the product of a function of the pre- and post-synaptic neuron, scaled with a learning factor μ. The most commonly used functions for f and g are the identity function and the transfer-function. These are given by:

$$\Delta w_{ij}(t) = \mu a_i(t) a_j(t) \tag{4.40}$$

$$\Delta w_{ij}(t) = \mu o_i(t) o_j(t). \tag{4.41}$$

The former states that the change of the synaptic weight is directly dependent on the pre- and post-synaptic activity. The latter is comparable and states that the change of the synaptic weight depends on the output of the pre- and post-synaptic neurons.

The basic Hebb'ian Learning Rule is unstable, because the synaptic strength grows unbounded. For a proof, the reader is referred to Dayan and Abbott (2001). To avoid the unbounded growth, upper and lower saturation conditions must be introduced. This, however, leads to synapses which remain at saturation and hence, the network loses the ability to differentiate the input signals (Dayan & Abbott, 2001). The Hebb'ian Learning Rule is more a principle than an actual practical mechanism. For this reason, many adaptations and workarounds have been introduced. For an overview, the reader is referred to (Dayan & Abbott, 2001; Arbib, 1995; D. E. Rumelhart, McClelland, & the PDP Research Group, 1986; Gerstner & Kistler, 2002a).

The BCM Rule – Bienenstock, Cooper, and Munro

The synaptic modification rule by Bienenstock et al. (1982) is a model for stimulus selectivity in the primary visual cortex of primates and humans. Neurons are modelled with a linear firing-rate, but may also have a positive-valued sigmoid-shaped function. The rule is given by

$$\tau_w \frac{d\vec{w}}{dt} = v\vec{u}(v - \Theta_v),$$

where Θ_v is a threshold on the post-synaptic neuron. The varying threshold avoids unbounded growth. One of the functions for Θ_v follows

$$\tau_t \frac{d\Theta_v}{dt} = v^2 - \Theta_v.$$

Other than the basic Hebb'ian Learning Rule, the BCM rule introduces competition among synapses. If some synapses are strengthened, this leads to an increase of the post-synaptic firing-rate, and consequently to an increasing threshold, and as a result, heterosynaptic weakening occurs.

According to the intention of the BCM rule (stimulus selectivity) applications have so far been shown in feed-forward networks (Cooper, 1986) trained with natural images (Law & Cooper, 1994).

Evolution and Local Learning – Floreano and Mondada

To the best of the author's knowledge, Floreano and Mondada (1996) are the first to use local learning mechanisms in a recurrent neural network to control an autonomous robot. In their approach, a genetic algorithm is used to evolve a set of properties for dynamic synapses in a neural network of fixed structure. The set of properties of a synapse is:

1. Synapse is driving or modulating (synapse type).
2. Synapse is excitatory or inhibitory (synapse sign).
3. The strength of a synapse is governed by one of the following rules:
   (a) Plain Hebb
   (b) Post-synaptic Hebb
   (c) Pre-synaptic Hebb
   (d) Co-variance
4. Possible learning rates are $r \in \{0.0, 0.3, 0.7, 1\}$.

Driving synapses correspond to the synapses introduced in the methods chapter (see sec. 3.2). Their values accumulate to the activity of the post-synaptic neuron. Modulating synapses dampen the output of the post-synaptic neuron. If no damping is present, the output follows the sigmoidal transfer-function.

The robot platform is the Khepera robot (Mondada et al., 1993) which is also used in this work (see chap. 5). In its basic form, it is equipped with eight infra-red proximity sensors, of which two are mounted to the back and six are equally distributed around the front of the robot, spanning approximately 180° of sensor coverage. Two wheels accelerate the robot.

The task to solve by evolution is an obstacle-avoidance behaviour. The environment is a corridor designed as a loop. The network consists of eight input neurons (one for each infra-red sensor), one hidden, and two output neurons (driving the two motors of the robot). Each input sensor is connected to the hidden and both output neurons. The hidden neuron is connected to itself and the output neurons:

$$\mathcal{C} = \begin{pmatrix} o_8 \\ o_9 \\ h_{10} \end{pmatrix} = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 1 \end{pmatrix}.$$

The result of their experiment is a wall-following behaviour. The synapses are modulated by the set of plasticity rules, determined by evolution, until a distance of approximately two cm to the wall on the right hand side is established. This distance is then kept constant. When tested in an open environment, the controller fails to maintain a straight trajectory.

This approach by Floreano and Mondada is the first conducted experiment in this context. There are two aspects which may be improved upon. First, the chosen experimental set-up does not require any adaptation. An obstacle-avoidance or wall-following behaviour is well-described by a static neural network. Second, the chosen model is comparably complicated. A network may have up to eight different types of synapses (four plasticity rules and two synapse types), sixteen if the sign of the synapse is taken into account. A full understanding of the underlying dynamics of such a system is very difficult.

Homeostatic Adaptation – Di Paolo

Another model for synaptic plasticity in recurrent neural networks, within the context of evolutionary robotics, was introduced by Di Paolo (2000). His work is motivated by Ashby's Homeostat (Ashby, 1954), Turrigiano's work on synaptic plasticity in biological systems (Turrigiano, 1999), and neuro-psychological experiments on visual distortion (Welch, 1974).

As with Floreano and Mondada, Di Paolo uses a genetic algorithm to evolve a recurrent neural network. The task is photo-taxis (light-seeking) in an open-ended, two-dimensional environment without any obstacles, and with randomly placed light sources which are activated sequentially. A mobile, wheel-driven robot, similar to the Khepera, is equipped with two light intensity sensors. After successful evolution of a light-seeking behaviour, the position of the two sensors are exchanged, in analogy to the vision inversion experiments in neuro-psychology.

Di Paolo uses a genetic algorithm on a structurally fixed, fully connected eight-neuron network (two input neurons, two output neurons, four hidden neurons) with a continuous-time neuron model. For each synapse, a set of parameters is encoded in the genes. Among these parameters are the learning rate, a linear damping factor for the plasticity rules, and an encoding for one of four possible plasticity rules:

1. R0: Hebb or Anti-Hebb'ian Learning

2. R1,2: potentiation (R1) or depression (R2) depending on the pre- and post-synaptic activity with respect to a threshold

3. R3: no plasticity

For a synapse, the corresponding plasticity rule is only activated when the firing-rate (output) of the post-synaptic neuron exceeds the boundaries of a defined interval ($o_i(t) \in [0.119, 0.881]$ with standard sigmoid transfer-function). The fitness-function favours controllers which spend the majority of the time close to a light source, but also those that required less regulation (inactive plasticity rules) during the evaluation, i.e. neurons whose firing-rate would remain longer within the defined boundaries.

Analyses of the evolved neural networks show that about half of the solutions are able to adapt to the inversion of the sensory configuration. After inversion of the sensors, the controller shows a behaviour which can be compared to Ashby's description of the Homeostat's reaction to distortion. Continous changes in the behaviour occur until a stable state is found and photo-taxis is re-established.

The approach is interesting and the results are convincing. In contrast to the previously discussed experiment by Floreano and Mondada, Di Paolo's set-up requires adaptivity. However, as with Floreano and Mondada, Di Paolo requires a set of different plasticity rules for each synapse, and a trigger mechanism as well.

Homeostatic Learning – H. Williams

The approach introduced by H. Williams (2004) is inspired by the work of Turrigiano (1999) and Di Paolo (2000). The main critics of Williams concerning the work of Di Paolo's approach is that the learning rules operate only locally on one synapse, and that there is no synaptic scaling as reported by Turrigiano. Consequently, Williams introduces a homeostatic plasticity rule for synaptic scaling in artificial neural networks. In addition, Williams also discusses a model for intrinsic plasticity, which is the regulation of the firing-rate of a neuron (output regulation), in contrast to synaptic scaling, which regulates the inputs to maintain a stabilised firing-rate. Both methods are presented and discussed in randomly-parametrised recurrent neural networks (H. Williams, 2004).

Two sorts of experiments are presented. First, the output dynamics of fully-connected, randomly parametrised networks are analysed for randomised inputs, followed by experiments with the same network architecture for a photo-taxis task. In the remainder of this section, only the latter is discussed.

The experimental set-up is comparable to the experiment presented by Di Paolo (see previous section). A two-wheeled Khepera-like robot is equipped with two light intensity sensors and placed in a featureless, open-ended, two-dimensional environment with a randomly-placed light source. A fully-connected recurrent neural network with six continuous-time neurons (two input neurons, two output neurons and two hidden neurons) is randomly-parametrised and the behaviour is observed in simulation. The random parameters are the initial weights and biases, the plasticity facilitation ρ, and the node learning rate μ for both methods:

$$\text{synaptic scaling:} \quad \Delta w = \rho\mu|w|$$

$$\text{intrinsic plasticity:} \quad \Delta\Theta = \rho\mu$$

To avoid unbounded growth, all weights are artificially limited within an interval $w \in [-8, 8]$.

The resulting photo-taxis behaviour of the system is only discussed descriptively, i.e. by means of the observed trajectories of the robot in the simulated environment, and not the manner in which the behaviour relates to the internal dynamics of the plastic synapses and neurons. The task is sufficiently solved, which means that solutions find the light source. Once found and approached, the robot withdraws and then cycles around the light source. Robots with synaptic scaling show smooth trajectories compared to those with intrinsic plasticity.

Williams' conclusion is that plasticity in continuous-time recurrent neural networks makes the networks much more sensitive to input signals. An assumption is made that homeostatic learning is a good substrate for artificial evolution, but that evolution would take much longer because of the increased number of parameters and because plastic networks need longer trials to enable plasticity to configure the network.

The presented approach by Williams is simpler compared to Di Paolo's and Floreano and Mondada's approaches. Williams uses only a single plasticity rule (either synaptic scaling or intrinsic plasticity) for the entire network. Randomly parametrised networks solve a photo-taxis behaviour. Like Di Paolo, Williams also uses a trigger mechanism and artificially limited weights.

Homeokinesis – Der and Pantzer

Another method related to the principle of homeostasis, which was introduced by Ashby as a model for adaptivity, is the approach by Der and Pantzer (1999). In their approach, homeokinesis is the governing principle. The idea is not to stabilise a stationary state (homeostasis) but a definite internal kinetic regime (homeokinesis).

The method operates on the prediction error of a self-model of the system. From the current state of a discrete-time dynamical system (a mobile robot), the self-model predicts the next state. Compared with the actual state obtained from the sensor values, this allows the computation of a prediction error. Briefly, a gradient descent method adapts the parameters of the self-model. The method has thus far been applied to wheel-driven robots (Der & Pantzer, 1999) and simulated snakes with a higher degree of motor complexity (Der et al., 2005). In any case, only networks with a single layer of neurons are trained.

Although the motivation and inspiration by Ashby's Homeostat governs both methods, Der and Pantzer's homeokinesis, and the Self-Regulating Neuron presented here, the homeokinetic principle is not a local learning rule, due to the use of the gradient descent method, and therefore, does not compare well to the Self-Regulating Neuron model.

Temporal sequence learning – Porr and Wörgötter

The final approach, presented as related work, is a local learning mechanism in the robotics context. The method introduced by Porr and Wörgötter (2003) is differs greatly from the methods discussed above. The neural network structure is closely related to the subsumption architecture of Brooks (Brooks, 1991a). A set of bandpass-filters is fed with δ-pulses as input. The overall output of the system is given by the weighted sum of the filter outputs (Porr, 2003).

Each filter produces a damped oscillation with a different frequency as a response to a presented δ-pulse. The different frequencies define the maximal time window in which temporal sequences can be learned. A special pathway is hand-designed and is referred to as the reflex pathway. In an obstacle-avoidance task, the reflex is triggered by a collision sensor and as a result induces an avoidance behaviour. The bandpass-filters, which are also referred to as prediction pathways, receive pulsed inputs from distance sensors. Hence, the prediction pathways receive input before the reflex pathway, which is related to the collision sensor. With a Hebb'ian learning method, the weights of all (predictive and reflex) pathways are modified until the temporal sequence of the range sensors and the collision sensor is learned, such that the reflex pathway finally does not contribute to the behaviour. This means that the controller has learned to predict the consequences of the stimulus, present in the prediction pathways, with respect to the reflex pathway.

Learning then continues and the pathway with the earliest signal dominates. This results in robots which finally show a circular trajectory for which the prediction pathways have a constant (zero) relation to each other (Porr & Wörgötter, 2003). In a more recent experiment, the same single-neuron, feed-forward architecture is used for line-following, using reduced camera information ("Chained learning architectures in a simple closed-loop behavioural context.", 2007).

Two main concerns must be stated at this point. First, the behaviour is developed by a predefined, hand-crafted reflex pathway. Second, the set of bandpass-filters is chosen artificially by the experimenter and defines the maximal time window which can be learned. This raises the questions of how this learning mechanism scales to other tasks which cannot be based on only one reflex or any reflex (e.g. walking behaviour), and how critical the limitation of the pre-defined time window is with respect to autonomous adaptive robots.

Discussion

The Hebb'ian Learning Rule is a general principle and less a working learning method. Nevertheless, it is the fundamental principle which governs the other learning mechanisms which have been presented. The SRN model also includes a Hebb'ian term, which is seen when the transmitter and receptor of the equation for synaptic weight $c_{ij}\xi_i(t)\eta_j(t)$ are replaced by their definitions (see eq. 4.14 and eq. 4.16).

The BCM rule is self-regulating in the sense that unbounded growth is avoided by a sliding threshold. Due to its motivation as a model for stimulus selectivity in the primary visual cortex, it is used either with a linear transfer-function or a strictly positive sigmoid, and thus far, only in feed-forward networks for image processing. There are no applications for recurrent networks embedded in the sensori-motor loop. The SRN model is not designed to model a function of a specific brain area but is designed as a general model for synaptic plasticity. Hence, its use is not limited to feed-forward networks or applications detached from the sensori-motor loop.

Floreano and Mondada are the first to use evolution and learning to generate a behaviour for an autonomous robot. Their method requires eight different types of synapses and the task solved by this method does not require any adaptation. In the SRN model, every neuron and every synapse is governed by the same homeostatic principle. In this sense, the SRN model is more general and minimal in comparison to that of Floreano and Mondada.

Di Paolo uses four regulating methods and an activation and deactivation mechanism for the plasticity rules. The SRN model does not require such a trigger mechanism. It is active at every instant during the life span of an autonomous agent.

H. Williams uses only one of two plasticity methods for the entire network. The first was designed as a model for synaptic scaling, the second as a model for intrinsic plasticity. Like Di Paolo, he also uses a trigger mechanism and also artificially limits the synapse weight. The SRN model, although divergent for a discussed configuration, is not limited artificially.

Both, Di Paolo and Williams only discuss a positive tropism. Other than a light source, their environment is featureless. A light seeker with obstacle avoidance will be presented later in this work (see chap. 6).

The homeokinesis principle by Der and Pantzer is a gradient descent method and hence a global learning rule. The SRN method is local. Every neuron is modulated only according to its internal activation.

The last method to be discussed is that of temporal sequence learning (also called ISO learning) by Porr and Wörgötter. Their approach is inspired by Brooks subsumption architecture. An ISO-learning network is a one layer feed-forward network with bandpass-filtered inputs. The synaptic weights are modulated by a Hebb'ian Learning rule until the temporal sequence of the bandpass filtered inputs is learned. Behaviours are learned with respect to a manually designed reflex behaviour. The SRN model is designed to work in recurrent neural networks of arbitrary structure and does not require a reference system to adapt to.

This concludes the discussion of related work. The chapter closes with conclusions and is followed in the next section with the transient analysis of the SRN model, while it is used to control a mobile robot in the sensori-motor loop.

4.4 Conclusions

This chapter introduced the Self-Regulating Neuron as an extension of the standard additive neuron model. The constant synaptic weight w_{ij} is replaced by pre- and post-synaptic neuron properties, which are referred to as the transmitter and receptor strength of a neuron. The transmitter reflects the internal neuron activity as a regulated firing-rate, and the receptor is modulated in order to regulate the neuron's activity towards the defined target value. Each neuron is a homeostatic unit, and the overall behaviour of a recurrent neural network results from the local interactions between the units. After the discussion of the three-dimensional Self-Regulating Neuron model, the dynamical properties were analysed mathematically and numerically. It was shown that the dynamics of a single SRN is comparable to the standard additive neuron, but that it also shows novel properties such as hysteresis for positive self-coupling, and co-existing attractors in the case of negative self-coupling. The chapter closed with a discussion of related work with respect to the SRN model.

Chapter 5

Experiments on Plasticity Parameters

The previous chapter introduced the SRN model and discussed its attractors for isolated neuro-modules, i.e. neuro-modules which are neither situated nor embedded and are, therefore, isolated dynamical systems. In this chapter the transient, i.e. behaviour-relevant dynamics of the Self-Regulating Neuron are analysed. Neuro-modules are now considered as embedded and situated systems, constantly driven by external stimuli in the sensori-motor loop.

Different obstacle-avoidance controllers for the Khepera robot with increasing structural complexity of the underlying neural network are analysed with respect to the behaviour relevance of the plasticity parameters β, γ, and δ. All experiments and analyses have been conducted in simulation and the behaviours have been verified on the physical platform.

The chapter is divided into three sections. The first section describes the experimental design, including the controller set-up, the simulation environment, and the physical robot platform. The second section presents the analysis of different obstacle avoidance controllers. The controllers are derived from two well-known controller from literature, the Braitenberg vehicle 3b (Braitenberg, 1984), and the MRC (Hülse & Pasemann, 2002). The controllers are chosen such that the structural complexity increases incrementally with respect to the size of the network and the plasticity parameters affecting the transient dynamics (behaviour). For each neural network, the parameters are varied and the corresponding transients and behaviours are analysed. The results of the analysis will lead to initial parameter settings used for the evolution of recurrent neural networks (chap. 6: Artificial Evolution of SRN-Controllers) and to implications for the modification of the evolutionary algorithm ENS^3, which are discussed in the third section. The chapter closes with a discussion of the experiments and their results.

5.1 Experimental Design

The behaviour of a system depends on the morphology, the controller and the environment (Pfeifer & Scheier, 1999; Pfeifer & Bongard, 2006; A. Clark, 1996). Thus, these three aspects provide the basics of the set-up which is used in all experiments which are presented in this chapter. This section covers this experimental set-up.

Figure 5.1: KHEPERA ROBOT. A) Khepera I Robot (front), B) Khepera II Robot (front), robot pictures taken from (K-Team, 2005), C) Schematic view of the sensors and actuators, redrawn from (K-Team, 1999). Green: The two actuators (DC motors). Red: the eight proximity and light intensity sensors. The indices of the sensors correspond the hardware specifications.

5.1.1 Morphology

The morphology of a robot includes the shape, the arrangement of the sensors and actuators, and their characteristics. Because the experiments are conducted in simulation and with the physical platform, both are discussed in this section.

There are two reasons to conduct the experiments and analysis in simulation. Setting up an experiment is less time consuming and can be reproduced with higher accuracy. This second property is relied on, when the dynamics of the controllers are analysed. Nevertheless, following Brooks approach and Tani's argumentation (see sec. 2.3), all behaviours are validated on a physical platform against the natural environment. In order to minimise the cost of porting the controller from simulation to the physical platform the simulation must be chosen appropriately, and the differences between simulation and physical robot must be known.

The section begins with the presentation of the physical platform, followed by the chosen simulator and concluded with a discussion of the differences between the two.

Robot Platform

The physical robot platforms used for the experiments in this chapter, are the Khepera I Robot (Mondada et al., 1993) and its successor the Khepera II (see fig. 5.1). The Khepera is a two-wheeled differential drive robot with a diameter of about 5.5cm (Khepera I). Its most prominent feature is its small size. This feature dispenses with the need for a large environment, thus, enabling experiments to be performed on a desk, for example.

The standard sensor and actuator configurations for the Khepera I and Khepera II robot are listed in table 5.1. Of the eight distance sensors, two are attached to the rear, and six are equally distributed around the front, spanning a sensor range of approximately 180° (see fig. 5.1 C and fig. 5.2) and a maximal distance of 50mm for the Khepera I, and 100mm for the Khepera II (see tab. 5.1).

Simulation Platform

The Khepera 2.0 simulator (Michel, 2005) is an open source software used to simulate the Khepera I robot on Linux/UNIX operating systems. It includes a graphical environment editor

	Khepera I	Khepera II	Khep. Simulator
Processor	Motorola 68331 16 Mhz	Motorola 68331 25 MHz	
Com.	Std Serial Port up to 38 kbps	Std Serial Port up to 115 kbps	
Update Freq. via serial port	≈ 80 Hz	≈ 125 Hz	
Motion	2 DC motors with incremental encoder (approx 10 pulses per mm)	2 DC motors with incremental encoders (approx 12 pulses per mm)	
Speed	Min: 2 cm/s Max: 60 cm/s	Min: 0.02 m/s Max: 1 m/s	Max: 40 cm/s assuming Khep. I
Sensors	– 8 IR proximity sens. 50 mm range – 8 light sensors	– 8 IR proximity sens. 100 mm range – 8 light sensors – Power Consumption sens.	– 8 IR proximity sens. 29 mm range – 8 light sensors
Diameter Height	55 mm 30 mm	70 mm 30 mm	50 mm n.a. (2D Sim.)
Weight	≈ 70 g	≈ 80 g	no inertia

Table 5.1: KHEPERA I/II SPECIFICATIONS. Khepera I/II Specifications (K-Team, 2005) and comparison to the Khepera 2.0 Simulator. For further information see (K-Team, 2005).

(see fig. 5.3), and an API (application programming interface) enabling robot control programs, written in C, to be executed by the robot. This API was used to include a communication interface which connects the simulator to the ISEE framework (see app. A). Table 5.2 provides the specifics for the simulated robot model. Differences between the simulated model and the physical platform are listed in table 5.1.

Discussion

The simulated and physical Khepera robot are similar in size, maximal speed and sensor configuration. Only the range of the proximity sensors and the lack of inertia in the simulation are different.

The shorter range of the sensors in the simulation is of no consequence when the controller is ported to the physical platform, because obstacles are detected earlier in the latter. Porting problems emerge when the environments differ e.g. in corridor sizes. The differences in the sensor

Property	Symbol	Implementation
Robot Diameter	d_R	50 mm
Wheel Diameter	d_w	10 mm
Noise	n_v	10% wheel speed
	n_Θ	5% orientation
Pose calculation		
Orientation	$\Delta\Theta$	$\Delta\Theta = (m_L - m_R)/200 \cdot n_\Theta$
Position	Δx	$\Delta x = +\frac{1}{4}(m_L + m_R) \cdot n_v \cdot \cos(\Theta)$
	Δy	$\Delta y = -\frac{1}{4}(m_L + m_R) \cdot n_v \cdot \sin(\Theta)$
		$m_L, m_R \in [-10, 10]$ angular wheel velocity
Sensor implementation		
Infra-red	s_i	fixed value out of 15 positions (see fig. 5.2)
		range ± 25 degree
		max distance 29 mm
Ambient Light	L_i	$[d = (250 - sqrt((dx)^2 + (dy)^2))/250.0]_0^1$
		$L_i = [500 - \sum_j \cos(1.5 \cdot \alpha) \cdot d \cdot 450]_{500}^{50}$
		range ± 60 degree
		max distance 25 cm

Table 5.2: KHEPERA SIMULATOR MODEL. Simulated Khepera in the Khepera 2.0 Simulator (Michel, 2005). The equations are derived from the source code.

implementation are shown in figure 5.2. The missing inertia of the simulator is not a drawback because the inertia of the physical robot is negligible with respect to the weight and maximal speed of the Khepera robot. In conclusion, this means that controllers analysed in simulation can be expected to show similar behaviour when applied to the physical platform.

5.1.2 Controller

All control structures are implemented using the same controller set-up which is discussed in this section. Every neural network has two input and two output neurons (see fig. 5.4), for which the pre- and post-processing is presented below.

Pre-processing

The two input neurons I_0 and I_1 represent two virtual proximity sensors. Each sensor value is calculated as the mean at three of the corresponding proximity sensors of the left and right front

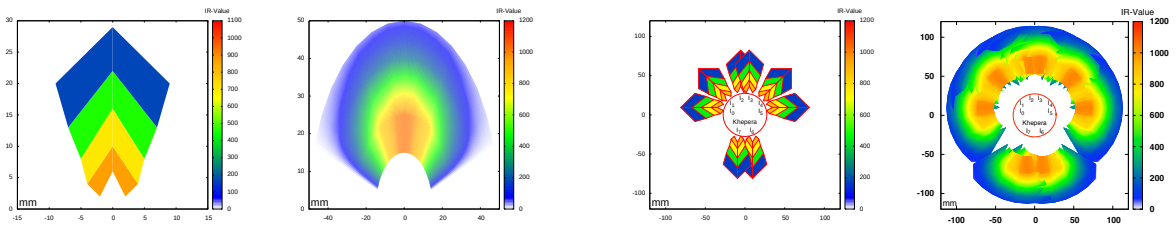


Figure 5.2: Comparison of real and simulated Khepera proximity sensors. From left to right: Single sensor range, simulated and real, robot sensor configuration, simulated and real robot. The sensor configurations are taken from the source code of the simulator and the sensor specifications of the physical sensors (Siemens SFH900). The simulated sensors are realised by 15 coordinates, each assigned a distance value. The maximal value of all coordinates that intersect with an obstacle is returned as the measured distance. The real sensor emits an infra-red beam and calculates the distance using the measured reflection angle.

side of the robot. The equation for the sensors is given by:

$$I_k = \frac{2}{3}\left(\sum_{j=3k}^{2+3k}\left(\frac{s_j}{s_{max}}\right)\right) - 1 \quad \in [1,-1], \qquad k \in \{0,1\} \tag{5.1}$$

where j denotes the index of the j-th front sensor s_j (see fig. 5.1C), and s_{max} the maximal possible value returned by a sensor. The resulting input is -1 if no obstacle is detected, or 1 if an obstacle is as close as possible. Measured distances in between are mapped linearly onto the interval. The input sensor value range of $I_{0,1} \in [1,-1]$ is chosen because it spans the non-saturated domain of the hyperbolic tangent transfer-function.

Post-processing

The output neurons denoted by O_2 and O_3, for the left and right motor respectively, deliver values in the interval $O_{2,3} \in [-1,1]$, which have to be post-processed in order to control the robot. The values are multiplied by a constant factor v, which is referred to as the *speed factor*, and are given by:

$$m_{L,R} = vO_{2,3},$$

where $m_{L,R}$ is the command passed to the left and right motor, respectively.

5.1.3 Environment

Two different evaluation environments are used in the experiments (see fig. 5.3 B/C). The first is a bounded featureless environment. For a reactive behaviour, the turning angles are constant if the system is in the same state before an obstacle is encountered. In an empty bounded environment, the distances between obstacles is maximised, so that the SRN neuron can settle

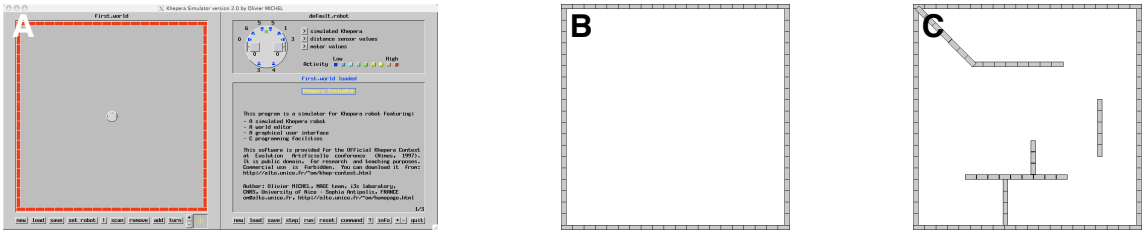


Figure 5.3: KHEPERA SIMULATOR AND EVALUATION ENVIRONMENTS. A) The Khepera 2.0 Simulator, B) The empty environment, C) Standard Environment, with a sharp corner added in the upper left corner. The colour scheme of the environments are changed for better visibility. The empty environment is used to analyse the transient dynamics of the SRN model. It ensures maximal distance between obstacles supporting the analysis of the effect of the plasticity parameters isolated from the history of the system. The second environment visualises the obstacle-avoidance and exploration behaviour. Note the sharp corner in the upper left region of the environment, and the obstacles, which have a small profile (e.g. right hand side of the environment), when approached accordingly.

towards its asymptotically stable state. This enables the system's response to an obstacle to be cleared from its history, so that the effect of the plasticity parameters can be better analysed.

A second advantage is that different parameter settings lead to discriminative behaviours, which then draw clearly distinguishable traces in the empty environment. This enables the detection of differences, already at the behaviour level before analysing the transients.

The second environment is a bounded environment with different classes of obstacles (see fig. 5.3). There are two significant types, the sharp corner in the upper left, in which controllers with a poor exploration behaviour get stuck (see next section), and obstacles which show a small profile when approached accordingly. The latter are not handled well by controllers with a poor obstacles-avoidance behaviour.

5.2 Obstacle Avoidance

In the previous chapter, different isolated neuro-modules were analysed with respect to their attractors. In this section, neuro-modules controlling a robot in the sensori-motor loop performing an obstacle-avoidance task are analysed with respect to their transient dynamics. For each structure, the plasticity parameters β, γ, δ are varied and their effect on the behaviour-relevant dynamics is determined. The controllers, which are presented here, are chosen with increasing structural complexity of the underlying neural network and increasing influence of the plasticity parameters β, γ, and δ.

The sections begins with a SRN implementation of the Braitenberg vehicle 3b (Braitenberg, 1984) followed by the MRC (Hülse & Pasemann, 2002).

The Braitenberg vehicle is selected as the first controller, because it is minimal with respect to its structure and because it is constructed of two disjunct neuro-modules from the previous chapter. Each neuro-module consists of one input and one output neuron (see fig. 5.4 B). The dynamics of this system is only related to the plasticity parameter of the receptor β, as the input neurons are not governed by the SRN model (see chap. 4). The controller is then extended by

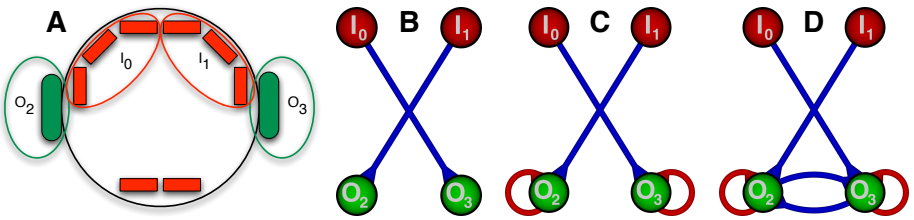


Figure 5.4: SRN-Braitenberg and SRN-MRC controllers. A) Schematics of the Khepera robot, B) Original Braitenberg vehicle (without recurrent connections), C) Braitenberg vehicle with positive self coupling of the output neurons, D) SRN-MRC. For a discussion, see text.

excitatory recurrent connections for the output neurons, which include the previously omitted plasticity parameters γ and δ controlling only one additional synapse.

The second chosen control structure is the MRC, which is the smallest static neuro-controller known to perform the obstacle-avoidance task with the ability to handle sharp corners. It is also a gradual increase of complexity compared to the extended Braitenberg vehicle, because the plasticity parameters γ, δ now control two synapses.

In the following discussion, the transient of the system, which is related to an occurrence of an obstacle is called the *echo* of the system. It is defined by the interval beginning with the system's state when the first presence of the stimulus at the input neuron(s) is detected and ending when the system has settled at its asymptotically stable state when the obstacle is no longer present. Echoes may overlap.

5.2.1 Braitenberg vehicle 3b without recurrent connections

The Braitenberg vehicles 3a/b (Braitenberg, 1984) are minimal implementations of positive and negative tropism. Depending on the coupling of the sensors and motors (ipsilateral vs. contralateral) and the type of connection (excitatory vs. inhibitory) a sensory stimulus either increases or decreases the corresponding motor (see fig. 5.5). This results in a turn being made, either towards, or away from, the stimulus, i.e. positive or negative tropism.

The experiments in this section are conducted with the SRN implementation of the Braitenberg vehicle 3b with inhibitory connections (see fig. 5.4 B). The Braitenberg vehicle 3b has ipsilateral connections. According to the pre-processing, inhibitory ipsilateral connections lead to the following behaviour. A stimulus at the left sensor reduces the right motor speed, resulting in a right turn. The reduction of the wheel speed remains as long as the stimulus is present. The vehicle turns away from the stimulus until it is no longer sensed and then continues with its translational movement.

The dynamics of the system is given by the following set of equations

$$\begin{aligned} \mathcal{C} &= \begin{pmatrix} c_2 \\ c_3 \end{pmatrix} = \begin{pmatrix} 0 & 0 & 0 & -1 \\ 0 & 0 & -1 & 0 \end{pmatrix} \\ (i,j) &\in \{(2,1),(3,0)\} \\ a_i(t+1) &= -\xi_i(t) I_j \end{aligned}$$

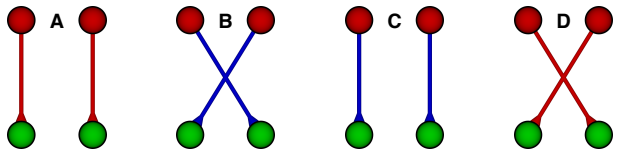


Figure 5.5: BRAITENBERG VEHICLES. The vehicle has two inputs (red) and two actuators (green). Assume that the sensors detect light intensity and that the actuators control two motors. The first two controllers realise the same behaviour. When a light source is detected at the left sensor, in the first case, the ipsilateral motor is excited, in the second the contralateral motor is inhibited. Both cases result in a negative tropism, such that the vehicles turns away from the stimulus. The third and forth structures realise a positive tropism. When a light source is detected by one sensor, in the first case the ipsilateral motor is inhibited, in the second the contralateral motor excited. In both cases the vehicle turns towards the light source.

$$\begin{aligned} \xi_i(t+1) &= \xi_i(t)(1+\beta g_i(t)) \\ g_i(t) &= \tau(a^*)^2 - \tau(a_i(t))^2 \\ w_i(t) &= -\xi_i(t) \\ \Delta\xi_i(t) &= \xi_i(t+1) - \xi_i(t) \\ &= \xi_i(t)(1+\beta g_i(t)) - \xi_i(t) \\ &= \beta\xi(t)g_i(t). \end{aligned} \tag{5.2}$$

The transient dynamics of this system is only related to one of the three plasticity parameters: β (see eq. 5.2). In the previous chapter, it was discussed that the parameter β has no effect on the asymptotic receptor value ξ^*. In contrast to the fixed point analysis of the previous chapter, this chapter is concerned with the transient and therefore behaviour-relevant dynamics. As the parameter β controls the growth and decay of the receptor, changes in the values of β effect the transients of the system. Figure 5.6 visualises this effect. In both plots a peek stimulus is presented to the input neuron (red). Both output neurons (green) have comparable values, and very similar transients towards the asymptotically stable state. However, the number of iterations (transient length) is higher for the smaller value of β. This corresponds to the results of the analyses of the previous chapter, which showed that receptor fixed point coordinate ξ^* is independent of parameter β, but also demonstrates that the transient behaviour is affected by β. It is this effect which is analysed with respect to an obstacle-avoidance behaviour.

Two representative controllers with different values of β are chosen for this analysis. The corresponding Braitenberg vehicles are referred to as B1 for $\beta = 0.1$ and B2 for $\beta = 0.01$ (see fig. 5.7). Obstacle-avoidance behaviours with other valus of β did not differ significantly from those shown for B1 and B2.

The parameter β determines the transient behaviour of the system. This means that the echo of a presented stimulus differs for the two presented controllers. In order to analyse the differing behaviours of the Braitenberg vehicle for variations of the parameter β, comparable conditions must be ensured each time an obstacle is sensed. Hence, both controllers are tested in the empty environment. The results of the experiment for both controllers, B1 and B2 are shown in figure 5.7.

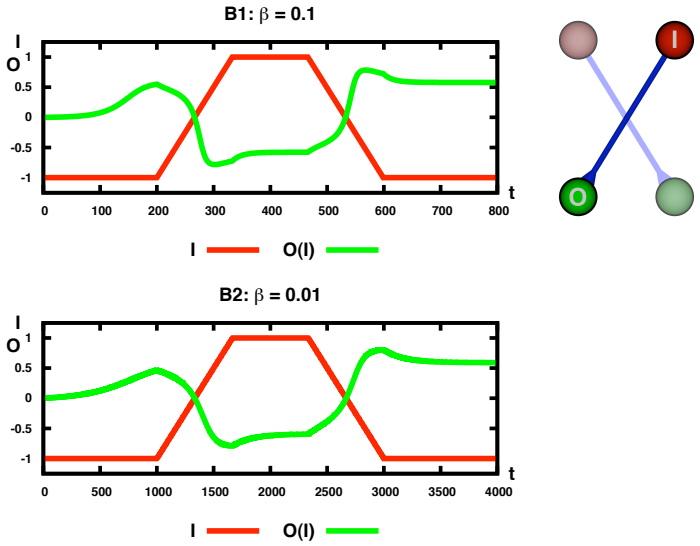


Figure 5.6: Peek plot comparison of vehicles B1 and B2. Peek plots for different values of β. Upper: B1 with $\beta = 0.1$, Lower: B2 with $\beta = 0.01$. Both plots show a very similar transient behaviour but the number of iterations in the lower plot is one order of magnitude larger. This is consistent with the findings of the previous section which showed that β does not affect the transmitter coordinate of the fixed point η^*. The plot also shows how the transients of the vehicles change for different settings of β.

Discussion of the observed behaviour

The given obstacle avoidance task is sufficiently solved by both controllers (see fig. 5.7 A/B). The resulting behaviours differ in their reaction to detected obstacles. The controller B1 shows a reduced turning angle, compared to B2. This is the result of an overshooting of the output neuron of B1. The transient plots (see fig. 5.7 C/D) of the controller are discussed first for B1 and then compared to B2 in the following paragraphs.

When no obstacle is present ($9600 \lesssim t \lesssim 9800$), all relevant controller properties (I_1, O_2, w_{12}) settle at their asymptotically stable values. As soon as an obstacle is sensed ($t \approx 9800$), the input I_1 increases, and because of the inhibitory connection w_{12}, the activity of the output neuron O_2 decreases. The output of O_3 is not affected by the changes in the input neuron I_1, so that the reduction of O_2 results in obstacle-avoidance behaviour of the vehicle. The turning behaviour causes a decrease of the sensor input I_2. The output O_2 follows again, but because the absolute value of w_{12} is increased, the output neuron overshoots ($t \approx 9850$). When it overshoots, it is larger, compared to the value of O_3 and, therefore, the vehicle now turns towards the obstacle. As the output O_2 settles, the trajectory of the vehicle stabilises to a straight movement.

In contrast, the output O_2 of the vehicle B2 does not overshoot significantly. Hence, once it turns away from the obstacle, it remains on a stable, straight trajectory. This explains why the controller with a larger value for $\beta = 0.1$ shows a smaller turning angle compared the controller with a value $\beta = 0.01$. This difference between the behaviours of B1 and B2 can be observed in simulation and on the physical platform.

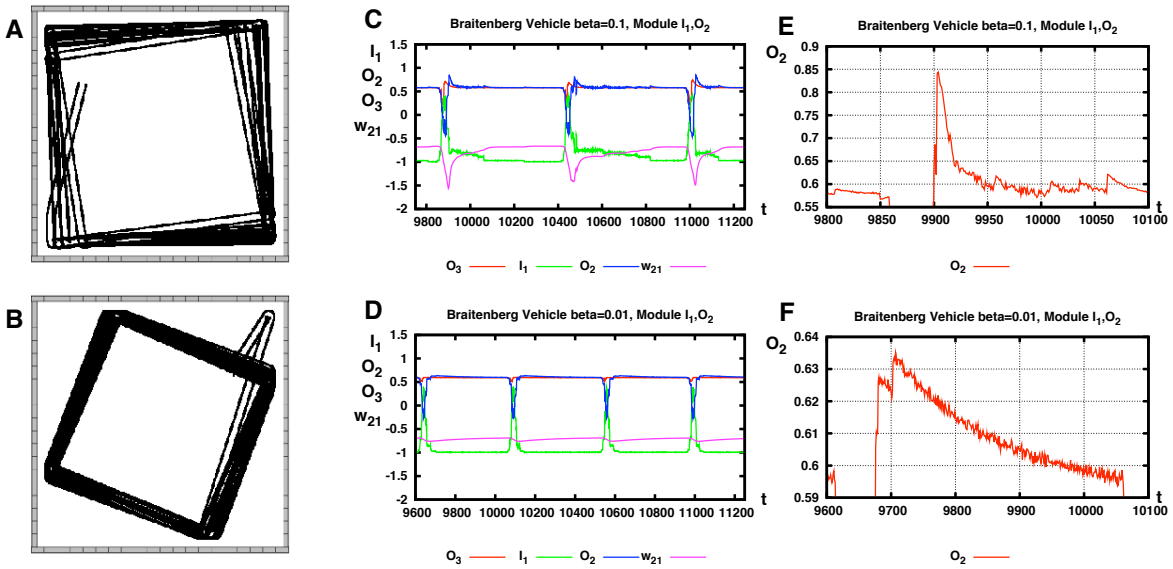


Figure 5.7: Behaviour comparison of vehicles B1 and B2. A,C,E) Braitenberg vehicle B1, $\beta = 0.1$. B,D,E) Braitenberg vehicle B2, $\beta = 0.01$. The trajectory of B1 (A) shows a smaller turning angle of the vehicle as a response to an encountered obstacle comparing to B2 (B). The transient plots of B1 (C) and B2 (D) show the cause for the difference in behaviour. The output neuron of B1 overshoots, while the same neuron for B2 shows a comparably stable behaviour. This is pointed out in two plots, in which the echo of the stimulus in the output neurons is zoomed into. It is shown that the echo in B1 (E) has a higher amplitude and shorter duration compared to B2 (F).

There is a second distinguishing property of the transients which is not observable in the behaviour. If the noise induced by the sensor is neglected, the echo of the presented stimulus in the synaptic connection of B2 is significantly larger compared, to that of the echo in B1. There is a notion of a memory effect, observable in B2, which will be discussed in the following section.

In summary, small values of β result in small changes but a long echo, while large values of β result in large changes but a short echo. The resulting behaviours are oppositional to the amplitude of the echo: B1 shows a smaller turning angle compared to B2.

5.2.2 Braitenberg vehicle 3b with self-connections

In this section the Braitenberg vehicle from the previous section is extended by excitatory recurrent connections on the output neurons (see fig. 5.4 C). This extension includes the previously excluded plasticity parameters γ and δ with minimal changes in controller structure and influence of the parameters. Only one additional synapse is included, and both newly introduced parameters only affect this single synapse.

This extended Braitenberg vehicle consists of two identical neuro-modules of type E which was presented in the previous chapter (see fig. 4.5). As mentioned previously, the ratio δ/γ determines the receptor coordinate of fixed point ξ^*. The behaviour of the module depends on the

transients of the system, and therefore, also on the magnitude of the parameters. The dynamics of the system are analysed with respect to varying magnitude and ratio of the parameters.

To isolate the effect of the plasticity parameters γ and δ, the parameter β is kept constant and comparably small for all experiments. A value of $\beta = 0.001$ is chosen, for all experiments which are presented from here one, as it is a magnitude smaller than the value presented in the previous section, and is, therefore, neglected in the following analysis.

The dynamics of the controller are given by the following set of equations:

$$
\begin{aligned}
\mathcal{C} &= \begin{pmatrix} c_2 \\ c_3 \end{pmatrix} = \begin{pmatrix} 0 & -1 & 1 & 0 \\ -1 & 0 & 0 & 1 \end{pmatrix} \\
(i,j) &\in \{(2,1),(3,0)\} \\
a_i(t+1) &= \Theta_i + \xi_i(t)\eta_i(t)\tau(a_i(t)) - \xi_i(t)I_j(t) \\
\xi_i(t+1) &= \xi_i(t)(1+\beta g_i(t)) \\
\eta_i(t+1) &= (1-\gamma)\eta_i(t) + \delta h_i(t) \\
g_i(t) &= \tau(a^*)^2 - \tau(a_i(t))^2 \\
h_i(t) &= 1 + \tau(a_i(t)) \\
a_i^* &= \Theta_i + \xi_i^*\eta_i^*\tau(a^*) + \xi_i^* I^* \\
\eta_i^* &= \frac{\delta}{\gamma} h_i^* \\
\xi_i^* &= \frac{a_i^* - \Theta_i}{\eta_i^*\tau(a_i^*) - I_j^*} \\
&= \frac{\gamma}{\delta}\frac{a_i^* - \Theta_i}{h_i^*\tau(a_i^*) - I_j^*} \\
w_{ii}^* &= \xi_i^*\eta_i^* \\
w_{ij}^* &= -\xi_i^*.
\end{aligned}
$$

A set of experiments are conducted with different magnitudes and ratios of γ and δ (see tab. 5.3). The initial parameter set $\gamma = 0.01$ and $\delta = 0.015$ was determined empirically in the previous chapter. First, the plasticity parameter δ is varied in order to change the ratio, i.e. the transmitter coordinate of the fixed point η^*. Then, the experiments are repeated with varying values for γ in order to alter the magnitude as well. The ratios $\delta/\gamma = 1.5$, 0.5 and 2 are chosen as they result in an obstacle-avoidance behaviour in the majority of the cases (see tab. 5.3). The initial ratio of 1.5 is the empirically determined value from the numerical fixed-point analysis of the previous chapter. The magnitude of γ was increased and decreased by one order. These nine representative values for γ and δ were chosen for presentation here. Other parameter settings were analysed but did not result in significantly different behaviour or did not show any exploration and obstacle-avoidance behaviour at all.

Discussion of the observed behaviour

The figure 5.8 shows the trajectories for all nine parameter configurations. From the plots, three classes of behaviours can be derived. The first class is a wall-following (parameter setting

	γ	δ		γ	δ		γ	δ
A	0.01	0.015	D	0.1	0.015	G	0.001	0.0015
B	0.01	0.005	E	0.1	0.15	H	0.001	0.0005
C	0.01	0.020	F	0.1	0.20	I	0.001	0.002

Table 5.3: EXPERIMENTAL SET-UP FOR γ AND δ. Parameter configurations for γ and δ for the Braitenberg vehicle with recurrent connection. The values are chosen beginning with the empirically determined values from the previous section $(\gamma, \delta) = (0.01, 0.015)$. The magnitude is increased and decreased by one order and the ratios of 1.5, 2, and 0.5 are chosen for experimentation, whenever the result in an obstacle-avoidance an exploration behaviour. For $\gamma = 0.1$ the ratios lead to a turning on the spot, so that other setting are presented.

$(\gamma, \delta) \in \{\mathrm{D}\}$). The second class is defined by those parameter settings that result in a constant turning angle: $((\gamma, \delta) = \{\mathrm{A}, \mathrm{B}, \mathrm{E}, \mathrm{F}, \mathrm{H}, \mathrm{G}\})$. The third class of behaviours is defined by the parameter settings that result in varying turning angles and, therefore, a better exploration of the environment, as more of the environment is covered by the trajectory of the vehicle $((\gamma, \delta) = \{\mathrm{C}, \mathrm{I}\})$.

To understand the difference in the behaviour of the modules, the transients of one of the neuro-modules within each controller is plotted (see fig. 5.9). The same neuro-module is depicted in all the plots, as the plotted data was captured while the robot was turning left within the environment. As the controller is symmetric, equivalent plots are received for the other neuro-module of the controller, i.e. when the robot turns right within the environment. What follows is a discussion of the three distinguishable classes of behaviors which have been identified from the trajectory plots seen in figure 5.9.

Wall-follower $(\gamma, \delta) \in \{\mathrm{D}\}$: The transient plot for the wall-following behaviour (see fig. 5.9 D) differs significantly from the other classes of behaviours. The mean absolute value of the strength of the input synapse w_{30} is larger and constantly growing. For the first set of obstacles ($t < 3250$), the transients are comparable to those of the other controllers. The recurrent connection w_{33} is subcritical (no hysteresis) and the duration of the echo is very small. The vehicle only turns as long as the stimulus is present. There is no overshooting of the output-neurons of the B2 controller, as discussed in the previous section. The wall-following behaviour results from the very small turning angles of the controller's avoidance behaviour.

There is a second effect observable in the transient plot. As soon as the absolute value of the synapse reaches a threshold value, the transients differ significantly from the transients of the other parameter settings. The reason for this discrepancy is that the synapse w_{30} amplifies the noise of the sensor which is present at the input neuron. If the synaptic connection grows large enough, this noise directly effects the behaviour of the system, resulting in the stochastic behaviour which is also seen in the trajectory (see fig. 5.8 D).

Constant turning angle $(\gamma, \delta) \in \{\mathrm{A}, \mathrm{B}, \mathrm{E}, \mathrm{F}, \mathrm{H}, \mathrm{G}\}$: The second class of behaviours is defined by the set of vehicles that show constant turning angles in response to an obstacle. For the

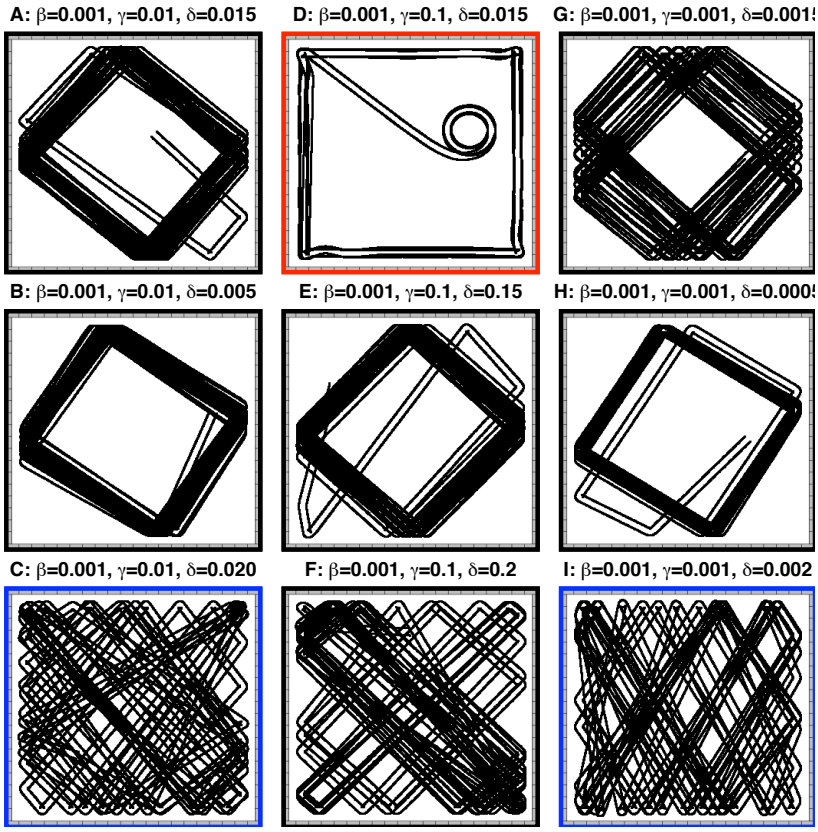


Figure 5.8: Trajectory plots for each pair γ, δ. Trajectory plots of all nine parameter settings. All plots are captured in simulation. Three classes of behaviours can be derived from the plots: wall-follower (D, red border), constant turning angle (A, B, E, F, G, H, black border), and varying turning angles (C, I, blue border). A speed factor of five was chosen to produce behaviours which are easier to distinguish.

controllers of this class of behaviours, all of the transient plots show the same properties which can be described as:

- (approximately) constant strength of the input synapse w_{30}
- only a short duration of the echo in the synapse w_{33} and in the output neuron O_3.

Because of these two properties the system is in the same state each time an obstacle is detected. Therefore, the reaction to an approaching obstacle does not differ, resulting in the observable constant turning angles. These are larger compared to those found in the plots of controller D, and result in a deflection of the obstacle and not in a wall-following behaviour.

Varying turning angle $(\gamma, \delta) \in \{\mathrm{C}, \mathrm{I}\}$**:** For the third class of behaviours, the exploring behaviour with varying turning angles, the two controllers show different properties of the transients. The controller with the parameter setting I has the same properties as the controller

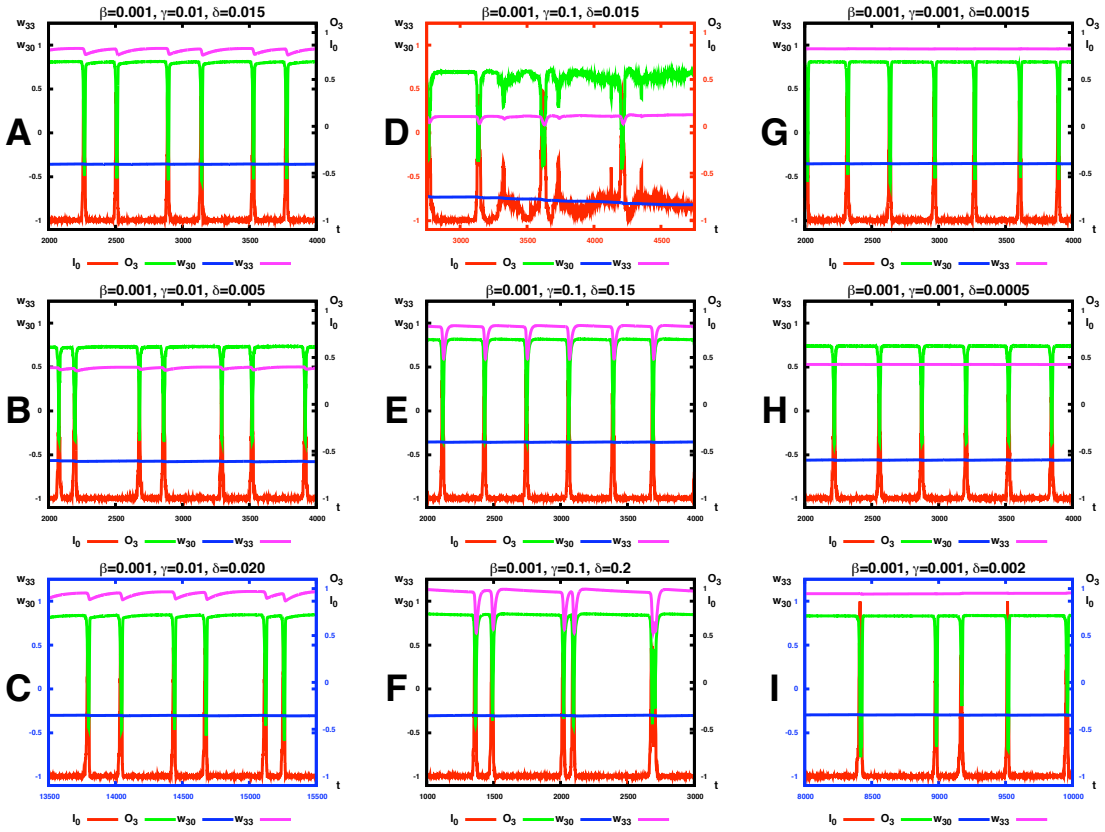


Figure 5.9: TRANSIENT PLOTS OF THE VEHICLES A–I. Transient plots of all nine parameter configurations. The border colour corresponds to the classes derived from the trajectory plots. For the discussion of the transients, the reader is referred to the description given in this section.

which produced the second class (short echo, $\dot{w}_{30}(t) \approx 0$ for almost every $t \in \mathcal{T}$). The difference in the behaviour-relevant dynamics is that the mean strength of the recurrent synaptic connection is constantly overcritical $w_{33} > 1$. An overcritical, positive, recurrent connection results in a hysteresis of the neuron (Pasemann, 1993). For the MRC (Hülse & Pasemann, 2002), it is the overcritical positive recurrent connection of the output neurons which determines the turning angle of the system.

The different turning angles in the trajectory plot result from the pre-processing (see eq. (5.1) and fig. 5.11). As mentioned previously, the virtual sensors are given by the mean value of the three corresponding sensors. When the vehicle approaches the wall with a large angle of incidence (AOI), two properties lead to an echo with a high amplitude. First, the sensors return higher values sooner, and second, more sensors are returning values unequal to zero (see fig. 5.11). Hence, while turning, at least two sensors detect the obstacle for a longer duration compared to the case where the approach is made with a smaller angle of incidence. The higher amplitude of the echo results in a larger turning angle of the system. This is additionally supported by the hysteresis, which drives the output neuron into saturation. This behaviour is shown in the

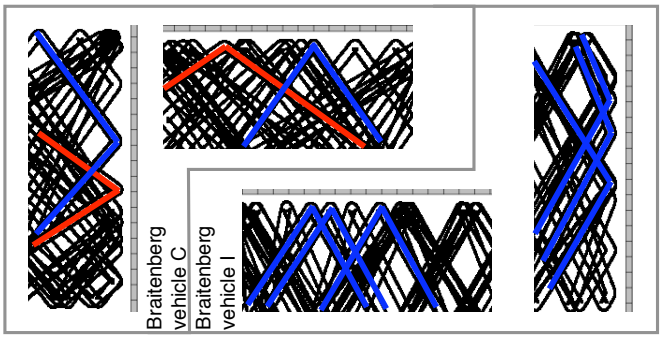


Figure 5.10: Comparison of trajectories of vehicles C and I. Differences between the trajectories of the vehicles C (left) and I (right). It can be seen that vehicle C shows different turning angles, where vehicle I has different turning angles for different regions of the environment only, here visualised for the upper and left boundary.

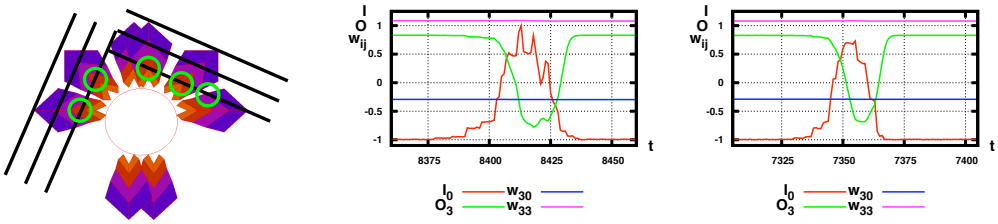


Figure 5.11: Analysis of the behaviour of vehicle I. From left to right: Sensor activation for different angles of incidence (AOI), transient plot for large AOI, transient plot for small AOI. The sensor activation shown for the AOIs measured from the trajectory plot of the controller. The three bars are equally distributed and range from no activation of any sensor to the detection of a close object from both AOIs. The transient plots show that the input peek for the high AOI (centre) is larger compared to that of a small AOI (right).

trajectory plot (see fig. 5.8 I) in the upper and lower boundary of the environment. The turning angle is larger than 90° so that the angle of incidence for the upper and lower boundary of the environment is larger compared to the angle found for the left and right boundary. In the latter regions, the corresponding sensors reach higher values later, and fewer sensors are activated, which results in a lower amplitude of the echo. This is equivalent to a smaller turning angle. The value of the overcritical self-connection is close to one, so that the width of the hysteresis domain can be neglected in this context. It still implements an output curve similar to that produced by a binary switch so that the output neuron is either in upper or lower saturation of the transfer-function. This is because an overcritical connection leads to two stable fixed points, separated by an unstable fixed point (Pasemann, 1993).

The controller defined by the parameter setting C shows different transients compared to the controller defined by the parameter setting I. The synaptic strength of recurrent connection w_{33} is overcritical but varies each time an obstacle is sensed. The duration of the echo is

large enough so that the synaptic strength does not reach its asymptotically stable state before the vehicles reach the next obstacle. In contrast to all the other controllers which have been presented, the value of the overcritical synaptic weight can even be decreased below one, leading to a breakdown of the hysteresis (Pasemann, 1993). The history of the system influences the dynamical properties of the controller which is equivalent to the behavioural response (turning angles) of the vehicle to encountered obstacles. This explains the good exploration behaviour of the system. As for the Braitenberg vehicle B2 of the previous section, this effect can be considered as a form of memory.

It must be noted, that the memory and, therefore, the parameter setting for which this effect occurs, is highly dependent on the environment and the morphology of the system. If the environment is larger, or the motors slower, the time between approaching obstacles would increase, causing the effect to vanish.

Discussion

The previous sections presented transient analysis depending on two subsets of the plasticity parameters. For each set of experiments, configurations were found which showed memory effects for the given morphology within the environments.

Based on the findings from the initial experiments, presented above, a subsequent experiment is designed to test the hypothesis of the memory effect. This is done by evaluating the performance of the Braitenberg vehicle, with recurrent connection and the extracted parameter setting $(\beta, \gamma, \delta) = (0.01, 0.01, 0.02)$ within the second environment (see fig. 5.12). The trajectory and the transients of one of the neuro-modules are shown in figure 5.12. This Braitenberg vehicle shows a novel behavioural property compared to those discussed previously. It is able to escape sharp corners. The minimal neural controller known to perform equally well in the task is the MRC (Hülse & Pasemann, 2002). It requires two more synapses forming an inhibitory loop between the output neurons (see fig. 5.14).

The transient plot (see fig. 5.12) of the Braitenberg vehicle illustrates the cause of the observable behaviour. The plot is captured as the robot enters the upper sharp corner of the environment. When the vehicle enters the corner, it first shows the wiggling behaviour known from Braitenberg vehicle 3b. If it senses an obstacle to its left, it turns right. It then senses the wall on the right side, causing it to turn left. In contrast to the standard Braitenberg vehicle with static synapses, the continuous presence of a stimulus increases the absolute values of the synaptic connections (w_{30}, w_{33}), until the recurrent connection (w_{33}) is overcritical. The parameters are small enough (longer transients), causing the hysteresis to exist long enough, such that the vehicle is able to escape the sharp corner. After escaping, the absolute synaptic weights decay to their asymptotically stable values.

In order to assure that this behaviour is the result of synaptic plasticity, three Braitenberg vehicles with static synapses are evaluated. The controllers are chosen with synaptic weights equivalent to the minimal, the mean, and the maximal values of the synaptic connection of the SRN Braitenberg vehicle of this section. All three controllers showed poor obstacle avoidance or exploration behaviour (see fig. 5.13).

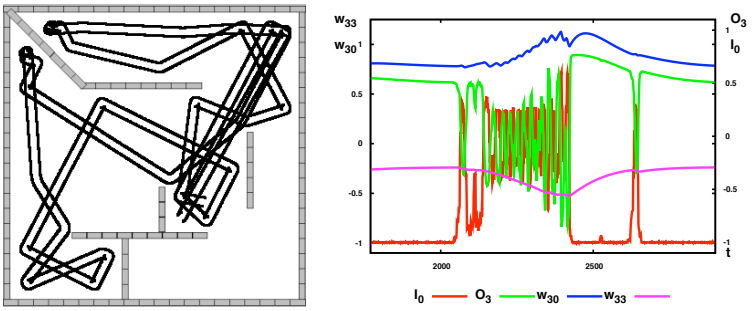


Figure 5.12: TRAJECTORY AND TRANSIENT PLOT FOR ADAPTIVE BRAITENBERG VEHICLE. Trajectory and transient plot for the Braitenberg vehicle resulting from the experiments. The configuration of the vehicle is $\beta = 0.01, \gamma = 0.01, \delta = 0.02, v = 3$. The trajectory plot (right) visualises the novel behavioural property. The vehicle is able to escape sharp corners, but still displays a good exploration behaviour. The transient plot (right) visualises how this is achieved. The constant presence of a stimulus raises the absolute strength of the synaptic connection w_{33} until it is overcritical, resulting in a hysteresis domain. Additionally, the absolute strength of the synapse w_{03} is increased so that changes in the output neurons are large enough to enable the vehicle to escapes the sharp corner. After leaving the region, the properties of the controller settle, ensuring a good exploration behaviour.

5.2.3 Minimal Recurrent Controller with Self-Regulating Neurons

The previous chapter discussed the transient dynamics for different settings of the plasticity parameters β, γ, and δ in a minimal control structure for obstacle-avoidance. For each neuron, the parameter β controls two incoming synapses (w_{ji}, w_{ii}) and the parameters γ and δ control one outgoing synapse (w_{ii}). The controller is also minimal with respect to the structural influence of the plasticity parameters. The dynamics of only one isolated output neuron is modulated by the different parameter settings, as there is no structural coupling, and the maximal loop length is one.

Consequently, the next step is to introduce a minimal structure with a loop of length two. The MRC (Hülse & Pasemann, 2002) is such a minimal obstacle-avoidance network for a two-wheeled robot (see fig. 5.14). In this section, this structure is implemented with the SRN model for three reasons:

1. Increase of structural complexity with respect to the Braitenberg vehicle with minimal increment.

2. Determine how the SRN model changes the structure–function relationship for well known minimal static structures.

3. Derive possible rules for the variation operator of the evolutionary algorithm ENS^3.

The first reason was already addressed at the beginning of this chapter. The second and third refer to alterations of the variation operator of the evolutionary algorithm ENS^3 (see sec. 3.3.1) which may be required. The SRN model differs significantly from the standard additive neuron model for which ENS^3 was originally designed. A different neuron model is very likely to

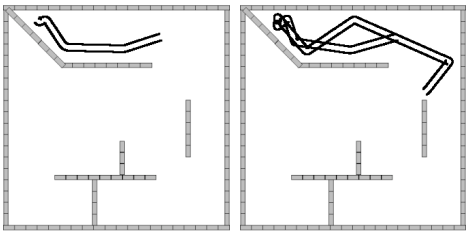


Figure 5.13: Behaviour comparison of static controllers derived from the adaptive Braitenberg vehicle. Evaluation of novel behavioural property. Three static controllers were tested, with minimal ($w_{33}, w_{03} = (0.741, -0.238)$), maximal ($w_{33}, w_{03} = (1.126, -0.525)$), and average values ($w_{33}, w_{03} = (0.933, -0.382)$) derived from the SRN Braitenberg vehicle. The classifications of minimal, average and maximal refer to the absolute values of the weights ($|w_{33}|, |w_{03}|$). The controller with minimal and average synaptic strength showed the same poor exploration behaviour (both shown in the left trajectory plot). The reason is the missing hysteresis domain, responsible for large turning angles. The controller with maximal synaptic weights was able to escape the sharp corner, but showed poor obstacle avoidance behaviour. The higher velocity of the system leads to shorter reaction time. If only one sensor detects an obstacle, the threshold to initiate a turning is reached later. This causes the controller to collide with the obstacle with a low profile (compare figure 5.12). Both these factors cause the poor behaviour.

demand other structures in order to operate comparably well. The analysis in this section leads to implications for the variation operator which favour structures on which the SRN performs well.

The section begins with the analysis of the symmetric MRC with the Self-Regulating Neuron model (SRN-MRC), which was derived from the original asymmetric MRC by removing the ipsilateral connections. The asymmetric SRN-MRC is considered at the end of this section resulting in the implications for the variation operator (see sec. 3.3.1).

Symmetric SRN-MRC

The original MRC (Hülse & Pasemann, 2002) is a minimal controller for an obstacle-avoidance behaviour of a two-wheel differential drive robot. Similarly, to the Braitenberg vehicle, it has two input and two output neurons and requires the same pre- and post-processing. In addition to the Braitenberg vehicle with recurrent connections, found in the previous section, the original MRC has four additional synapses, two ipsilateral connections between the input and output neurons, and an inhibitory even loop between the output neurons. In this section, the ipsilateral connections are removed for two reasons. First, the absolute strength of the synapses of the static asymmetric MRC can be neglected with respect to their contribution to the behaviour. Second, they cause an asymmetry in the control structure. The second aspect is discussed in the next section.

The output neurons of the static MRC have overcritical positive self-connections and an even inhibitory loop of length two (see fig. 5.14). The overcritical connections are responsible for a hysteresis effect which determines the turning angle of the vehicle for an encountered obstacle (Hülse & Pasemann, 2002). The even loop realises a co-existing larger hysteresis domain

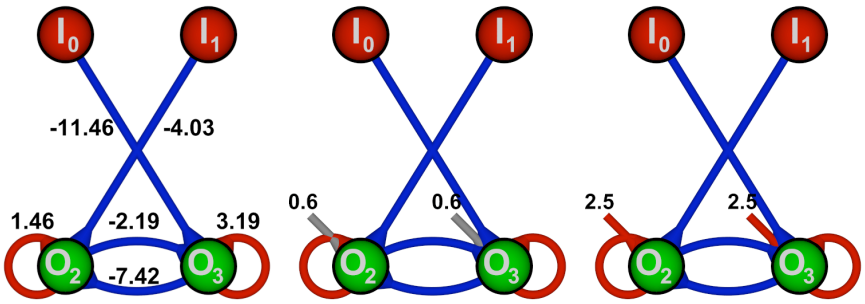


Figure 5.14: Different implementation of the MRC. From left to right: Structurally symmetric MRC (Hülse & Pasemann, 2002), Symmetric SRN-MRC with static bias, SRN-MRC with regulated bias. The SRN-MRC (centre) is generated from the static MRC by converting the connectivity matrix from $w_{ij} \in \mathbb{R}$ to $c_{ij} = \text{sign}(w_{ij}) \in \{-1, 0, 1\}$. A bias is required to achieve the desired behaviour (see text). It can be introduced in two possible ways, conventionally and through a bias neuron which is connected by regulated synapses. Both controllers are analysed with respect to their behaviour in this section.

which enables the vehicle to escape sharp corners with a turning angle of approximately 180°. The larger hysteresis is necessary, because the turning behaviour must continue even when the stimulus is not present at one of the sensors any more.

The symmetrised SRN-MRC presented here because it is the next incremental step of structural complexity compared to the Braitenberg vehicle with recurrent connection. The plasticity parameters of the output neurons control one additional incoming and one additional outgoing synapse. Further, the output neurons are structurally coupled to each other and no longer isolated from other SRNs. The symmetric SRN-MRC is defined by the following set of equations:

$$
\begin{aligned}
\mathcal{C} &= \begin{pmatrix} c_2 \\ c_3 \end{pmatrix} = \begin{pmatrix} 0 & -1 & 1 & -1 \\ -1 & 0 & -1 & 1 \end{pmatrix} \\
(i,j,k) &\in \{\{2,1,3\},\{3,0,2\}\} \\
a_i(t+1) &= \Theta_i + \xi_i(t)\eta_i(t)\tau(a_i(t)) - \xi_i(t)\eta_k(t)\tau(a_k(t)) - \xi_i(t)I_j(t) \qquad (5.3) \\
\xi_i(t+1) &= \xi_i(t)(1+\beta g_i(t)) \\
\eta_i(t+1) &= (1-\gamma)\eta_i(t) + \delta h_i(t) \\
g_i(t) &= \tau(a^*)^2 - \tau(a_i(t))^2 \\
h_i(t) &= 1 + \tau(a_i(t)) \\
a_i^* &= \Theta_i + \xi_i^* I^* \\
\eta_i^* &= \frac{\delta}{\gamma} h_i^* \\
\xi_i^* &= -\frac{a^* - \Theta_i}{I_j^*} \\
w_{ii}^* &= \xi_i^* \eta_i^* \\
w_{ij}^* &= -\xi_i^* \\
w_{ik}^* &= -\xi_i^* \eta_k^*. \qquad (5.4)
\end{aligned}
$$

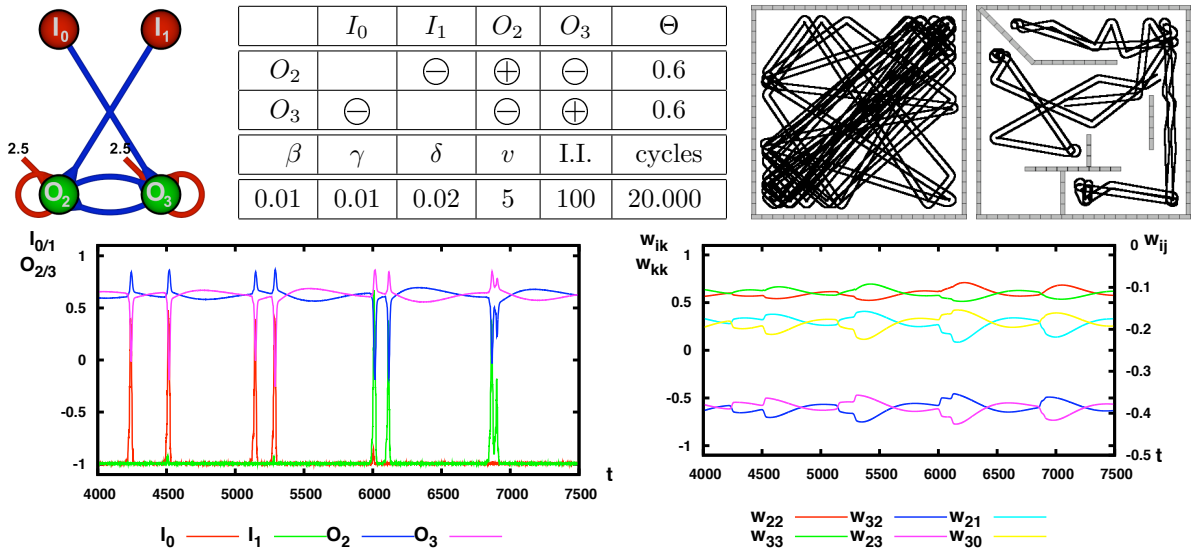
The activation function (see eq. 5.3) illustrates the discussed incremental structural increase of the controller. For each output neuron one additional term is added. This does not imply an incremental increase of the dynamical properties of the resulting controller, as a small modification of the structure of a neural network does not imply a small change of its dynamical properties. A two-neuron module can already show (qualitatively) every possible type of attractor, namely fixed point, periodic, quasi-periodic and chaotic attractors (Pasemann, 2002; Pasemann, Hild, & Zahedi, 2003), in contrast to the single neuron, which can only show co-existing fixed points (hysteresis phenomena) and period-2 oscillations (Pasemann, 1993). Therefore, an increment in structural complexity does not correspond to a incremental increase of the dynamical properties. Next, the observed behaviour of the SRN-MRC is discussed.

Observed behaviour – symmetric SRN-MRC structure: First experiments with the SRN-MRC showed a rotation of the vehicle on the spot around its own axis without any translational movement. Two circumstances lead to this behaviour. First, the sensor values include noise, which is passed over the input neurons to the output neurons. As a result, the output neurons are in different initial conditions. Second, the even inhibitory loop amplifies the difference as follows: The neuron with a larger initial activation will more strongly inhibit the other neuron, reducing its inhibitory effect which was imposed by the loop of length two. This is a feedback loop which drives the neuron with an initially higher-activation value towards the upper target value and the neuron with an initially lower-activation value towards the lower target value. This explains the rotational behaviour produced by the controller.

Observed behaviour – SRN-MRC with bias: The solution is a bias value which is large enough to compensate this effect. There are two methods may be used to introduce a bias value to the system. One involves adding a constant value to the activation equations ($\Theta_i \neq 0$), and the other involves adding a bias neuron, which is connected to the output neurons over dynamic synapses. The bias neuron is realised by an input neuron with a bias, which is not connected to any sensor. The differences between the attractors resulting from both methods were discussed in the previous chapter (see chap. 4) in which the bifurcations diagrams for the single neuron and the input-output neuro-module were analysed. The behaviour of both SRN-MRC variants is discussed next.

Fixed bias: The symmetric SRN MRC shows a good exploration behaviour in the square environment. The exact configuration of the controller, the trajectory and the transients are shown in figure 5.15. The trajectory plots indicate a memory effect of the controller. In the lower right corner the vehicle draws identifiable non-parallel traces, which are an indication of different turning angles. The transient plots support this observation. Close to time step $t \approx 6750$, two successive stimuli increase the synaptic weights before they are able to converge towards their asymptotically stable values. Therefore, the vehicle shows a different reaction (turning angle) to the second obstacle which it encounters, compared to the first.

To validate the memory effect imposed by the parameter set derived from the experiments with the Braitenberg vehicle in the previous section, other parameter settings are evaluated (see fig. 5.16). Variations of β are chosen for presentation here, but similar results are observed



	I_0	I_1	O_2	O_3	Θ
O_2		$\ominus$	$\oplus$	$\ominus$	0.6
O_3	$\ominus$		$\ominus$	$\oplus$	0.6
β	γ	δ	v	I.I.	cycles
0.01	0.01	0.02	5	100	20.000

Figure 5.15: Symmetric SRN-MRC with constant bias. From left to right and top to bottom: Controller, configuration table, trajectory plot in the empty and standard environment, transient plots of the neuron output and synaptic weights for the empty environment. In the table of the configuration, v refers to the speed factor, and I.I. to the number of initial iterations. The latter refers to the number of iterations for which the sensor values are present and the neural network is processed without applying the output to the motors. This amount of time is required for the controller to converge before a coordinated behaviour can be observed. From the trajectory plot in the empty environment it can be seen that the SRN-MRC with fixed bias draws non-parallel traces (see lower left corner). This implies different values for the synaptic weight when an obstacle is sensed. The transient plots support this observation. Around the time step of 6750, it can be seen that the synaptic strength depends on the time between two consecutive obstacles. This difference leads to the varying turning angles. The trajectory plot in the standard environment demonstrates the exploration and obstacle-avoidance behaviour of the controller in a non-trivial environment.

for the other parameters. The plots (see fig. 5.16) show that the memory effect disappears when the speed is varied (see fig. 5.16 top), and when the magnitude of β is changed (see fig. 5.16 centre/bottom).

Regulated bias: The symmetric SRN-MRC with a regulated bias value shows a good exploration behaviour in the square environment. The precise configuration, the trajectory and the transients of the controller are shown in figure 5.17. The SRN-MRC with regulated bias differs in both the trajectory and the transient plot from the SRN-MRC with fixed bias. The trace of the vehicle in the trajectory plot shows constant turning angles similar to those discussed for the Braitenberg vehicle I of the previous section. Variations of the parameter setting and the speed factor did not result in significantly different traces. The cause for the behaviour is visualised in the transient plot. The duration of the echo is too short to realise a memory effect for the given morphology (motor speed, sensor range, etc.) and environment (size of the square world).

The equations of the SRN-MRC with a bias neuron connected with excitatory synapses differ from the previous and read as follows:

$$
\begin{aligned}
\mathcal{C} &= \begin{pmatrix} c_2 \\ c_3 \end{pmatrix} = \begin{pmatrix} 0 & -1 & 1 & -1 \\ -1 & 0 & -1 & 1 \end{pmatrix} \\
(i,j,k) &\in \{\{2,1,3\},\{3,0,2\}\} \\
a_i(t+1) &= \xi_i(t)\eta_i(t)\tau(a_i(t)) - \xi_i(t)\eta_k(t)\tau(a_k(t)) - \xi_i(t)I_j(t) + \xi_i(t)I_\Theta \qquad (5.5) \\
\xi_i(t+1) &= \xi_i(t)(1+\beta g_i(t)) \\
\eta_i(t+1) &= (1-\gamma)\eta_i(t) + \delta h_i(t) \\
g_i(t) &= \tau(a^*)^2 - \tau(a_i(t))^2 \\
h_i(t) &= 1 + \tau(a_i(t)) \\
a_i^* &= \xi_i^*\eta_i^*\tau(a_i^*) - \xi_i^*\eta_k^*\tau(a_k^*) - \xi_i^* I_j^* + \xi_i^* I_\Theta^* \\
\xi_i^* &= \frac{a_i^* - \Theta_i}{\eta_i^*\tau(a_i^*) - \eta_k^*\tau(a_k^*) - I_j^* + I_\Theta^*} \\
\eta_i^* &= \frac{\delta}{\gamma} h_i^* \\
\xi_i^* &= \frac{a^*}{-I_j^* + I_\Theta^*} \\
w_{ii}^* &= \xi_i^*\eta_i^* \\
w_{ij}^* &= -\xi_i^* \\
w_{ik}^* &= -\xi_i^*\eta_k^*. \qquad (5.6)
\end{aligned}
$$

The difference of the SRN-MRC with fixed and regulated bias value is shown in the activation function (compare equations (5.3) and (5.5)). The latter is regulated by the receptor equation, which is the reason for the necessarily high value of $I_\Theta = 2.5$ compared to $\Theta_i = 0.6$ in the first case (see fig. 5.15 and fig. 5.17).

Asymmetric SRN-MRC

Up to this point different controllers with increasing structural complexity and minimal possible increments, were analysed with respect to their transient dynamics. All the networks which were presented, starting with the Braitenberg vehicle, were symmetric. Because the SRN model differs significantly from the standard additive neuron (SAN) model, it operates differently on comparable structures. This was already indicated in the previous section for the Braitenberg vehicle with recurrent connections, which was able to escape sharp corners with the SRN model, while failing to do so with the SAN.

This difference in the structure–function relationship has implications for the evolutionary algorithm ENS^3 which was initially designed for the SAN model. In this section, the originally published asymmetric MRC (see fig. 5.18) is converted to the SRN model by using the sign of each synapse, not considering the absolute value of the weight. Asymmetric here refers to the resulting asymmetric connection matrix $\mathcal{C}$. From the figure 5.18 it can be seen, that the ipsilateral connections of the original MRC are small and neglectable compared to the other

synapses. For the standard MRC these synapses do not contribute significantly to the behaviour of the system. For the asymmetric SRN-MRC the ipsilateral connections will have influence and it is this influence which is of interest here. The control structure is chosen for analysis because it is the next incremental step towards the symmetric MRC. An analysis of the resulting dynamics of the SRN-MRC will lead to recommendations for modifications of the variation operator of the evolutionary algorithm ENS^3.

Observed Behaviour: Without bias the asymmetric SRN-MRC rotates around its own axis without any translational movement. Introducing bias values results in a rudimentary obstacle-avoidance and exploration behaviour. The behaviour and the required bias values reflect the asymmetry of the structure. The exact configuration, the transient and the trajectory plots of the controller are shown in figure 5.18.

The resulting behaviour and the required bias values are best understood by analysing the rows of the connectivity matrix $\mathcal{C}$. For the output neuron O_3, the sum over the incident synapses is zero, for the neuron O_2 it is minus two:

$$\sum_j c_{3j} = 0 \tag{5.7}$$

$$\sum_j c_{2j} = -2. \tag{5.8}$$

At this point it must be noted that for any Self-Regulating Neuron all incident synapses are identical if the pre-synaptic neurons are in the same state. This is an effect of the receptor, which regulates all incomming synapses (see eq. 4.6).

In the case of the asymmetric SRN-MRC and the absence of an obstacle ($\{I_0, I_1\} = \{-1, -1\}$) this means that the input neurons have no effect on the output neuron O_3 as they cancel each other (see eq. 5.7). For the output neuron O_2, there is as increased influence, compared to the symmetric SRN-MRC, as the input is doubled (see eq. 5.8). The output neuron is driven towards the upper target value. The even inhibitory loop between the two output neurons has the same feedback effect, as discussed for the symmetric SRN-MRC, leading to constantly low value for output neuron O_3 and constantly high values for output neuron O_2. To compensate for this, the bias value of O_3 must be chosen significantly higher. A value of $\Theta_3 = 1.2$ suffices for the given rudimentary obstacle-avoidance and exploration behaviour, however, it is too large to allow the activity of the neuron to be regulated towards the target value. This explains why the asymmetric SRN-MRC can only turn left to avoid obstacles and, therefore, shows an overall poor behaviour compared to the controllers discussed previously in this chapter.

5.3 Conclusions

The analysis of the symmetric and asymmetric SRN-MRC enable the formulation of three implications for the modification of the evolutionary algorithm ENS^3:

1. The absolute value of the row sums of the connectivity matrix should be either zero or one. For the previous controller, this rule applied and led to good behaviours. For the

asymmetric MRC, this rule is violated, leading to a comparably bad behaviour due to necessary high compensations in the bias.

2. The original ENS^3 algorithm does not allow multiple synapses from a pre-synaptic to a post-synaptic neuron. For the same state of all pre-synaptic neurons, all incoming synapses on the post-synaptic neuron have the same absolute value. From experiments on the Aplysia it is known that synaptic growth is essential for long-term plasticity. Synaptic growth increases the number of connections of two neurons in a path of neurons (Squire & Kandel, 1998) and is one of the basic mechanisms for conditioning (see chap. 2.4). For the evolutionary algorithm ENS^3 this means that multiple synaptic connections of the same type (excitatory or inhibitory), should be allowed between two neurons. Mathematically this changes the trinary synapse type $c_{ij} \in \{-1, 0, 1\}$ to $c_{ij} \in \mathbb{Z}$. Elements of $c_{ij} \in \mathbb{R}$ are also possible. When $c_{ij} \in \mathbb{Z}, \mathbb{R}$ the search space is increased.

3. To enable the neurons to regulate towards both target values, the absolute bias should have the same order of magnitude as the target value. The assumption is that the order of magnitude of the bias value is related to a required row sum of the connectivity matrix. If the absolute value of the sum is larger than one, the neuron has more synaptic input which it can use to compensate for large bias values.

These implications are open and will be discussed at the end of this work (chap. 7: Discussion).

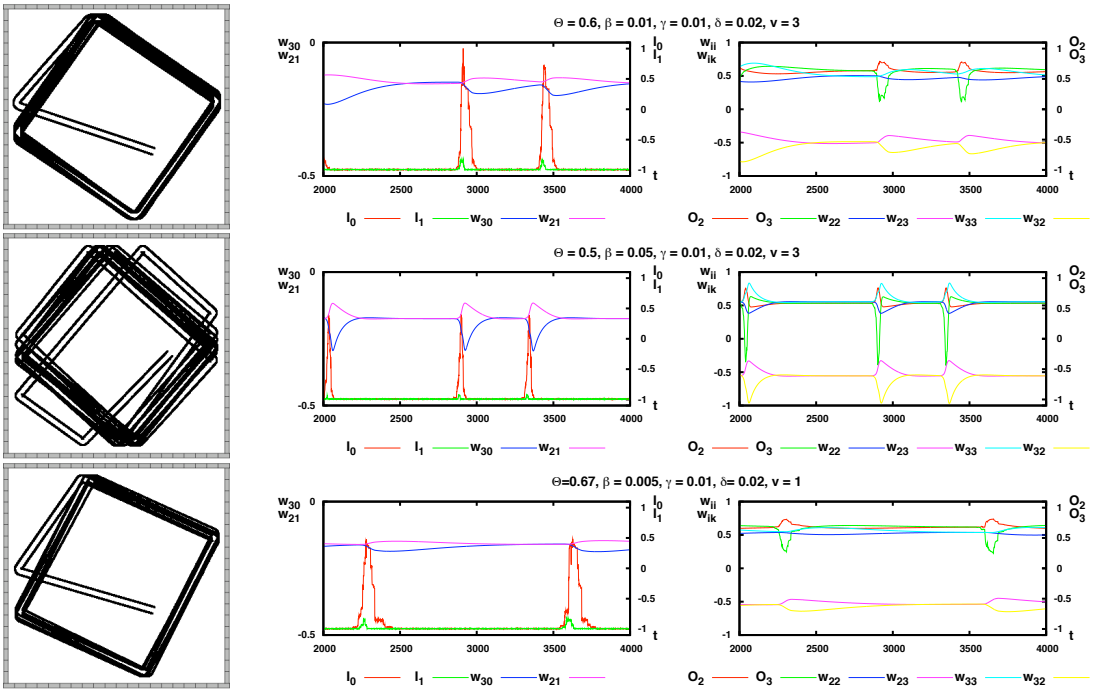
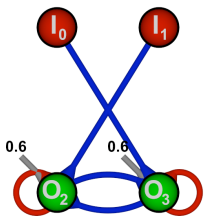
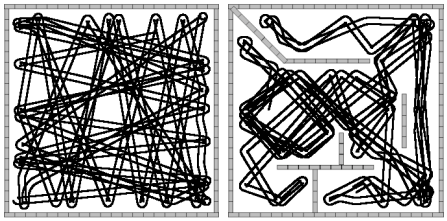
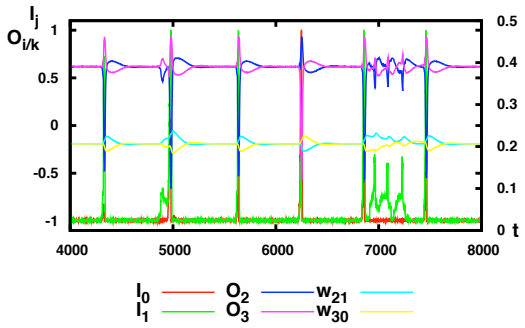


Figure 5.16: Symmetric SRN-MRC. Symmetric SRN-MRC with fixed bias with different settings of β. This figure shows the trajectory and the transient plots for different settings of β (the parameter configuration for each plot is shown above it). All three settings show an obstacle avoidance behaviour in the empty environment. Of particular interest, are the differences between the transient plots. The upper configuration is discussed in detail in the text and is shown here for reference. For the second configuration (centre, $\beta = 0.05$), the echo has a higher amplitude and a shorter duration resulting in an overshooting behaviour of the output neurons similar to the discussed behaviour of the Braitenberg B1 vehicle was presented in the previous section. For the third configuration (bottom, $\beta = 0.005$), the echo is of a small amplitude and a long duration, comparable to the transients of the Braitenberg vehicle I, also presented in the previous section. The synaptic weights of the controller can be considered as almost constant. This figure demonstrates exemplarily for β that the set of parameters, derived from the experiments with the Braitenberg vehicles and chosen for experimentation in this section (upper plot), lead to the desired transients for the MRC, which show memory effects for the given setting (morphology and environment).



	I_0	I_1	O_2	O_3	Θ
O_2		$\ominus$	$\oplus$	$\ominus$	$\oplus$
O_3	$\ominus$		$\ominus$	$\oplus$	$\oplus$
β	γ	δ	v	I.I.	cycles
0.01	0.01	0.02	5 (3)	500	20.000





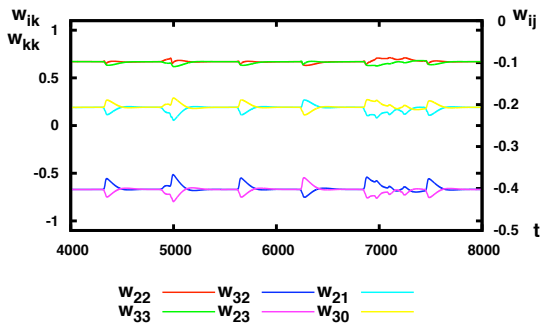
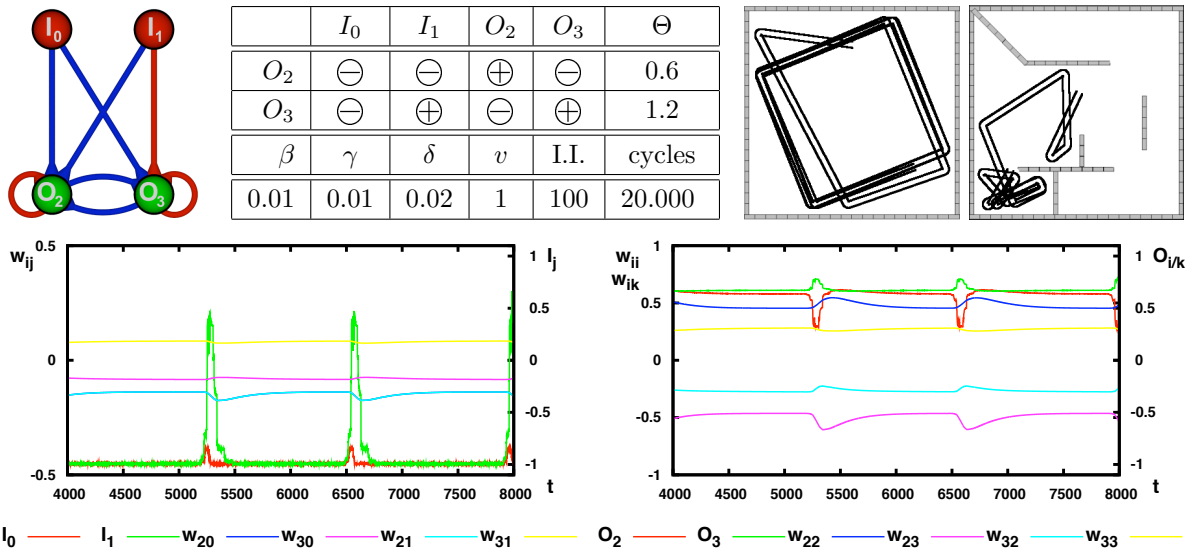


Figure 5.17: Symmetric SRN-MRC with regulated bias. From left to right and top to bottom: Controller, configuration table, trajectory plot in both, the empty and standard environments, transient plots of the neuron output and synaptic weights for the empty environment. In the configuration table, v refers to the speed factor, and I.I. to the number of initial iterations. The latter refers to the number of iterations for which the sensor values are present and the neural network is processed without applying the output to the motors. This time is required for the controller to converge before a coordinated behaviour can be observed. The speed factor in brackets refers to the speed factor in the standard environment, which is differs in order to produce the observed behaviour. In contrast to the SRN-MRC with static bias, this controller show constant turning angles in the empty environment. The cause is visualised in the transient plots, which shows echoes that are small in amplitude and duration. This corresponds to an approximately static controller. The constant turning angles are also observable over a wide range within the standard environment.



	I_0	I_1	O_2	O_3	Θ
O_2	$\ominus$	$\ominus$	$\oplus$	$\ominus$	0.6
O_3	$\ominus$	$\oplus$	$\ominus$	$\oplus$	1.2
β	γ	δ	v	I.I.	cycles
0.01	0.01	0.02	1	100	20.000

Figure 5.18: Asymmetric SRN-MRC with constant bias. From left to right and top to bottom: Controller, configuration table, trajectory plot in both, the empty and standard environments, transient plots of the neuron output and synaptic weights for the empty environment. In the table of the configuration, v refers to the speed factor, and I.I. to the number of initial iterations. The latter refers to the number of iterations for which the sensor values are present and the neural network is processed without applying the output to the motors. This time is required for the controller to converge before a coordinated behaviour can be observed. The asymmetric MRC was derived from the original published MRC (Hülse & Pasemann, 2002) by converting the connectivity matrix of the synapse strength. The asymmetric SRN-MRC requires an asymmetric distribution of the bias values. The bias of the output neuron O_3 is twice as large compared to that of O_2 which results from the asymmetric connectivity matrix (see text for detailed explanation). The asymmetric SRN-MRC can only avoid obstacles by turning left, which explains the poor obstacle-avoidance and exploration behaviour.

Chapter 6

Artificial Evolution of SRN-Controllers

The previous chapters presented the analysis of the attractor landscape and the transient dynamics for different configurations of the Self-Regulating Neuron model, decoupled and coupled from the sensori-motor loop. In this chapter, artificial evolution is used to generate control structures to solve a task, given the morphology of the robot and the environment. The first experiment is the pole-balancer, a standard benchmarking problem for trainable controllers (Barto et al., 1983; Geva & Sitte, 1993; Riedmiller, 1996; Pasemann, 1997a; Spong, 1998). The pole-balancer or cart-pole experiment is chosen to demonstrate how a zero output of a controller for an input, converging towards zero, can be realised with the SRN model. This answers the question whether the diverging behaviour of the SRN model, a desirable property in the case of the Braitenberg vehicle's obstacle-avoidance behaviour (see sec. 5.2), limits the model otherwise.

The second experiment is a light-seeker that has to find a light source under varying ambient light conditions. The controller has only two light intensity sensors and cannot distinguish between ambient light and a light source purely through the sensor values.

Both experiments are presented separately in two sections. Each section begins with a detailed description of the experimental and evolutionary set-up, followed by a discussion of the evolved neuro-controller. The controller was generated using the evolutionary algorithm ENS^3 (see chap. 3) implementation within the ISEE environment (see app. A). The controllers presented here were selected by virtue of the simplicity of their structures. Naturally, the structure evolution algorithm also produced intermediate structure of larger scale and complexity[†].

The chapter begins with the pole-balancer, followed by the light-seeker, and closes with a discussion of both experiments.

6.1 SRN Pole-balancer

This section describes the pole-balancer or cart-pole experiment as an example of how the diverging behaviour of the SRN model with an input close to zero can be handled. The cart-pole experiment is optimal for this purpose, because the task is to minimise the input (cart

[†]Scale and complexity refer to the number of neurons and synapses in the structure.

track limits	± 2.4 m
failure angles	$\pm 12^\circ$ (deg)
gravity	$-9.91\ \frac{\mathrm{m}}{\mathrm{s}^2}$
length of pole	1 m
mass of the cart	1.0 kg
mass of the pole	0.1 kg
magnitude of the control force (F)	10.0 N
integration time step (Δt)	0.001 s (0.02 s)

Table 6.1: Standard Cart Pole parameters. The parameters are taken from Geva and Sitte (1993). The standard integration time step of 0.02s is adjusted to 0.001s to match the requirements of the SRN model, which has longer transients compared to the SAN.

position and velocity, pole angle and angular velocity) together with the output (force applied to the cart) in a minimal, non-trivial experimental setting. In what follows, the experimental and evolutionary set-up are presented first, followed by a discussion of the evolved controller with the best performance.

6.1.1 Experimental set-up

The cart-pole experiment is a standard benchmark for neural network learning algorithms (Geva & Sitte, 1993) and is most often used in the form first described by Barto et al. (1983). An inverted pendulum is mounted onto a cart, which can move along a limited track in only one dimension. The controller fails, if either the pole deviates too far from its centre, or the cart reaches the boundaries of the track (see tab. 6.1). The goal is to balance the pole while finding and then remaining at the centre of the track.

The standard cart-pole parameters are taken from Barto et al. (1983) and Geva and Sitte (1993) with a modification to the integration time step (see tab. 6.1).

The inputs for the controller are the position x of the cart, the velocity $\dot{x}$ applied to the cart, the pole angle Θ, and its angular velocity $\dot{\Theta}$ (see tab. 6.2). The controller's output is the force applied to the cart. As the SRN model does not allow stabilisation to occur around zero for a zero input, a different controller concept is required. Instead of a single output neuron determining the cart force, two output neurons are chosen. The sum of both determines the resulting force (see tab. 6.2). Theoretically, the resulting summed force is doubled compared to the standard benchmark problem. Analysis of the evolved controller will show that the output neurons have opposite signs at every time step, so that this is never the case, and hence, the controller setting remains within the definition of the benchmark problem.

The cart-pole was simulated with the PoleBalancer program, an implementation of the equations given by Geva and Sitte (1993) and Pasemann and Dieckmann (1997b), initially written by Martin Hülse as part of the ISEE package with modifications by the author of this work to match the requirements of the SRN model.

Name	Symbol	Input-Neuron	Pre-Processing
Cart position	x	I_0	$I_0 = \frac{x}{2.4}$
Pole Angle	Θ	I_1	$I_1 = \frac{15 \cdot \Theta}{\pi}$
Cart velocity	$\dot{x}$	I_2	$I_2 = \frac{\dot{x}}{2.4}$
Pole angular velocity	$\dot{\Theta}$	I_3	$I_3 = \frac{15 \cdot \dot{\Theta}}{\pi}$
Name	Symbol	Output-Neuron	Post-Processing
Cart force, Agonist	f_0	O_4	$f_0 = 10 \cdot O_4$
Cart force, Antagonist	f_1	O_5	$f_1 = 10 \cdot O_5$
Cart force, Resulting	f	$O^+ = O_4 + O_5$	$f = f_0 + f_1$

Table 6.2: CART-POLE CONTROLLER INPUT/OUTPUT. Equations are taken from Pasemann and Dieckmann (1997b).

6.1.2 Evolutionary set-up

The evolution was initialised with an empty network, only consisting of input and output, but no hidden neurons and no synaptic connections. Corresponding to the sensor and actuator configuration, the initial controller consisted of four input and two output neurons. To reduce the search space for this experiment, the plasticity parameters β, γ, δ were not exposed to variation and were fixed to the setting derived in the previous chapter.

The fitness-function published by Pasemann and Dieckmann (1997b) leads to solutions, which use high oscillations of the output neuron as a control strategy. Although the task was sufficiently solved, solutions which consume less energy, in terms of the deviation of the output neurons, are favoured. Hence, the fitness-function was altered, and is given by

$$F = \sum_{t=1}^{T} f(t) \tag{6.1}$$

$$f(t) := 1 - c_0|x(t)|_0^1 - c_1|\Theta(t)|_0^1 - c_2\mathcal{F}(t) - c_3|O^+(t)| - c_4\mathcal{Z}(t) \tag{6.2}$$

$$f : \mathbb{N} \mapsto \mathbb{R}$$

$$T \in \mathbb{N} \quad \text{[Life span of an individual, defined by experimenter]}$$

and $f(t)$ and has the following characteristics:

1. Each time step of the life time of the controller is rewarded $(+1)$.
2. The distance of the cart to the centre of the track is punished $(-c_0|x(t)|_0^1)$.
3. The deviation of the pole from its centre is punished $(-c_1|\Theta(t)|_0^1)$.
4. The integral of the applied force is punished at every time step $(-c_2\mathcal{F}(t))$.
5. The energy consumed at every time step is punished $(-c_3|O^+(t)|)$.
6. The number of sign changes for each output neuron is punished $(-c_4\mathcal{Z}(t))$.

where the terms of $f(t)$ are given by:

$$
\begin{aligned}
c_k \in \mathbb{R}^+, &\qquad \text{Fitness-function coefficients, open to user interaction.}\\
i, j \in \{4, 5\} &\qquad \text{Indices of the output neurons.}\\
\Delta t &\qquad \text{Integration time step (see tab. 6.1).}\\
|y|_0^1 &:= \frac{|y|}{\max\{y\}}, \qquad y \in \mathbb{R}\\
\mathcal{F}(t) &:= \sum_{s=0}^{t} O^+(s)\Delta t\\
\mathcal{Z}(t) &:= \begin{cases} 0 & : \ \operatorname{sign}(O_i(t)) = \operatorname{sign}(O_i(t-1)), \forall i \\ 1 & : \ \operatorname{sign}(O_i(t)) = \operatorname{sign}(O_i(t-1)) \text{ and} \\ & \quad \operatorname{sign}(O_j(t)) \neq \operatorname{sign}(O_j(t-1)), \forall i, j : i \neq j \\ 2 & : \ \operatorname{sign}(O_i(t)) \neq \operatorname{sign}(O_i(t-1)), \forall i \end{cases}.
\end{aligned}
$$

The c_k are coefficients that are open to user interaction, so that the weighting of the fitness-function terms can be changed during the evolution. This is one technique to avoid the bootstrap problem, which is known as incremental evolution (Nolfi & Floreano, 2000). During the first generations the coefficients are chosen such that the task is not too difficult, and then adapted as soon as the performance of the population increases. In this case, at the beginning of the evolution, only the life time was rewarded and the punishment terms were added sequentially, when the average performance of the population increased to match the new affordances (less oscillation, less energy consumption, etc.).

The evaluation time was set to 25,000 time steps (25 seconds), of which the first 5000 iterations were not taken into account for the fitness-function, i.e. the fitness-function was calculated only for the last 20,000 time steps. This corresponds to the initial convergence phase for the controller, in which actions which would normally be punished (e.g. large oscillations) and would therefore, lead to overall low fitness, are permitted. This technique, of the so called *warm-up steps* or *warm-up phase* is another way to avoid the bootstrap problem. Consider two different controllers during the first generations, of the evolution. Both controllers survive almost equally long, but the first controller has zero output, so that the pole falls towards the failure condition almost immediately. The second controller survives longer, because it oscillates and therefore stabilises the pole for a short period of time. Obviously, the second controller should be favoured by evolution, but because the first has zero output, it is not punished, while the second is punished for high oscillations. Without warm-up steps, the first, undesired controller would receive a higher fitness value. These considerations also hold for the selection of good controllers throughout the evolution process and not only for the first generations. Without warm-up steps, the population is likely to collapse, with respect to the fitness distribution.

To avoid specialisation and elitism, every controller was evaluated 15 times with random initial conditions. An evaluation is referred to as a try. The conditions were the same for each controller in one generation. The overall fitness is then given by the sum of the fitness value of each try.

At the beginning of each evaluation time, 500 initial iterations are processed. For this initial set of iterations, the initial condition (input values) are presented to the controller and the

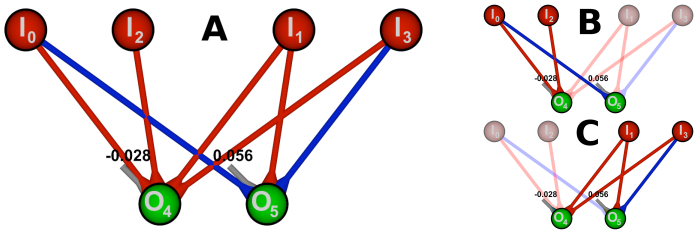


Figure 6.1: CART-POLE NEURAL NETWORK AND ITS SUB-MODULES. A) Evolved neural network solving the task. B) Cart positioning sub-module, C) Pole balancing sub-module. For a detailed discussion see text.

controller is iterated, but the output is not applied to the motor of the cart. The simulation is frozen during the initial iterations. This time is required by the SRN model in order to converge against values which enable good performance. The initial set of iterations and the integration time step depend on the choice of the plasticity parameters as they determine the modulation rate of the synapses (see sec. 5.2).

In addition to the two abortion criteria defined by the standard cart-pole experiment (track length and maximal deviation of the pole), another abortion criteria was introduced. The evaluation is terminated if computational limitations are reached. This occurs if the computation leads to NaN (not a number) due to a diverging receptor strength. In the chosen Java (Sun Microsystems, 2007) implementation the hyperbolic tangent returns NaN for values larger than approximately 710.

6.1.3 Results

This section discusses the best controller generated by artificial evolution. As the artificial evolution algorithm ENS^3 alters parameters and the structure of the neural networks. Controllers with very different topology and size[†] were generated. The selected controller is chosen for presentation here because of its performance and minimal structure.

This solution is the overall best-performing controller with respect to different criteria (see fig. 6.2), namely:

1. Fitness value as defined by Pasemann and Dieckmann (1997b) (see fig. 6.2 A).
2. Time successfully balancing the pole (see fig. 6.2 B).
3. Fitness value with respect the fitness-function given above (see eq. 6.1 and fig. 6.2 C).

Observed behaviour

The presented controller is able to balance the pole and, as required, does not use oscillation to stabilise it (see fig. 6.3 A and fig. 6.4 A). Due to the agonist and antagonist configuration of the controller, one neuron stabilises at the upper target value, while the other stabilises at the

[†]Size refers to the number of synapses and hidden neurons.

lower target value in order to stabilise the pole (see fig. 6.4 A.II). The sum of both neurons O^+ is close to zero, when the pole and the cart are centred (see fig. 6.4 A.III).

The behaviour of the controller can be classified into five cases, depending on the initial condition and the evaluation time observed (see fig. 6.2 and fig. 6.3). Each of the cases is described in the following and related to the equations given by the neural network structure in the following section.

1. Evolved Case: For the first case (see fig. 6.3 A), the transients of the variables of the controller and the cart are shown for an initial condition with the highest achieved fitness value ($x(0) = -0.24, \Theta(0) \approx 0.079$). The plot shows that the pole is successfully stabilised after 5,000 time steps for the remaining 20,000 time steps. According to the evolutionary set-up, this leads to the highest possible fitness. The output neurons O_4 and O_5 stabilise at the lower and upper target value, respectively, such that O^+ is close to zero. The evolution of the synaptic strength over time (see fig. 6.4 A) indicates, that this case is not stable, as the synapses diverge. This leads to the second case.

2. Switching of output neurons and period-2 oscillation: The second case occurs after time step 50,500 of the same initial conditions as the first case (see fig. 6.3 B and fig. 6.4 B). First, both output neurons converge to zero, followed by a period-2 oscillation with varying amplitude and mean value (time step $\approx$ 50,750 - 51,000), until the oscillation decreases and the output neurons converge towards the target values, but in reversed order:

$$\begin{aligned}(O_4(t \gtrsim 5000), O_5(t \gtrsim 5000)) &\approx (-a^*, a^*)\\ (O_4(t \gtrsim 52000), O_5(t \gtrsim 52000)) &\approx (a^*, -a^*).\end{aligned}$$

The reason for the switching is indicated by the input values $I_0(t) \propto x(t)$ and $I_1(t) \propto \Theta(t)$ (see fig. 6.4 B.I). The inversion of the output neurons happens when the two inputs are almost equal. The convergence of the output neuron towards zero constantly increases the error term of the receptor function $\xi(t)$ (see eq. 4.8) which leads to the divergence of the synaptic strength. The large absolute values of the diverging synapses cause the overshooting reaction of the controller to small variations of the input which explains the oscillatory behaviour. This holds until the output neurons converge to their target values $\pm a^*$ enabling the receptor strengths to converge to their asymptotically stable values, and finally stabilise the entire system.

3. Quasi-periodic behaviour: The third case (see fig. 6.3 C and fig. 6.5 C) is shown for a different initial condition ($x(0) = 0.72, \Theta(0) \approx 0.026$). In this case, the weights of the synapses are small compared to those of the previous case. What follows is that small deviations of the pole angle and small values of the pole angle velocity are followed by the output neurons. The resulting output variation is not large enough to catch the pole immediately, but large enough to keep it from falling towards the failure condition (see tab. 6.1). Hence, the output neurons follow the pole angle deviation in a quasi-periodic transient, resulting from the sensori-motor loop, until the pole is finally stabilised again.

4. Period-2 oscillation followed by switching of the output neurons: This case ($x(0) = 0.24, \Theta(0) \approx -0.079$) is similar to the second case. The difference is the order in which oscillation and switching occurs (see fig. 6.3 D and fig. 6.5 D). In the fourth case, first the period-2 oscillation is observed, followed by the switching of the output neurons. The reason for the switching is the same as for the second case: the sum of the cart-pole position and the pole angle inputs is almost zero. However, the reason for the oscillation differs. The weights of the synapses in the fourth case have grown large over time before the inputs sum up to zero (see fig. 6.5 D). Small disturbances of the pole angle and pole angle velocity result in large changes of the output of the output neurons, resulting in an overshooting behaviour. This overshooting behaviour is the reason for the period-2 oscillation. While the cart-pole is kept stable by the overshooting behaviour, the inputs sum up to zero, as previously described, which is an indication for the switching of the output neurons. The oscillation decreases as the output neurons reach their target values.

5. Irregularity in phase-space plot: This phenomenon is only visible, if the phase-space plot is created with a high resolution. An explanation of the behaviour can be found if the transients are compared to the transients of a closely-related initial condition (see fig. 6.6 and fig. 6.7). The plots show the transients for two different but similar initial conditions ($x_E(0) = 1.919, \Theta_E(0) = -0.0342, x_F(0) = 1.9, \Theta_F(0) = -0.03$). The only significant difference is visible in the transients of the synaptic strengths. In the case of the initial condition for which the controller fails (E), their absolute values are smaller compared to those of the initial condition for which the pole is successfully balanced (F). This leads to the conclusion, that a stable behaviour can only be achieved if the synapses have reached a threshold value, before the inputs demand for a stabilisation behaviour. This also depends on the bias values. For changes of the biases, both cases (E & F) lead to a stabilisation behaviour. The phenomenon also occurs for neuro-controller with static synapses (Pasemann, personal communication, 30. July, 2007). It is assumed that this is related to asymmetry in the control structure and the equations defining the cart-pole experiment. To fully analyse and understand this effects, a mathematical analysis, which includes the sensori-motor loop, is required. This means that the interaction of the controller, the body and the environment has to modelled and taken into account as a single system.

In the following section, the description of the observable behaviour is related to the controller dynamics, i.e. the equations given by the neural network structure.

Explanation of the observed behaviour

The structure of the controller (see fig. 6.1 A) reveals how the stabilisation behaviour is achieved. This is best visualised when the input neurons are clustered and considered separately. The first cluster contains the input neurons $I_0 \propto x$ and $I_2 \propto \dot{x}$ and the output neurons O_4, O_5 (see fig. 6.1 B). The speed of the cart (I_2) only modifies one output neuron (O_4). Depending on the sign of the output neurons, the resulting speed (O^+) is either increased or decreased as a result of the changed position input (I_0). Which of the two cases applies depends on the input of the current location of the cart (I_0). If the cart position is positive, O_4 is positive and O_5 is negative. A positive speed input (I_2) will increase the speed (positive O^+), while a negative

input will decrease the current speed (negative O^+). Summarising, the sub-module consisting of I_0, I_2, O_4, O_5 will drive the cart towards the centre (zero coordinate) of the track.

The same considerations also hold for the sub-module I_1, I_3, O_4, O_5 (see fig. 6.1 C). If the pole angular velocity (I_3) is positive, O_4 is positive and O_5 is negative. A positive pole angle (I_1) will further increase the cart-pole force, while in the case of a negative pole angular velocity it will decrease the cart-pole force. This sub-module realises a centring of the pole. Compared to the cart-position sub-module, the pole angle has a doubled influence, which reflects that the centring the pole has a higher priority than the centring of the cart. The overall behaviour of the controller is then given by the superposition of the two sub-modules.

To understand the cases of the behaviour of the controller presented above, the underlying equations must be taken into account. The dynamics of the controller are given by:

$$\begin{aligned}
a_i(t+1) &= \Theta_i + \xi_i(t) \sum_{j \in \mathcal{I}} c_{ij} I_j(t), \quad \forall j \in \mathcal{I}, t \in \mathcal{T} : \eta_j = 1, \quad i \in \{4,5\} \\
\xi_i(t+1) &= \xi_i(t)(1 + \beta(\tau(a^*)^2 - \tau(a_i(t))^2) && (6.3) \\
a_4(t+1) &= \Theta_4 + \xi_4(t)\big(+ I_0(t) + I_1(t) + I_3(t) + I_2(t)\big) && (6.4) \\
a_5(t+1) &= \Theta_5 + \xi_5(t)\big(- I_0(t) + I_1(t) - I_3(t)\big) && (6.5) \\
O^+(t) &= O_4(t) + O_5(t) && (6.6) \\
&= \tau(a_4(t)) + \tau(a_5(t)) \\
&= \tau\left(\Theta_4 + \xi_4(t)\big(+ I_0(t) + I_1(t) + I_3(t) + I_2(t)\big)\right) + \\
&\quad \tau\left(\Theta_5 + \xi_5(t)\big(- I_0(t) + I_1(t) - I_3(t)\big)\right) \\
\Theta_4 &= -0.029 \\
\Theta_5 &= +0.056.
\end{aligned}$$

The effect of the small bias values is understood if the solution for O^+ is analysed (see eq. 6.6). It is rewritten in the following form:

$$\begin{aligned}
\Sigma_4(t) &:= +I_0(t) + I_1(t) + I_3(t) + I_2(t) \\
\Sigma_5(t) &:= -I_0(t) + I_1(t) - I_3(t) \\
\Longrightarrow O^+ &= \tau\left(\Theta_4 + \xi_4(t)\Sigma_4(t)\right) + \tau\left(\Theta_5 + \xi_5(t)\Sigma_5(t)\right). && (6.7)
\end{aligned}$$

Two cases are compared with respect to the consequences of O^+, bias values equal and unequal to zero.

First, it is assumed that both bias values are zero $\Theta_4 = \Theta_5 = 0$. In this case the cart force is zero $O^+ = 0$ if:

$$\begin{aligned}
&\Theta_4 = \Theta_5 = 0 \\
&\Rightarrow O^+(t) = 0 \quad \Leftrightarrow \quad \begin{cases} \forall i \in \mathcal{I} : I_i(t) = 0 \\ \xi_4(t)\Sigma_4(t) = -\xi_5(t)\Sigma_5(t) \end{cases}. && (6.8)
\end{aligned}$$

It follows from equation 6.8 that there are two possible solutions for $O^+(t) = 0$ under the assumption that the bias values are zero:

$$\forall i \in \mathcal{I} : I_i(t) = 0 \qquad (6.9)$$

$$\xi_4(t)\Sigma_4(t) \quad = \quad -\xi_5(t)\Sigma_5(t). \tag{6.10}$$

The first case (see eq. 6.9), stating that all inputs are zero at the same time (see eq. 6.9), is critical. If this case occurs, it means that the pole angle is zero ($I_1 = 0$) and the pole angular velocity is zero ($I_3 = 0$), and the cart remains with with no velocity at the centre of the track ($I_0 = 0$, $I_2 = 0$). This is an unstable fixed point of the system. If it is not otherwise disturbed, the system will remain unchanged in this case, leading to diverging receptor strength, as at each time step, the increment of the receptor is maximal (see eq. 6.3). The changes in the synaptic strength do not influence the activation of the neuron, because the input is zero, and hence, the increment of the activation is zero too. In a technical application this leads to numerical limits, so that this case leads to the abortion of the evaluation (see above).

The second case (see eq. 6.10) follows from the rewritten solution for the output O^+ (see eq. 6.7), and is not critical. At least one input is not equal to zero, therefore, there is sensory input which influences the internal states of the neural network and also its output and, consequently, its behaviour. Hence, this condition is not stable in the same sense as discussed for the first case, and cannot lead to diverging receptors.

Next, the same considerations are drawn for the case in which the bias values are not zero:

$$\Theta_4, \Theta_5 \neq 0$$

$$\forall i \in \mathcal{I} : I_i(t) = 0 \quad \Rightarrow \quad O^+(t) \neq 0 \tag{6.11}$$

$$O^+(t) = 0 \quad \Leftrightarrow \quad \Theta_4 + \xi_4(t)\Sigma_4(t) = -\Theta_5 - \xi_5(t)\Sigma_5(t). \tag{6.12}$$

The first case (see eq. 6.11) is not critical. Although all inputs are zero, the output is not (see eq. 6.11). At least the cart position and cart velocity inputs are altered as a result and will not remain zero. This also holds for the second case, in which an output equal to zero (see eq. 6.12) is the result of non-zero inputs, which means that the system is externally-driven and will, therefore, not remain in the critical state. In conclusion, the bias values are small enough to not contribute to the overall dynamics of the system, but large enough to avoid a diverging behaviour of the output neurons. Next, the observed cases are discussed in relation to the equations given by the neural network.

The activation equations of the agonist and antagonist, as well as the solution for O^+ (see eq. (6.4) to (6.6)) reveals when the switching behaviour occurs namely at two distinct points, given by:

$$O_4 : \quad \xi_4(t)\Sigma_4(t) = \Theta_4$$
$$O_5 : \quad \xi_5(t)\Sigma_5(t) = \Theta_5.$$

Close to these two points, the error term $g(t)$ of the corresponding output neuron is maximal (see eq. 6.3). Hence, the growth of the synapse is maximal, leading to the observable overshooting behaviour (period-2 oscillation) of the controller. This holds, until the mean value of the output neurons approach the target values, so that the synapses converge towards their asymptotically stable values. This is the explanation of the second case described above, the switching of the output neurons, followed by a period-2 oscillation.

In the third case, due to the different initial conditions, the synapse strength grows large sooner, such that the oscillation occurs before the switching of the output neurons.

Discussion

This section presented a controller for the cart-pole benchmark problem with four inputs. The experiment is well-suited to demonstrate how the diverging behaviour of the SRN model for an input approaching zero, which is a desirable feature in the obstacle-avoidance task (see sec. 5.2), but undesirable in other cases, can be challenged if required. In the cart-pole experiment, the controller has to produce a zero output for inputs close to zero, which is precisely the setting at which divergence of the receptor occurs for the single neuron (see chap. 4). This was achieved by implementing a neural network analogy of the agonist and antagonists configuration found in biological systems (Klinke & Silbernagl, 2005). The evolved controller is a pure feed-forward structure, but it shows dynamical features, such as period-2 oscillations and quasi-periodic transients, as a result of the sensori-motor loop. The SRN neural network sufficiently solves the task for the required evolutionary setting, i.e. it was able to balance the pole, with minimal energy consumption after the given warm-up steps, for the maximal evaluation time. The balancing behaviour was described by the structure and the equations given by the neural network.

The experiment also shows that the diverging behaviour of the output neuron for inputs converging to zero cannot be compensated for completely but can be minimised with the agonist/antagonists configuration. There is a period of time for which the controller oscillates when the inputs sum up to zero. This underlines that it is not possible to achieve zero output for zero input with the SRN model. The question posed at the beginning of this section was if this property limits the application of the SRN model.

First, similar solutions can also be found in the literature for neuro-controller with static synapses (Pasemann, 1997a). Second, the principle of rapid rhythmic inputs driving slow effectors is found in biology (Morris & Hooper, 1998) and also used and discussed in the context of robotics (Hülse, 2007), where high oscillatory motor signals are filtered by the low-pass filter properties of the actuators. The task was sufficiently solved with the SRN model and showed oscillatory behaviour only for a subset of initial parameters and, otherwise, only for a short period of time. It is, therefore, concluded that this property does not limit the SRN model for possible applications.

The controller was evolved with fixed plasticity parameters, which were taken from the previous chapters. The next section discusses an evolution of an adaptive controller with variation of the plasticity parameters.

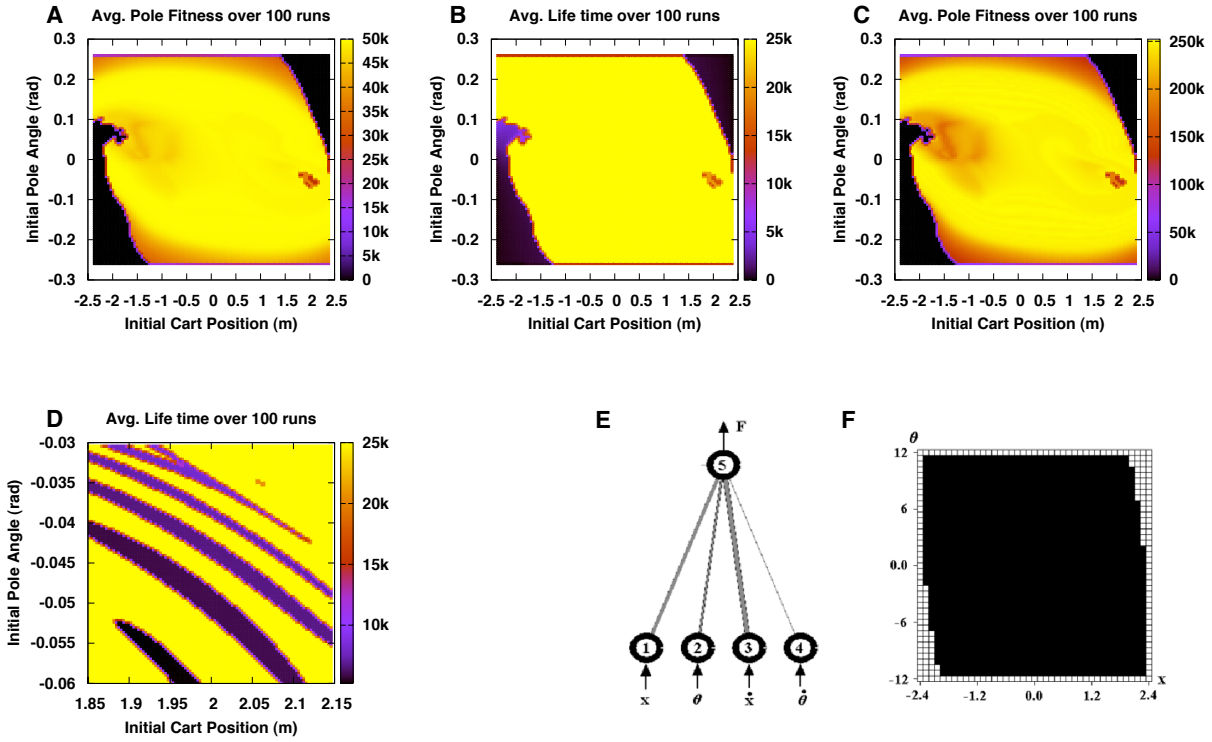


Figure 6.2: Best evolved controller and comparison with a static solution. A-E) Initial conditions plotted against A) Fitness-function defined by Pasemann and Dieckmann (1997b), B) Life time C) Fitness-function given in the text (see eq. 6.2). The plot D is an enlargement of the irregularity by a clipping of the plot B. Plot E shows the neural network by Pasemann and Dieckmann (1997b), plot F the corresponding phase-space. The latter two plots are taken from Pasemann and Dieckmann (1997b). For each of the plots A-C, the initial cart position and the initial pole angle were varied over 100 equally distributed values over the valid intervals (for plot D over the reduced intervals). For each setting, 100 randomly initialised instances of the controller were evaluated. The given fitness value at each coordinate is the average of 100 runs. The fitness phase-space reveals an irregularity in the fitness values for a certain set of initial conditions (enlarged in plot D). This phenomenon is discussed in the text. Figure F shows black for a successful balancing of the pole for more than 120 seconds while white encodes a balancing of less than 30 seconds. The resolution of the plot is 40 by 40. Plot F is best compared with plot B, although the resolutions differ significantly.

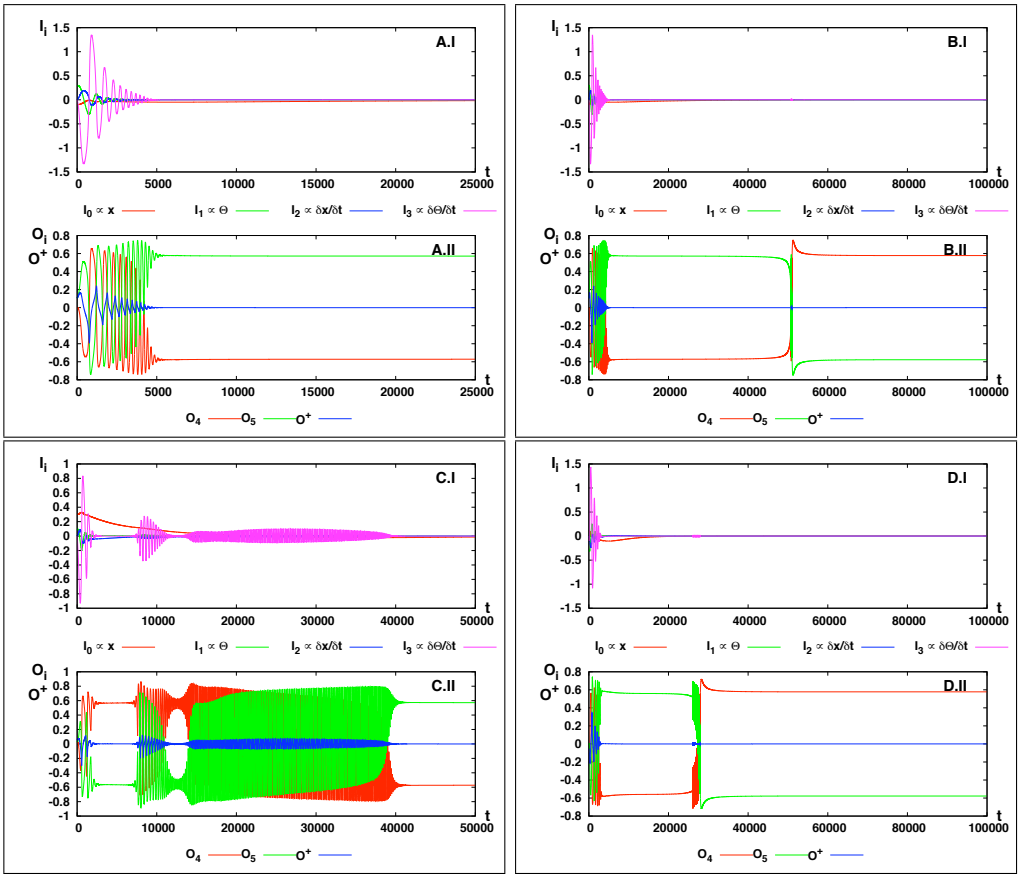


Figure 6.3: Cart-pole transients for different initial conditions. Four cases of different behaviour of the same controller with varying initial conditions and evaluation time. A) Initial condition $x(0) = -0.24, \Theta(0) \approx 0.079$ and evaluation time equal to the evolutionary set-up (25,000 iterations), B) Same initial condition as A but with increased evaluation time (100,000), C) Initial condition $x(0) = 0.72, \Theta(0) \approx 0.026$, D) Initial condition $x(0) = 0.24, \Theta(0) \approx -0.079$. All plots were generated with $a_i(0) = o_i(0) = 0, \xi_i(0) = \eta_i(0) = \varepsilon, \forall i \in \mathcal{N}$.

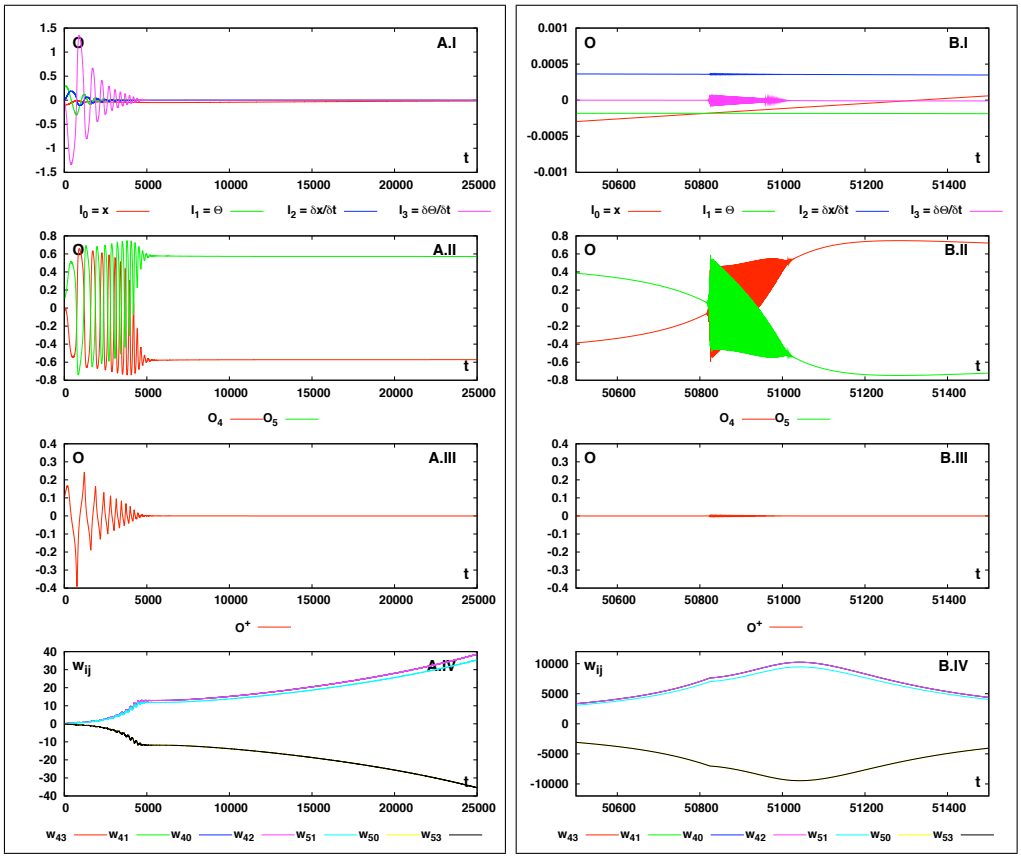


Figure 6.4: Cart-pole transients for the conditions A and B. Detailed plot of the first two cases A and B. A) Plot for the conditions present during evolution. The plot shows that after the warm-up phase of 5,000 iterations, the cart is centred and the pole is balanced (A.I), the output neurons have reached their asymptotically stable values (A.II), and that the summed output O^+ is close to zero (A.III). But the plot of the synaptic strength (A.IV) clearly indicates a diverging behaviour (for discussion see text). The result of the diverging behaviour is shown on the right-hand side (B.I and B.IV), with the same initial conditions, but increased evaluation time, compared to the evolution setting. It is shown that as soon as the cart position input (I_0), and the pole angle input (I_1) approach zero (B.I), the output neurons converge towards zero (B.II) followed by a period-2 oscillation and finally a convergence towards their asymptotically stable values, but in reversed order. This is the effect of the large synaptic strength (B.IV). For a detailed discussion, see text.

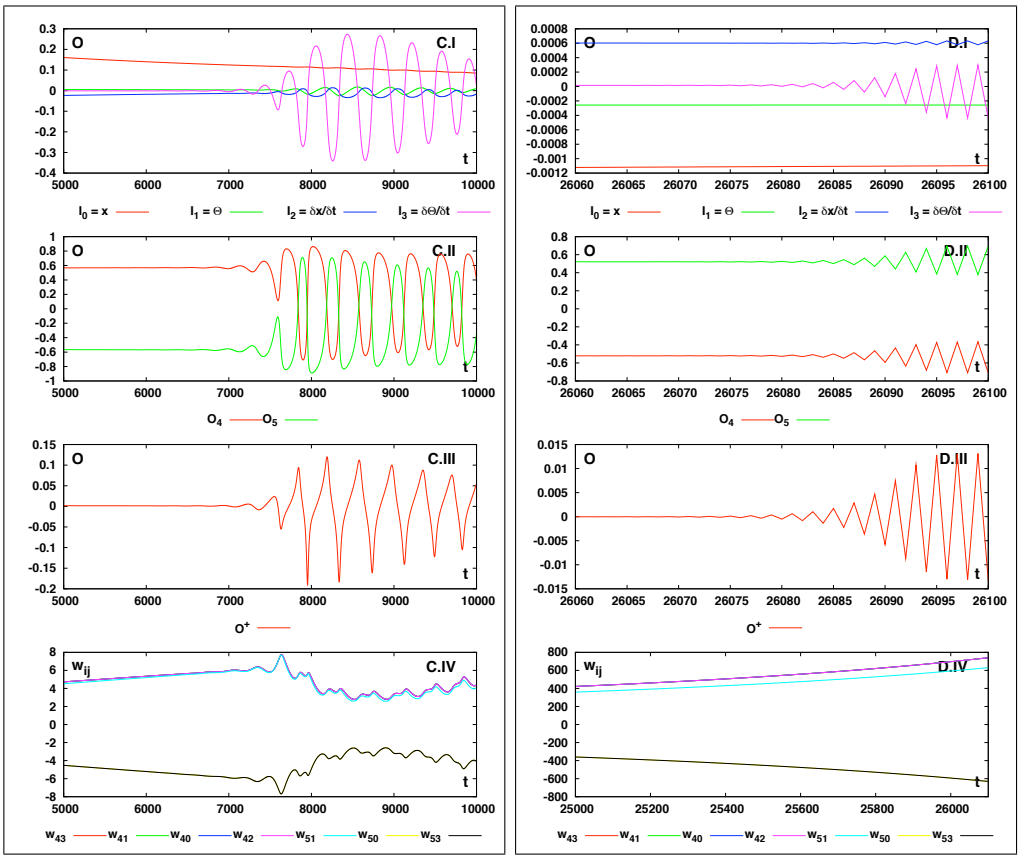


Figure 6.5: Cart-pole transients for the conditions C and D. Detailed plot of the two cases C and D. C) The output neurons, and hence, the summed output O^+ show a quasi-periodic behaviour (C.II & C.III), which is the result of the comparably small synaptic strength (C.IV). D) The synaptic strength of the controller is large compared to the other cases, such that small variations of the inputs (D.I) lead to an overshooting behaviour of the output neurons (D.II, D.III). For a detailed discussion, see text.

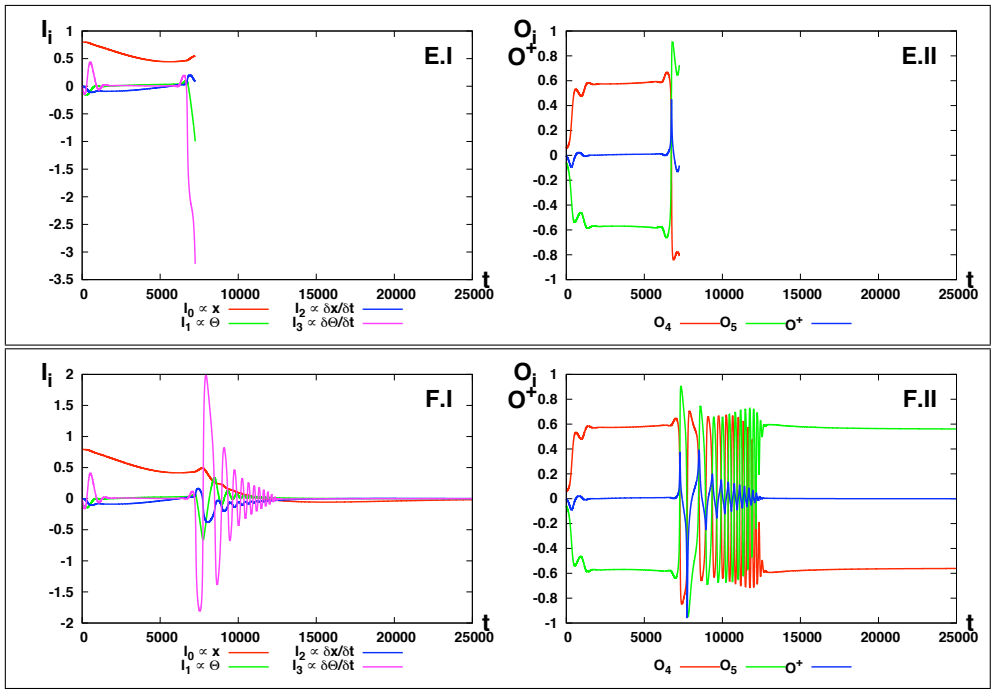


Figure 6.6: Cart-pole transients for the conditions E and F. Comparison between two initial conditions, one within the irregularity region of the phase-space (E), the other closely-related to it (F). The initial conditions are E: $x(0) = 1.919, \Theta(0) = -0.0342$, and F: $x(0) = 1.9, \Theta(0) = 0.03$. A detailed plot of the two initial conditions is shown in the figure 6.7

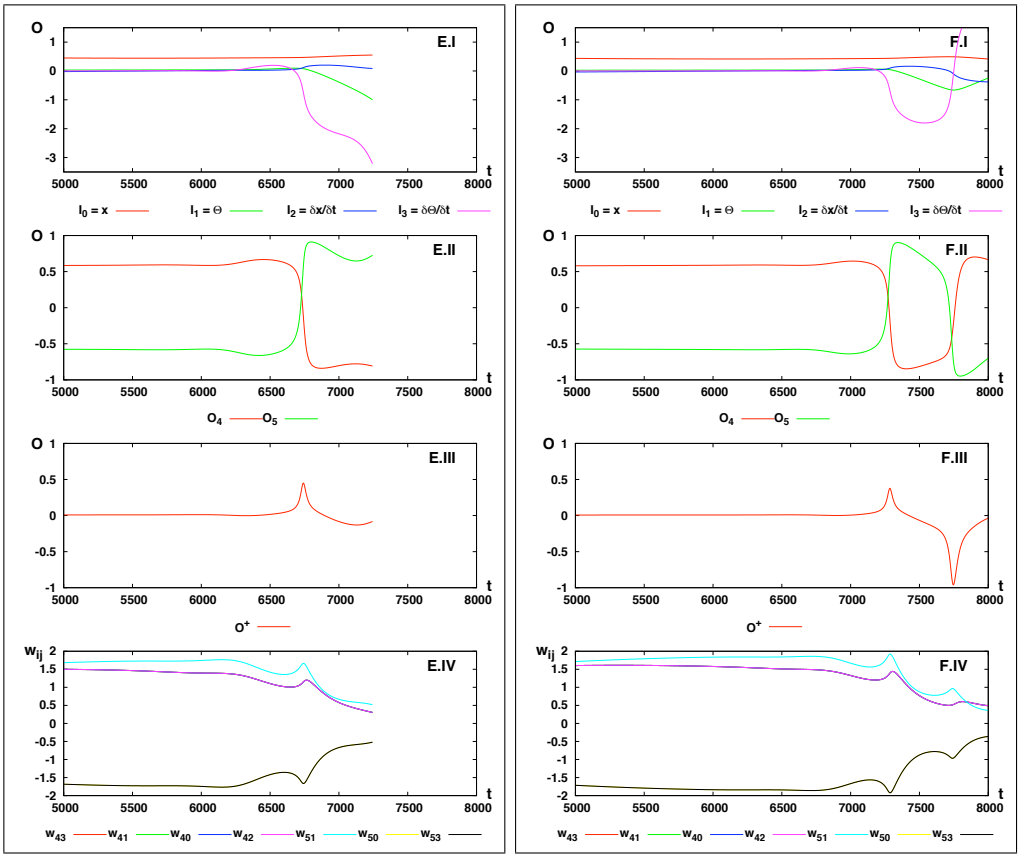


Figure 6.7: Detailed cart-pole transients for the conditions E and F. Detailed plot of the two cases E and F. Only the synaptic strength (compare E.IV and F.IV) differ. The comparison indicates that there is a minimal required strengths which has to be fulfilled in order to balance the pole. For a detailed discussion, see text.

6.2 SRN Adaptive Light-Seeker with Ambient Light

The previous section discussed an experiment in which an input of zero must be regulated towards an output of zero. It was shown that this cannot be achieved in total, but that the solution solves the task well in comparison to solutions with the standard additive neuron model. In this section, a controller is evolved which shows the adaptation properties of the SRN model.

In the light-seeker experiment, first described by Grey Walter in 1950 (Walter, 1950, 1951), an autonomous agent, equipped with light sensitive sensors, has to find a light source. The analogy often drawn to biology is that the light source is a food reservoir and the light sensitive sensors are sensors that enable the agent to detect this reservoir. Approaching the light is then interpreted as finding food and recharging. Nearly all experiments within this domain have been conducted within a static ambient light environment, often in the dark, which means that if the agent is not heading towards the designated light source, the sensors do not sense any light. A sensed light intensity, therefore, always relates to a light source. A systematic analysis of the light-seeking behaviour in the context of fusion and expansion techniques in the field of evolutionary robotics and recurrent neural networks is given by Hülse (2007).

In this experiment, ambient light is included in the environmental set-up. Ambient light means that there is no specific light source, but that, sensors read non-zero light intensity values, independently of the position or heading of the robot, when no light source is detected. The robot is a variation of the Braitenberg vehicle described in the previous chapter, with two proximity sensors for negative tropism (obstacle-avoidance) and two additional sensors for positive tropism (light-seeking).

The experiment is divided into two parts. First, a light-seeker is evolved, then in the next step, a varying ambient light is included in a simpler environmental set-up. Ambient light is only varied from generation to generation, to ensure comparability of the fitness values within a population.

This section begins with a detailed description of the environmental and experimental set-ups, and follows with a discussion of the results.

6.2.1 Experimental Set-up

The Khepera Simulator of the previous chapter was not used in this experiment for two reasons. First, in the Khepera Simulator, a light source is not blocked by a wall. This means, that if there is a wall between the light source and the robot, the robot will detect it with the same intensity as if there was no wall present. Second, and more importantly, The Khepera Simulator does not allow randomised placement of the objects within the environment.

Therefore, based on a swarm simulation of the ASM II robot (I Support Learning, 2007), a new simulation was written in YARS (app. A: ISEE). The initial YARS description file was provided by Steffen Wischmann[†] for swarm experiments (see fig. 6.9). YARS is based on the open-source physics engine ODE (Smith, 2005), which is likely to show unstable behaviour, when the proportions of the simulated rigid bodies are very small. Examples of properties are the mass and dimension of a body. To increase the stability of the simulation, but the ASM Robot, and not the Khepera robot, was chosen for simulation.

[†]No reference available.

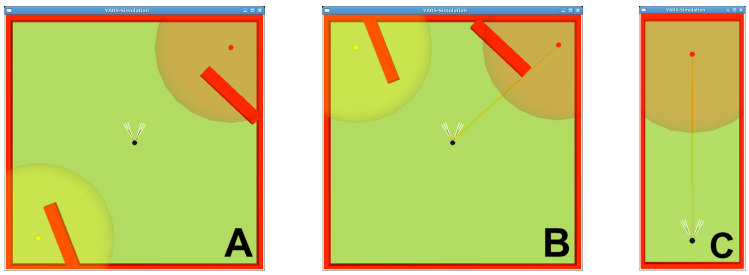


Figure 6.8: SRN Light-Seeker Environments. A: Environment with randomised placement of objects and light sources. B: Same environment, different random positioning. C: Environment used for the evolution of the SRN Adaptive Light-Seeker. The circles indicate the distance to the light sources where the measured light has 10% of its maximal intensity.

Environment

Initially a more complex and randomised environment was chosen (see fig. 6.8A), with the following characteristics:

1. The light source is not visible from the initial position of the robot.
2. Obstacles are replaced randomly for each new generation.
3. Light sources are replaced randomly for each new generation.

The first property ensures that individuals are only rewarded when they actively search for the light source. The second and third properties ensure that the solution strategy cannot be a specific behavioural pattern, such as e.g. searching the environment in left or right turns with increasing diameter. As the light source and obstacle positions vary, this sort of behaviour leads to a low fitness.

For simple light-seeking and obstacle-avoidance tasks, this environment led to good results in a small amount of time (about 5-6 generations), when the evolution is started with the Braitenberg vehicle from the previous chapter as an initial structure. However, it was not well-suited for evolution with varying ambient light, for the following reason. In order to avoid specialisation, each individual was evaluated 15 times, and the sum of the fitness values was used for the selection operator. In this comparably large environment, up to 5,000 cycles are required for the robot to detect the light source without any ambient light. This number must be increased significantly if ambient light is included in the setting. The evaluation time took too long, thus a simpler environment (see fig. 6.8 B), capturing the main characteristics of the more complex environment had to be designed. In this second, simpler environment, the evaluation time required for an individual to find the light source, even when ambient light, is reduced to approximately 500 cycles. The light source is randomly placed in a certain interval and is not detectable from the start position of the robot. The circle in the figure indicates where the measured light intensity of the light source is 10% of the maximal intensity (see fig. 6.8 A/B/C and fig. 6.9, right hand side). The sensors have a Gaussian-distributed noise of 5%, so that the

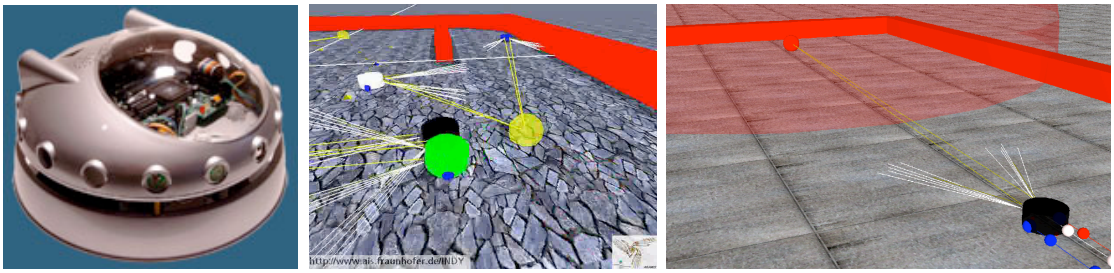


Figure 6.9: ASM-II Robot and YARS Simulation. Left: The ASM II robot, image taken from official homepage (I Support Learning, 2007), Centre: The swarm robot simulation written by Steffen Wischmann. Screen shot taken from the video available at (Relais d'information sur les science la cognitin, 2007). Right: SRN Light-Seeker with ambient light simulator.

circle indicates the maximal distance from which the light source is clearly distinguishable from the noise.

For the light-seeking behaviour, each light source had a maximal light intensity of 1, and the ambient light was varied between 0.5 and 1. These values were chosen to avoid evolving the controller in the usual dark environment in which a sensed light intensity is automatically related to a light source. The measured light intensity l_i for each light sensor is calculated in YARS using the following equation:

$$\begin{aligned} l_i &= \sum_{j=0}^{L-1} \frac{L_j}{1+r_j^2}\sqrt{\alpha_j} \\ \alpha_j &= \begin{cases} 0 & |\omega_j| > \omega_{max} \\ \frac{|\omega_j|}{\omega_{max}} & |\omega_j| \leqslant \omega_{max} \end{cases} , \end{aligned} \tag{6.13}$$

where L is the number of light sources in the environment, L_j is the maximal intensity of the j-th light source, r_j is the distance of the j-th light source to the sensor, ω_j is the relative angle between the position of the light source and the light sensor in relative coordinates of the robot, and ω_{max}, the maximal relative angle for which the light sensor still detects the light.

For a description of the implementation, see table 6.3 and figure 6.8 C.

6.2.2 Evolutionary Set-up

This subsection begins with the set-up of the controller, followed by the discussion of the fitness-function.

Controller

The robot has two proximity and two light intensity sensors, each represented by one input neuron. Distance and light intensity are mapped such that when no obstacle is present or no light is detected, the corresponding input value is -1. For the minimal possible distance of an obstacle with respect to the proximity sensor, and the maximal measured light intensity with

	Radius	Mass	Fric. coeff. μ_x	Fric. coeff. μ_y	Torque	Max. velocity
Body	10 cm	0.95 kg	0	0	–	–
Differential wheels	6 cm	0.05 kg	15	0	0.5 N	10 rad/s
Supporting wheels	0.1 cm	1.0 kg	0	0	0	0

Table 6.3: PHYSICAL PROPERTIES OF THE SIMULATED ASM-II ROBOT. The 1 kg value for the weight of the supporting wheels is required by the physics engine to avoid numerical instabilities. This was compensated for increasing the torque and the maximal velocity of the differential wheels until the behaviour of the simulated and physical robot were comparable. The friction coefficients are given with respect to the physics engine ODE, which is used in YARS. A description of the parameters can be found in the ODE documentation (Smith, 2005).

respect to the sensor range of the light sensor, the corresponding values are +1. In YARS, a measured light intensity value is the sum of the distance dependent light intensities of all light sources in range of the sensor, added to the ambient light. Let $l_a \in [0.5, 1]$ denote the ambient light value, L be the number of light sources in the environment, and l_i, the light value of the i-th light source with respect to the current position and orientation of the current robot (see eq. 6.13), then the measured light values at the sensor l_j is given by

$$\begin{aligned} l_i &\in [0,1] \\ l_j &= l_a + \sum_{i=0}^{L-1} l_i \end{aligned}$$

Let $x_l, x_r \in \mathbb{R}$ denote the value returned by the left and right distance sensor, and $l_l, l_r \in \mathbb{R}$, the value returned by the left and right light intensity sensor, respectively, then the pre-processing is given by:

$$\begin{aligned} I_{0,1} &= 2\frac{x_{l,r}}{\max\{x_{l,r}\}} - 1 \\ \Rightarrow I_{0,1} &\in [-1,1], \\ I_{2,3} &= 2\frac{l_{l,r}}{\max\{l_{l,r}\}} - 1. \end{aligned}$$

The additive light intensity is not a realistic model of a physical LDR (Light Dependent Resistor), which has a non-linear activation curve and reaches saturation for light intensities over a certain (manufacturer dependent) threshold. Nevertheless, this model was chosen because of its simplicity and its non-linear characteristic which is comparable to a logarithmic function, and does not increase the difficulty of the problem.

The light sources were placed at large distances to each other in the more complex first environment. This ensured that only one light source at a time could be detected by the robot. It then follows that

$$l_a \in [0.5, 1] \text{ and } l_i \in [0, 1] \quad \Rightarrow \quad l_{l,r} \in [0.5, 2]$$

$$\Rightarrow I_{2,3} \quad \in \quad [-1,1].$$

As the goal of the robot in the light-seeking task is to find the light source and stop in front of it, the agonist/antagonist configuration of the previous experiment is chosen. The overall output is the sum of the corresponding output neurons. In contrast to the cart pole, the maximal speed of each wheel is limited. The summed output is mapped linearly to the wheel velocity interval. As the maximal wheel velocity is naturally limited by the system, a summed absolute output value, larger than one, is cut off by the physics of the motors. Let M_l and M_r denote the motor command for the left and right motor, and O_l^+, O_r^+, the output of the two left and right motor neurons (in the following abbreviated with $M_{l,r}$ and $O_{l,r}$, see also figure 6.10), then the post-processing is given by

$$\begin{aligned} O_l^+ &= O_4 + O_6 \\ O_r^+ &= O_5 + O_7 \\ \Rightarrow O_{l,r}^+ &\in [-2,2] \\ M_{l,r} &= \left(O_{l,r}^+(\max\{M_{l,r}\} - \min\{M_{l,r}\}) + \min\{M_{l,r}\}\right)\Big]_{\min\{M_{l,r}\}}^{\max\{M_{l,r}\}}. \end{aligned}$$

In contrast to the previous section, the plasticity parameters were exposed to evolution in the following manner. One set of plasticity parameters was open for variation, and for each neuron the parameters were set according to this set. Let $\alpha_v, \beta_v, \gamma_v$ denote the set of varied plasticity parameters, then:

$$\forall i \in \mathcal{N} : \{\alpha_i, \beta_i, \gamma_i\} \quad = \quad \{\alpha_v, \beta_v, \gamma_v\}.$$

Fitness-Function

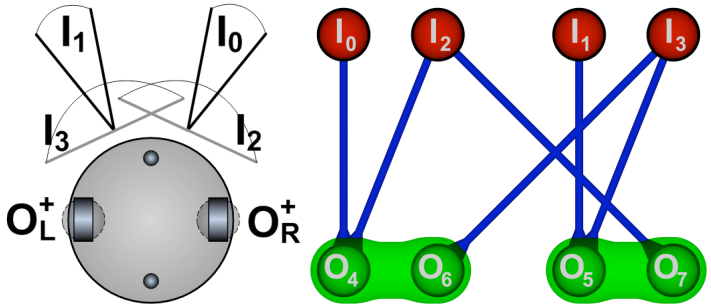
As described above, the environment itself punishes undesired static solution strategies, such as circular search. Therefore, the fitness-function is kept very simple. It is the summed values of the light sensors. For convenience, the ambient light is subtracted from the light sensor's values in the fitness-function (not in the pre-processing of the controller), therefore, it is not part of the fitness. Let T denote the maximal evaluation time, then the fitness-function reads:

$$F := \sum_{t=0}^{T-1} \big((l_l(t) - l_a(t)) + (l_r(t) - l_a(t))\big) \tag{6.14}$$

$$= \sum_{t=0}^{T-1} (l_l(t) + l_r(t) - 2l_a(t)). \tag{6.15}$$

Evaluation abortion criteria

Two abortion criteria are introduced, in which the evaluation time is terminated and the next try or individual is evaluated. The first criterion corresponds to a failure of the obstacle-avoidance task (negative tropisms). The evaluation is aborted, if the robot collides with an obstacle. The light sources are not considered as obstacles, i.e. the robot may pass through them.



	Pre-Processing	Range
IR Proximity sensor, right	$I_0 = 2\frac{x_r}{\max\{x_r\}} - 1$	$[-1.1]$
IR Proximity sensor, left	$I_1 = 2\frac{x_l}{\max\{x_l\}} - 1$	$[-1.1]$
Light intensity sensor, right	$I_2 = \frac{l_r}{\max\{l_r\}} - 1$	$[-1.1]$
Light intensity sensor, left	$I_3 = \frac{l_l}{\max\{l_l\}} - 1$	$[-1.1]$

Figure 6.10: SRN Adaptive Light-Seeker. Upper left figure: Schematics of the simulated ASM-II robot. Note that the ordering of the sensors is reversed compared to the Khepera Simulator of the previous chapter. The original ASM-II was modified by adding two Sharp infra-red proximity sensors (Sharp GP2D12). Upper right figure: The evolved controller with the best performance in minimal structural size. It is symmetric, strictly feed-forward, has only one layer, and uses only inhibitory connections. Bottom: Table of the input neurons and their pre-processing.

The second abortion criterion results from the fact that the YARS simulation, due to the underlying physics engine, is much slower in comparison to the Khepera 2.0 Simulator (Michel, 2005), which was used in the previous chapter. In order to decrease the evolution time, and in response to a failure of the exploration and light-seeking behaviour (positive tropisms), the evaluation is aborted if the robot has not moved a specific distance from its initial position after a given number of time steps. The distance and the number of time steps are parameters which are open to modification by the experimenter during runtime. For this evolution, a minimal distance of one meter had to be travelled in less than 50 evaluation cycles (five seconds).

6.2.3 Results

The neuro-controller which is presented here is minimal with respect to its structure[†] and performed best with respect to the given fitness-function (see eq. 6.15). The controller is a pure symmetric feed-forward structure (see fig. 6.10). In the figure, the ordering of the input neurons is changed to underline the symmetry of the neural network.

[†]Evolution produced intermediate solutions of higher structural complexity. One such preceding result of evolution with comparable fitness but higher structural complexity is shown in the figure 3.2.

Observed Behaviour

The controller solves the task for the range of ambient light values which were present during the evolution ($l_a \in [0.5, 1]$). It also solves the task for ambient light values which are slightly higher (e.g. 1.1, see fig. 6.15), but not for values smaller than 0.5 (especially not without ambient light $l_a = 0$, see fig. 6.11). For values between 0.5 and approximately 1.1 the behaviour of the controller varies (see figures 6.11 to 6.15). For smaller ambient light values, the robot's trajectory is straight and the velocity is higher. For higher values, the trajectory indicates a chaotic movement and a slower overall velocity, i.e. the robot requires significantly more time to reach the light source. In all cases, the robot approaches the light source and slows down to almost zero velocity. When the evaluation time is significantly increased from the evaluation time during the evolution, the robot will pass the light source, and continue with obstacle avoidance until the light source is perceived again. The explanation for this behaviour is given in the next section.

Analysis of the transients of the observed behaviour

A first understanding of the behaviour of the controller is derived from the transients of the system while the agent acts in the sensori-motor loop. The transients give good indications, which are then related to the equations of the neural network in a second step. As the transients are very noisy (explanation follows below), and therefore, difficult to read, a smoothing method was chosen. For each transient plot (see figures 6.11 to 6.15) the upper two figures show the raw data for the trajectory shown of the left-hand side, while the lower two figures show the same transients, but smoothed with Bezier spline curves (Encarnação et al., 1996) using the *smooth bezier* method provided by gnuplot (T. Williams & Kelley, 2007). The method is implemented such that a Bezier curve is approximated with connected endpoints. The degree of the Bezier curve is given by the number of data points.

There is a high amount of noise on the output neurons in all plots where the ambient light lies within the trained range ($l_a > 0.5$). This is the result of the noise which is present at the input neurons. For each sensor, a Gaussian-distributed noise with a standard deviation of 5% is added to the sensor value. If the light intensity input neurons I_2 and I_3 are not close to zero, the noise is amplified by the strength of the two synapses w_{63} and w_{72}, whose absolute value is larger than one. The absolute strength of the other incident synapses are smaller than one, but they also contribute to the noisy output. If at least one of the light intensity input neurons is close to zero, the noise is amplified by the diverging behaviour of the SRN model. The diverging response of an input-output neuro-module, without recurrent connection to an input crossing zero, was previously discussed (see. chapters and sections 4, 5 and 6.1). In this setting, the output neurons and synapses do not diverge, for two reasons. First, the robot acts in the sensori-motor loop. This was previously discussed for the Braitenberg vehicle (see sec. 5). The oscillations are part of the behaviour of the robot, and alter the sensory input so that the input value of zero will not remain for a long enough duration of time. In the case of an ambient light smaller than one ($l_a \in [0.5, 1[$), the case of a light sensor input close to zero occurs only when the light source is detected. The robot is, therefore, approaching the light source and hence, the light intensity neurons' values I_2 or I_3 are further increasing. The second reason is independent of the sensori-motor loop, and is explained for an ambient light value of one ($l_a = 1$, see fig.

6.13). Here, a noisy input close to zero is constantly present at the sensors. This does not lead to a divergence of the synaptic strength, because the process of incrementing and decrementing occur equally often with, on average, the same strength. Therefore, they cancel each other (see fig. 6.16). The strength of the synapse is anti-proportional to the size of the noise interval around zero, as the error-dependent term of the receptor returns larger increments for smaller values of the activation (see eq. 4.8). Hence, the synapses grow faster and larger when values closer to zero are presented.

The transient plots (see figures 6.11 to 6.15) show that as soon as the light source is detected by the sensors, the absolute value of the synaptic connections w_{63} and w_{72} grow, leading to a decrease of the output of the output-neurons O_6 and O_7. The robot slows down. When the measured light intensity is large enough, the output neurons O_6 and O_7 regulate towards the lower target, and, as a result, the summed output O_L^+ and O_R^+ is close to zero. This explains the slowing down and stopping of the robot when a light is detected.

As O_6 and O_7 are close to their target values, the synaptic connections w_{63} and w_{72} decrease in amplitude, decreasing the influence of the light sensors. Hence, the absolute value of the activity of the output neurons O_6 and O_7 is smaller than the target value, resulting in summed outputs O_L^+ and O_R^+ which remain small but positive. This explains why the robot slowly approaches the light source, but passes it, if no obstacle is present.

In the case of an ambient light of zero (or smaller than 0.5), the pre-processing does not push the output neurons O_6 and O_7 towards the lower target value. This is discussed at the end of the next section, which formally, describes the observations.

Analysis of observed behaviour

To understand how the behaviour described above is achieved by the controller, the transients (see figures 6.11 to 6.15) must be analysed in relation to the structure of the neuro-controller. The dynamics are given by the following set of equations:

$$i \in \mathcal{O}, \qquad j \in \mathcal{I} \tag{6.16}$$

$$\mathcal{C} = \begin{pmatrix} -1 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \tag{6.17}$$

$$\beta = 0.279 \tag{6.18}$$

$$\Theta_i = 0 \tag{6.19}$$

$$a_i(t+1) = -\xi_i(t) \sum_{j \in \mathcal{I}} c_{ij} I_j(t) \tag{6.20}$$

$$w_{ij}(t) = \xi_i(t) \tag{6.21}$$

$$\xi_i(t+1) = \xi_i(t)(1 + \beta(\tau(a^*)^2 - \tau(a_i(t))^2)) \tag{6.22}$$

$$O_L^+(t) = \tau(a_4(t)) + \tau(a_5(t)) \tag{6.23}$$

$$= \tau\big(-\xi_4(t)(I_0(t) + I_1(t))\big) + \tau(-\xi_5(t) I_1(t)) \tag{6.24}$$

$$O_R^+(t) = \tau(a_6(t)) + \tau(a_7(t)) \tag{6.25}$$

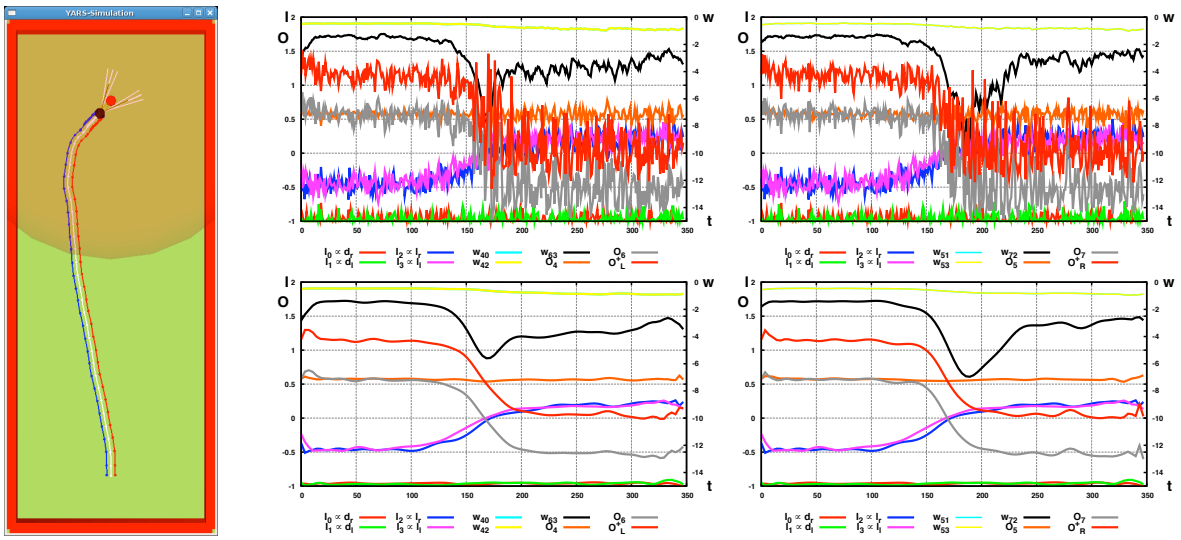


Figure 6.11: SRN Adaptive Light-Seeker with ambient light intensity of 0.5. Left: Trajectory; Right: Transients of the neuro-modules. The upper two transient plots show the raw transients of the two neuro-modules controlling the left and right wheel speed, respectively, the lower two are the Bezier splines for which the data points of the upper two plots are the points of the Bezier curve. This method is also used in subsequent transient plots. The noise on the output neurons is the result of the noise on the input neurons, amplified by the synaptic connections. The noise on the output neurons is increased, when the light intensity input neurons I_2 and I_3 approach zero. The behaviour of the robot can be described as follows: when no light is seen, it moves along an almost straight trajectory. As soon as the light source is detected, the robot approaches it. The initial turning angle corresponds to the random initialisation of the neural network, combined during the initial 500 iterations.

$$= \quad \tau\big(- \xi_6(t)(I_2(t) + I_3(t))\big) + \tau(-\xi_7(t)I_3(t)). \tag{6.26}$$

Four different cases will be described in the following paragraphs:

1. No light source detected.
2. Light source detected by the left sensor.
3. Light source detected by both sensors.
4. Maximal values detected by both sensors.

Due to the symmetry of the neural network, the two cases where the light source is detected by a single sensor, i.e. the right sensor and the left sensor, are analogous and are not discussed separately. To describe these cases in terms of the equations given above, a few definitions are required:

$$\begin{aligned} I_j^c &:= \text{constant value of input neuron } j \\ \Sigma_i^c &:= \sum_j c_{ij} I_j^c \quad \text{input to output neuron } i \end{aligned}$$

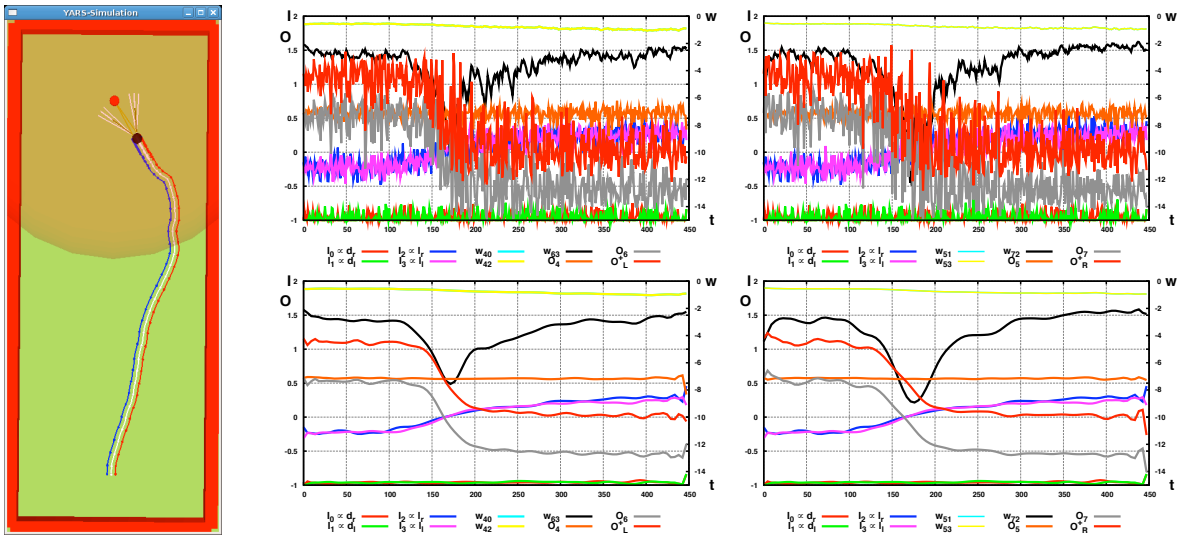


Figure 6.12: SRN Adaptive Light-Seeker with ambient light intensity of 0.75. Left: Trajectory, Right: Transients of the neuro-modules. The behaviour of the controller is in this case, comparable to the previous one (see fig. 6.11). The robot also locates and approaches the light source, but its trajectory is not as straight compared to the previous setting. For explanation, see text.

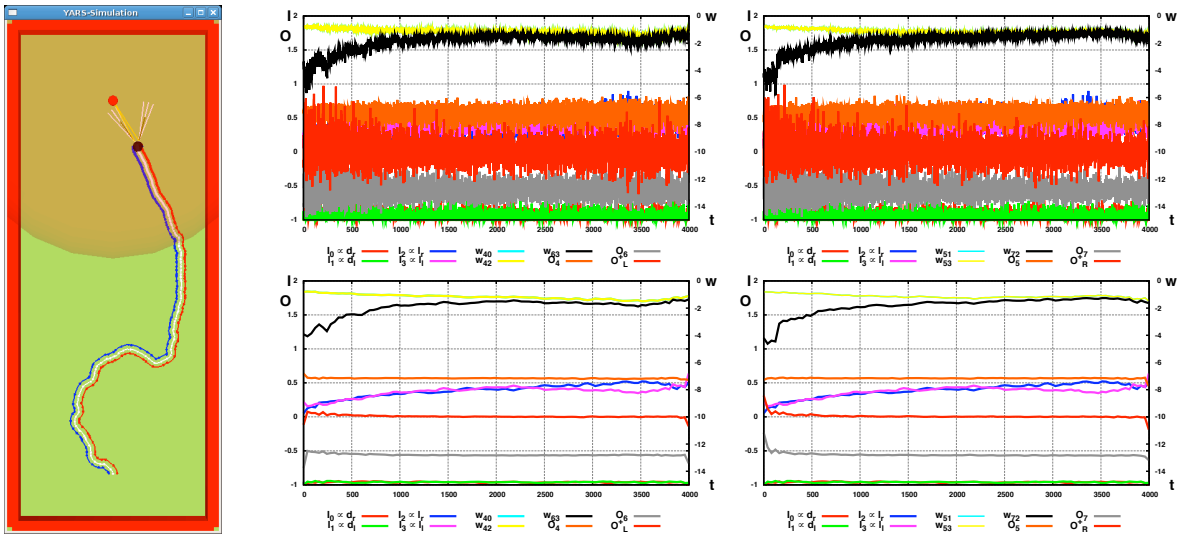


Figure 6.13: SRN Adaptive Light-Seeker with ambient light intensity of 1.0. Left: Trajectory, Right: Transients of the neuro-modules. The controller still finds the light source, but its trajectory resembles a chaotic or random search path. This results from the light intensity input neurons which are close to zero when no light source is detected and hence, lead to comparably large synapses. For explanation, see text.

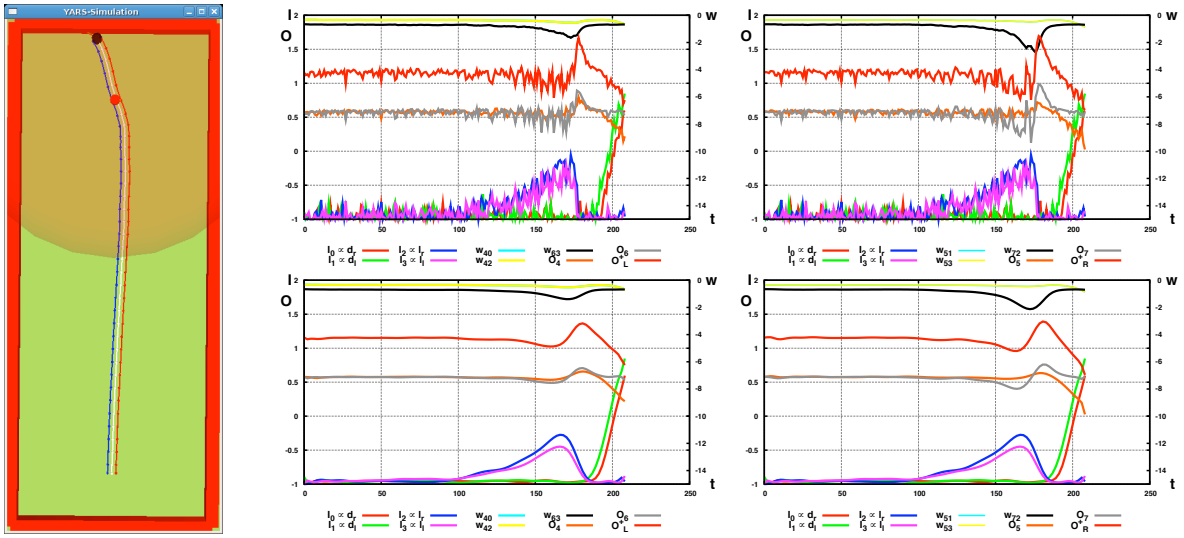


Figure 6.14: SRN Adaptive Light-Seeker with ambient light intensity of 0. Left: Trajectory, Right: Transients of the neuro-modules. The controller completely fails. Although it shows a minimal approaching behaviour, it can neither stop in front of the light source, nor avoid obstacles. For explanation, see text.

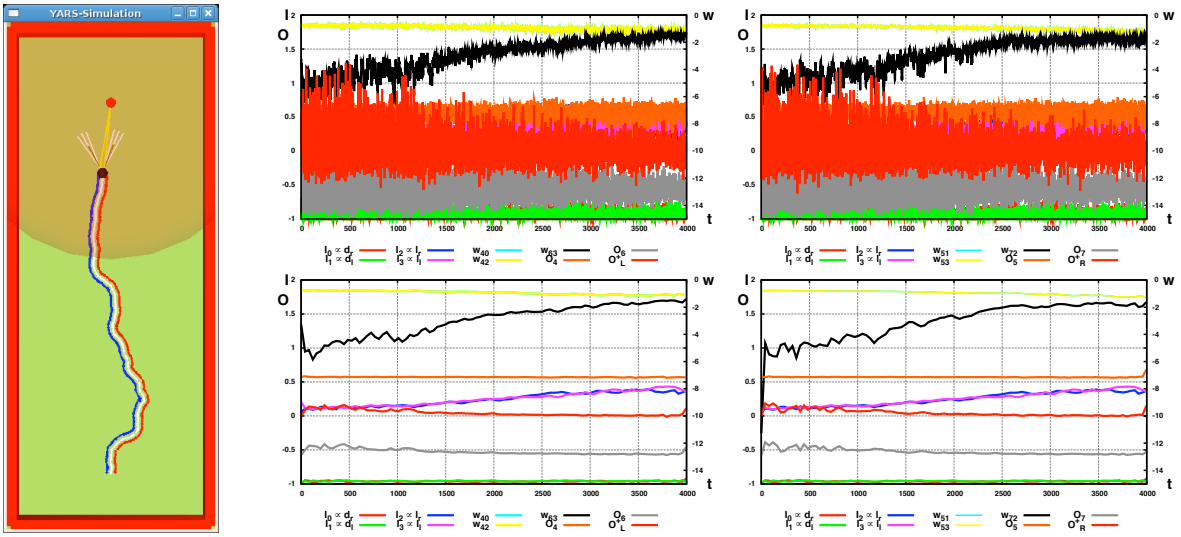


Figure 6.15: SRN Adaptive Light-Seeker with ambient light intensity of 1.1. Left: Trajectory, Right: Transients of the neuro-modules. The controller still finds the light source under this condition, but its trajectory resembles a chaotic or random search path. This results from the light input neurons, which are close to zero, if no light source is detected and, hence, result in comparably large and, therefore, noise-amplifying synapses. Compared to the other situations, in this condition, the robot keeps a constant distance to the light source. For explanation, see text.

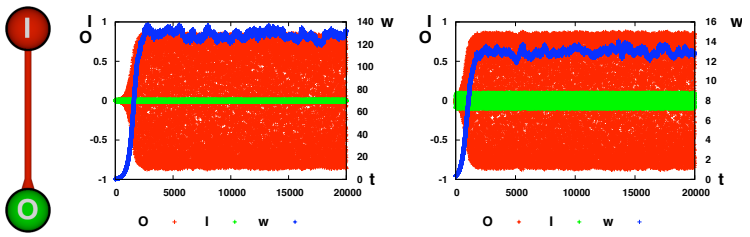


Figure 6.16: INPUT-OUTPUT MODULE WITH NOISY INPUT. Left: Visualisation of the input-output neuro-module without recurrent connection. Centre: Transients of the module for a noisy input with a maximal deviation of 0.01. Right: Transients of the module for a noisy input with maximal deviation of 0.1. The two transient plots show that the synaptic weights do not diverge although the input remains close to zero over a long period of time. The large values of the synaptic weights, which result from the input close to zero, lead to an amplification of the noise, which in turn leads to strong deviations of the output of the output neuron. However, the amplification is not large enough to cause the activation of the output neuron to reach technical limitations which would lead to an abortion of the evaluation during evolution (see text below).

$$
\begin{aligned}
a_i^* &= \xi_i^* \Sigma_i^c \\
\Rightarrow \xi^* &= \frac{a_i^*}{\Sigma_i^c} \\
\vec{O} &:= (O_L^+, O_R^+) \\
\vec{I} &:= (I_j^c) \\
\vec{\Sigma} &:= (\Sigma_i^c) = (-I_0^c - I_2^c, -I_1^c - I_3^c, -I_3^c, -I_2^c) \\
\vec{a} &:= (a_i^*) \\
\vec{\xi} &:= (\xi_i^*).
\end{aligned}
$$

For simplicity, the calculations are performed without ambient light, and a constant measured light intensity of -0.5 which corresponds to a distance from the light source of approximately 1.7 meters (see eq. 6.13). The calculations for the four selected cases are as follows:

$$
\begin{aligned}
1): \quad \vec{I} &= (-1,-1,-1,-1) && (6.27) \\
\Rightarrow \vec{\Sigma} &= (2,2,1,1) && (6.28) \\
\Rightarrow \vec{\xi} &= \left(\frac{1}{2}a^*, \frac{1}{2}a^*, a^*, a^*\right) && (6.29) \\
\Rightarrow \vec{a} &= (a^*, a^*, a^*, a^*) && (6.30) \\
\Rightarrow \vec{O} &= (2\tau(a^*), 2\tau(a^*)) && (6.31)
\end{aligned}
$$

$$
\begin{aligned}
2): \quad \vec{I} &= (-1,-1,-0.5,-1) && (6.32) \\
\Rightarrow \vec{\Sigma} &= (1.5,2,1,0.5) && (6.33) \\
\Rightarrow \vec{\xi} &= \left(\frac{2}{3}a^*, \frac{1}{2}a^*, a^*, 2a^*\right) && (6.34) \\
\Rightarrow \vec{a} &= (a^*, a^*, a^*, a^*) && (6.35) \\
\Rightarrow \vec{O} &= (2\tau(a^*), 2\tau(a^*)) && (6.36)
\end{aligned}
$$

$$
\begin{aligned}
3): \quad \vec{I} &= (-1,-1,-0.5,-0.5) && (6.37)\\
\Rightarrow \vec{\Sigma} &= (1.5,1.5,0.5,0.5) && (6.38)\\
\Rightarrow \vec{\xi} &= \left(\frac{2}{3}a^*,\frac{2}{3}a^*,2a^*,2a^*\right) && (6.39)\\
\Rightarrow \vec{a} &= (a^*,a^*,a^*,a^*) && (6.40)\\
\Rightarrow \vec{O} &= (2\tau(a^*),2\tau(a^*)) && (6.41)
\end{aligned}
$$

$$
\begin{aligned}
4): \quad \vec{I} &= (-1,-1,0.5,0.5) && (6.42)\\
\Rightarrow \vec{\Sigma} &= (0.5,0.5,-0.5,-0.5) && (6.43)\\
\Rightarrow \vec{\xi} &= (2a^*,2a^*,2a^*,2a^*) && (6.44)\\
\Rightarrow \vec{a} &= (a^*,a^*,-a^*,-a^*) && (6.45)\\
\Rightarrow \vec{O} &= (0,0) && (6.46)
\end{aligned}
$$

At first, the fixed point equations of the first three conditions (see eq. (6.27) to (6.41)) indicate that no matter where a light source is detected, the resulting behaviour is a straight forward movement of the robot (see eqs. (6.31), (6.36), (6.41)). To understand the behaviour, the solutions for the receptors, which are equivalent to the synaptic weights (see eq. 6.21), must be taken into account (see eqs. (6.29), (6.34), (6.39)). They must be read as follows. The third and fourth values are the synaptic connections from the right light sensor I_3 to the right motor O_6 and the left sensor I_2 to the left motor O_7, respectively. The four situations reveal that the asymptotically stable values for the synaptic connections are different. If light is detected at one side, the corresponding weight is higher, although the output neuron is regulated towards the target value a^*. This means, that in the sensori-motor loop, the sensitivity towards changes of the light sensors is increased. Consider the second case form example (see eq. (6.32) to (6.36)). If the robot moves and the light sensor values remain constant, the robot will continue its straight movement. However, if the value I_2 changes, it has a higher influence on the motor output, due to the higher value of ξ_7 compared to I_2 and ξ_6. Therefore, the robot is more sensitive to changing light intensities at its right side. Changing light intensities within the environment only occur, if the robot senses a light source while it is moving. For ambient light, the measured light intensity is constant and independent of the position and orientation of the robot. Hence, the fixed point equations show, how ambient light increases the synaptic weights, and as a result the sensitivity of the controller to changing light intensity, but does not influence its trajectory. This explains the adaptivity of the controller to varying ambient light settings. The third situation clarifies why the robot slows down while it approaches the light source, but finally passes it, if no obstacle is present. While approaching the light source, the increasing intensity repeatedly reduces the output of the output neurons. This reduction is compensated for by the homeostatic behaviour of the output neurons. The result is an approaching towards the light source, and passing by it. The fourth condition (see eq. (6.42) to (6.46)) shows at which distance to the light source ($\vec{I} = (-1,-1,0.5,0.5)$) the robot will stop ($\vec{O} = (0,0)$). If no obstacle is present, then the halting of the robot depends only on the measured light intensity values. As they are given by the sum of the intensity of the ambient light and of the light sources, the time at which the robot stops and the stopping distance of the robot, with respect to a light source in this set-up ($l_i \in [0,1]$), depends on the ambient light configuration. An example of such a configuration is shown in figure 6.15. The fourth condition also gives the maximal ambient light setting for which the controller will function. Any ambient light value larger than 1.5 will result in a failure of the controller (not shown).

The reason why the controller does not react to the light source for ambient light values smaller than 0.5 is a consequence of the pre-processing. The input values remain negative and

hence, the output neurons O_6 and O_7 remain positive. The robot is able to slow down, but is not able to stop, as $O^+_{L,R}$ are always positive.

The very high oscillations for an ambient light source value of 1 is explained by the diverging behaviour of the SRN model for inputs close to zero. In this case, if no light source is detected, the light intensity inputs are zero $(I_2, I_3) = (0, 0)$, and the noise on the input neurons is amplified by the strong synaptic connection.

6.2.4 Discussion

The presented controller is a pure feed-forward structure. A comparable structure with the standard additive neuron model, would not solve the given task. It is known from literature, how solutions for this task generally look (Pasemann, Hülse, & Zahedi, 2003; Hülse et al., 2004; Hülse, 2007). One solution (Pasemann, Hülse, & Zahedi, 2003) utilises a hidden neuron with overcritical inhibitory self-connection, and a sub-structure called *chaotic 2-module* (Pasemann, 2002). Analysis of the controller revealed that the stopping behaviour, as a reaction to a detected light source, is the result of the complex dynamics domain of the controller. Considering neural networks as parametrised dynamical systems (Pasemann, 1996), the inputs drive the controller into different dynamical domains, each of which can be understood as a different behaviour pattern. In the case of the static light-seeker, it is driven into an oscillatory domain, enabling the robot to remain at constant distance to the light source. The solution with an oscillatory domain in the static case requires, in contrast to the feed-forward structure presented in this section, at least one recurrent connection. For other solutions with static synapses, the principle is similar. Adaptivity to varying ambient light conditions cannot be realised by a static strict feed-forward network, as it has only one global fixed point per input value, and therefore, only one reaction to each light intensity configuration. It is not adaptable.

6.3 Summary

This chapter presented two evolved SRN controllers for two different tasks. The first controller solved a standard benchmark control problem, the pole-balancing, in which a cart has to balance a pole in a bounded one-dimensional environment. This problem was chosen to demonstrate that the diverging behaviour does not limit the SRN model's applicability . A control structure inspired by the agonist/antagonist configuration in biological systems solved the task. The solution performed well compared to neuro-controllers known from literature.

The second experiment demonstrated the adaptivity of the SRN model. A robot has to find a light source in an environment, in which it cannot distinguish between a light source and ambient light in its sensor information. The solution takes advantage of the sensori-motor loop, and the homeostatic property of the SRN model. For constant input, the output neurons regulate towards their asymptotically stable values, but the synaptic connections realise a higher sensitivity towards changing light intensities, i.e. light sources. Solutions with the standard additive neuron model solve the task by keeping a constant distance to the light source. This is the result of a switching mechanisms triggered by a certain threshold value of the light sensors. Such a static solution would not perform well in the presented setting.

In what follows, the presented work is discussed and finally closed with conclusions.

Chapter 7

Discussion

Most research in the field of artificial intelligence can still be classified as GOFAI, which at its core, implies symbolic representation and rule-based processing of the symbols. In robotics, the sense-model-plan-act paradigm is currently the most favoured. Even if the models are biologically motivated, the results do not provide insights about the basic mechanisms of signal processing in biological nervous systems.

The fields of Behaviour-Based Robotics and Embodied Artificial Intelligence understand cognitive agents as embodied and situated entities, whose morphology directly influences their perception, and which are strongly coupled to their environment. Intelligence is the result of the interaction between the brain, body and environment. In Behaviour-Based Robotics, programmed solutions for behaviours, although proof of good engineering, are models which are weakly related to biology, e.g. Brooks subsumption architecture, and are, therefore, of the same biological relevance in understanding the brain as GOFAI approaches.

In Embodied Artificial Intelligence, which originated from Behaviour-Based Robotics, the situation is different. Research focuses on the interaction between the body, the control system and the environment. The control systems are also mostly pre-programmed, based on findings in biology, and do not provide new insights on the basic principles of neural signal processing in the sensori-motor loop.

Evolutionary Robotics, in general, uses recurrent neural networks of fixed structure, only changing the network parameters to generate solutions for a given task. The result is that behaviours are biologically relatable and often convincing, but again, provide no insights on the underlying principles which lead to the observed behavioural properties of interest.

The approach followed in this work is closely-related to the fields mentioned above. Embodiment and situatedness are understood as crucial elements for intelligence, and artificial evolution is used to generate solutions for a given task. The difference is that possible solutions are not pruned by pre-defined neural structures, as the structure itself is open to variation. This results in sparsely connected neural networks, which permit a full mathematical understanding of the behaviour-relevant dynamics. Generalising these to other problem domains is the result of formalising general principles of neural signal processing from the insights gained through the analysis of the solutions.

Adaptivity and learning were deemed early on as crucial factors for intelligence in the ap-

proaches discussed above, but taking GOFAI as an example, have only been taken into account in applications with very narrow ranges, as in automated pattern recognition and classification in specific domains for example.

In the field of Embodied Artificial Intelligence, biologically-inspired models of adaptivity and learning have only recently been introduced, and even then, only for very specific tasks (only positive or only negative tropism) and require artificial limitations, a high structural complexity, or strong assumptions on the pre-processing of the presented data. None of the methods which use adaptation allow the full analysis and understanding of the governing behaviour-relevant principles of neural signal processing in artificial systems, the results of which could reflect back onto the understanding of biological systems.

Hence, robotics is not used here as an engineering discipline, but rather as a method to understand intelligence. From this very general aim, a more specific one was targeted, which arose from the most prominent features of biological systems which are still widely lacking in artificial systems, namely adaptation and learning.

Following the Behaviour-Based Robotics approach that minimal systems must be fully understood first before systems of higher complexity are taken into account, a basic form of synaptic plasticity, a biological concept, was modelled here. In the context of cybernetics, it is not the underlying bio-chemical mechanisms that are of interest, but the principles which lead to a certain behaviour of the system. The standard additive neuron defines the level of modelling as it already shows dynamical features known from biological nervous systems, such as co-existing attractors (e.g. leading to the hysteresis phenomenon) and chaos. An extension of the standard neuron was proposed in this work as a model of short-time plasticity by synaptic scaling.

The model was inspired by Ashby's Homeostat, a machine consisting of homeostatic units, the local interactions of which lead to an adaptive behaviour of the entire system. In the model presented in this work, every neuron is a homeostatic unit, and the behaviour of an agent is the result of the local interactions of these Self-Regulating Neurons.

In contrast to the approaches mentioned above, this work is concerned with the understanding of general principles of neural signal processing. Consequently, the model was analysed analytically and mathematically as a disembodied dynamical system. This provided insights on the dynamical reservoir of a single neuron and small neuro-modules. The next step was to analyse the behaviour-relevant transient dynamics with respect to variations of the plasticity parameters. For experimentation, well-known structures from literature were chosen with increasing structural complexity. The result of the experiments and the insights gained on the transient dynamics of the SRN in the sensori-motor loop was an obstacle-avoidance behaviour which performs comparably well with respect to the best minimal static solution known from literature. With the plasticity property of the SRN model, the controller is structurally less complex but equally good at exploration and obstacle-avoidance in environments with sharp corners, in which static controllers of the same structural complexity fail. It solves the task by adapting the synaptic strengths to the different environmental conditions it encounters.

The insights gained from the experiments led to the artificial evolution of two controllers, one for the standard benchmark problem of pole-balancing, and one for a light-seeker which adapts to different ambient light situations. Both controller show that the diverging behaviour of the SRN model does not limit its application.

With respect to networks with the standard additive neuron model, the pole-balancer solves

the task comparably well. The ambient light-seeker demonstrates a behavioural novelty. The pure feed-forward structure can distinguish between ambient light and a light source as a result of the homeostatic property of the Self-Regulating Neuron and the interaction with the environment.

The result of this work is a minimal and general plasticity rule for recurrent neural networks of arbitrary structure, which is related to findings of homeostatic plasticity in biological nervous systems. The dynamics of a single neuron and small neuro-modules were discussed. Compared to related plasticity rules for artificial neural networks, it is not limited to feed-forward networks, does not require a trigger mechanism or artificial limitations, and due to its generality, allows the full-understanding of the behaviour-relevant dynamics as minimal neural networks can be generated by structure evolution, thus, enabling such analyses.

The model shows the limitations of pure synaptic weight dynamics. Such a form of synaptic modulation (in all of the discussed approaches) is comparable to short-term plasticity described for biological systems. STP is an adaptivity mechanism, as re-adaptation is necessary for a previously encountered situation which was not present for some period of time. No matter how large or small the plasticity parameters of any purely synaptic weight-affecting mechanism are, after presenting a stimulus for some time, all traces of this stimulus affecting the synapses in the system will not be found in the system after some other period of time has elapsed. The duration of both periods is determined by the plasticity parameters. The time required to show a reaction to a presented stimulus is in the same order of magnitude of time as required for the system to recover from it. This is the definition of STP. Learning is classified as a form of plasticity in which the changes in the reaction to a stimulus is present for much longer than the duration for which training has occurred. This form of plasticity is referred to as long-term plasticity (LTP) and is, mainly, related to structural changes in biological nervous systems. This might be achieved by different rates for growth and decay. Only LTP, and hence, memory would be present without the behavioural property of adaptation. Another problem would be that either long-term potentiation or depression, but not both, would be available. Both forms of plasticity (STP and LTP) are found in biological systems, and it is their interplay which gives rise to complex adaptive systems. STP was modelled in this work, LTP must be modelled on top of it. What is proposed is a method of synaptic growth and regression, or for short, synaptogenesis, based on the signal flow along a synaptic pathway as it is found in experiments, such as those conducted with the Aplysia.

Considering STP and LTP as two basic levels of learning, then the form of LTP described above may be related to the third level of learning, specifically, learning over generations. So far, artificial evolution and evolutionary robotics are stochastic, gradient-based search methods in the parameter space (of neural networks). It does not seem biologically plausible to generate networks using random structural changes, as is done in the artificial evolution algorithm used in this work. The genes within the DNA of biological systems are too few to encode the connections of the neurons in the brain. It is assumed that rules for the formalisation and elimination of the neurons and their connections are encoded in the genes. This is supported by findings in children, for example, who have more neurons and synapses when they are born, the numbers of which decrease as the child grows older.

It is assumed that it is not the connections, but rather the rules of generating and reducing neurons and synapses that are encoded in the genes. Such synaptogenesis rules should be

formalised and used in artificial evolution. It is plausible to assume that biology has found the best solutions, and hence, the best principles for neural signal processing in complex adaptive systems. The current methods are not able to reproduce similar solutions without restricting the evolutionary process. One example of differences in biological and artificial neural systems, is the symmetry found in the sensory and motor system of biological systems, which is not taken into account at all in artificial evolution. If the assumption that biology has found the best solutions is wrong, better solutions could be found. In both cases, the generating evolutionary methods should be changed, and not restricted to force narrow-ranged solutions. In current research, the generating methods of stochastic nature are kept unchanged and restrictions, such as the evolution of neuro-modules for specific brain functions, are used instead. This leads to the same faults GOFAI has faced in history, as it is hoped that the highly specified sub-solutions will produce an overall solution once proper fusion techniques are found. The new form of artificial evolution could be the evolution of rules to assemble and disassemble neural structures during the life-time of an artificial agent. The rules should be based on different local mechanisms, taking local properties of neurons, their synapses and neighbours into account. They then define the LTP mechanisms which are built upon the present STP methods.

Currently, ongoing research focuses on possible connectivity rules for plastic SRN networks, on which such new generative methods could be based. Future work will include regulatory terms for the plasticity parameter α and the bias value, which are presently static. The plasticity parameter α controls the target value, and could be related to proprioceptive sensors (e.g. energy reservoir or consumption sensor) to alter the behaviour of the agent. Similar to the sliding threshold of the BCM rule, a regulatory bias could serve as a natural limitation factor for the diverging behaviour of the SRN model, making it biologically more plausible.

Appendix A

ISEE

For the artificial evolution and analysis of the dynamical properties of recurrent neural networks presented in this work, a software package, designed and mainly written by the author of this work was used. The software package is called ISEE (*integrated structure evolution environment*) and is presented in this appendix.

ISEE (Zahedi & Hülse, 2008) is a package implemented around the ENS^3 algorithm (see sec. 3.3). Another tool, providing the same algorithm called CEN[†] was available at the time, but was limited by its lack of a general communication interface. This meant that a substantial implementing effort was required for new experiments. Additionally, the tool only supported the standard additive neuron model. Extending it in order to evolve recurrent neural networks with other models, such as the proposed SRN model, would have required large changes within the source code. Therefore, the software package ISEE was created. ISEE is a not a single application, but a software package consisting of three main software tools for artificial evolution and the analysis of dynamical features of recurrent neural networks (see fig. A.1). Additional tools support the analysis of dynamical systems, and the handling of the logging files produced during evolution.

The first tool is *EvoSun*[‡], which is the implementation of the ENS^3 algorithm (Dieckmann, 1995; Pasemann et al., 2001). The most prominent feature is its GUI, which provides interactive access to all evolution control parameters (see sec. 3.3) and the possibility to monitor the relevant indicators of the evolutionary process during runtime.

The second tool is *Hinton*. It connects EvoSun to a simulated or a physical robot and is the implementation of the evaluation process. Decoupled from EvoSun, Hinton is a tool used to analyse the internal dynamics of recurrent neural networks in the sensori-motor loop while the network controls the simulated or physical robot within its environment.

The third tool is named YARS (*yet another robot simulator*). It is a real-time simulator based on the physics engine ODE (Smith, 2005). The rendering is done by OpenGL (Group, 2007). Other rendering engines, such as Ogre (Streeting et al., 2006) and drawstuff (Smith, 2005) were also included in earlier versions of YARS. Providing an evaluation environment in YARS only requires a single XML description file. This minimises the time and implementation effort

[†]CEN was written by Ulrich Steinmetz. There are no publications available.

[‡]Author of EvoSun is Martin Hülse.

required to set-up a new experiment. The XML-file includes the description of the environment, and a set of robots. For each robot a separate communication port is created. Through a handshake mechanism, YARS and Hinton exchange all the necessary information about the sensor and motor configuration automatically. With the exception of the XML-file specifying the experiment, no further implementation effort is required.

This section begins with a functional overview of tools included in the ISEE package. The software tools are explained, in as much detail as necessary, to understand how the experiments were conducted and how the analytic results were obtained. In addition to the three previously mentioned tools, two more are then introduced for the analysis of recurrent neural networks, namely *Brightwell* and *Analyser*.

For the sake of completeness and for the interested reader, this section is followed by a description of the concept of ISEE as well as a more detailed description of all the ISEE tools.

The following appendix (see app. B) explains how to implement new experiments and analysis methods within the ISEE framework. This appendix closes with an overview of applications of ISEE in different research projects (see fig. A.19ff).

A.1 Overview of the main tools

This section introduces the five main components of the ISEE software package. All of the experiments and analyses in this work were conducted with these five tools. Hence, this section is sufficient to understand how the results presented in this work were obtained. The five main tools of ISEE are *EvoSun*, *Hinton*, *Analyser*, *Brightwell*, and *YARS*.

EvoSun is the implementation of the evolutionary algorithm ENS^3 (Dieckmann, 1995). The software tool EvoSun provides a graphical user interface (GUI) which displays all evolution control parameters. These parameters, which can be altered during runtime, include the probabilities with which neurons and synapses are added and removed, the probabilities with which bias values and synaptic weights are altered, the costs for neurons and synapses, etc. (see sec. 3.3). This is a significant difference to other approaches, such as genetic algorithms, in which the evolutionary process runs as a batch process. This software tool was designed and written by Martin Hülse[†]. Although the author of this work did not contribute to EvoSun, it is presented first, as it is the starting point in running an artificial evolution (see fig. 3.3).

Hinton provides two main functionalities. First, during artificial evolution, it evaluates the behaviour of a recurrent neural network with respect to a fitness-function. Second, when decoupled from the artificial evolution, Hinton provides analysis tools to determine the behaviour-relevant dynamics and the structure–function relationship of recurrent neural networks.

The evaluation occurs during the evolution in combination with EvoSun and a simulated or physical robot. In case of evaluation, Hinton receives recurrent neural networks from EvoSun. Each network is evaluated with respect to a fitness-function implemented in Hinton (see app. B.3). Once the evaluation is terminated, the generated fitness-value is returned to EvoSun and used as a parameter for the selection operator (see sec. 3.3).

[†]No publication available.

The analysis occurs separately from the artificial evolution. In this set-up, a recurrent neural network is loaded into Hinton from the file system. Hinton provides a GUI which supports the alteration of the structure and parameters of the recurrent neural network during runtime. Interactively changing the network properties and its structure is referred to as lesion experiments. Observing the results of the lesions in the behaviour is a method used to determine the structure–function relationship and the individual contribution of the synapses and neurons. To support this method, a tool that plots the transients of the internal states of the recurrent neural network is included in Hinton. It is named Analyser and is discussed hereafter.

To evaluate or analyse a controller, Hinton can connect either to a simulated or a physical robot. It is also possible to switch between both settings during runtime, a feature which is necessary in order to close the gap between the simulated and physical robot. Comparing the differences in the behaviours indicates required changes to the simulation properties. This issue is discussed in detail in the following section (see sec. A.3.5).

An export function is included, which converts recurrent neural networks to program code for specific robot hardware platforms or third party tools. Currently supported are Atmel AVR Assembler (Atmel, 2008), C/C++ (Stroustrup, 2000), Java (Sun Microsystems, 2007), IConnect (MICRO-EPSILON, 2006), GML (Himsolt, 1997), YSocNet (Ghazi-Zahedi, 2001), GermanTeam C++ module code (*GermanTeam2004*, 2004).

Analyser: The Analyser (Rosemann, 2004) provides two visualisation methods which enable the user to analyse the structure–function relationship of recurrent neural networks. These methods are the graph visualisation and transient plot (see fig. A.7).

The graph visualisation method displays the structure of the recurrent neural network, where the current states of the network properties are colour-coded. For a neuron, the neuron colour corresponds to the current activity, and for a synapse the colouring corresponds to the sign of the synapse. The synaptic strength is indicated by the width of the line representation. Different graph layout algorithms are implemented and can be applied to arrange the neurons conveniently.

The transient visualisation method plots the internal states of the recurrent neural network similarly to EEG plots. When combined with lesion experiments, they allow the behaviour-relevant dynamics, as well as their relation to the parameters and structures of the recurrent neural network, to be determined.

Brightwell is a tool which is used to analyse the dynamical properties of a recurrent neural network. In contrast to the *Analyser*, the network is analysed independent of the sensori-motor loop. Different methods, such as the Bifurcation-diagram, Iso-periodic plot, First-return map, Liapunov exponent, etc., may be used in the analysis process. Adding new analysis methods is possible with a minimal implementation effort (see app. B.5). All the plots within this work, which visualise the dynamical properties of recurrent neural networks, are generated with Brightwell and the Analyser.

YARS is a robot simulator. In YARS a simulation is defined by a single XML-file, which includes the environmental description and the description of a set of robots. The description

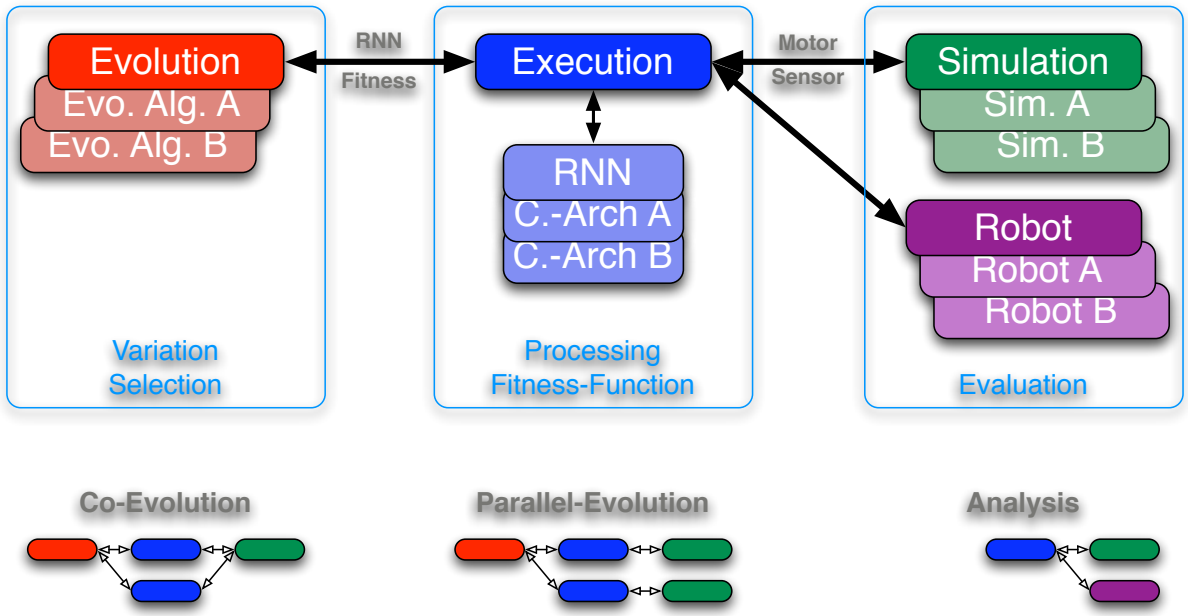


Figure A.1: ISEE CONCEPT. The upper part of the figure shows how the artificial evolution is realised with the three distinctive tools. The variation and selection operators are implemented in an evolution program. This program (left-hand side) generates RNN and sends them to an execution program (centre). The execution program receives sensory input from a simulator or robot, processes the network, calculates the fitness and returns motor command to the evaluation environment. The fitness value is returned to the evolution program. During artificial evolution, a simulator (right-hand side) provides an evaluation environment. The schematics below show different possible applications. Left: Co-evolution. Centre: Parallel distributed evolution: Right: Transient analysis in the sensori-motor loop.

file includes the physical properties of all the objects of the environment and the robot. For the robot, the description file also includes the specifications of the actuators and sensors.

The initial version of the XML-grammar was used as a basis for a project-wide simulation definition language in the German Research Foundation Priority Program 1125 "Cooperating teams of mobile robots in dynamic environments" (DFG-SPP 1125 "Kooperierende Teams mobiler Roboter in dynamischen Umgebungen") (Zahedi et al., 2005).

For each robot defined in the XML-file, a communication connection between YARS and Hinton is created, through which all information about the sensor and motor configuration of the robots are exchanged automatically. Therefore, in addition to the XML-file, no further implementation is required. For a detailed description of how an experiment is specified, the reader is referred to the following appendix (see app. B.7).

A.2 ISEE Specification

The previous section presented an overview of the main software tools included in the ISEE package. In the remainder of this appendix a more comprehensive list and a more detailed

description of the tools is presented. First however, the design and architectural considerations that led to the concrete realisation of ISEE is presented.

The requirements for a software package for the structural evolution of recurrent neural networks were derived from knowledge gained from previous experiments (Dieckmann & Pasemann, 1995; Hülse et al., 2001; Lara et al., 2001; Pasemann, 1998, 1997a; Pasemann & Dieckmann, 1997b, 1997a; "Evolving brain structures for robot control", 2001; Pasemann et al., 2001; Nolfi & Floreano, 2000).

A.2.1 Design Considerations

The design considerations consist of four different parts:

1. The desired features.
2. The estimated end user characteristics.
3. The target operating systems.
4. The target robot platforms.

Desired features

The desired features can be grouped according to the different phases of an artificial evolution experiment. These phases are the set-up, the evolution, and the analysis of the results. A fourth group of necessary features is added, which are general and required in the three phases.

1 Set-up phase:

- 1.1 Setting up an experiment should not require a priori knowledge of possible solutions.
- 1.2 Support for extending an existing recurrent neural network by adding new behavioural properties. This can be done in at least two ways[†]:
  - 1.2.1 By fusion of two or more existing recurrent neural networks.
  - 1.2.2 By expansion of an existing controller.
- 1.3 At least three different neuron models must be supported:
  - 1.3.1 Standard additive neuron model.
  - 1.3.2 Chaotic neuron model.
  - 1.3.3 Self-Regulating Neuron model.

2 Experimental phase:

- 2.1 Support for on-line monitoring of the evolutionary process.
- 2.2 Interactive control of the evolutionary process by variation of the process parameters during runtime.

[†]For a comprehensive overview of fusion and expansion techniques see (Hülse, 2007)

2.3 Parallel evolution of populations is required to conduct predator-prey experiments as described by Nolfi and Floreano (1998) and in order to decrease the search space (Markelić & Zahedi, 2007).

2.4 Evolution of arbitrary behaviours for arbitrary robotic platforms should be supported.

3 Analysis phase:

3.1 Different analysis methods are required:

3.1.1 Transient plots of the dynamics of the recurrent neural network in the sensori-motor loop.

3.1.2 Visualisation of the dynamical properties (Bifurcation diagram, Isoperiodic plots, etc.) of the recurrent neural network independent of the sensori-motor loop.

3.1.3 Lesion experiments and parameter variation of the recurrent neural network during runtime to determine the behaviour relevant sub-structures.

4 General requirements:

4.1 Artificial evolution processes many individuals until a solution is generated. This requires an efficient and fast implementation.

4.2 The system should be open with respect to the robot platform and desired behaviours. This requires the possibility to easily add and modify:

4.2.1 Fitness-functions.

4.2.2 Simulations.

4.2.3 Neuron or network models.

4.2.4 Methods to visualise dynamical properties of recurrent neural networks.

End User Characteristics

After identifying the desired features, the next step is to analyse the user group.

1. The user group should not be limited to programming experts. This means that setting up experiments must be possible with minimal or no special programming knowledge. Therefore, the system has to provide very simple interfaces.

2. The ISEE package must be available for the most common operating systems. Otherwise, users who are bound to specific operating systems are excluded from using ISEE.

Operating System / Hardware

The software package ISEE was created much earlier then YARS. At the time, different simulators were only available for specific operating systems. Even now, some simulators are still more applicable to simulate specific robot platforms and tasks. As an example, the Khepera 2.0 Simulator (Michel, 2005) is still the fastest simulator for Khepera-like robots and the obstacle-avoidance task (see chap. 5), but it requires an UNIX operating systems and the X11 library (Michel, 2005).

In order to provide users with the flexibility to choose any simulator, ISEE should not be restricted to either Linux/UNIX, Windows, MacOS or Solaris operating system, but support at least these four.

A.2.2 Architectural Strategies

The architectural strategies presented in this section are derived from the design considerations mentioned above.

- The evolutionary process is divided into distinctive software tools:
  - The selection and variation operators are separated from the evaluation operator (see fig. A.1).
    - The separation has two advantages:
      1. The evaluation tool can be used, independently of the remainder of the system, to analyse the dynamics of the recurrent neural network in the sensorimotor loop.
      2. Several populations can be spread over several evaluation tools and evaluated in the same simulation environment (predator-prey experiment), without any extra implementation cost.
    - The separation has one disadvantage:
      1. Several tools have to be executed and set-up for an experiment. This disadvantage is resolved through the use of start-up scripts and configuration files.
- Java was chosen as the programming language. It is supported on the most common operating systems, specifically on Windows, Linux/UNIX, SunOS, and MacOS.
- Whenever possible, standards are used. For example, XML is chosen as the logging file format for the evolutionary process, as there exist a variety of well-documented and supported parsing libraries for Java, C/C++, and other programming languages.
- The UDP protocol was chosen for communication between Hinton and the simulator, as the communication is time-critical. Both software tools run locally on a single computer. Locally transmitted UDP-packages are not lost. Therefore, there is no need for the error correction mechanisms of the TCP/IP protocol. Even if a package is lost, for a robust controller this should be of no consequence. A UDP package contains values representing sensor and motor data.
- TCP/IP communication between EvoSun and Hinton. This communication requires support for the exchange of larger packages and is not time-critical as the neural networks must be exchanged. This is well-supported by the TCP/IP protocol.
- Implementations of fitness-functions, simulator/robot communication, neuron models, and analysis methods are loaded dynamically and automatically from specified directories (see fig. A.2). Each implementation has to extend a superclass which provides the necessary

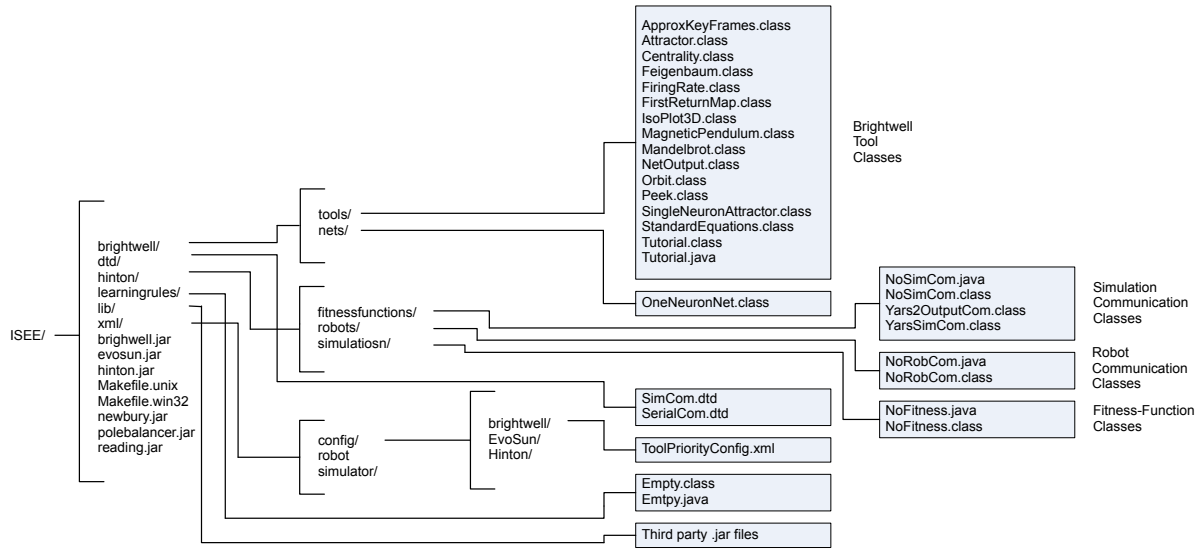


Figure A.2: ISEE DIRECTORY STRUCTURE. This figure shows how ISEE is organised in the file system. From left to right, directories and sub-directories. Blue boxes: Implementations of functionality such as fitness-functions, communication classes, etc. that are loaded dynamically at start-up.

basic functionality. To set up a new experiment, the user need only address the desired additional functionality.

A.3 ISEE Tools

The previous section discussed the design considerations and architectural strategies for an artificial evolution framework. This software specification led to the implementation of ISEE as a collection of distinctive software tools. The complete list of tools included in ISEE is given in table (see tab. A.1). Next, all the tools available in ISEE are presented and discussed.

A.3.1 Cholsey

The first tool to be presented is not a tool that can be executed. It is the implementation of the recurrent neural network, named *Cholsey*. During the artificial evolution, a large number of recurrent neural networks are processed with a considerably large number of iterations. The chosen implementation of the neural network is time-critical with respect to the artificial evolution, hence, it is discussed separately.

A recurrent neural network can be considered as a directed graph or digraph (see sec. 3.3.1). There are two possible ways to represent a graph, namely an adjacency-matrix and an adjacency-list (Cormen et al., 1990).

A.3.1 Definition: Adjacency-Matrix, by Cormen et al. (1990)

Cholsey	The neural network implementation.
EvoSun	Implementation of ENS^3 algorithm and visualisation of the evolutionary process.
Hinton	The neural network processor and implemenation of evaluation and analysis processes of the recurrent neural networks.
Analyser	Neural network visualisation and transient plotter.
YARS	Yet Another Robot Simulator.
Brightwell	Visualisation of the dynamical properties of the recurrent neural networks.
Reading	GUI for the evolution log-files.
NewBury	Merges a set of recurrent neural networks in one file.
Beaumy	Extracts a new initial set of recurrent neural networks from an existing evolution log-file.

Table A.1: ISEE Tools. The tool's name, with the exception of EvoSun, Analyser and YARS, the name are derived from castles that exist or existed in the birth district of the author, Bournemouth, and were taken from (Britannia.com, 2006).

Let $G = (V, E)$ be a graph. Then the adjacency-matrix representation of the graph G consists of a $|V| \times |V|$ matrix

$$\begin{aligned} A &= (a_{ij}) \qquad i, j = 0, 1, \ldots, |V| - 1 \\ &\textit{such that} \\ a_{ij} &= \begin{cases} 1 & \textit{if } (i, j) \in E, \\ 0 & \textit{otherwise.} \end{cases} \end{aligned}$$

In the case of a neural network the matrix entries $a_{ij} \in \{0, 1\}$ are replaced by the synaptic weights $w_{ij} \in \mathbb{R}$.

A.3.2 Definition: Adjacency-List, by Cormen et al. (1990)
Let $G = (V, E)$ be a graph. Then the adjacency-list representation of a graph G is an array Adj of $|V|$ lists, one for each vertex in $u \in V$. For each $u \in V$, the adjacency list $Adj[u]$ contains all the vertices v such that there is an edge $(u, v) \in E$.

For a neural networks this means that each neuron contains a list of all outgoing synapses.

Which representation should be chosen depends on two factors:

1. The purpose of the representation.
2. The density of the graph.

One possible criteria which may be used to determine the choice of representation is the type of visualisation of a recurrent neural network. It will be discussed later in this appendix (see

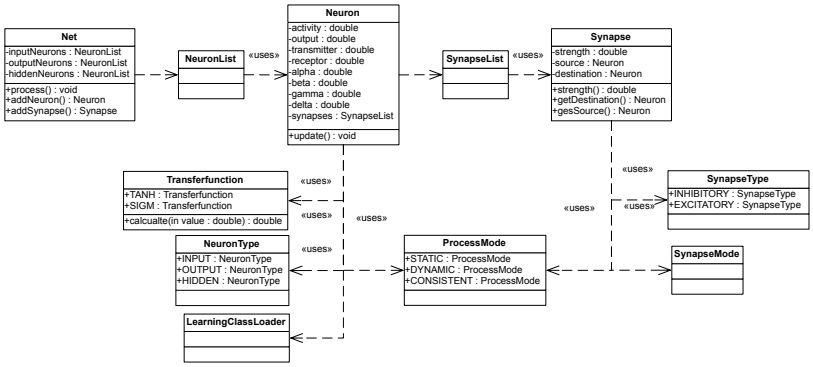


Figure A.3: CHOLSEY UML DIAGRAM. In this concept, the neuron is the central unit. All neurons are managed in the class *Net*. Each neuron has access to all incoming synapses and can, therefore, calculate its internal states autonomously. The class *LearningClassLoader* dynamically loads all available learning rule implementations and provides them over a generic interface as singletons (Gamma et al., 1995). The strength function of the class *Synapse* either returns the assigned constant strength value or dynamically calculates the strength as a product of the pre- and post-synaptic neuron properties.

sec. A.3.3), that both methods, a table (adjacency matrix) and a graph view (adjacency list) of the recurrent neural network are useful for the analysis of the dynamical properties of the controller.

As stated at the beginning of this section, the implementation of the recurrent neural network is a time-critical aspect of artificial evolution. Compared to the visualisation of a recurrent neural network this has a higher priority with respect to the choice of representation.

If the computational cost is considered, the question of which representation is more suitable, matrix or graph, depends on the expected density of the recurrent neural networks that are generated by the evolutionary algorithm. Consider the following two examples.

A sparse graph ($|E| \ll |V|^2$) is represented by a sparse matrix. Iterating through a network represented by a sparse matrix results in many unnecessary computations, as a large number of weight and, therefore, entries w_{ij} in the matrix are zero and do not contribute to the activity. In this case an adjacency list would be computationally cheaper.

A dense graph ($|E| \approx |V|^2$) is represented by an array with large adjacency lists. Processing a list is computationally more expensive then accessing an array. In the case of a dense graph an adjacency matrix representation would be computationally cheaper.

Is is now clear, that the chosen implementation of Cholsey depends on the type of structures that are generated by the selected evolution strategy. Using the ENS^3 algorithm for structural evolution of recurrent neural networks, the resulting networks are likely to be sparsely connected, due to the approaches followed in this work. That is, evolved controllers are analysed with respect to their behaviour-relevant dynamics, in order to extract generalisable principles of neural signal processing (see sec. 3.3). Structurally small networks, i.e. sparsely connected networks can be fully analysed and understood, and are, therefore, preferred. By setting neuron and synapse costs, and carefully choosing the structure modification parameters of the algorithm ENS^3, sparse networks can be favoured during the artificial evolution.

Algorithm 2 Implementation of Cholsey.

Require: $N(t) := \{n_i(t)\}_{i=0,\ldots,|N|-1}$ {set of all neurons}
Require: $n_i(t) := \{\Theta_i, a_i(t), o_i(t), \mathcal{S}_i\}$
Require: $\mathcal{S}_i := \{w_{ij}\}_{j=0,1,\ldots}$ {set of all incoming synapses}
1: **for all** $n_i(t) \in N(t)$ **do**
2: $a_i(t+1) = \Theta_i + \sum_{j=0}^{|S_i|-1} w_{ij} \cdot o_j(t)$
3: **end for**
4: **for all** $n_i \in N$ **do**
5: $o_i(t+1) = \tau(a_i(t+1))$ {where $\tau(x)$ denotes the transfer-function}
6: **end for**

Algorithm 3 Implementation of Cholsey with the SRN model.

Require: $N(t) := \{n_i(t)\}_{i=0,\ldots,|N|-1}$ {set of all neurons}
Require: $n_i(t) := \{\Theta_i, a_i(t), o_i(t), \mathcal{S}_i(t)\}$
Require: $\mathcal{S}_i(t) := \{w_{ij}(t)\}_{j=0,1,\ldots}$ {set of all incoming synapses}
Require: $\mathcal{C}_i(t) := \{c_{ij}(t)\}_{j=0,1,\ldots}, c_{ij} \in [-1, 1]$ {sign of incoming synapses}
Require: $w_{ij}(t) := c_{ij} \cdot \xi_i(t) \cdot \eta_j(t)$
1: **for all** $n_i \in N$ **do**
2: $a_i(t+1) = \Theta_i + \sum_{j=0}^{|S_i|-1} w_{ij}(t) \cdot o_j(t)$
3: **end for**
4: **for all** $n_i \in N$ **do**
5: $o_i(t+1) = \tau(a_i(t+1))$ {where $\tau(x)$ denotes the transfer-function}
6: update $\xi_i(t+1) = f_\xi(a(t+1))$ {Transmitter strength calculation (see chap. 4)}
7: update $\eta_i(t+1) = f_\eta(a(t+1))$ {Receptor strength calculation (see chap. 4)}
8: **end for**

Hence, an adjacency-list was chosen as the representation structure of the neural network. In contrast to the definition of an adjacency-list given above (see def. A.3.1), each neuron contains a list of all *incoming* synapses. This enables the calculation of the state of a neuron based on information stored locally in each neuron (see alg. 2). A UML diagram of the structure is shown in figure A.3, where the class *Net* is the root of the structure, equivalent to the array *Adj* described in the definition A.3.1.

Using the SRN model (see chap. 4), the synaptic weight is the product of a pre- and post-synaptic neuron property (ξ, η). The implementation of a network (see alg. 2) can easily be extended to support other synapse models or local learning rules (see alg. 3).

A.3.2 EvoSun

The software tool *EvoSun* (see fig. A.4) was developed by Martin Hülse and is the implementation of the ENS^3 algorithm (Dieckmann, 1995). It is an improvement upon the previously used software tool CEN[†] which was initially written by Uli Steinmetz. The prominent feature

[†]No publications available.

of EvoSun is its ability to monitor and influence the evolution process during runtime. The parameters which determine the probabilities with which neurons and synapses are added and removed, the changes of synaptic weights and the bias values, are accessible through a GUI (see fig. A.4).

The monitoring functionality displays the development of all relevant evolution properties. Examples of these properties are the fitness of the best individual, the average fitness of the population and its standard variation, the age of the oldest individual in the population, etc. In particular, the oldest individual is of interest as it has survived several of the presented environmental configurations, it is likely to have generalisation properties with respect to the given problem. Although it might not have the highest fitness, it should, therefore, be considered in the analysis.

A log-file of the evolution is written by EvoSun. In each generation a minimum of two individuals is stored: the fittest and the oldest individual of the current population. Any other number $n > 1$ of individuals can be stored. In this case, the n fittest individuals plus the oldest are stored in the log file.

A.3.3 Hinton

The Hinton tool processes the neural network. Additional functionality depends on the context. There are two possible uses:

1. Evaluation of a recurrent neural network during the evolution process.
2. Analysis of the behaviour-relevant dynamics and the structure–function relationship of a recurrent neural network in the sensori-motor loop.

This section will introduce the concept of Hinton before presenting the functionality provided for both the uses mentioned above; namely evaluation and analysis of recurrent neural networks.

Concept

The graphical user interface (GUI) of Hinton is divided into three regions, left, centre, and right (see fig. A.5). Each of the regions represents one aspect of the functionality and is labelled accordingly:

The left region groups all functionality relating to receiving and loading a recurrent neural network.

The central region groups all functionality relating to the execution, evaluation and analysis of the networks.

The right region groups all functionality relating to the communication with a simulated or physical robot.

The graphical division of the GUI also represents the architecture of Hinton (see fig. A.6). This section does not discuss the architecture in full detail, but covers the aspects relating to setting up an experiment.

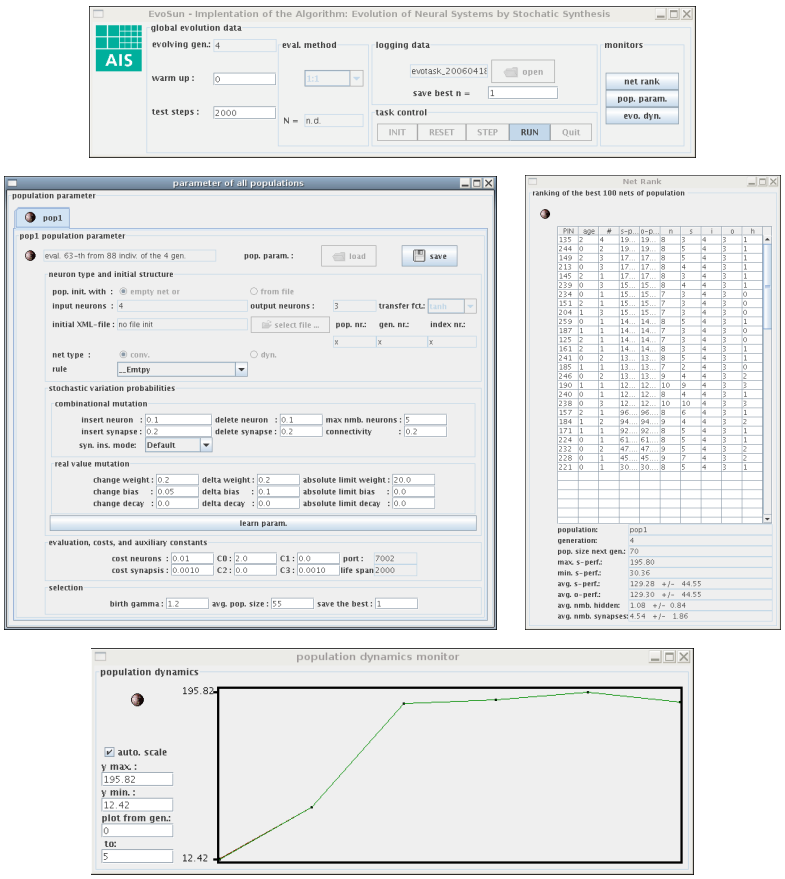


Figure A.4: EvoSun GUI. Top: *EvoSun Control Panel*. This panel controls the number of evaluation steps (cycles), the number of test steps in which no evolution is performed (warmUpSteps), the number of clients for parallel evolution, and the logging of the evolution. Right: *Parameter Control Panel*. This panel provides an interface to interactively control the evolution during runtime. All evolution parameters, such as the probability with whihc neurons and synapses are added and deleted, can be altered here during the evolution. Left: *NetRank Panel*. This panel shows the selected individuals of the previous generation and the number of offspring which were generated from each of them. Bottom: *Evolution Dynamics Panel*. This panel provides the functionality to monitor of the evolution process. A number of plots may be generated, e.g. the fitness of the best individual, age of the best individual, age of the oldest individual, average fitness and its variation.

The entry point is the class *HintonMain*. It initiates the *control* package, which initialises all other packages and handles the dependencies between them.

The *broker* package is responsible for the communication between Hinton and the evolution software tools. Currently, CEN and EvoSun are supported. The communication includes the current individuals generated by the evolution software, and the fitness value evaluated by Hinton.

The *ambassador* package handles the communication between Hinton and the simulated or physical robot. From the perspective of Hinton, both are handled equally (see app. B.2).

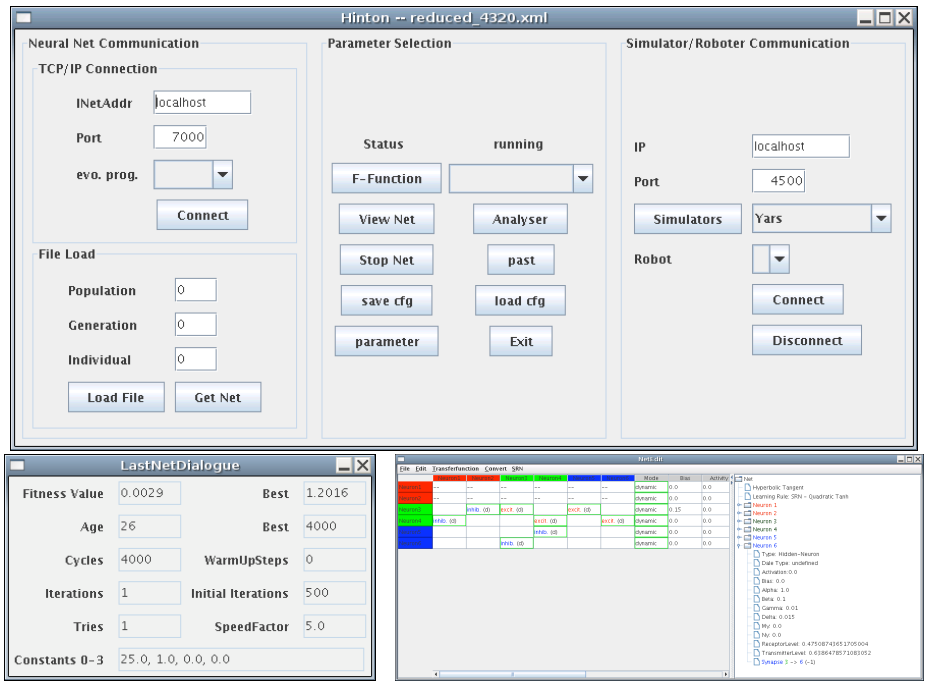


Figure A.5: HINTON GUI. This plots shows the GUI provided by Hinton. Top: *Hinton Main GUI*. The GUI is divided into three parts. The left hand side provides an interface to configure the connection to an evolution program, and to load an individual from the file system. The centre provides the functionality for analysing and evaluating the recurrent neural network. The pull-down menu lists all available fitness-functions. The buttons invoke the Analyser, the NetEditor and the LastNetDialogue. The left hand side provides an interface to connect to a simulator or the physical robot. Pull-down menus list the available interfaces. Bottom left: *LastNetDialogue*. The LastNetDialogue is used during evolution. It displays all evolution parameters, the data of the best and the last individual of the current population. Bottom right: *NetEditor*. This GUI is an interface to the network structure and network parameter. It provides functionality to change the network during runtime. Neurons and synapses can be added or removed, and bias values and synaptic weights can be changed. It is not available during evolution.

The communication consists of receiving sensor data from the robot, and sending the motor commands that were generated by the neural network.

The *executor* package provides the functionality required to process, evaluate and analyse a recurrent neural network. Sensor data is fed as input to the network, and motor commands are generated from the network output.

The exchange of information between these three packages is controlled by two interfacing classes, *RobotStruct*, *ProcessParameter* (see app. B.4). The first stores the motor and sensor data, network input and output, and the relation which input/output neuron is mapped to which sensor/motor, respectively. The latter stores all data received from the evolution software. This data includes the current neural network and the number of evaluation cycles. For a full list of available parameters the reader is referred to the next chapter (see app. B.4.1).

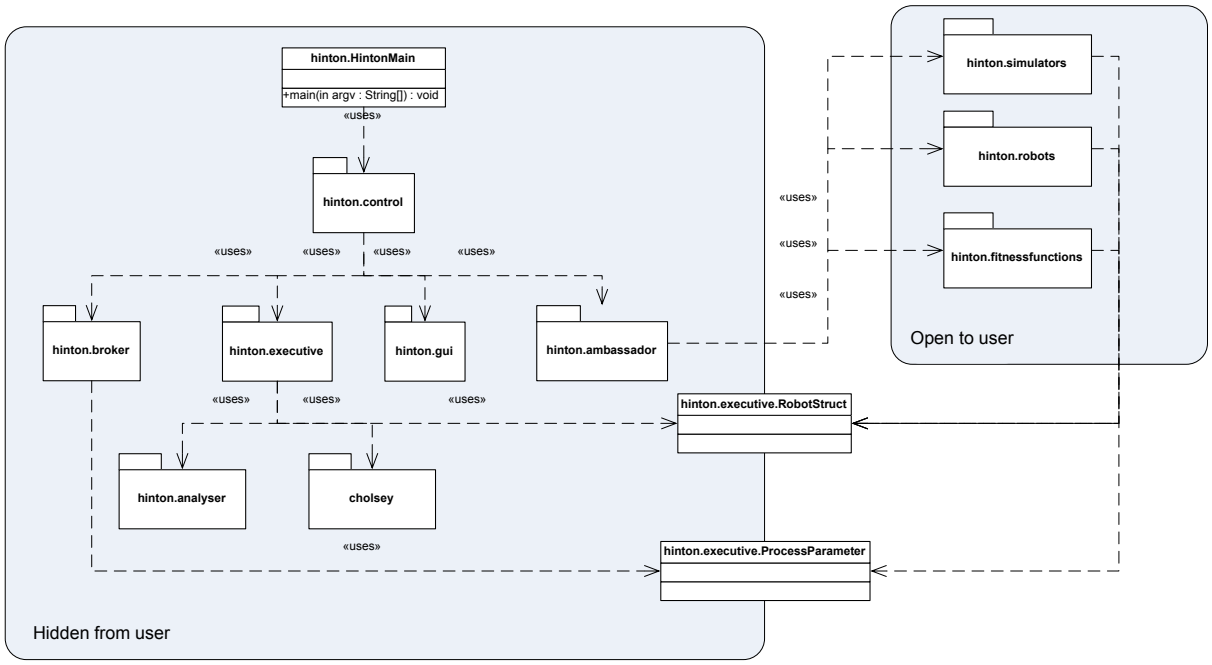


Figure A.6: HINTON UML DIAGRAM. This diagram shows the concept of Hinton, which is divided into several packages. The package *hinton.control* initialises all other packages and handles their dependencies. The *hinton.broker* package is responsible for the communication with the evolution program. The *hinton.executive* package takes care of the fitness-functions, the recurrent neural network processing and their analysis. The *hinton.ambassador* package handles the communication between Hinton and the simulated or physical robot. The *hinton.analyser* package is the implementation of the Analyser tool. These packages require no user input. For the purpose of setting up a new experiment, Hinton provides three extension points: *hinton.simulators*, *hinton.robots*, *hinton.fitnessfunctions*. The interface between the extension points and Hinton is provided in form of two classes: *hinton.executive.RobotStruct* and *hinton.executive.ProcessParameter*. These two classes provide all necessary information (see app. B.4).

In each of these three packages, the functionality of Hinton can be extended to match the requirements of the experiment of interest. For a detailed description of how to extend Hinton, the reader is referred to the next chapter (see app. B).

Analysis of recurrent neural network

During analysis, Hinton provides two different methods to visualise and manipulate a recurrent neural network during its control of a robot in the sensori-motor loop, a neural network editor (see fig. A.5) and a transient plot tool.

The neural network editor is divided into a matrix view (see fig. A.5b, left) and a tree view (see fig. A.5b, right) of the neural network. The matrix view enables change to be made to the structure and the parameters of the neural network. Changes of the parameter values are passed to the network during runtime, so that the effects of the changed parameter on the behaviour of the robot are immediately observable. This type of analysis is referred to as lesion experiment.

The editing tool also provides functionality to export recurrent neural networks. A network can be exported to program code for specific robot hardware platforms or third party tools. Currently supported are exporters to Atmel AVR Assembler (Atmel, 2008), C/C++ (Stroustrup, 2000), Java (Sun Microsystems, 2007), IConnect (MICRO-EPSILON, 2006), GML (Himsolt, 1997), YSocNet (Ghazi-Zahedi, 2001), GermanTeam C++ module code (*GermanTeam2004*, 2004).

The *Analyser* is as an independent tool, although it is incorporated into Hinton. The Analyser is used to display the transients of the neural network parameters. It is discussed separately in the next section (see sec. A.3.4).

Evaluation

During the evaluation phase, the analysis functions described above are not available. The only additional display function is the *LastNetDialogue* (see fig. A.5). This GUI shows the current set of evolution process parameters and the fitness of the last individual in comparison to the best individual of the current population. This allows the monitoring of the effect of changes made to the fitness-function coefficients, and the relation of the generated fitness value to the desired behaviour of the robot.

Acknowledgements

The author wants to especially thank Björn Mahn for his contributions to Hinton. He implemented early version of the network editing GUI, the network exporting, and the dynamical loading of fitness-functions and communication classes, and also contributed to subsequent refactoring of the tool, bug fixes and functionality extensions, which substantially increased the overall quality of Hinton.

A.3.4 Analyser:

Every evolved controller is analysed with respect to behaviour-relevant dynamical properties. The *Analyser* is a tool to visualise the structure and the transient dynamics of a recurrent neural network while it controls a robot in the sensori-motor loop.

Visualisation Methods

The Analyser offers two different visualisation methods, a graphical visualisation of the network structure and its current state, and transient plots of network properties, which are comparable to EEG plots (see fig. A.7). Both methods are discussed in the following paragraphs.

Graph visualisation: The graph visualisation displays the structure of a recurrent neural network. In contrast to the matrix visualisation, it is more intuitive. An important advantage is that the network can be visually organised interactively to emphasise the relationship and role of a particular neuron or substructure of the neural network.

When the network controls a robot, the colour coding of a neurons indicate its current output value. Blue indicates the lower domain range and red the upper domain range. For the

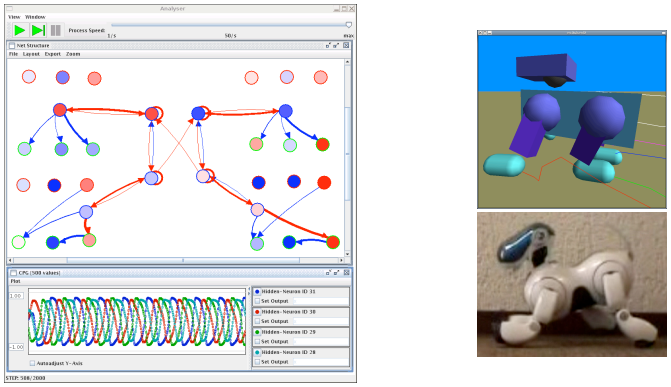


Figure A.7: ANALYSER. This figure shows the Analyser with an example of an application. In this case the Analyser is used together with a YARS-simulation of the Aibo platform (Markelić & Zahedi, 2007; Markelić, 2005). Left: Analyser. The Analyser plots the structure of a recurrent neural network as a digraph. The colour coding of the neuron represents the layer and the current output. Neurons with a red/green/blue outline are input/output/hidden neurons, respectively. Blue fill colour indicates low, red fill colour high output. The colour coding of the synapse represents the synaptic sign: blue refers to an inhibitory, red to an excitatory synapse. The width of the line representation indicates the synaptic strength. Below the network structure, the transients of the central pattern generator are plotted. The depicted network is a neural implementation of a fast quadruple weakling behaviour for the Aibo robot, visible on the lower right-hand side of the figure. The upper right-hand side of the figure shows the YARS simulation of the Aibo robot.

hyperbolic tangent, this is blue for -1 and red for 1. For the standard sigmoid, it is 0 and 1, respectively. The outline of the neurons indicates its layer. The colour coding is identical to the colour coding in the network editing tool discussed in the previous section (see fig. A.5). The outline colour red/green/blue relates to an input/output/hidden neuron, respectively.

The colour coding of the synapses corresponds to the sign of the synaptic weight. Blue synapses are inhibitory, red synapses excitatory. The width of the synapse corresponds to the absolute value of the synaptic strength. Thicker synapses indicate a stronger connection. In the case of dynamically changing synapses, the width of the synapse varies with the dynamics of the synaptic weight.

Plot visualisation: The graph visualisation, discussed above, displays the structure of a recurrent neural network and the states of the neurons and synapses. Changes of the neuron and synapse state are difficult to follow over time by the graph methods. Therefore, the *Analyser* provides a transient plot method, which is referred to as plot in the following section.

A plot displays the transients of any set of network properties (see fig. A.7). Displayable network properties are the output of a neuron, and the synaptic strength of a synapse.

The plot provides indications about the dynamics generating the behaviour of the robot. Examples for such dynamical properties are oscillations and hysteresis effects (see sec. 3.2). Together with lesion experiments and dynamical systems analysis of the recurrent neural network, the structure–function relationship can be extracted.

The current plot can be saved as a white-space-separated list for further analysis using third party software tools.

Acknowledgements

The Analyser was written by Michael Rosemann as part of his master thesis (Rosemann, 2004).

A.3.5 YARS

The software YARS (*yet another robot simulator*), is a mobile robot simulator which includes a physics engine (Zahedi et al., 2007), referred to as a physical (robot) simulator for short. It provides an evaluation environment for evolutionary robotics.

Two different approaches are use to evaluate the individuals of a population in artificial evolution: evaluation on the physical platform and evaluation in simulation. Both methods are found and discussed in literature (Nolfi & Floreano, 2000; Hülse et al., 2004; Floreano, 2000; Chaumont et al., 2007; J. F. Walker & Oliver, 1997; J. Walker et al., 2003; Jacobi et al., 1995; Harvey et al., 1997, 2005; Jin & Branke, 2005; Miglino et al., 1995; Lipson, 2005; Pollack et al., 1999; Pollack & Lipson, 2000; Nolfi et al., 1994). In this work, a simulator is used for the evaluation during the artificial evolution. There are four main reasons for using a simulator:

1. During evolution, hardware-damaging behaviours are likely to occur.

2. A simulator can run faster than real-time, so that the required time to evolve a controller is reduced.

3. The state of the simulator can be precisely set by the experimenter, so that each individual within a generation has the same initial conditions. This increases the comparability of the fitness values of the individuals in a population.

4. For the analysis of the behaviour-relevant dynamics, it is essential to know and control all the parameters (morphology, environment, etc.). This is easily achieved through the use of a simulator.

However, these advantages only hold, if the gap between simulator and reality does not lead to significant differences of the behaviour of a controller in simulation and on the physical platform. Closing the gap between simulation and reality stands in contrast to simulation speed, the second advantage listed above. The faster a simulator is, the less precise it is, which increases the gap between simulation and reality.

In contrast, for other experiments, a physical simulation is required. One example for such an experiment is behaviour-control for walking machines (see fig. A.7). Without the experience of falling, a system cannot learn to walk. Other examples (see fig. A.19ff) of experiments that require a physical simulator are gravity driven systems, such as micro.adam and micro.eva (Popp, 2005; Wischmann et al., 2005) .

For this reason a new simulator based on ODE (Open Dynamics Engine) was built. ODE is an open source physics engine, initially developed by Russell Smith (Smith, 2005). It is fast, numerically stable and well-documented. Numerically stable in this case means, that the simulation will not crash, if the internal physics runs into computational singularities. Typically, these singularities result in exploding bodies or bodies that catapult from location to location. One case in which these singularities might occur is if bodies collide at very high speed. For

evolution, this type of simulator behaviour is ideal, because it indicates hardware-damaging behaviour. Such a behaviour is not desired, and can easily be caught in the software and punished by the fitness-function. For the individual which follows, the running simulation is simply reset. This is only possible, if the simulation itself does not crash, even if the internal physics leads to uncontrollable results.

This advantage comes with a trade-off. ODE uses a first-order or Euler integrator (Strogatz, 1994), which is fast and numerically stable, but not precise in calculating the physics. For actuators, ODE uses a linear model. The maximal available force is used to linearly increase the velocity of the actuator until the desired velocity is reached. More realistic motor models are not included, and need additional implementation. The only available friction model is Coulomb friction. These properties of ODE result in a fast, but imprecise simulation. With respect to the requirements stated earlier, this means that there is a gap between simulator and reality, which must be considered.

In the approach followed here, this is not a drawback. The goal is to generate controllers, which are robust with respect to the given hardware. There are two approaches to achieve this kind of robustness, closed-loop control in the sensori-motor loop, and evolving the controller on an abstraction of the target platform. How these two methods account for robustness is explained in the next.

Consider a wheel driven robot with an obstacle-avoidance behaviour. If the motors of the physical robots are slower compared to the simulated motors, the turning behaviour is slower. Therefore, the robot will turn slower when an obstacle is detected by the sensors. Because it turns slower, the sensor stimulus is present for a longer duration within the system. The robot will turn for a longer period of time. The overall behaviour, obstacle-avoidance, is preserved, although it is not equivalent to the simulated behaviour in all aspects.

This does not hold for all configurations. If the motors of the physical robot are much faster, then the robot will turn faster. Because of possible inertia, it might turn too far, such that the obstacle is present again. This means, that the well-performing obstacle-avoidance behaviour in simulation, shows a rotating behaviour on the physical robot.

This example demonstrates an important aspect of closing the gap between simulator and reality. The simulator does not have to provide a precise model of the physical properties of the system, but it should capture the principles of the target hardware. What a sufficient abstraction is, can only be determined when the physical and simulated robots are compared on the behaviour level. This does not mean that the artificial evolution is run until a final solution is obtained, and then ported to the target platform, but rather that the intermediary results are validated against the target platform periodically during the evolution. This needs to be done, until the principles of the target platform are captured in the simulator (see fig. 3.3).

With this approach, the advantages discussed previously, namely, decreased evolution time, preventing hardware from damage, and increased comparability of individuals, hold true and justify the use of a simulator for the evaluation during the artificial evolution.

Different simulators were written for different experiments (see fig. A.19ff), basically from scratch. An error in one simulator was likely to be reproduced in the following simulators. Also, for each experiment, the communication to Hinton was written from scratch, and especially adapted to the simulated robot platform. A walking machine with many degrees of freedom requires a different communication compared to a wheel-driven system. Nevertheless,

the programmed simulations were necessary to gain enough experience in programming simulators based on ODE. This experiences finally lead to YARS, a general physical robot simulator based on ODE.

As stated in the specifications (see sec. A.2), ISEE should require only minimal — if any — programming knowledge. This must hold true in particular for the simulator YARS, because setting up a simulation is essential for the experiment. This means that no programming in C/C++ and no compilation by the user should be necessary in order to set-up a new experiment. We chose the Extensible Markup Language (XML) (Arbouzov et al., 2004) as the current user interface. This was done for four reasons:

1. XML can be written with almost any editor.
2. If the keywords of the description language are chosen with care, it is human readable and does not require any knowledge in programming languages such as C/C++.
3. With XSLT (Extensible Stylesheet Language Transformations) (J. Clark et al., 1999) there exists a method to transform a description form e.g. VRML (web3D Consortium, 1997) or X3D (web3D Consortium, 2006) into an XML file for YARS, and vice versa.
4. There are many, good open-source and freeware, context-sensitive XML editors available for every operating system. Writing a general graphical editing tool for YARS, based on the information available in the XML Schema grammar is possible without much implementation effort (currently a work in progress).

The goal was to provide a minimal and human readable interface. The description languages VRML and X3D were not chosen, because they are too extensive in their possibilities, and require advanced programming knowledge. A possible advantage of VRML and X3D is the availability of a variety of graphical development tools, but as VRML and X3D are designed to specify graphical scenes, they do not provide any possibility to describe robot sensors, such as an infra-red proximity sensor. This requires heavy extensions in order to serve as a general robot description language, which would make the description languages proprietary, such that graphical development tools could not be used any more. Hence, another language was developed, which is minimal but rich enough to describe an experiment and includes description elements for actuators and sensors: the Robot Simulation Markup Language (RoSiML) (Zahedi et al., 2005).

The first version of RoSiML was used in the German Research Foundation Priority Program 1125[†] as a general description language for all used simulators. It was agreed on RoSiML as the common description language so that experiment descriptions would be exchangeable between project members, independent of the simulation system. An overview of RoSiML is available in the appendix (see app. B.7).

RoSiML offers the possibility of a comprehensive experiment description. One RoSiML file includes the simulator set-up (camera position, window size, etc.), the environment description, and the description of the robots. In particular, the description does not limit the number of robots, such that a simulation of swarms is possible.

[†]German Foundation Priority Program 1125 "Cooperating teams of mobile robots in dynamic environments" (DFG-SPP 1125 "Kooperierende Teams mobiler Roboter in dynamischen Umgebungen")

For each robot, YARS automatically creates a communication port and exchanges the sensor and motor configuration with Hinton. Although this communication protocol was programmed for Hinton, it is open, so that other control programs can connect to YARS. Hinton and YARS typically run locally on one machine. Local communication is lossless, so that there is no need for the TCP/IP flaw control. Hence, for maximal communication speed, UDP is the chosen communication protocol.

Other simulators and physics engines, such as Havok (Havok.com Inc, 2006), Vortex (CM-Labs Simulations Inc, 2006), Webots (Michel, 2006), Adams (Software, 2006), Darwin2k (Leger, 2006) etc. were reviewed by the author but not chosen because of either cost, speed, or missing usability for evolutionary robotics.

Acknowledgements

Steffen Wischmann contributed to the implementation of the early version of the YARS core. The generic communication interface between YARS and Hinton was designed and implemented by Björn Mahn. Verena Thomas implemented the dynamical loading of control programs and the virtual camera sensor. Arndt Twickel wrote a tutorial and was an extensive tester. He became the co-author of YARS.

A.3.6 Brightwell

The previous sections discussed the software tools *Hinton* and *Analyser* with respect to the analysis of the dynamics of a recurrent neural network in the sensori-motor loop.

A recurrent neural network, understood as a dynamical system, can show a rich reservoir of dynamical properties, such as fixed point attractors, oscillations, quasi-periodic, and chaotic attractors. From this reservoir, only a subset of properties is selected if the recurrent neural network controls a robot in the sensori-motor loop.

If the dynamical reservoir of the recurrent neural network is known, the transients shown by the *Analyser* are better understood, if they are related to the reservoir by taking the sensor information into account.

The *Brightwell* tool is designed to visualise the dynamical properties of a recurrent neural network decoupled from the sensori-motor loop. It provides methods to visualise different dynamical properties. In a similar manner to the concept of fitness-function and communication classes in Hinton, Brightwell provides functionality to add new visualisation methods with minimal implementation effort (see app. B.5).

The following section describes the visualisation methods which are currently provided by Brightwell.

Brightwell tools

The available tools are divided into two groups, those which process recurrent neural networks, and those which visualise the behaviour of other dynamical systems. The second group is added for educational purposes. The intention is to provide an interface, such that students can implement any dynamical system and experiment with its parameters. This enables insights

to be gained on how the plots are generated, what they show, and the behaviour of dynamical systems.

The discussion begins with the tools for the analysis of the dynamical properties of recurrent neural networks. It is followed by the description of visualisation methods for other dynamical systems. Each tool description begins with the tool name which is given in bold characters.

Bifurcation Diagram: A bifurcation diagram shows the converged state of a recurrent neural network for a range of one of the system parameters. The properties of such a diagram are shown in figure A.8 by means of the single chaotic neuron (Pasemann, 1997b), which is given by the following equation:

$$a(t+1) \quad = \quad \Theta + \kappa a(t) + w\sigma(a(t)).$$

In this case, the bias value Θ of the system is varied over the interval $\Theta \in [0, 8]$. The bifurcation diagram for this example (see fig. A.8 centre) shows how the dynamics of the single chaotic neuron differ for different setting of Θ. For $\Theta \in [0, \approx 1.5]$ the plot shows that the system has a fixed point attractor. For values of $\Theta \in [\approx 1.5, \approx 4]$ a period-doubling route to chaos and finally chaos is observed. The state for which the quality of the attractor of the system changes, is called a bifurcation point. In this example, the plots show such a bifurcation from a fixed point attractor to a period-2 attractor for $\Theta \approx 1.5$.

A bifurcation diagram is generated as follows (see alg. 4). The dynamical system is set to its initial state. This can be a random initialisation or a user-defined state. For this initial state, a given number of convergence iterations are performed. After the convergence iterations, a given number of plot or draw iterations follow. This means, that after each draw iteration the current state of the system is plotted. This way oscillations or chaos, for which the system takes different values for iterations for one specific parameter configuration is visualised. An example is the period-2 oscillation of the single chaotic neuron for $\Theta \geqslant 6$ (see fig. A.8). After the draw steps are performed, the parameter is increased according to the defined parameter domain (x-range). Depending on the user's choice, the system is either reinitialised to its user-defined initial condition, reinitialised to randomised values, or kept in the current state. For the new parameter setting and the selected initial state for the system, the convergence and plot iterations are repeated. In order to visualise hysteresis or co-existing attractors, the system must be processed at least twice, once from the lower border to the upper, and once from the upper to the lower border.

For recurrent neural networks, the parameters open to variation are the bias term of a neuron and the weight of a synapse. The parameters of the other neurons and synapses are kept constant. The *Bifurcation Diagram* tool provides a graphical user interface, such that the user can set the algorithms parameter (see fig. A.8). The standard visualisation displays the bifurcation diagram of the average output over all neurons. The average may hide the dynamical properties of isolated neurons, which can differ significantly. Therefore, any set of neurons can be selected and displayed in isolation.

In some cases, it may not be possible to resolved whether the system is quasi-periodic or chaotic simply through visualisation. A good indication method is the Lyapunov exponent (Thomson & Stewart, 2002; Strogatz, 1994). If the exponent is negative, the system is neither

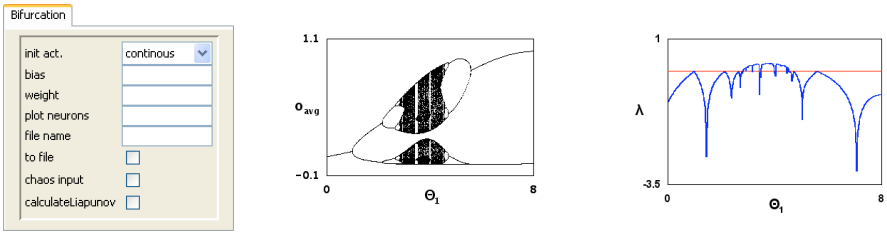


Figure A.8: Bifurcation Plot Tool. From left to right: Tool Parameter GUI, an example plot. The GUI provides an interface to the tool parameters, which are the initial activation function, the parameter open to variation, selection of neurons to be additionally plotted, file output, chaotic input, and calculation of the Lyapunov exponent. Possible initial activation functions are continuous, random, and resetted. This function determines how the neural network is reset for each new parameter configuration (see alg. 4). For the tool, either a bias of a neuron or one synapse is varied over the x-range. The chaotic input can be used to analyse the convergence behaviour system. The Lyapunov exponent can be calculated for the entire network and is displayed in an additional window. The example shows a single chaotic neuron with $\kappa = 0.6, w = -16$, and the standard sigmoid transfer-function (Pasemann, 1997b).

chaotic nor quasi-periodic. If it is zero, the system is quasi-periodic, while a positive exponent indicates a chaotic behaviour. The Lyapunov exponent is drawn in addition to the bifurcation diagram (see fig. A.8 left).

The last option allows the logging (to a file) of the states of the recurrent neural network as a white-space-separated list of values. This file can be used with third-party tools.

Iso-periodic Plot: An iso-periodic plot is a sketch of the dynamical properties of a system seen in figure A.9 left as it is applied to a neuron-module, in which the colours encode the periodic attractors existing for corresponding points in a two-dimensional subspace (Pasemann, 2002).

The iso-periodic plot is generated as follows (see alg. 5): For each setting of the two selected parameters, a system is reinitialised according to the initial state setting. As for the bifurcation diagram a defined number of convergence iterations are performed. After convergence, the system is iterated until the current state of the system is repeated. The equality of a state is defined by the maximal difference (see fig. A.9) between the values of two consecutive states. If it difference is small enough, the states are assumed to be equal. The period of the attractor is then given by the number of iterations until the periodic point is repeated. A maximal number of iterations, and therefore, maximal displayed period is set. All higher periods are coloured alike and denoted with "ch" in the legend, which indicates chaos. This is, however, only a place-holder for higher periods and does not mean that the corresponding parameter set leads to chaos. It also occurs when the number of convergence iterations is set too small during the calculation of the period, such that the attractor is not reached.

The tool allows the generation of plots for any two combinations of parameters, bias–bias, bias–synapse, synapse–synapse.

It should be noted that hysteresis cannot be visualised with the current version of the *Iso-periodic Plot*. For that, at each coordinate, the algorithm has to be executed at least twice for

Algorithm 4 Bifurcation Diagram.

Require: $N(t) := \{n_i(t)\}_{i=0,\dots,|N|-1}$ {set of all neurons}
Require: $C :=$ number of convergence iterations
Require: $D :=$ number of draw iterations
Require: $x_{min}, x_{max}, \Delta x$, the range and step size
for $x = x_{min}$ to x_{max} step Δx **do**
  for all $n_i(0) \in N(0)$ **do**
    reinit($n_i(0)$) {depending on the selected method}
  end for
  for $i = 0$ to C **do**
    process($N(i)$) {One iteration of the network (see sec. A.3.1)}
  end for
  for $i = 0$ to D **do**
    process($N(i)$) {One iteration of the network (see sec. A.3.1)}
    $\overline{o}(i) = \frac{1}{N} \sum_i^{|N|} o_i(i)$
    plot $(x, \overline{o}(i))$
    for all selected $n_j \in N(i)$ **do**
      plot$(x, o_j(i))$
    end for
  end for
end for

all attractors with a period of one. The fixed point attractor of the system must be calculated for an upper and lower setting of the initial state. Comparing the resulting fixed points shows hysteresis, when they are unequal.

Transient Plot: The bifurcation and iso-periodic plot methods, described above, visualise the asymptotic set of states of a recurrent neural network for varying parameters. The visualisation methods do not provide any insights about the transient behaviour of the system and they do not give any information about the trajectory or orbit, as in the case of the bifurcation diagram, the visualisation does not provide any information about the order of the states. Consider a quasi-periodic attractor or higher-order attractor (larger than three). In this case, the bifurcation diagram does visualise the elements of the asymptotic set, but not the sequence in which a neuron iterates over them. The iso-periodic plot only visualises the quality of the attractor, its set, but not its orbit.

The *Transient Plot Tool* displays the transients and orbits of the neuron outputs for any set of neurons (see fig. A.10). For a selected initial condition, the recurrent neural network is iterated according to the number of convergence iterations specified by the user. The tool's GUI enables the selection of the iteration step from which the system state is plotted (see alg. 6). This can be used to visualise the transient, and transient length of the system, while approaching the attractor from a specific initial condition. Additionally, it may show only the orbit of the system once the attractor has been reached, e.g. the output behaviour of a quasi-periodic neuron.

The transients can be written as a white-space-separated list to a file for further analysis

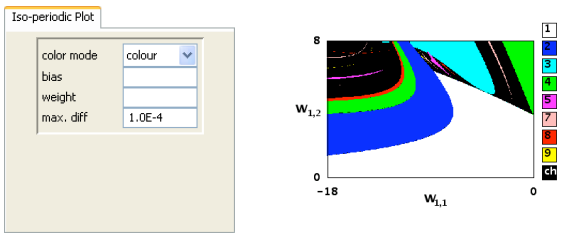


Figure A.9: ISO-PERIODIC PLOT TOOL. From left to right: Tool Parameter GUI, an example plot. The GUI provides an interface to the tool parameters. Parameters are the colouring method, the parameters open to variation, and the maximal difference for the period calculation. The colouring method is either colour or Gray values. Two parameters need to varied for the Iso-periodic plot. This can be any possible combination of bias values and synaptic strength. If both parameter are bias or synapse, the must given as comma separated list. For the maximal difference the reader is referred to the text and the described algorithm (see alg. 5). The example is the iso-periodic map of the two neuron module with $w_{21} = -6, \Theta_1 = -3, \Theta_2 = 4$, and the standard sigmoid transfer-function (Pasemann, 1997b).

with third party tools.

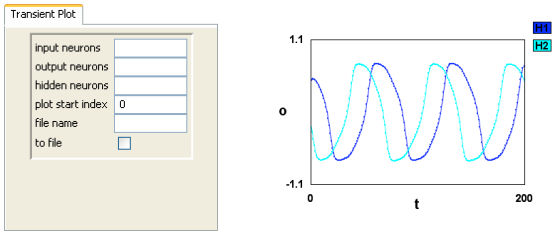


Figure A.10: TRANSIENT PLOT TOOL. From left to right: Tool Parameter GUI, an example plot. The GUI provides an interface to the tool parameters. Parameters include the neurons that will be plotted, the start index of the plot, and a file-name of the white-space-separated value output in ASCII format. The start index corresponds to the n-th iteration from which the plot begins. The number of iterations plotted is determined by the convergence iterations setting in the Tool Parameter Panel. The example plot shows the output of the two neurons of a SO(2)-Network (Pasemann, Hild, & Zahedi, 2003).

First-Return Map: The First-Return map is also known as a *Poincaré map* (Katok & Hasselblatt, 1999; Strogatz, 1994). Other authors distinguish between a *time-T map* and a Poincaré map (Alligood et al., 1996). The principle of both is very similar but differ in detail. For a *Poincaré map*, the complex structure of an attractor of a continuous-time dynamical system is captured by the intersections of its orbit with a sub-manifold (the Poincaré section). For a three-dimensional system, this is a plane. The motion will appear as a sequence of points, and e.g. a periodic attractor will appear as a repeating sequence of points while a quasi-periodic attractor will appear as a closed orbit on this plane (Lakshmanan & Rajaseekar, 2003). The difference between a Poincaré map and a time-T map is that in the first case, the plane must not necessarily capture points which are equally-spaced in time whereas the time-T map is

Algorithm 5 Iso-periodic Map.

Require: $N(t) := \{n_i(t)\}_{i=0,\ldots,|N|-1}$ {set of all neurons}
Require: $C :=$ number of convergence iterations
Require: $maxPeriod :=$ the maximal tested period
Require: $maxDiff :=$ upper limit to consider to states as equal

```
for x = x_min to x_max step δx do
  for y = y_min to y_max step δy do
    for all n_i(0) ∈ N(0) do
      reinit(n_i(0)) {depending on selected method}
    end for
    for i = 0 to C do
      process(N(i)) {(see sec. A.3.1)}
    end for
    for i = 0 to maxPeriod do
      process(N(C + i)) {(see sec. A.3.1)}
      Δ_i = Σ_j^N |o_j(C + i) − o_i(C + i − 1)|
      if Δ_i < maxDiff then
        plot (x, y, i) {i denotes the period}
        break
      end if
    end for
  end for
end for
```

stroboscopic, mapping the values of the variables at equal time intervals (Alligood et al., 1996).

For discrete-time systems, as they are used here, the first-return map is constructed by plotting the state $x(t)$ against the state $x(t+1)$, which relates to a stroboscopic capturing. The *First-Return Map* tool plots the average over all outputs $(\overline{o(t)}, \overline{o(t+1)})$, or the output of any tupel of selected neurons $(o_i(t), o_j(t+1)), i, j = 0, \ldots, |N| - 1$ in the two-dimensional plane.

In order to visualise co-existing attractors, the tool restarts the system from random initial conditions. For each new initial condition, a different colour is used to plot the attractor. This allows co-existing attractors to be distinguished visually.

Peek Plot: The *Peek Plot* tool plots the transient response of a recurrent neural network to a peek stimulus (see fig. A.11). The stimulus is represented by a varying bias value for a given neuron of the neural network. The peek is defined by a minimum and maximum value, the number of peek iterations, and a function determining the shape of the peek. Implemented functions are peek (rectangle shaped pulse) and ramp (trapezoidal shaped pulse) (see fig. A.11b).

Firing Pattern Plot: The *Firing Pattern* tool is used to visualise the periodic behaviour of a recurrent neural network (see fig. A.12). In contrast to the Transient Plot tool, only the sign of the output values are plotted, and not the actual neuron output values. This enables the visual determinations of the period and the phase-shift of the neuron output patterns.

Algorithm 6 Tranisent Plot.

Require: $N(t) := \{n_i(t)\}_{i=0,\dots,|N|-1}$ {set of all neurons}
Require: $C :=$ number of convergence iterations
Require: $P :=$ plot index
Require: $P < C$
1: init($n_i(t)$) {depending on selected method}
2: **for** $i = 0$ to P **do**
3: process($N(i)$) {(see sec. A.3.1)}
4: **end for**
5: **for** $j = 0$ to $C - P$ **do**
6: process($N(j)$) {(see sec. A.3.1)}
7: **for** $i = 0$ to $|N|$ **do**
8: plot $(j, o_i(t))$
9: **end for**
10: **end for**

Centrality Tool: The *Centrality Tool* visualises the centrality of each neuron within a neural network. The centrality measurement is known from the analysis of social networks (Brandes, 2000; Hanneman & Riddle, 2005), in which the centrality defines the importance of an actor within a group of actors. The known measurements are:

- Degree centrality.
- In-degree centrality.
- Out-degree centrality.
- Closeness centrality.
- Betweenness centrality.
- Flow-Betweenness centrality.

The *Centrality Tool* enables the selection of any set of the methods listed above (except flow-betweenness). An overall classification is given by the average of the selected centralities. The results can be displayed visually (see fig. A.13), or written as a table to an ASCII-file for further analysis using third-party tools. In the following paragraphs, the centrality methods, listed above, are detailed.

Degree / In- / Out-degree centrality The three degree centralities measure the different aspects of the connectivity of a neuron in a neural network. The *degree centrality* takes into account all synapses connected to a neuron, whereas the *in-degree* and *out-degree* centrality only take into account the number of incoming and outgoing connections, respectively.

A.3.3 Definition: Degree / In- / Out-degree centrality

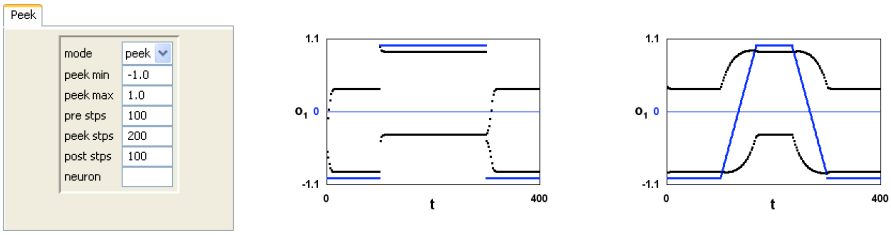


Figure A.11: PEEK TOOL. From left to right: Tool Parameter GUI, two example plots. The GUI provides an interface to the tool parameters. Parameters include the peek method, minimal and maximal peek value, pre-peek-, peek-, and post-peek-steps, and the neuron to be varied by the peek. Two peek methods are currently implemented and shown here. For both plots the same network is used. An input neuron is connected to an output neuron. The synaptic strengths are $w_{21} = 1, w_{22} = -1.5$. The peek (centre) changes the activation of the input neuron from the minimal value (-1) to the maximal value (+1) as a step function. The ramp (right) changes the activation of the input neuron between the minimal value (-1) and the maximal value (+1) through a linear increase and decrease. The plots show the difference in behaviour for the two methods.

Let $G = (V, E)$ be the graph describing the neural network. Then the three degree centralities are then defined as

$$\begin{aligned} deg(v) &:= |\{u \in V | (u,v) \in E \vee (v,u) \in E\}| \\ indeg(v) &:= |\{u \in V | (u,v) \in E\}| \\ outdeg(v) &:= |\{u \in V | (v,u) \in E\}| \\ C_{deg}(v) &= \frac{deg(v)}{|V|^2 - 1} \\ C_{indeg}(v) &= \frac{indeg(v)}{|V|} \\ C_{outdeg}(v) &= \frac{outdeg(v)}{|V|} \end{aligned}$$

Closeness centrality The description of the *Closeness centrality* concept is given here. Consider two different synaptic pathways. Let t be a destination neuron, and r and s two source neurons. If the synaptic pathway $\overline{rt}$ is shorter than the synaptic pathway $\overline{st}$, then the output of the neuron r is considered to have a higher influence on t. The closeness centrality measures the degree of shortest pathways a neuron has within a network.

A.3.4 Definition: Closeness centrality
Let $G = (V, E)$ be the graph describing the neural network. Then the closeness centrality is defined as

$$\begin{aligned} dist(v,t) &= \textit{length of the shortest path between } v \textit{ and } t \\ C_C(v) &= \sum_{t \in V \setminus \{v\}} dist(v,t) \end{aligned}$$

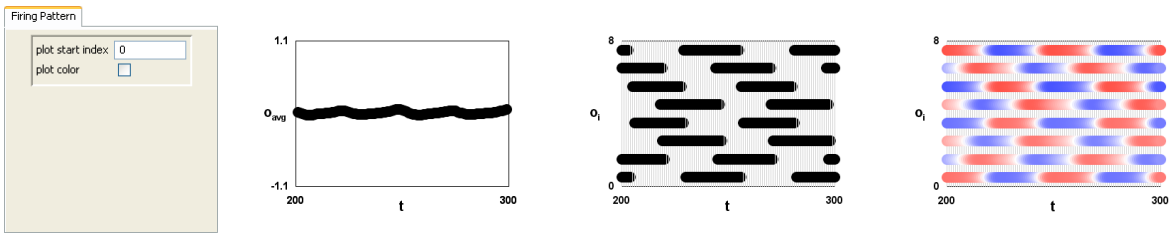


Figure A.12: Firing Pattern Tool. Firing Pattern Tool. From left to right: Tool Parameter GUI, three example plots. The parameters for the tool are the start index from which the plot starts, and whether colour should be used. The neural network is iterated according to the convergence iterations setting. The plot starts when the network has been iterated *index* times. The example plot has been created with two coupled SO(2)-networks (Pasemann, Hild, & Zahedi, 2003) which were used as a CPG for an Aibo (Markelić & Zahedi, 2007; Markelić, 2005) and an Aibo-like morphology (Klaassen et al., 2004). The centre plot shows the average over all neurons. The two plots on the right-hand side show the firing pattern for all neurons. The neurons are ordered from bottom to top with increasing neuron index. The x-axis is the iteration step of the recurrent neural network. White-space denotes an output below 0 for the hyperbolic tangent transfer-function and below 0.5 for the standard sigmoid transfer-function (in the black and white images). Black dots denote an output above 0 or 0.5, respectively. In the coloured image, red corresponds to a high activation (+1) and blue to a low activation (−1 in the case of the hyperbolic tangent, 0 in the case of the standard sigmoid).

$$C_C(v) = \begin{array}{l} \textit{smaller values indicate higher centrality} \\ \dfrac{1}{\sum_{t \in V \setminus \{v\}} dist(v,t)} \\ \textit{higher values indicate higher centrality} \end{array}$$

Betweenness centrality The *Betweenness centrality* measures how many shortest paths between any two neurons s, t pass a neuron v. The concept originates from communication pathways between actors. The larger number of shortest communication pathways between two actors are controlled by a third actor, the more powerful the third actor is.

A.3.5 Definition: Betweenness centrality
Let $G = (V, E)$ be the graph describing the neural network. Then the betweenness centrality is defined as

$$\begin{aligned} \sigma_{st} &= \textit{number of shortest paths between } s \textit{ and } t \\ \sigma_{st}(v) &= \textit{number of shortest paths between } s \textit{ and } t \textit{ over } v \\ C_B(v) &= \sum_{s \in V \setminus \{v\}} \sum_{t \in V \setminus \{v,s\}} \frac{\sigma_{st}(v)}{\sigma_{st}} \end{aligned}$$

Flow-Betweenness centrality The *Flow-Betweenness centrality* measures how much information or signal flow passes an actor, with respect to the maximum flow (Cormen et al., 1990) in the network. The maximal flow between two neurons can be defined by the strength of the synapse connecting them.

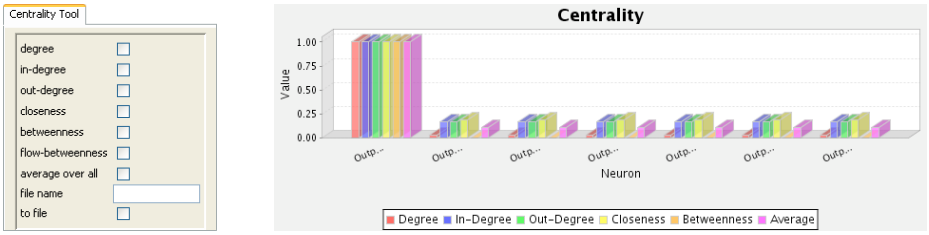


Figure A.13: CENTRALITY TOOL. From left to right: Tool Parameter GUI, an example plot. The GUI provides the list of available centrality measurements. From this list any subset can be selected. In addition the overall average over the selected set of centralities can be chosen as well. If a file-name and the check box are filled, the results are written to an ASCII-File. On the right-hand side, an example is given for a star network. This is a configuration, in which one neuron is bidirectionally connected to every other neuron. The other neurons have no further connections. The plots was generated with the JFreeChart library (Viklund, 2006).

A.3.6 Definition: Flow-Betweenness centrality

Let $G = (V, E)$ be the graph describing the neural network. Then the betweenness centrality is defined as

$$\begin{aligned} f_{st} &= \textit{maximum flow between } s \textit{ and } t \\ f_{st}(v) &= \textit{maximum flow between } s \textit{ and } t \textit{ over } v \\ C_F(v) &= \sum_{s \in V \setminus \{v\}} \sum_{t \in V \setminus \{v,s\}} \frac{f_{st}(v)}{f_{st}} \end{aligned}$$

This centrality measurement is currently not included in Brightwell.

First-Return Map (File): The *First-Return Map (File)* tool (see fig. A.14) is similar to the First-Return Map tool described above. In contrast, this tool does not generate the data by processing a recurrent neural network, but takes the data from an input file. This file is a white-space-separated table of values. The user specifies which column defines the x- and y-axis. The tool plots the y value of the line $i + 1$ against the x value of the line i.

Magnetic Pendulum: The *Magnetic Pendulum* tool is the first educational tool found in the Brightwell application. The magnetic pendulum is a chaotic dynamical system. Two or more magnets are placed on a plane around the centre of a swinging pendulum. The pendulum is then displaced from its centre and released to swing. The trajectory is affected by the forces of the magnets. At which magnet the pendulum finally rests is highly dependent on the initial position and initial velocity of the pendulum. Small variations of the initial condition result in a different outcome. The magnetic pendulum is a good visualisation of fractal basins (see fig. A.15). It is given by the following equations take from Dickau (2008):

$$\ddot{x}(t) + R\dot{x}(t) - \sum_{i=1}^{S} \frac{x_i - x(t)}{\left(\sqrt{(x_i - x(t))^2 + (y_i - y(t))^2}\right)^3} + Cx(t) = 0$$

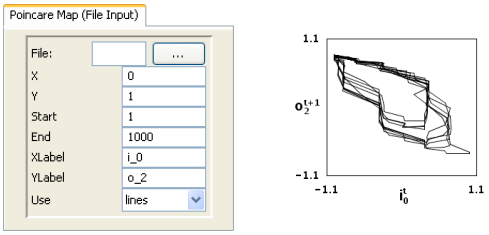


Figure A.14: First-Return Map (File). From left to right: Tool Parameter GUI, and one example plot. The parameters define which columns of the input file are selected for plotting (X/Y), which rows are selected (start/end) and how the axes will be labelled. The last parameter determines whether lines or points are used for plotting. The plot on the right hand side shows a First-Return map for a Braitenberg controller with the SRN model (see sec. 5.2.2).

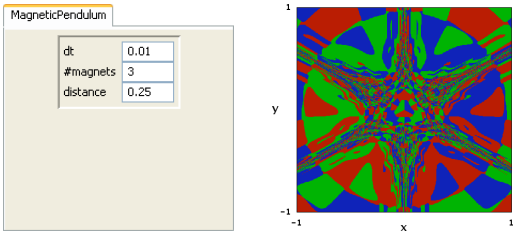


Figure A.15: Magnetic Pendulum. From left to right: Tool Parameter GUI, and one example plot. The tool parameters are the time step dt used by the Runge-Kutta method, the number of magnets, which are equally distributed around the centre, with the distance (3rd parameter) defined by the user. The example plot shows the basins for the parameters visible in the left figure. The plot was generated with 5000 convergence iterations. The colouring is determined by the magnet, with the smallest distance to the end position of the pendulum. At each coordinate, the pendulum is started with zero initial velocity: $\dot{x}(0) = 0, \dot{y}(0) = 0$.

$$\ddot{y}(t) + R\dot{y}(t) - \sum_{i=1}^{S} \frac{y_i - y(t)}{\left(\sqrt{(x_i - x(t))^2 + (y_i - y(t))^2}\right)^3} + Cy(t) \quad = \quad 0,$$

where S is the number of magnets, (x_i, y_i) the coordinate of the i-th magnet, $(x(t), y(t))$ is the position of the pendulum at time t, d is the distance between the pendulum and the plane containing the magnets, R the friction, and C the spring parameter. The parameters were chosen as follows:

$$\begin{aligned} R &= 0.2 \\ C &= 0.5. \end{aligned}$$

The Runge-Kutta method (Strogatz, 1994) is used to calculate the dynamics.

Standard Equations: The *Standard Equations* tool is the second educational tool. It is designed as a selection of dynamical systems found in standard literature (Strogatz, 1994; Thomson

& Stewart, 2002). The goal is to provide explorative access in order to understand the effect of various parameters on a dynamical system (see fig. A.16).

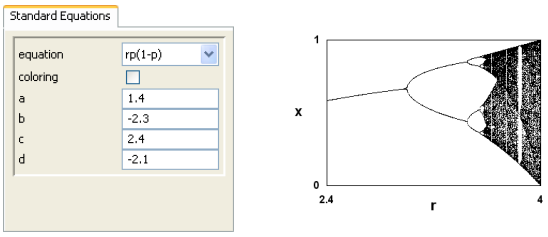


Figure A.16: Logistic map. From left to right: Tool Parameter GUI, an example plot. The GUI allows the selection of the standard dynamical system from a pull-down menu. The parameters a, b, c, d are intended to be used for newly implemented systems. The right-hand shows the resulting logistic map as an example.

Tutorial: The last tool to be introduced is the *Tutorial* tool. It is not an implementation of an analysis or visualisation method, but rather, it provides a template and description of the process which is required to add new visualisation tools in the Brightwell framework. The plot of the tool shows the various visualisation functions which are currently implemented. The interface, shown in figure A.17, displays the various user interface functions which have been implemented. The source code of this tool is documented, with a detailed description of the methodology for the creation of a new tool for Brightwell.

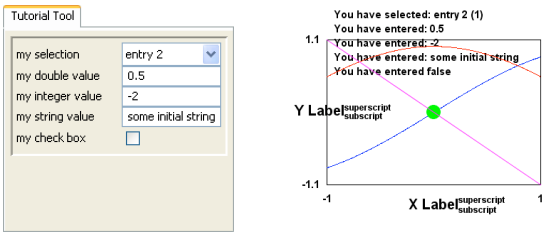


Figure A.17: Tutorial Tool. From left to right: Tool Parameter GUI, an example plot. The GUI shows the possible interface functionality (see app. B.5). The plot on the right-hand side shows the various draw methods which are implemented.

Approx Key Frames: The *Approx Key Frames* tool was contributed by Bernhard Klaassen. It was developed during a research project which was concerned with the automatic calculation of parameters of a recurrent neural network, so that the periodic behaviour would match a given set of key frames (Klaassen et al., 2004).

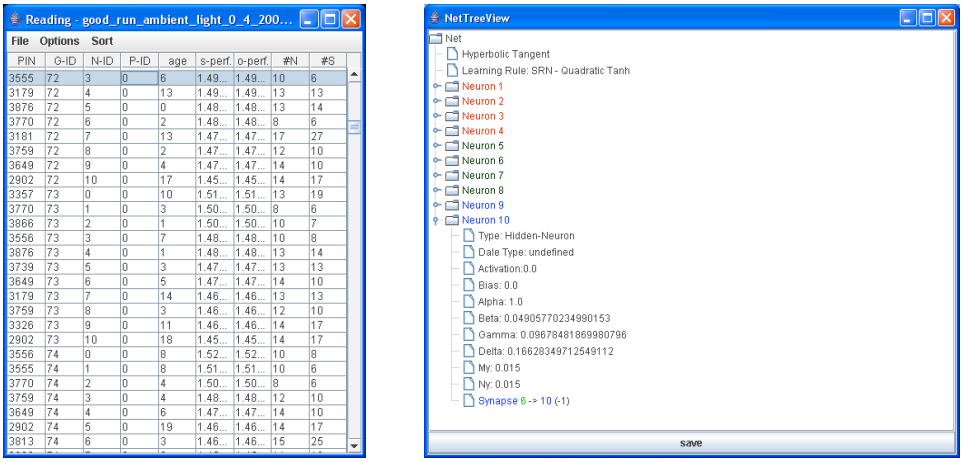


Figure A.18: READING. From left to right: Reading GUI, Individual Network View.

A.3.7 Reading

The tool *Reading* is designed to visualise an evolution log-file written by EvoSun. The log-file for an evolution typically grows very large. In order to find a specific individual, or generation, the user may use any editor and search for the generation. This method is no longer applicable for a file which already contains a few generations and individuals, as the clarity cannot be provided by a text editor. Therefore, a tool was created, which enables the visualisation of an evolution log-file (see fig. A.18). The GUI allows access to individual nets, and enables the sorting of the list of individuals by generation, identification number, fitness, and other network properties. Selected networks can be exported and used by other ISEE tools.

A.3.8 Newbury

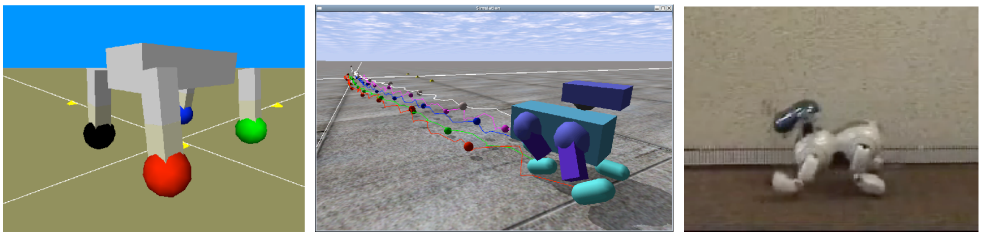
Newbury is a program designed to generate a new initial population file from a set of neural networks. The set of neural networks are individual XML files which are specified for Newbury through the command line. These files are then merged into one new evolutionary logging file, which can then be used by EvoSun as an initial population file.

A.3.9 Beaumy

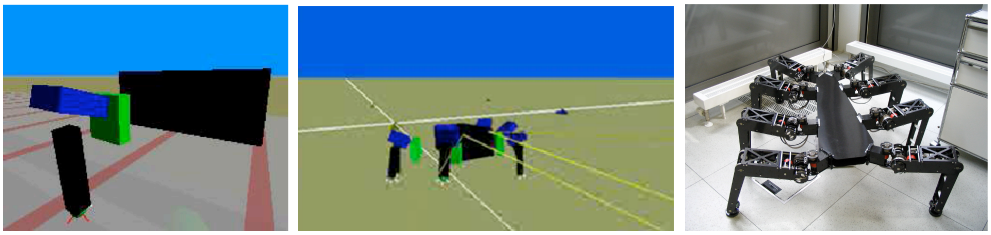
Beaumy is a tool which is used to extract the best individuals of an evolution log-file to form a new initial file for EvoSun. The parameters allow the user to select the number of generations, and the number of individuals from each generation for the new initial population file. The numbering of the generations is backwards, starting from the last generation in the evolution log-file.

A.4 Examples of projects implemented with the ISEE framework

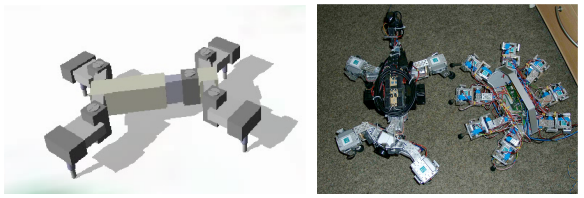
This section gives an overview of projects, which have been implemented with the ISEE framework. Not all projects were developed entirely within the framework. Some, such as the RunBot, Scorpion and others existed before ISEE, but simulations were programmed to either provide an evaluation environment for artificial evolution, or to test their controller structures in simulation for the relevant platform. Each project is described below along with a figure of the simulation and the physical robotic platform which was used.



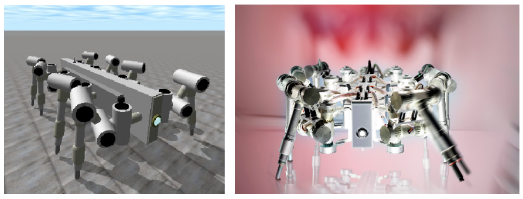
A.19.1 AIBO. Left: CALT (complex adaptive legged toys) project. The goal of the project was to develop a modular control concept of pluggable walking machines of arbitrary structure (Klaassen et al., 2004). Centre: Simulated Aibo in YARS. Right: the physical robot platform to which the evolved controller was ported. The projects goal was to understand and evolve a fast locomotion for a four-legged robot in the RoboCup domain (Markelić & Zahedi, 2007; Markelić, 2005).



A.19.2 OCTAVIO. Octavio (by Manfred Hild and Torsten Siedel) is a modular eight-legged walking machine. Each leg has its own power supply and control unit. Only low bandwidth communication is possible between the legs, which allows synchronisation. Left: First different controllers for single-leg control were evolved (Twickel & Pasemann, 2006) and then combined in a walking machine (centre). Right: The physical platform.

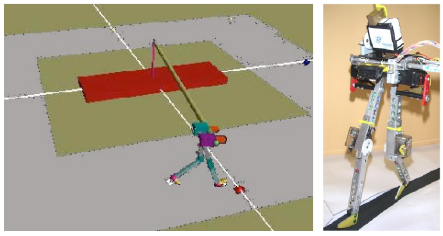


A.19.3 AMOS. AMOS is an umbrella term for a set of walking machines. Shown here are a four-legged and an eight-legged instance of AMOS. The four legged machine (by Poramate Manoonpong) is designed to evaluate the use of artificial whiskers as multi-sensori input (Manoonpong, 2007).

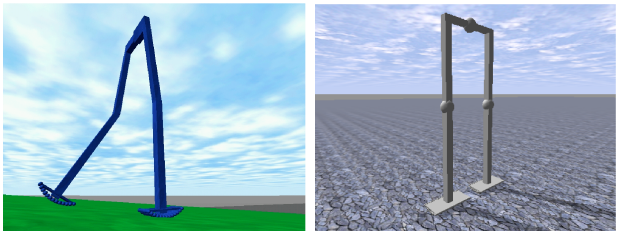


A.19.4 SCORPION. Scorpion (by Frank Kirchner) is a biologically inspired eight-legged walking machine. A simulation for Scorpion was written by Poramate Manoonpong. The image of the Scorpion robot was taken from the project page of the University of Bremen (Spenneberg, 2008).

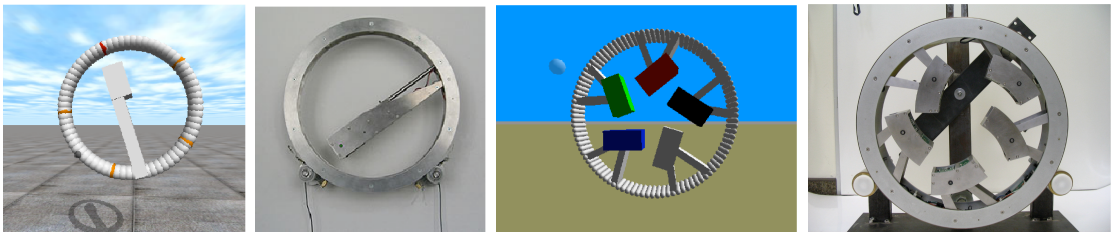
Figure A.19: ISEE PROJECT EXAMPLES I/IV.



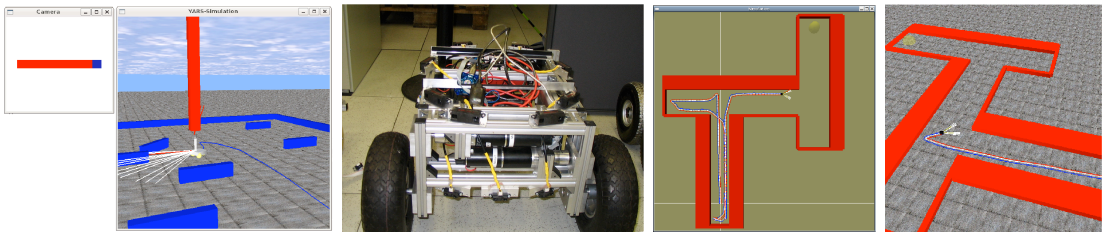
A.20.1 RunBot. RunBot (Geng et al., 2006) is the fastest two-legged walking machine. The image of the robot was taken from the project page at the BCCN in Göttingen. A simulation was written by Poramate Manoonpong in YARS.



A.20.2 Passive and dynamic walker. Steffen Wischmann first evolved the morphology of a passive dynamic walker (left). From that experiment, an active walker was evolved which requires only very limited actuation (Wischmann & Pasemann, 2004).

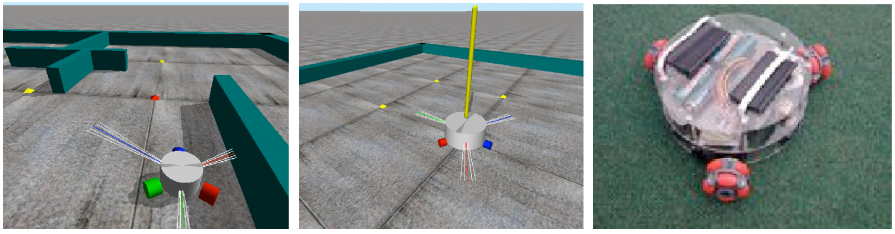


A.20.3 micro.adam and micro.eva. Two art projects, by Julius Popp (Popp, 2005). Left: simulation of and the physical micro.adam, right: simulation and physical micro.eva. Both are gravity driven robots, which rotate around the centre by displacement of their centre of gravity. micro.eva was used to compare centralised and decentralised control strategies (Wischmann et al., 2005).

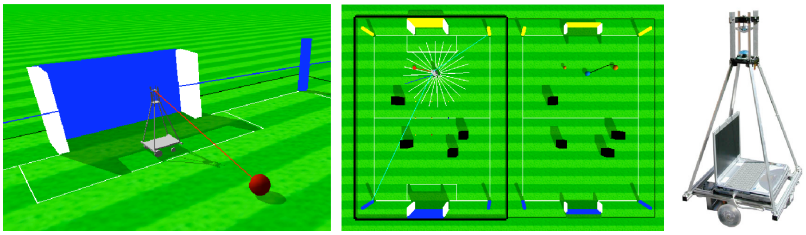


A.20.4 DFG OUTDOOR project. This project is funded by the German Research Foundation Priority Program 1125. In an outdoor scenario, teams of mobile robots are supposed to cooperate on a large field. Communication in such a scenario is difficult. The robots should be able to detect the intention of team-mates by observation of their actions, using the concept of mirror neurons. The two figures on the left-hand side show an experiment, implemented by Verena Thomas (Thomas, 2008) in YARS, to evolve a neuro-controller which is able to distinguish between ego-motion and motion within the environment. The figure in the centre is the physical OUTDOOR platform, based on the Volksbot (Wisspeintner & Bose, 2005). The two figures on the right show a simulated environment, a maze with randomised hallway width, used to evolve a controller with memory.

Figure A.20: ISEE project examples II/IV.

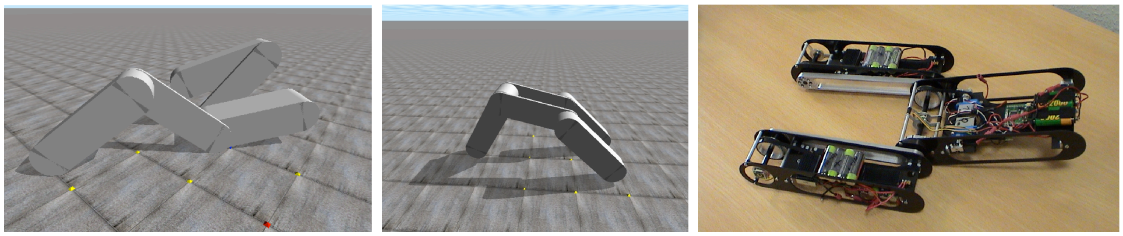


A.21.1 Robertino with and without pole. Robertino (Verbeek, 2008) is a holonomic platform (right-hand side). The two figures to the left show simulations (by Björn Mahn) used to evolve a neuron controller to control a Robertino with and without a pole (Mahn, 2003).

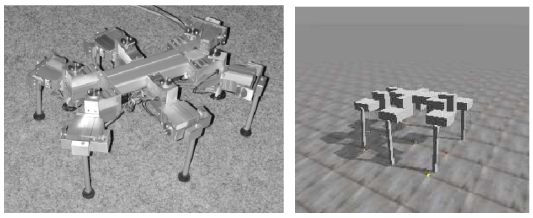


A.21.2 RoboCup. The images are taken from two different projects, both related to RoboCup. In both projects, a controller for a RoboCup behaviour was evolved. On the left-hand side, is a simulation based on ODE written for goalkeeping behaviour for the AIS/Musashi MidSize RoboCup Team (Zahedi et al., 2004). Next, the evaluation environment (by Björn Mahn) for a striker behaviour. Two neuro-controllers were evolved and then fused to create an overall striker behaviour, a ball-seeker, and a goal-shooting behaviour. The striker project was led by Martin Hülse.

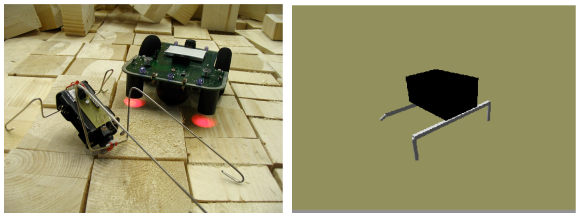
Figure A.21: ISEE project examples III/IV.



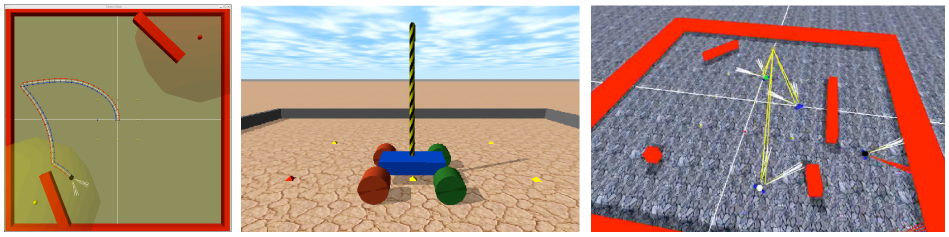
A.22.1 Uffington. This robot platform consists of five segments which are connected by four independent actuators. The original platform was designed for Mars exploration. The two figures on the left show the simulation, the figure on the right shows the redesigned physical platform (by Mirko Mannes).



A.22.2 Morpheus. Morpheus (by Jörn Fischer) is a six-legged walking machine (left) with two degrees of freedom in each leg (Fischer, 2003). Equipped with a single infra-red sensor, it is able to explore an office environment. Right: The simulation environment used to evolve new behaviours.



A.22.3 Ted and Do:Little. Ted (by Manfred Hild) is a walking machine with only two degrees of freedom controlled by a SO(2)-Network (Pasemann, Hild, & Zahedi, 2003). Left: The physical machine, together with Do:Little (by Manfred Hild and Jan Karabasz), left, the simulation (by Steffen Wischmann).



A.22.4 Other projects From left to right: Light-seeker with randomised environment in YARS, pole-balancer implementation in ODE, and swarm robots implementation in YARS.

Figure A.22: ISEE project examples IV/IV.

Appendix B

Howto's

This appendix provides guidelines on how to use and extend ISEE for artificial evolution and analysis of recurrent neural networks.

In order to set-up an experiment, two steps are required. The first step is to include an evaluation environment within the ISEE framework. This can be a simulation in YARS, a third party simulator, or a physical robot. The second step is to implement a fitness-function which evaluates the generated controller.

Both steps are discussed in this appendix, followed by a description of the process required to extend ISEE through the neural network exporter, learning rules, and Brightwell visualisation methods.

The description is written for the reader without expert knowledge in programming, and may include information not required by experts.

The following typographical conventions are used in this appendix: *function*, **class name**, COMPONENT, `gui element`, variable/attribute.

B.1 Overview

The introduction mentioned two steps, including an evaluation environment, and defining a fitness-function. Both these steps require modifications of Hinton. Therefore, this section gives an overview of the process sequence for Hinton. This is necessary to understand the concept of the communication and fitness-function classes and why the implementation of different functions are required. The process sequence is described on the basis of the function calls between the different classes of Hinton (see fig. B.1).

A brief introduction of the relevant classes is given (see tab. B.1), followed by the sequence of interactions between them.

The process is discussed with respect to artificial evolution. When Hinton is used as an analysis tool, decoupled from artificial evolution, the process may be considered equivalent. In this case, only one individual is processed within each generation and it is not altered over the generations. The steps are visualised in the figure B.1 and the step number refer to the number given in the figure.

Net	Implementation of the neural network.
RobotStruct	Data exchange container for sensor and motor data of the simulated/physical robot.
FitnessFunction	Implementation of a fitness-function.
Ambassador	Implementation of a communication class, either for the simulated or physical robot.
SimCom	Superclass of communication implementation, providing basic functionality required by every communication. In this section a simulated robot is chosen, but the process is equivalent for physical robots.
Processor	Processing of the neural network, communication with the robot, calculation of the fitness-function.
ProcessorThread	Controls the overall evolutionary process.

Table B.1: Classes used within the Hinton main loop. On the left-hand side: Names of the classes which are used during the main loop for the evaluation and processing of a controller in Hinton. Right-hand side: Brief description of the functionality of the class.

Step 1: The initial conditions for each individual in a generation must be equal, to assure that the fitness values are comparable. Hence, a method must be provided, which sets the initial conditions at the beginning of each new generation. This method, named *setNewStartPosition*, is part of the ambassador implementation (see sec. A.3.3), and is called by the **Processor-Parameter**.

Step 2: The evaluation of an individual begins with the setting of the initial conditions. For this reason the **ProcessorThread** calls the *reset* method implemented in the ambassador. It is responsible for setting the initial conditions to those previously defined in the *setNewStart-Position* method.

Step 3: The third step is to reset the fitness-function before an individual is evaluated. This is done by the **ProcessorThread**. It calls the *reset* method of the fitness-function implementation.

Step 4: After resetting the initial conditions and the fitness-function, the **ProcessorThread** calls the *run* method of the **Processor**. The **Processor** is the neural network processing implementation. Besides processing the neural network, the **Processor** is the implementation of the sensori-motor loop, as described in the following steps.

Step 5: There are two different methods which may be used to start the sensori-motor loop. The first possibility is to start it with the processing of the recurrent neural network and the generation of the motor commands. Using this method, the first action of the controlled robot

is only dependent on the initial parameter of the controller. In the case of recurrent neural networks, these parameters are the synaptic strength and bias values of the neurons.

In the implementation of Hinton, a second method was chosen. Before the recurrent neural network is processed, sensor values from the simulated or physical robot are used as input to the network. Then, the motor commands are generated and passed to the robot.

This method was chosen, because it adds another source of randomness to the evolution process. In the case of the first method which is described above, the first motor command is independent of the environmental setting. By making the motor commands of the controller dependent on the first sensor signals, the first command varies for the same controller in different environments. This increases the selection pressure for robust controllers.

In the process loop this means that the methods *send* and *update* of the AMBASSADOR are called by the **Process** before the network is processed. The send method typically sends data to the simulation/robot, and the update method receives data and fills the **RobotStruct** (see app. B.4.2). Pre- and post-processing of the data is implemented in *send/update* methods, where applicable.

Step 6: After the **RobotStruct** has been filled by the ambassador, the **Proccessor** receives one sensor value for each input neuron from the **RobotStruct** by calling the *getOutputDouble* method. The minimum of the number of input neurons $|I|$ and the number of sensors $|S|$ determines how the sensors are mapped to the input neurons. The first $n = \min(|I|, |S|)$ input neurons are fed by the first n sensor values. The order of the sensors is defined by an XML-file (see app. B.4.2). If the number of input neurons exceed the number of sensors, the input neurons activations are set to zero. If the number of sensors exceed the number of input neurons, the sensor values are not used in the network processing. However, they can still be used in the calculation of the fitness-function. Therefore, global sensors are typically added at the end of the list of sensors, so that they are not available for the neural network, but for the evaluation process using the fitness-function.

Step 7: Once the sensor values have been received, the next step is the processing of the recurrent neural network. The number of iterations of the recurrent neural network is defined in the **ProcessParameter** object. Typically only one iteration is performed, but any other number of iterations is possible. A common example for a different setting is three. This ensures that the newly applied sensor values reach the output neurons before motor commands are generated. Three iterations are used because the recurrent neural network consists of three neuron layers; input, output, and hidden.

Step 8: The values of the output neurons are fed back into the **RobotStruct**. The value for the output neuron to motor mapping is set equivalent to the value for the input neuron to sensor mapping (see step 7). If the number of motors exceeds the number of output neurons, the relevant values are set to zero.

Step 9: After the neural network has been processed, the **Proccessor** calls the *calculate* method of the fitness-function implementation. The fitness-function implementation has access

to all data that is made available by the **ProcessorParamter** and **RobotStruct**.

The experimenter has to ensure that the method *calculate* of the fitness-function implementation stores a valid fitness value at each step (see app. B.3).

Step 10: This step is optional and only applied, if the experimenter defines more then one try for each evaluation. For each try, the controller is evaluated against a different initial condition, which must be previously defined in the *setNewStartPosition* method. The overall fitness is the sum of the fitness values of all the tries. The method *nextTry* is called by the **Proccessor** and is typically implemented similarly to the *reset* method.

Summary

The steps one to ten define the framework in which the experimenter includes a new experiment. This is the framework, in which the communication, the fitness-function, the **RobotStruct**, and the **ProcessParameter** class are located.

B.2 How to write a Communication class

A communication class extends the class **SimCom** and must be located in the directory HINTON/SIMULATORS (see fig. A.2). From the superclass **SimCom** the communication class has access to all information available through the three data exchange classes, **RobotStruct**, **ProcessParameter**, and **RobotStatus** (see app. B.4).

This section explains which functions need to be implemented, and how they interact with the overall framework.

String getSimName

The method *getSimName* must provide a unique, human readable string. The string defined in this method is displayed in the `communication selection` pull-down menu available on the Hinton GUI (see fig. A.5). If no configuration file is found, the *getName* of the superclass **SimCom** adds extra character to the string provided by the *getSimName*, and therefore, the superclass method should not be overwritten.

String getConfigName

The method *getConfigName* returns a string that specifies the location of a configuration XML-file. This XML-file is read by the class **SimCom** when Hinton starts and is used to initialise the **RobotStruct** (see app. B.4.2). It defines the sensor to input-neuron and output-neuron to motor mappings.

void connect

The method *connect* is called as soon as the user activates the `connect` button on the simulation panel of Hinton (see fig. A.5). This method initialises the connection between Hinton and the simulation. Typically the attributes *port* and *ip*, which are available through the superclass

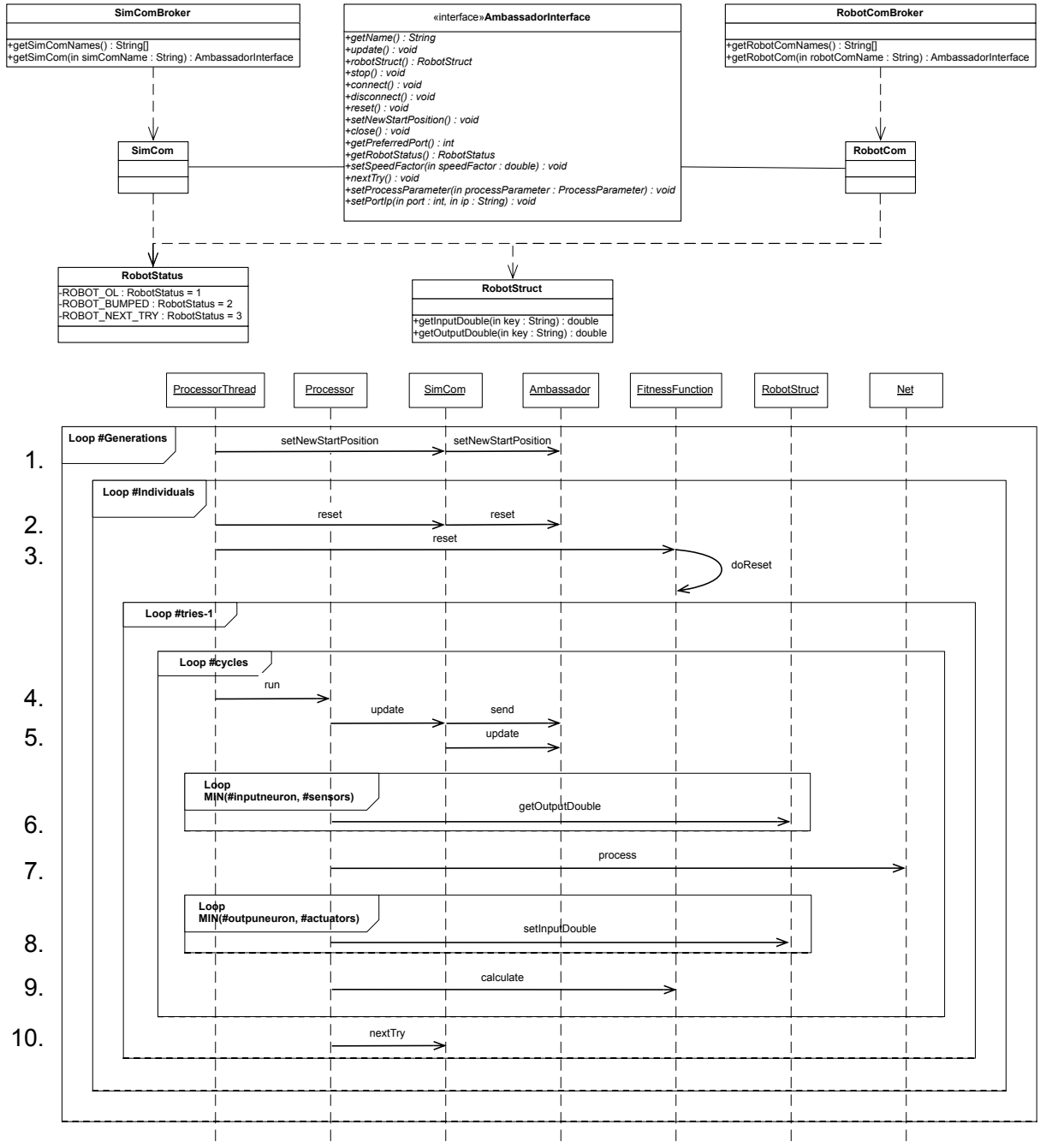


Figure B.1: Hinton UML class and UML sequence diagram.

SimCom, are used to identify the port and host ip of the simulator. These attributes are automatically filled with the values given through the GUI.

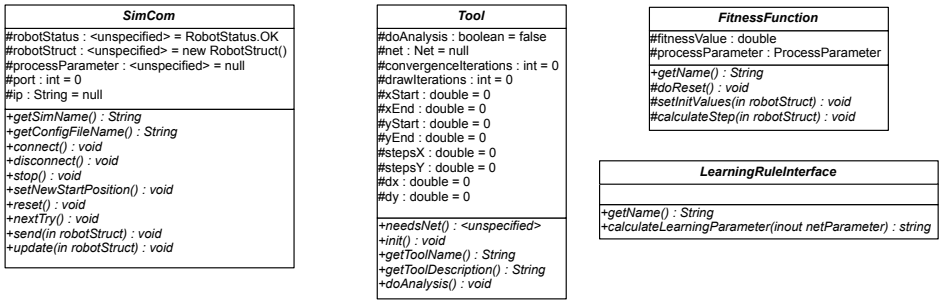


Figure B.2: Overview of the main extension clases. From left to right, and top to bottom: **SimCom**, **Tool**, **FitnessFunction**, **LearningRuleInterface**. This digram does not show all implemented functions and attributes. It only shows the available attributes, that can be accessed directly, and the abstract methods, that need to be implemented by the user.

void disconnect

The method *disconnect* is called by the GUI, as soon as the user presses the `disconnect` button on the simulation panel of Hinton (see fig. A.5). In this method, the user has to take care that the connection previously established in the *connect* method is closed. Typically this function sends a close command to the simulator, and frees the occupied socket.

void stop

The method *stop* is called as soon as the user presses the `stop net` button on the Hinton GUI. No further data is exchanged between Hinton and the simulator once this is done. This means that currently set motor commands are still controlling the robot. Hence, the robot might continue to act according to the previously set motor commands. This function can be used to send a stop command, ensuring that the robot halts.

void setNewStartPosition

The method *setNewStartPosition* is called once for each new generation (see fig. B.11). This function generates the initial conditions for all individuals. They are set only once, because they must be equal for all individuals in one generation. If they were not equal, the results of the evaluation of each individual would not be comparable. The evolution would then be a randomised search over the parameter-space.

void reset

The method *reset* is called once for each individual, before the evaluation begins (see fig. B.13). Typically, this method sends a reset signal to the simulator.

void nextTry

The method *nextTry* is only called if more than one try is defined by the experimenter. A try is similar to a new evaluation, but it is performed with the same controller. The overall fitness is the sum of the fitness values over all tries. The method typically sends a next-try or reset command to the simulator.

void send and void update

The methods *send* and *update* are redundant. There is no need to implement both. They are both provided, to make it possible to distinguish between the data exchange (send-method) from the data processing (update-method) methods. Typically, the *send* method is implemented such that it first sends the motor commands to the simulator, and waits for the updated sensor values returned by the simulator[†]. The new sensor values are then stored in the **RobotStruct**. This is then done in the *update* method. The pre- and post-processing are provided within these functions, where applicable.

B.3 How to write a Fitness-Function class

The fitness-function is a qualitative measurement of the behaviour of a robot performing a task within an environment. The result of the fitness-function is the fitness value, a scalar assigned to the behaviour. The comparison of the fitness values for different behaviours determine the differences of the quality of the behaviours with respect to the given task.

Based on the fitness values of the individuals in one generation, the selection operator selects the individuals that form the parents of the next generation (see sec. 3.3).

An implementation of a fitness-function extends the superclass **FitnessFunction**. To make the newly-created fitness-function available in Hinton, the fitness-function implementation must be located in the directory HINTON/FITNESSFUNCTIONS (see fig. A.2), from which it is automatically loaded during the start up of Hinton.

This section describes the methods that are required for fitness-function implementations.

String getName

The method *getName* must provide a unique, human-readable string. The string defined in this method is displayed in the fitness-function selection pull-down menu available on the Hinton GUI (see fig. A.5).

void doReset

The method *doReset* is called once before the evaluation of an individual begins. This function resets all variables and data structures. The only variable, which is reset by the superclass **FitnessFunction** and does not need to be taken care of, is *fitnessValue*.

[†]This does not violate the statement of how the sensori-motor loop is initialised. At the beginning, all outputs of the network are zero. Typically, these are stop commands, so that they do not initiate any action.

void setInitValues

The method *setInitValues* is called once before an individual is evaluated. This method is used to set the initial values of the fitness-function calculation. Examples are the starting coordinates of the robot, so that the distance can be measured, or the initial sensor values, from which the deviation is used for the fitness value calculation.

void calculateStep

The method *calculateStep* is called once after every update of the recurrent neural network. Because it can not be determined, when the evaluation ends, this method must guaranty, that after each calculation step, the variable *fitnessValue* is a valid representation of the current behaviour evaluation. The evaluation can end unexpectedly if the simulator sends a bump signal (see app. B.4.3). When the evaluation is terminated, the value currently stored in the variable *fitnessValue* is returned to the evolution program (EvoSun) and is used in the selection process.

B.4 Data-Exchange classes

This section discusses the data-exchange classes that provide all the available evolution and evaluation information. All classes are initiated once at start-up time of Hinton, and the data is updated during runtime. The objects are preserved.

B.4.1 ProcessParameter

The class **ProcessParameter** is a data container for the parameters that control the evaluation and which are provided by the evolutionary program (EvoSun) and the GUI of Hinton. It contains the current recurrent neural network and the process parameters that are available to the experimenter through the EvoSun (see fig. A.4) and Hinton GUI (see fig. A.5). The discussion of the parameters is divided into two groups, those related to EvoSun and those related to Hinton.

EvoSun parameters:

Constants C0 – C3 are double values, which can be used as coefficients in the fitness value calculation (see app. B.3).

Cycles The number of evaluation cycles. This is the maximal number of neural network iterations in the evaluation process.

Warm-Up Steps The number of the initial evaluation steps, for which the fitness value is not computed, i.e. for which Hinton does not call the *calculateStep* method (see sec. B.3) of the fitness function implementation. This can be used to allow the behaviour to converge, without any punishment in the fitness value. This was discussed in the evolution chapter (see chap. 6).

Net The current recurrent neural network.

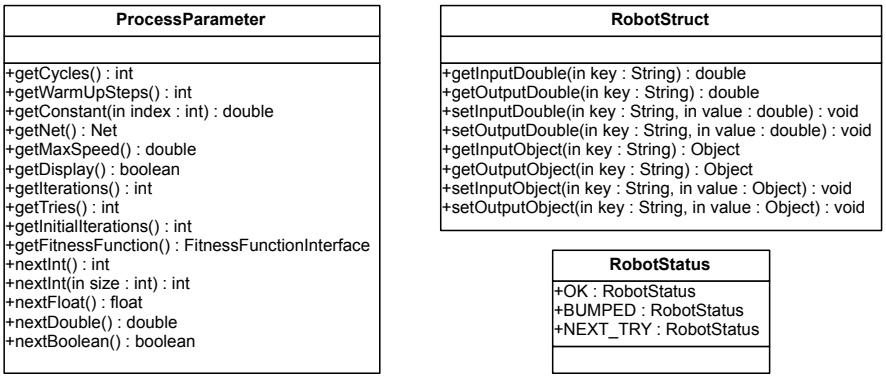


Figure B.3: DATA-EXCHANGE CLASSES. From left to right: **ProcessParameter**, **RobotStruct**, and **RobotStatus**.

Hinton parameters:

Display is a boolean value, which can be used in the fitness-function implementation to toggle the print out of debug information on the fitness value calculation.

Maximal Speed This parameter is used for the post-processing of motor commands. The transfer-function, and therefore, the output of the output-neurons is limited ($\sigma(x) \in]0, 1[$, $\tau(x) \in]-1, 1[$). In order to control the speed of a robot, higher speeds might be appropriate.

Iterations This parameter controls how often the network is processed after receiving the sensor information, and before sending the generated motor commands. This can be used to process the network several times, such that the sensor value, currently presented at the input neurons, reaches the output neurons before the motor command is generated.

Initial Iterations The initial iterations are performed once for each individual at the beginning of an evaluation. After presenting the first available sensor information, a number of initial iterations of the recurrent neural network are performed. This can be used to ensure, that the recurrent neural network has converged to a stable state, before controlling the robot (see chap. 6).

Fitness Function This is the current fitness-function instance. The reference to the fitness-function is typically used, to send the fitness value to the simulator. This is done, so that the simulator can plot the trajectory of the best individual within the current generation.

B.4.2 RobotStruct

The class **RobotStruct** is a representation of the sensor and motor data from the robot's perspective. Sensor values are considered as robot output and motor commands as robot input. For each of the data streams, input and output, the **RobotStruct** provides getter and setter

methods. The first set of commands work on double values (see fig. B.3), because the data type double is the standard data type in Hinton.

The second set of functions works on the Java data type **Object**. As **Object** is the superclass of every class in Java, these getter and setter methods are not restricted and can be used to store and pass any desired data type.

B.4.3 RobotStatus

The class **RobotStatus** is only available in the communication class **SimCom**. The **Processor** checks the attribute robotStatus after each send-update call of the communication implementation. The attribute robotStatus can only take one of the following values:

RobotStatus.OK If the attribute robotStatus is set to RobotStatus.OK, the evaluation is continued.

RobotStatus.BUMPED The bump originates from obstacle-avoidance experiments. In the evolution of an obstacle avoidance behaviour the evaluation is terminated if the robot has bumped into an obstacle. The concept of a *bump* is generalised to an external event that terminates the evaluation of an individual. Depending on the setting for the tries parameter (see app. B.1), either the next try is processed, or the next individual is evaluated.

B.5 How to write a Tool class

The program *Brighwell* (see sec. A.3.6) provides a set of analysis and visualisation tools. In this section they are simply referred to as tools. This section describes how a new tool is included in Brighwell.

A new tool extends the class *Tool*, and must be located in the directory BRIGHTWELL/TOOLS (see fig. A.2). In the following guide, the attributes, which are available from the superclass **Tool**, and the functions which need to be implemented, are discussed.

Available attributes

This section describes the attributes, that are available when the superclass **Tool** is extended. The attributes include data which the user specifies through the control panel of the Brightwell GUI (see sec. A.3.6).

convergenceIterations The number of neural network update iterations, before the plot begins.

drawIterations The number of plot iterations. For each coordinate, a number of convergenceIterations are first performed, followed by a number of drawIterations for which the current state of the systems is plotted on the **DrawingPanel** (see below).

doAnalysis Analysis algorithms are mostly preformed in loops. It is desirable that a user can interrupt a running analysis by an event. The variable *doAnalysis*, which is true by default, is set to false when Brightwell has caught the escape key. It can, therefore, be used to interrupt the current analysis.

net This is a pointer to the currently selected neural network (visible over the `NetEdit` GUI). It is `null` if no net was selected.

xStart/xEnd/yStart/yEnd Define the x- and y-domain of the plot.

stepsX/stepsY The resolution of the x- and y-domain.

dx/dy Domain divided by the resolution.

Available functions

A set of functions are predefined in the Tool superclass. These must be defined in the implementation of a new analysis tool.

needsNet This function only returns a boolean value. It must be overwritten. When the `run` button is pressed by the user, the value of this function is evaluated. If it is true, and no network was selected, a message dialogue is presented, reminding the user to select one. In this case, the analysis tool is not executed.

init This function is called, when the tool is first loaded at start-up time. It is used to initialise the tool GUI (see below, How to create an input panel).

getToolName This function returns a string, which is displayed in the tab selection menu of Brighwell.

getToolDescription This function returns a string, which is displayed in the tool tip, when the mouse is placed and kept over the tool name for a while. It can be used to give detailed information about the purpose of the tool.

doAnalysis This function is called, when the tool is executed.

How to create an input panel

Besides the globally available attributes that are accessible in every tool, such as the selected network and the number of convergence and draw iterations, each tool has specific parameter requirements. To be able to provide a graphical interface with minimum implementation effort, a set of functions are provided by the Tool superclass. These functions are derived from the Analyser which was written by Michael Rosemann (Rosemann, 2004).

For each input device (integers, doubles, check boxes, etc.), there is an add function, which places the corresponding input field on the tool's GUI, and a get function, which is used to retrieve the value from the input field. Both use a string as an identifier. The string is also used as a label for the input field.

The syntax for all input fields is similar and shown below:

```
public void addInteger(String name,
                int minValue, int maxValue, int defaultValue)
public int getInteger(String name)
```

```
public void addDouble(String name,
               double minValue, double maxValue, double defaultValue)
public double getDouble(String name)


public void addString(String name, String initialValue)
public String getString(String name)


public void addComboBox( String name, String[] entries,
               int selectedIndex)
public int getComboBoxIndex(String name)


public void addCheckBox(String name, boolean initialValue)
public boolean getCheckBox(String name)


public void addFileChooser(String name, String initialValue)
public String getFileChooser(String name)
```

B.5.1 How to to use the DrawPanel

Brighwell provides a simplified drawing panel. There are different ways to initialise such a panel, specific for neurons, synapses, etc.. The generic method is given here as an example. For a detailed list of all possible functions, please refer to (Zahedi & Hülse, 2008). This section simply provides a general overview of the available functionality.

```
public DrawingPlane getNewWindow(String name,
    double xMin, double xMax,
    double yMin, double yMax)
```

The name is displayed as the window name. The domain ranges for the x- and y-axis are given as additional parameters to the function. This is important as drawing coordinates are not given in global window coordinates (starting with 0,0 at the upper left corner of the drawing panel), but in coordinates with respect to the given domain range.

Different functions to draw lines and points are provide by a DrawingPlane.

```
public void drawPoint(double x, double y)

public void drawPoint(double x, double y, Color color)

public void drawLine(double x0, double y0, double x1, double y1)
```

```
public void drawLine(double x0, double y0, double x1, double y1,
    Color color)
```

As stated above, the coordinates are given with respect to the intervals which were defined when the panel was created.

B.6 How to write a Learning Rule class

For this thesis, an interface is required to enable the implementation of different learning methods with minimal implementation effort, and enable to dynamic loading of new classes without needing to re-compiling the entire framework. This allows the testing of different implementations and the subsequent comparison of the results. The learning rule class consists of two functions, which must be specified by the experimenter.

```
public String getName()

public void calculateLearningParameter(Vector netParameter)
```

The netParameter vector includes all neuron parameters $(a(t), o(t), \xi(t), \eta(t), \ldots)$ and must be updated in the *calculateLearningParameter* function. The *getName* function must return a unique human readable string. It is listed in a pull-down menu in the `NetEditor` (see sec. A.3.3).

Each new learning rule class must be placed in the directory LEARNINGRULES from which it is automatically loaded during the start-up of Hinton. The string provided by the *getName* function is used as identification such that a learning rule can be selected during runtime by the `NetEdit` dialogue box.

B.7 RoSiML

To define an experiment in YARS, an XML-based description language is provided (see app. A.3.5). This section gives an overview of the current state of RoSiML. It is under constant development, but the concept and the basic structure for objects and joints remain the same. The newest version of RoSiML and its documentation is provided online (Zahedi et al., 2007). An experiment description in RoSiML (robot simulation meta language) is divided into three parts, the simulator, environment, and movable objects (simply referred to as movables) description.

A movable is a set of compounds, which are divided into objects. An object is a geometrical primitive, and is connected to another by a joint. When YARS is started, it prints a list of all objects, motors and sensor, as a tree. A naming mechanism is implemented, such that every sensors and every motor is identified by a unique string, which is communicated to the control program during the handshake process. This identification can then be used to access the desired information.

The simulator description specifies general simulator specific configurations, such as the position of the global camera, the size of the simulator visualisation, etc.

The environment description specifies the static objects in the environment. Although they can be placed randomly, they remain immovable throughout the simulation.

There are different types of movables, which are labelled as controlled, active, passive, and moving. They differ in the way they are controlled.

passive A passive movable is not controlled at all. These objects can, for example, be used to simulate a ball in a RoboCup scenario (The RoboCup Federation, 2007).

active An active movable is controlled through a socket communication. Motor commands and sensor values are communicated over and UDP protocol (see app. A).

controlled A dynamically loaded program written in C++ controls the robot. The controller is identified by a name given as a human-readable string.

moving A dynamically loaded program written in C++ controls the robot. The controller is identified by a name given in a human-readable string. In contrast to the controlled movable, this type of object outputs are forces which are directly applied to the body of the movable.

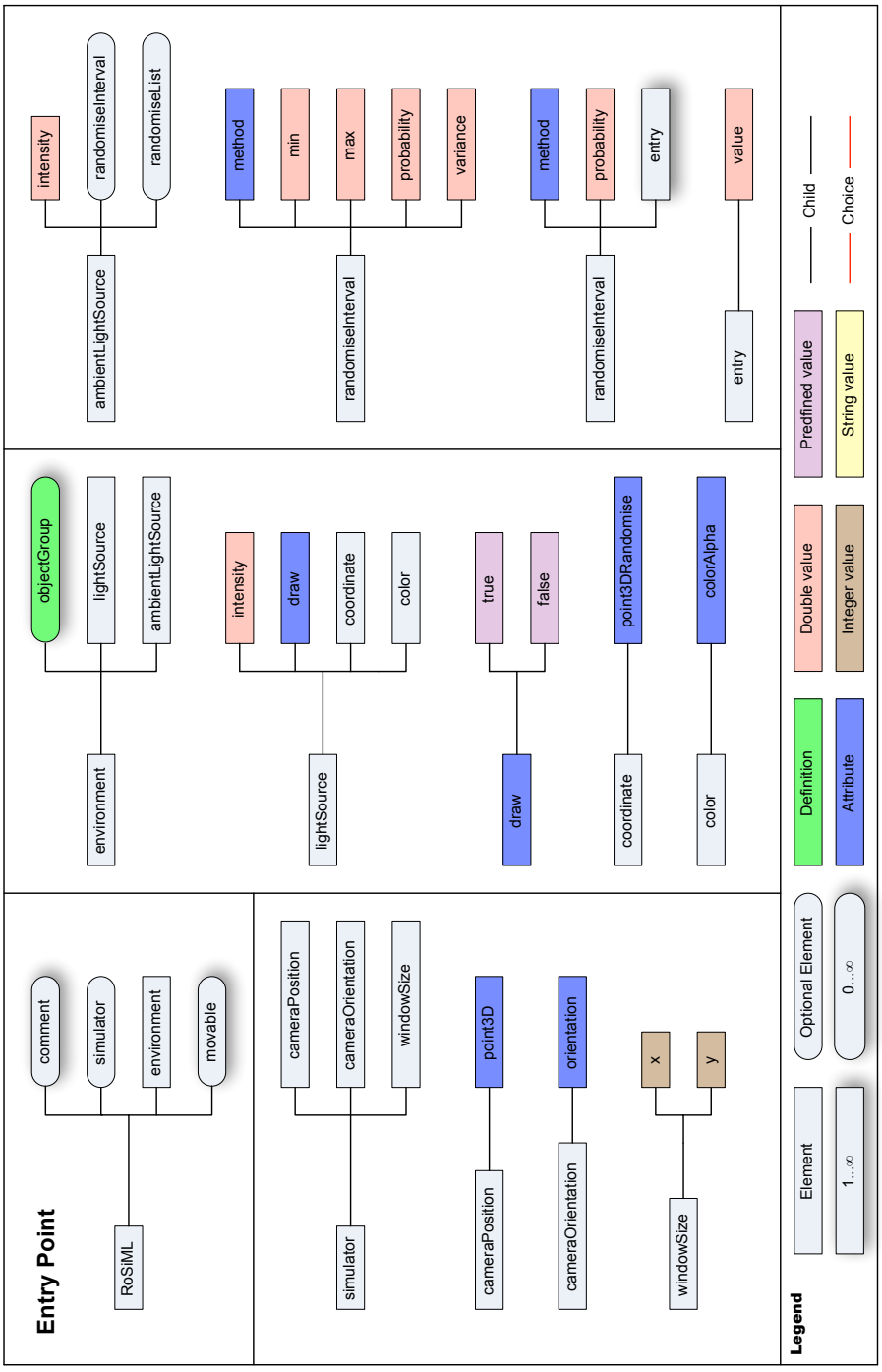


Figure B.4: RoSiML I/VI: Simulator and environment configuration.

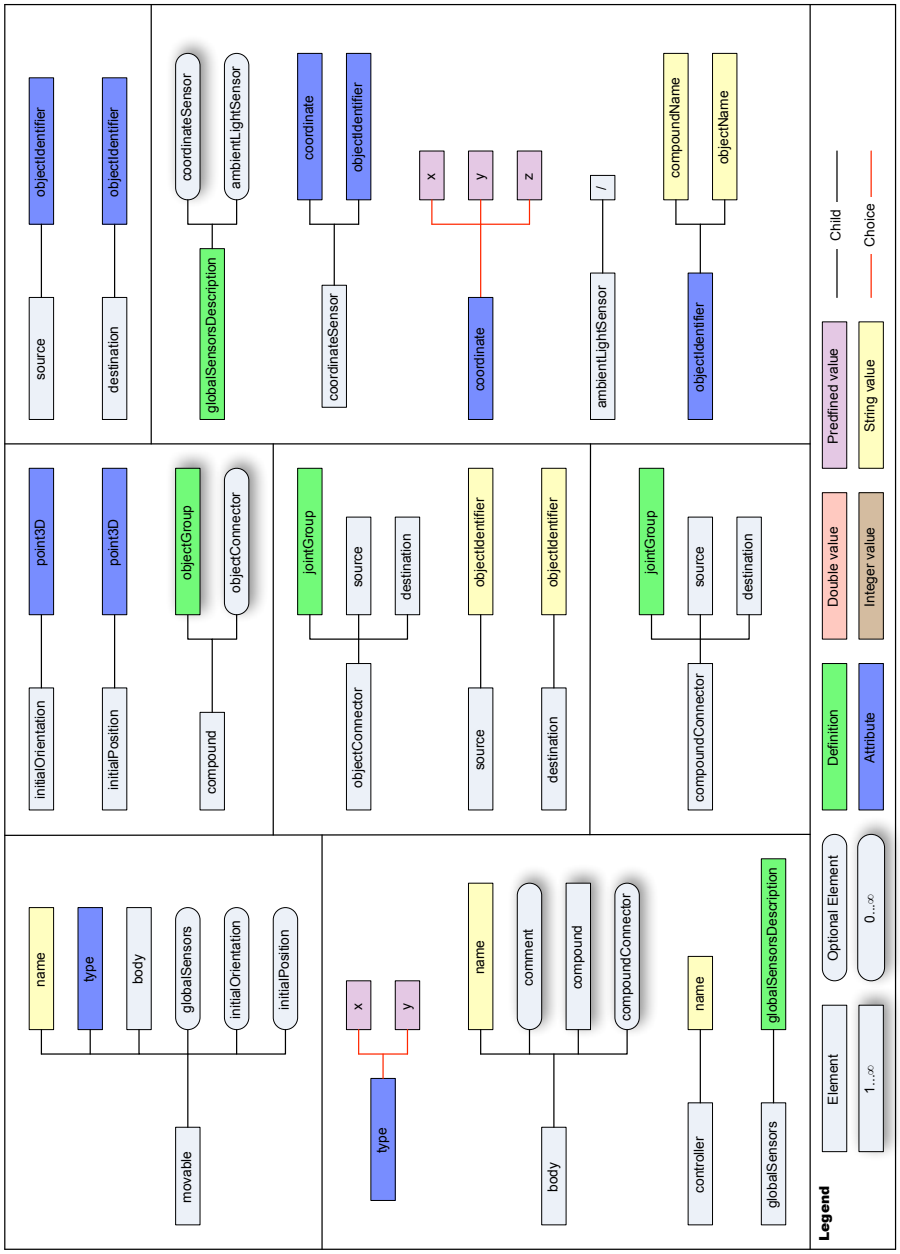


Figure B.5: RoSiML II/VI: Movable configuration.

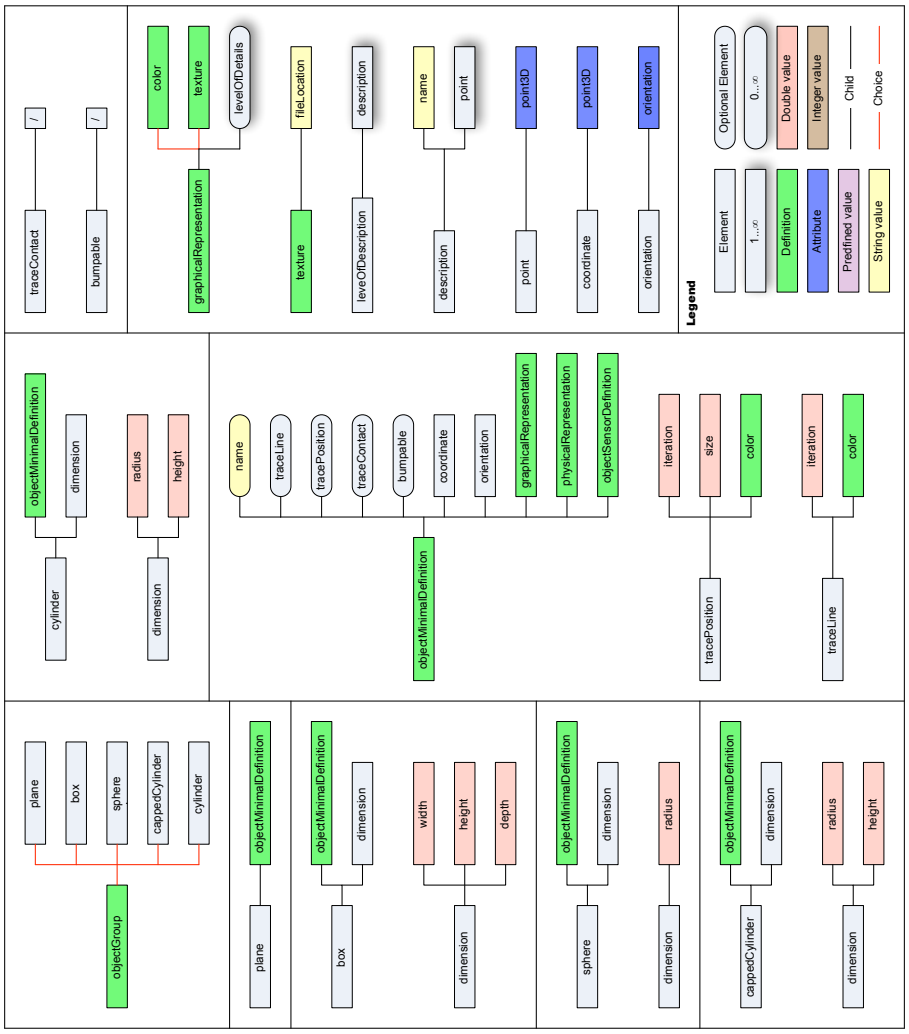


Figure B.6: RoSiML III/VI: Object configuration.

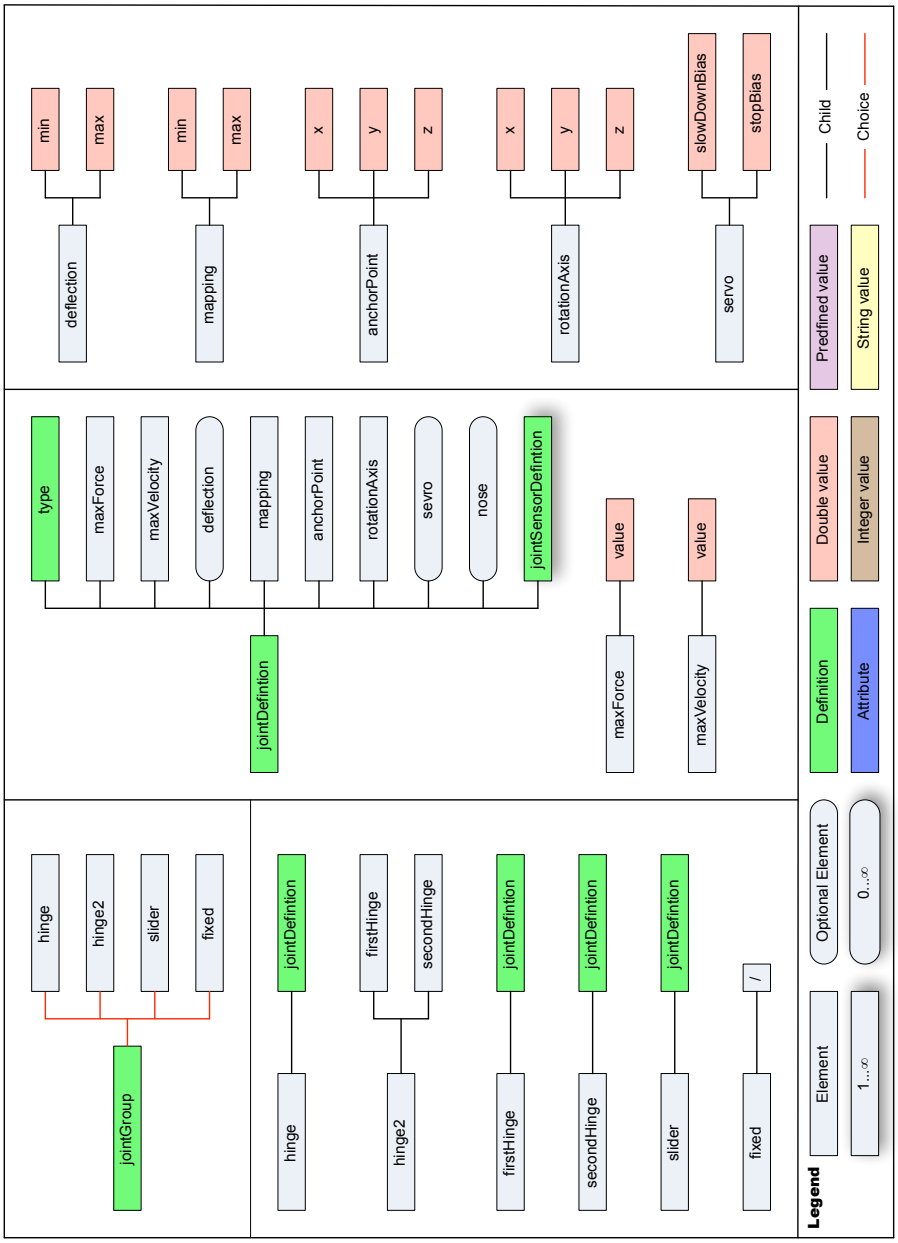


Figure B.7: RoSiML IV/VI: Joint configuration.

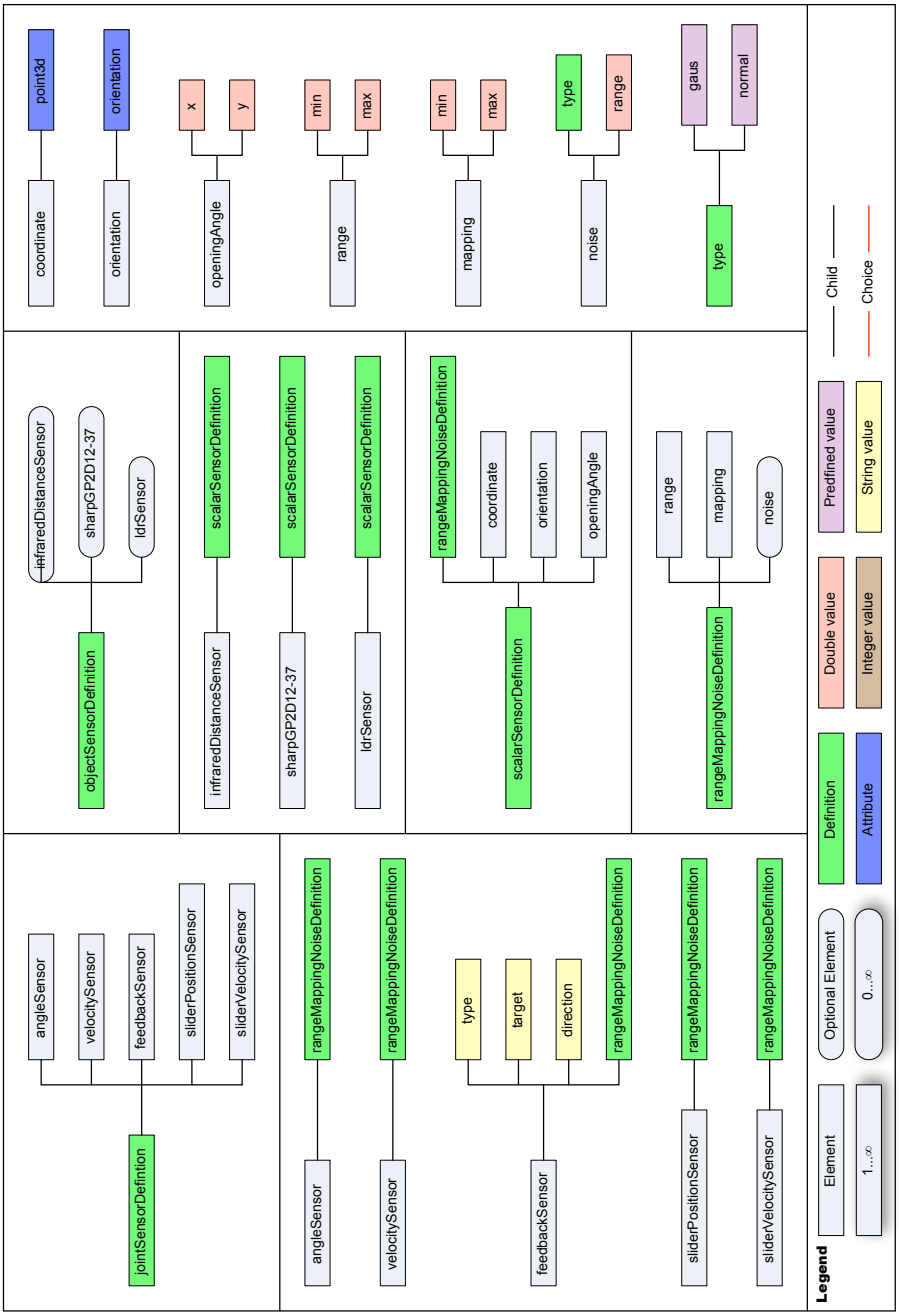


Figure B.8: RoSiML V/VI: Joint sensor configuration.

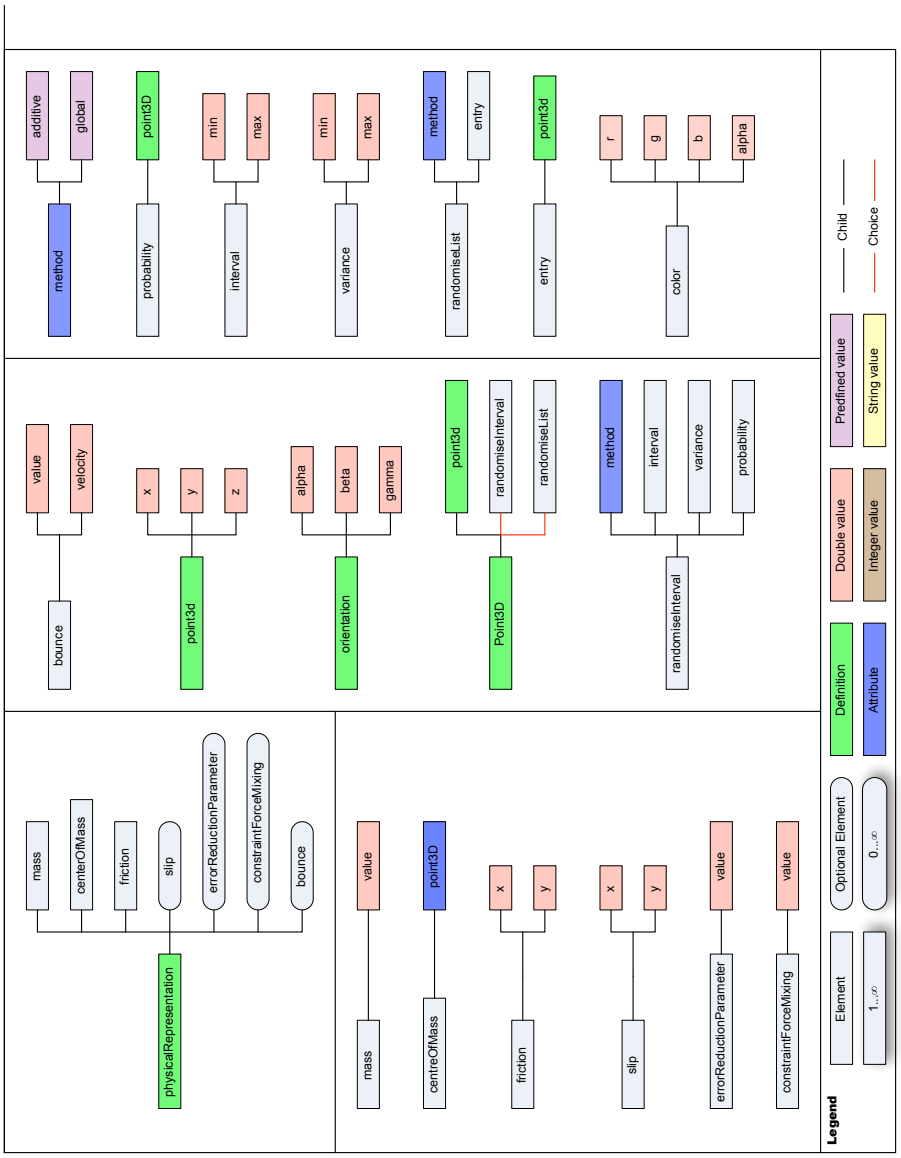


Figure B.9: RoSiML VI/VI: Physical object configuration.

Appendix C

SRN Model Stability Analysis

Single neuron with excitatory recurrent connection

This section presents the detailed calculations of the stability analysis for the single Self-Regulating Neuron with excitatory recurrent connection, the results have been previously presented and discussed (see chap. 4). Whenever necessary and possible, the calculations were conducted with the assistance of a computer algebra program.

The SRN with positive self-coupling is given by the following set of equations:

$$\begin{aligned} a(t+1) &= \Theta + \xi(t)\,\eta(t)\,\tau(a(t)) \\ \xi(t+1) &= \xi(t)\left(1+\beta\left(\tau(\mathrm{a}^*)^2 - \tau(a(t))^2\right)\right) \\ \eta(t+1) &= (1-\gamma)\,\eta(t) + \delta\,(1+\tau(a(t)))\,. \end{aligned}$$

Assuming that the activation coordinate of the fixed point is equal to the target value, the fixed point of the system $\rho^* = (a^*, \xi^*, \eta^*)$ is given by:

$$\begin{aligned} a^* &= \Theta + \xi^*\eta^*\tau(a^*) \\ \eta^* &= \frac{\delta}{\gamma}\,(1+\tau(a^*)) \\ \xi^* &= \frac{a^* - \Theta}{\frac{\delta}{\gamma}\,(1+\tau(a^*))\,\tau(a^*)}. \end{aligned}$$

To calculate the Jacobian matrix, the equations of the SRN are rewritten as follows:

$$\begin{aligned} \rho &:= (a, \xi, \eta) \\ \rho^* &:= (a^*, \xi^*, \eta^*) \\ f_a\colon \mathbb{R}^3 \mapsto \mathbb{R}, \quad f_a(\rho) &:= \Theta + \xi\eta\tau(a) \\ f_\xi\colon \mathbb{R}^3 \mapsto \mathbb{R}, \quad f_\xi(\rho) &:= \xi\left(1+\beta\left(\tau(a^*)^2 - \tau(a)^2\right)\right) \\ f_\eta\colon \mathbb{R}^3 \mapsto \mathbb{R}, \quad f_\eta(\rho) &:= (1-\gamma)\,\eta + \delta\,(1+\tau(a)) \\ f\colon \mathbb{R}^3 \mapsto \mathbb{R}^3, \quad f(\rho) &:= (f_a(\rho), f_\xi(\rho), f_\eta(\rho))\,. \end{aligned}$$

The Jacobian matrix of the SRN is then written as:

$$Df(\rho) = \begin{pmatrix} \frac{\delta f_a}{\delta a}(\rho) & \frac{\delta f_a}{\delta \xi}(\rho) & \frac{\delta f_a}{\delta \eta}(\rho) \\ \frac{\delta f_\xi}{\delta a}(\rho) & \frac{\delta f_\xi}{\delta \xi}(\rho) & \frac{\delta f_\xi}{\delta \eta}(\rho) \\ \frac{\delta f_\eta}{\delta a}(\rho) & \frac{\delta f_\eta}{\delta \xi}(\rho) & \frac{\delta f_\eta}{\delta \eta}(\rho) \end{pmatrix},$$

where

$$\begin{aligned} \frac{\delta f_a}{\delta a}(\rho) &= \xi\eta\left(1-\tau(a)^2\right) & \frac{\delta f_\eta}{\delta \xi}(\rho) &= 0 \\ \frac{\delta f_\xi}{\delta a}(\rho) &= -2\beta\xi\tau(a)\left(1-\tau(a)^2\right) & \frac{\delta f_a}{\delta \eta}(\rho) &= \frac{\gamma}{\delta}\frac{a^*-\Theta}{1+\tau(a)} \\ \frac{\delta f_\eta}{\delta a}(\rho) &= \delta\left(1-\tau(a)^2\right) & \frac{\delta f_\xi}{\delta \eta}(\rho) &= 0 \\ \frac{\delta f_a}{\delta \xi}(\rho) &= \eta\tau(a) & \frac{\delta f_\eta}{\delta \eta}(\rho) &= 1-\gamma \\ \frac{\delta f_\xi}{\delta \xi}(\rho) &= 1+\beta\left(\tau(a^*)^2-\tau(a)^2\right) \end{aligned}$$

resulting in

$$Df(\rho) = \begin{pmatrix} \xi\eta\left(1-\tau(a)^2\right) & -2\beta\xi\tau(a)\left(1-\tau(a)^2\right) & \delta\left(1-\tau(a)^2\right) \\ \eta\tau(a) & 1+\beta\left(\tau(a^*)^2-\tau(a)^2\right) & 0 \\ \frac{\gamma}{\delta}\frac{a^*-\Theta}{1+\tau(a)} & 0 & 1-\gamma \end{pmatrix}.$$

The stability analysis is conducted for the fixed point, hence, the Jacobian matrix is calculated for $\rho = \rho^*$:

$$Df(\rho^*) = \begin{pmatrix} \frac{(a-\Theta)\left(1-\tau(a^*)^2\right)}{\tau(a^*)} & -\frac{2\beta\gamma(a^*-\Theta)\left(1-\tau(a^*)^2\right)}{\delta(1+\tau(a^*))} & \delta\left(1+\tau(a^*)^2\right) \\ \frac{\delta}{\gamma}\left(1+\tau(a^*)\right)\tau(a^*) & 1 & 0 \\ \frac{\gamma}{\delta}\frac{a^*-\Theta}{1+\tau(a^*)} & 0 & 1-\gamma \end{pmatrix}.$$

A symbolic solution of the eigenvalues of the matrix given above is not possible, as the solutions grows too large in complexity. Therefore, in the next step, the plasticity parameters are substituted with empirically-determined values (see chap. 5):

$$\begin{aligned} a^* &:= \alpha \cdot \operatorname{arctanh}\left(\tfrac{1}{\sqrt{3}}\right) \\ \alpha &= +1 \quad \beta = 0.01 \\ \gamma &= 0.01 \quad \delta = 0.02. \end{aligned}$$

Note that a positive value for α is chosen, which means that in what follows the stability is analysed for the upper target value. This is equivalent to the positive fixed point $+a^*$. The

result is a Jacobian matrix of the SRN model at the fixed point depending on Θ:

$$Df(\rho^*) = \begin{pmatrix} -\frac{2\sqrt{3}\left(\Theta-\operatorname{arctanh}\left(\frac{\sqrt{3}}{3}\right)\right)}{3} & \frac{\frac{2}{300}\left(\Theta-\operatorname{arctanh}\left(\frac{\sqrt{3}}{3}\right)\right)}{\frac{\sqrt{3}}{3}+1} & \frac{1}{75} \\ \frac{2}{3}\sqrt{3}\left(\frac{\sqrt{3}}{3}+1\right) & 1 & 0 \\ -\frac{0.5\left(\Theta-\operatorname{arctanh}\left(\frac{\sqrt{3}}{3}\right)\right)}{\frac{\sqrt{3}}{3}+1} & 0 & 0.99 \end{pmatrix}.$$

Of this matrix, the eigenvalues λ_i can be computed as functions of Θ. For the purpose of readability, every additive term $a\Theta^b$ with $a < 10^{-4}$ and $b > 3$ is neglected in the presentation of the eigenvalue equations presented below, as stability only occurs for $|\Theta| < a^* < 1$ (see chap. 4), so that they are approximated by zero[†]. The functions for the eigenvalues are then given by:

$$\begin{aligned}
\lambda_1(\Theta) &= \left(-0.03543\Theta^2 - 0.05702\Theta^3 - 0.00672\Theta\right)^{\frac{1}{3}} \\
&\quad -0.3849\Theta \\
&\quad + \frac{0.1481\Theta^2 + 0.06137\Theta + 0.005364}{\left(-0.03543\Theta^2 - 0.05702\Theta^3 - 0.00672\Theta\right)^{\frac{1}{3}}} \\
&\quad +0.9168 \\
\lambda_2(\Theta) &= 0.9168 \\
&\quad - (0.5 + 0.866\mathrm{i})\left(-0.03543\Theta^2 - 0.05702\Theta^3 - 0.00672\Theta\right)^{\frac{1}{3}} \\
&\quad - \frac{(0.5 - 0.866\mathrm{i})\left(0.1481\Theta^2 + 0.06137\Theta + 0.005364\right)}{\left(-0.03543\Theta^2 - 0.05702\Theta^3 - 0.00672\Theta\right)^{\frac{1}{3}}} - 0.3849\Theta \\
\lambda_3(\Theta) &= 0.9168 \\
&\quad - (0.5 - 0.866\mathrm{i})\left(-0.03543\Theta^2 - 0.05702\Theta^3 - 0.00672\Theta\right)^{\frac{1}{3}} \\
&\quad - \frac{(0.5 + 0.866\mathrm{i})\left(0.1481\Theta^2 + 0.06137\Theta + 0.005364\right)}{\left(-0.03543\Theta^2 - 0.05702\Theta^3 - 0.00672\Theta\right)^{\frac{1}{3}}} - 0.3849\Theta.
\end{aligned}$$

The solutions for $|\lambda_i(\Theta)| = 1$ are calculated to determine the boundaries for the stable region, which are:

$$\forall i = 1,2,3 : |\lambda_i(\Theta)| \leqslant 1 \quad \Rightarrow \quad \Theta \in [-0.15, 0.65].$$

Note that the condition $\lambda_i = 1$ for at least one i and $|\lambda_i| < 1$ for all other i determines the value of Θ, at which the system state switches from stable to unstable (and vice versa) as at least a single eigenvalue is equal to one.

Figure C.1 shows the eigenvalues plotted against Θ. To create the plots, the Jacobian matrix was first solved for a certain value of Θ. A numerical calculation of the eigenvalues of the resulting matrix was then calculated. This was necessary, as singularities occurred, due

[†]Nevertheless, if these simplified eigenvalues are used for further calculations, the results differ significantly from those calculated with the unsimplified eigenvalues. Hence, in the following steps, the unsimplified equations are used for calculation.

α	β	γ	δ	lower target value $-a^*$		upper target value $+a^*$	
± 1	0.1	0.01	0.015	-0.6585	0.1958	-0.1958	0.6585
± 1	0.01	0.01	0.02	-0.6585	0.1559	-0.1559	0.6585
± 0.7	0.01	0.01	0.02	-0.2675	0.04232	-0.04232	0.2675

Table C.1: Results of the stability analysis for different parameter settings. For a detailed discussion of the results, the reader is referred to the chapter *Self-Regulating Neuron Model* (see chap. 4).

to technical limitations, if the eigenvalues were computed directly as functions of Θ. Slight differences are, therefore, observable between the calculation for the boundary conditions above and the values plotted in the figure.

Next, the same calculations are performed for the lower target value $\alpha = -1$, and therefore, for the lower fixed point $-a^*$. As the calculations are otherwise identical to those presented above, only the parameters and the solution are presented below. The chosen parameters are:

$$\begin{aligned} a^* &:= \alpha \cdot \operatorname{arctanh}\left(\tfrac{1}{\sqrt{3}}\right) \\ \alpha &= -1 \quad \beta = 0.01 \\ \gamma &= 0.01 \quad \delta = 0.02 \end{aligned}$$

and for the Jacobian matrix it follows that:

$$Df(\rho^*) = \begin{pmatrix} \frac{2\sqrt{3}\left(\Theta+\operatorname{arctanh}\left(\frac{\sqrt{3}}{3}\right)\right)}{3} & -\frac{\frac{2}{300}\left(\Theta+\operatorname{arctanh}\left(\frac{\sqrt{3}}{3}\right)\right)}{\frac{\sqrt{3}}{3}-1} & \frac{1}{75} \\ \frac{2}{3}\sqrt{3}\left(\frac{\sqrt{3}}{3}-1\right) & 1 & 0 \\ \frac{0.5\left(\Theta+\operatorname{arctanh}\left(\frac{\sqrt{3}}{3}\right)\right)}{\frac{\sqrt{3}}{3}-1} & 0 & 0.99 \end{pmatrix}$$

and consequently the simplified (see above) eigenvalues are:

$$\begin{aligned} \lambda_1(\Theta) &= 1 \\ \lambda_2(\Theta) &= 0.5773502692\Theta - 0.5\sqrt{1.333333333\Theta^2 - 4.025344939\Theta + 3.052578869} \\ &\quad +0.1148270019 \\ \lambda_3(\Theta) &= 0.5773502692\Theta + 0.5\sqrt{1.333333333\Theta^2 - 4.025344939\Theta + 3.052578869} \\ &\quad +0.1148270019 \end{aligned}$$

As before, the solutions for $|\lambda_i(\Theta)| = 1$ are calculated to determine the boundaries for the stable region, which are given by:

$$\forall i = 1,2,3 : |\lambda_i(\Theta)| \leqslant 1 \quad \Rightarrow \quad \Theta \in [-0.65, 0.15].$$

The eigenvalues for the lower target values are plotted against Θ with the method described above (see fig. C.1). The figure C.1 and the table C.1 also include the eigenvalues for different parameter settings. The calculations are omitted at this point, as they identical to the calculations presented above, except for the modifications made to the plasticity parameters $\alpha, \beta, \gamma, \delta$.

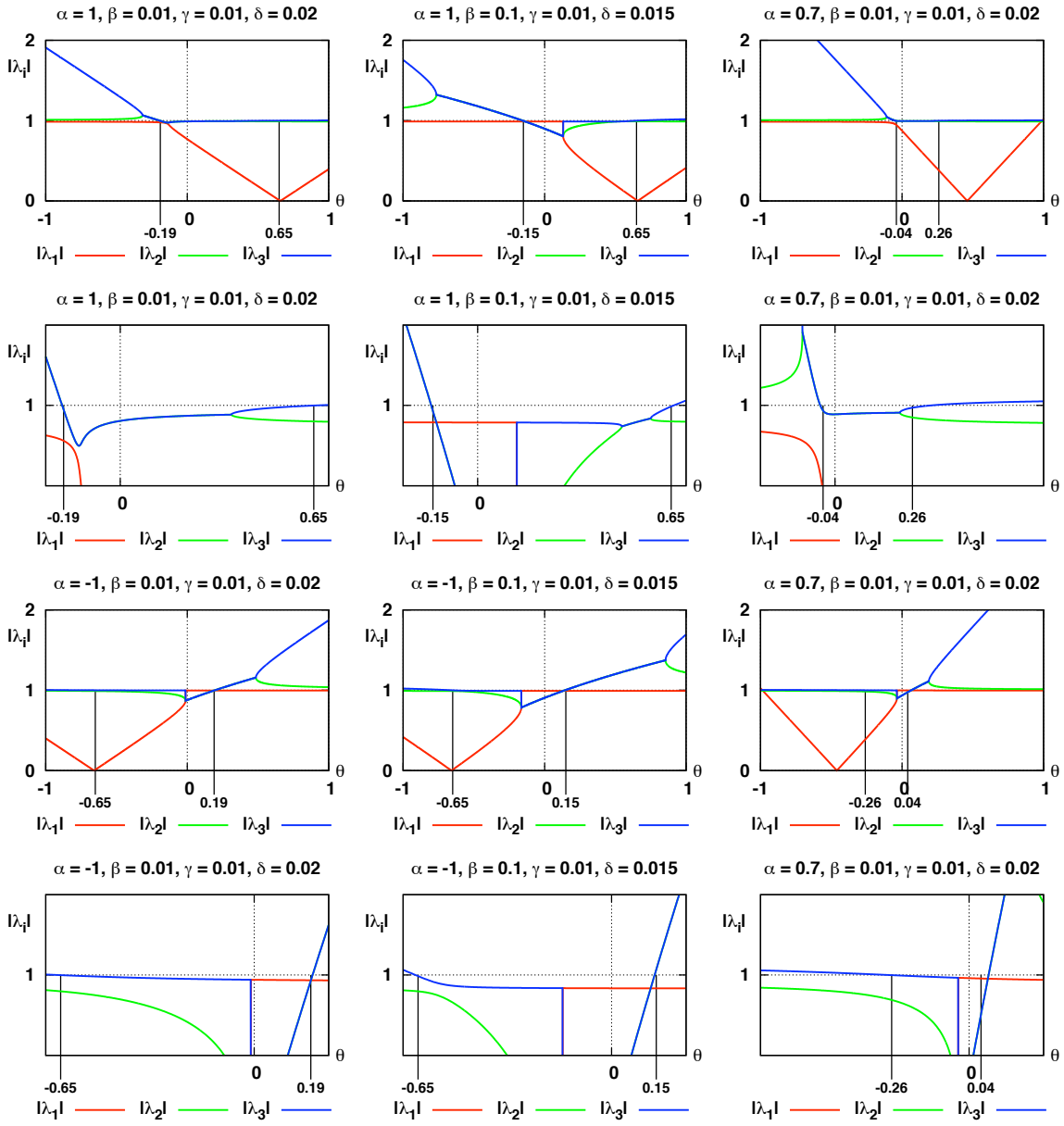


Figure C.1: Eigenvalues of the Jacobian matrix. From left to right (columns): $(\alpha, \beta, \gamma, \delta) = (1, 0.1, 0.01, 0.015)$, $(\alpha, \beta, \gamma, \delta) = (1, 0.01, 0.01, 0.02)$, $(\alpha, \beta, \gamma, \delta) = (0.7, 0.01, 0.01, 0.02)$. From top to bottom (rows): Eigenvalues for the positive target value for $\Theta \in [-1, 1]$. Clipping of the region of interest of the eigenvalue for the positive target value with $\Theta \in [-0.25 : 0.7]$, Eigenvalues for the positive target value for $\Theta \in [-1, 1]$. Clipping of the region of interest of the eigenvalue for the positive target value with $\Theta \in [-0.7 : 0.25]$. The chapter *Self-Regulating Neuron Model* (see chap. 4) discusses the different homeostatic regions with respect to the plasticity parameters. Differences in the values of the stable regions between the plots and the table (see last column) arise from the different methods of calculation. The explanation is given in the text.

List of Figures

List of Tables

List of Algorithms

Curriculum Vitae

PROFESSIONAL

MAY 2007 – PRESENT	PhD Scholarship at the Max-Planck Institute for Mathematics in the Sciences, Leipzig, Germany
MAR. 2002 – APR. 2007	Research Associate at the Fraunhofer Institute for Intelligent Analysis and Information Systems, Sankt Augustin, Germany, (formerly Fraunhofer Institute for Autonomous Intelligent Systems)
JAN. 2007 – APR. 2007	Research Associate in the fundamental research project MACS (EU-IST)
MAY 2006 – JAN. 2007	Research Associate in the fundamental research project XPERO (EU-FET)
JAN. 2006 – MAY 2007	Research Associate and deputy project leader in the fundamental research project OUTDOOR (DFG)
JAN. 2005 – DEC. 2005	Deputy Department Chief, department INDY (Intelligent Dynamics) (INDY was fused with the Autonomous Robots department Jan. 2006)
SEP. 2004 – SEP. 2005	Elected Member of the Institute Executive Committee (ILA)
SEP. 2001 – FEB. 2002	Software Engineer Robowatch Technologies, Berlin
AUG. 2001	Research Associate University of Tübingen, Graph-Visualisation, Professor Dr. rer. nat. Michael Kaufmann
DEC. 1998 – JUL. 2001	Software Engineer, 3X direct/3X Banktechnik AG, Göppingen

EDUCATION

OCT. 1995	Studied Computer Science at the University of Tübingen
JUL. 2001	Diploma Thesis “Analysis and Visualisation of Social Networks”, supervised by Professor Dr. rer. nat. Michael Kaufmann.

OTHERS

Successful completion of the Manager Trainee Program of the Fraunhofer AIS.
Project leader of different industrial and research projects at Fraunhofer AIS.
Participated in RoboCup 2003, Padua, Italy, Team AIS-Musashi, 7th place.
Participated in RoboCup 1998, Paris, France, Team Tübingen, 2nd place.

Publications

Journal Articles

Adaptive Behaviour Control by Self-regulating Neurons, Frank Pasemann, Keyan Zahedi, Marieke Rohde, Neural Networks, accepted

Book Articles

Representing Robot-Environment Interactions by Dynamical Features of Neuro-Controllers, Martin Hülse, Keyan Zahedi, Frank Pasemann, Anticipatory Behavior in Adaptive Learning Systems, Butz, M.; Sigaud, O., Gérard, P. (ed.), LNAI 2684, pp. 222-242, Springer, 2003

Adaptive Behavior Control wit Self-Regulating Neurons, Keyan Zahedi and Frank Pasemann, 50 Years of AI, M. Lungarella, F. Iida, J. Bongard, and R. Pfeifer (eds), Festschrift, LNAI 4850, Springer, pp. 196–205, 2007

Conference Proceedings

Yars: A physical 3d simulator for evolving controllers for real robots, Keyan Zahedi, Arndt von Twickel, and Frank Pasemann, In S. Carpin and et al., editors, *SIMPAR 2008*, LNAI 5325, pages 71—82. Springer, 2008.

An Evolved Neural Network for Fast Quadrupedal Locomotion, Irene Markelić, Keyan Zahedi, Advances in Climbing and Walking Robots, Proceedings of 10th International Conference (CLAWAR 2007), pp. 65–72, 2007

Evolving Neurocontrollers in the RoboCup Domain, Keyan Zahedi, Martin Hülse, Frank Pasemann, Robotik 2004, VDI-Berichte 1841, Düsseldorf, Germany, pp. 63–70, 2004

A Modular Approach to Construction and Control of Walking Robots, Bernhard Klaassen, Keyan Zahedi, Frank Pasemann, Robotik 2004, VDI-Berichte 1841, Düsseldorf Germany, pp. 633–640, 2004

SO(2)-Networks as Neural Oscillators, Frank Pasemann, Manfred Hild, Keyan Zahedi, Computational Methods in Neural Modeling, Proceedings IWANN 2003, LNCS 2686, Mira, J., and Alvarez, J. R., (Eds.), Springer, Berlin, pp. 144-151, 2003

Evolved Neurodynamics for Robot Control, Frank Pasemann, Martin Hülse, Keyan Zahedi, European Symposium on Artificial Neural Networks 2003, M.Verleysen (ed.), D-side publications, pp. 439-444, 2003

Robo-Salamander — an approach for the benefit of both robotics and biology, Ralph Breithaupt, Jochen Dahnke, Keyan Zahedi, Joachim Hertzberg, Frank Pasemann, 5th international Conference on Climbing and Walking Robots, Philippe Bedaud (Eds.), pp. 55-62, 2002

Workshop Proceedings

Representing Robot-Environment Interactions by Dynamical Features of Neuro-Controllers, Martin Hülse, Frank Pasemann, Keyan Zahedi, The 7th International Conference on the Simulation of Adaptive Behavior

Preprints

Adaptive Behaviour Control by Self-regulating Neurons, Frank Pasemann, Keyan Zahedi, Marieke Rohde, MPI MiS 2004, Nr. 55, (2004)

Reviewed Abstracts

In the search of principles underlying cognitive phenomena, Hülse, M., Zahedi, K., Wischmann, S., and Pasemann, F., Proceedings of the 50th Anniversary Summit of Artificial Intelligence (ASAI50), 2006.

Unreviewed

Team Design of AIS-Musashi 2003, A. Bredenfeld, V. Becanovic, Th. Christaller, I. Godler, M. Hülse, G. Indiveri, K. Ishii, J. Ji, H-U. Kobialka, N. Mayer, B. Mahn, H. Miyamoto, A.F.F. Nassiraei, F. Pasemann, P.-G. Plöger, P. Schöll, M. Shimizu, K. Zahedi, RoboCup-2003: Robot Soccer World Cup VII, Springer Verlag, 2003

Team Design and Evaluation of the T-Team of the University of Tuebingen for RoboCup '98, Michael Plagge, Boris Diebold, Richard Günther, Jörn Ihlendburg, Dirk Jung, Keyan Zahedi, Andreas Zell, RoboCup-98: Robot Soccer World Cup II, Springer Verlag, 1998

Diploma Thesis

Analysis and Visualisation of Social Networks, Keyan Mahmoud Ghazi-Zahedi, University of Tübingen, 2001

Invited Article

Evolution nicht-linearer Kontroller für mobile Roboter in dynamischen Umgebungen, Keyan Zahedi, Martin Hülse, Frank Pasemann, atp - automatisierungstechnische praxis, vol. 10, pp. 94-100, Springer, 2004

Others

ISEE - A Framework for the Evolution and Analysis of Recurrent Neural Networks for Embodied Agents, Martin Hülse, Steffen Wischmann, Keyan Zahedi, ECRIM News 64, Emergent Computing, pp 33-34, 2006

Proclamation

Hereby I confirm that I wrote this thesis independently and that I have not made use of any other resources or means than those indicated.

Osnabrück, January 2009

Bibliography

Abraham, & Shaw. (1992). *Dynamics, the geometry of behavior* (Second ed.). Addison-Wesley.

Abraham, R. H., Gardini, L., & Mira, C. (1997). *Chaos in discrete dynamical systems: a visual introduction in 2 dimensions* (P. Willin, Ed.). Santa Clara, CA, USA: TELOS.

Alligood, K. T., Sauer, T. D., & Yorke, J. A. (1996). *Chaos: An introduction to dynamical systems* (Vol. XVII). Heidelberg: Springer.

Alvarez, V. A., & Sabatini, B. L. (2007). Anatomical and physiological plasticity of dendritic spines. *Annual Review of Neuroscience*, *30*(1), 79–97.

Anderson, D., & Copeland, B. J. (2002). Artificial life and the chinese room argument. *Artificial Life*, *8*(4), 371–378.

Andronov, A., Leontovich, E., Gordon, I., & Maier, A. (1973). *Qualitative theory of second-order dynamic systems.* New York: John Wiley. (Jerusalem)

Angeline, P. J., Saunders, G. M., & Pollack, J. P. (1994, January). An evolutionary algorithm that constructs recurrent neural networks. *IEEE Transactions on Neural Networks*, *5*(1), 54–65.

Arbib, M. A. (1995). *The handbook of brain theory and neural networks.* Cambridge, MA, USA: MIT Press.

Arbouzov, L., Cowan, J., Fang, A., Grosso, P., Lanz, K., Le Hégaret, P., et al. (2004). *Extensible Markup Language (XML) 1.0* (Third ed.; F. Yergeau, T. Bray, J. Paoli, C. M. Sperberg-McQueen, & E. Maler, Eds.). W3C Consortium.

Arrowsmith, D. K., & Place, C. M. (1990). *An introduction to dynamical systems.* Cambridge, UK: Cambridge University Press. (Reprinted 1994)

Ashby, W. R. (1954). *Desgin for a brain.* London, UK: Chapman & Hall Ltd. (Reprinted with corrections, First published in 1952)

Ashby, W. R. (1956). *An introduction to cybernetics.* London, UK: Chapman & Hall, London. (Available online: `http://pcp.vub.ac.be/books/IntroCyb.pdf`)

Atmel. (2008). *Atmel AVR.* `www.atmel.com/atmel/acrobat/doc1022.pdf`.

Bäck, T. (1996). *Evolutionary algorithms in theory and practice: evolution strategies, evolutionary programming, genetic algorithms.* Oxford, UK: Oxford University Press.

Bailey, C., Giustetto, M., Huang, Y., Hawkins, R., & Kandel, E. (2002). Is heterosynaptic modulation essential for stabilizing hebbian plasticity and memory? *Nature Reviews in Neuroscience*, *1*, 11–20.

Bamon, R., & Roussarie, L. (1996). Dynamical systems. In R. Bamon, J.-M. Gambaudo, & S. Martinez (Eds.), (pp. 4–8). Paris: Hermann.

Barto, A. G., Sutton, R. S., & Anderson, C. W. (1983). Neuronlike adaptive elements that

can solve difficult learning control problems. *IEEE Transactions on Systems, Man, and Cybernetics*, *SMC-13*, 834–846.

Beer, R. (1996). Towards the evolution of dynamical neural networks for minimally cognitive behavior. In P. Maes, M. Mataric, J.-A. Meyer, J. Pollack, & S. Wilson (Eds.), *From animals to animats* (Vol. 4, pp. 421–429). Cambridge, MA: MIT Press.

Bi, G.-Q. (2002). Spatiotemporal specificity of synaptic plasticity: cellular rules and mechanisms. *Biological Cybernetics*, *87*, 319–332.

Bienenstock, E. L., Cooper, L. N., & Munro, P. W. (1982). Theory for the development of neuron selectivity: orientation specificity and binocular interaction in visual cortex. *The Journal of Neuroscience*, *2*(1), 32–48.

Bliss, T. V. P., & Lømo, T. (1973). Long-lasting potentiation of synaptic transmission in the dendate area of anaesthetized rabbit following stimulation of the prefrant path. *Physiology*, *232*, 551–556.

Bois-Reymond, E. D. (1848). *Untersuchungen Über Thierische Elektricität* (Vols. 1, 2 ed.). Berlin: Reimer.

Bongard, J. C., & Pfeifer, R. (2001, 7–11 July). Repeated structure and dissociation of genotypic and phenotypic complexity in artificial ontogeny. In L. Spector et al. (Eds.), *Proceedings of the genetic and evolutionary computation conference (GECCO-2001)* (pp. 829–836). San Francisco, California, USA: Morgan Kaufmann.

Box, G. (1957). Evolutionary operation: A method for increasing industrial productivity. *Applied Statistics*, *6*(2), 81–101.

Braitenberg, V. (1984). *Vehicles.* Cambridge MA: MIT Press.

Brandes, U. (2000, May). *Faster evaluation of shortest-path based centrality indices.* (Vol. 120; Preprint: Konstanzer Schriften in Mathematik und Informatik No. 120).

Brette, R., Rudolph, M., Carnevale, T., Hines, M., Beeman, D., Bower, J., et al. (2007, December). Simulation of networks of spiking neurons: A review of tools and strategies. *Journal of Computational Neuroscience*, *23*(3), 349–398.

Britannia.com, L. (2006, July). *Britannia.* `http://www.britannia.com/history/berks/index.html`.

Brooks, R. A. (1986, March). A robust layered control system for a mobile robot. *IEEE Journal of Robotics and Automation*, *2*(1), 14–23.

Brooks, R. A. (1989). *A robot that walks; emergent behaviors from a carefully evolved network* (Tech. Rep.). Cambridge, MA, USA.

Brooks, R. A. (1990). Elephants don't play chess. In P. Maes (Ed.), *Designing autonomous agents: Theory and practice from biology to engineering and back* (pp. 3–15). The MIT Press: Cambridge, MA, USA.

Brooks, R. A. (1991a). How to build complete creatures rather than isolated cognitive architectures. In K. Vanlehn (Ed.), *Architectures for intelligence.* (Erlbaum: Hillsdale, NJ).

Brooks, R. A. (1991b). Intelligence without reason. In J. Myopoulos & R. Reiter (Eds.), *Proceedings of the 12th international joint conference on artificial intelligence (IJCAI-91)* (pp. 569–595). Sydney, Australia: Morgan Kaufmann publishers Inc.: San Mateo, CA, USA.

Brooks, R. A. (1991c). Intelligence without representation. *Artificial Intelligence*, *47*(1–3), 139–159.

Cajal, S. R. y. (1892). A new concept of the histology of the central nervous system. *Hafner, 1977*, 7–29.

Cajal, S. R. y. (1906). The stucture and connexions of neurons. *Nobel Lectures: Physiology or Medicine, 1901–1921*, 220–253. (Elsevier, Amsterdam (1967))

Camel, J. E., Withers, G. S., & Greenough, W. T. (1986, December). Persistence of visual cortex dendritic alterations induced by postweaning exposure to a "superenriched" environment in rats. *Behavioral Neuroscience, 100*(6), 810–813.

Cannon, W. B. (1932). *The wisdom of the body.* New York: Norton.

Castellani, G. C., Quinlan, E. M., Cooper, L. N., & Shouval, H. Z. (2001, Oct). A biophysical model of bidirectional synaptic plasticity: dependence on AMPA and NMDA receptors. *Proceedings of the National Academy of Sciences of the United States of America, 98*(22), 12772–12777.

Chained learning architectures in a simple closed-loop behavioural context. (2007, Oct). *Biological Cybernetics, Online First.*

Champneys, A., Crutchfield, J., Doole, S., Elliot, D., Klingener, F., Kennel, M., et al. (2007, September). *Sci.nonlinear faq 2.0 (sept 2003).* `http://amath.colorado.edu/faculty/jdm/faq-Contents.html`.

Chaumont, N., Egli, R., & Adami, C. (2007). Evolving virtual creatures and catapults. *Artificial Life, 13*(2), 139–157.

Chiel, H., & Beer, R. (1997). The brain has a body: adaptive behavior emerges from interactions of nervous system, body and environment. *Trends in Neuroscience, 20*(12), 553–557.

Churchland, P. M., & Churchland, P. S. (1990, January). Could a machine think? *Scientific American, 262*(1), 26–31.

Clark, A. (1996). *Being there: Putting brain, body, and world together again.* Cambridge, MA, USA: MIT Press.

Clark, J., Lipkin, D., Marsh, J., Thompson, H., Walsh, N., & Zilles, S. (1999). *XSL Transformations (XSLT) Version 1.0* (J. Clark, Ed.). W3C.

Cliff, D. (1990). Computational neuroethology: a provisional manifesto. In *Proceedings of the first international conference on simulation of adaptive behavior on from animals to animats* (pp. 29–39). Cambridge, MA, USA: MIT Press.

CMLabs Simulations Inc. (2006, July). *Vortex.* `http://www.cm-labs.com/products/vortex/`.

Cole, D. (2004). The chinese room argument. In E. N. Zalta (Ed.), *The stanford encyclopedia of philosophy.* `http://plato.stanford.edu/`: Stanford Univiversity.

Conrad, M. (1969). *Computer experiments on the evolution of coadaptation in a primitive ecosystem.* Unpublished doctoral dissertation, Stanford University.

Cooper, L. N. (1986, Dec). Neuron learning to brain organization. *Cell Biophysics, 9*(1–2), 103–144.

Cormen, T. H., Leiserson, C. E., & Rivest, R. L. (1990). *Introduction to algorithms* (Twenty second printing (1999) ed.). Cambridge, MA, USA: MIT Press/McGraw-Hill.

Cowan, W., Südhof, T., & Stevens, C. (Eds.). (2001). *Synapses.* Baltimore, MD: Johns Hopkins Univ. Press.

Craig, A. M., & Shatz, C. J. (2001). Synapse formation and maturation. In W. Cowan, T. Südhof, & C. Stevens (Eds.), (pp. 571–612). Baltimore, MD: Johns Hopkins Univ. Press.

Davis, G. W. (2006, March). Homeostatic control of neural activity: From phenomenology to molecular design. *Annual Review of Neuroscience*, *29*(1), 307–323.

Dayan, P., & Abbott, L. F. (2001). *Theoretical neuroscience*. MIT Press.

Der, R., Hesse, F., & Martius, G. (2005). Learning to feel the physics of a body. In *Cimca '05: Proceedings of the international conference on computational intelligence for modelling, control and automation and international conference on intelligent agents, web technologies and internet commerce Vol-2 (CIMCA-IAWTIC'06)* (pp. 252–257). Washington, DC, USA: IEEE Computer Society.

Der, R., & Pantzer, T. (1999). Emergent robot behavior from the principle of homeokinesis. In A. Löffler, F. Mondada, & U. Rückert (Eds.), *Experiments with the mini-robot Khepera. Proceedings of the 1st international khepera workshop'99.* Heinz-Nixdorf-Institut Verlagsschriftenreihe.

Dickau, R. M. (2008, March). *Magnetic pendulum.* `http://mathforum.org/advanced/robertd/magneticpendulum.html`.

Dieckmann, U. (1995). Coevolution as an autonoumous learning strategy for neuromodules. In *Supercomputing in brain research: From tomography to neural networks* (pp. 427–432). World Scientific, London, Eds. Herrmann, H.J., Wolf, D.E., and Poeppel, E.

Dieckmann, U., & Pasemann, F. (1995, June). Coevolution as an autonomous learning strategy for neuromodules. In *Proceedings of the european conference on artificial life (ECAL'95).* Granada, Spain. (Postersession)

Dinse, H. R., & Merzenich, M. (2002). Adaptation of inputs in the somatosensory system. In M. Fahle & T. Poggio (Eds.), *Perceptual learning* (pp. 19–42). MIT Press.

Di Paolo, E. A. (2000). Homeostatic adaptation to inversion of the visual field and other sensorimotor disruptions. In J.-A. Meyer, A. Berthoz, H. Floreano D. and. Roitblat, & S. Wilson (Eds.), *From animals to animats 6. proceedings of the VI international conference on simulation of adaptive behavior.* Cambridge, MA: MIT Press.

Di Paolo, E. A. (2003). Organismically-inspired robotics: Homeostatic adaptation and natural teleology beyond the closed sensorimotor loop. In K. Murase & T. Asakura (Eds.), *Dynamical systems approach to embodiment and sociality* (pp. 19–42). Adelaide, Australia: Advanced Knowledge International.

Dreyfus, H. L. (1972). *What computers can't do. the limits of artificial intelligence.* Harper and Row.

Dreyfus, H. L. (1992). *What computers still can't do. a critique of artificial reason.* Harper and Row. (First published: What computers can't do: The limits of artifial intelligence, 1972, second edition with corrections and additions, 1979. Third edition: What computers still can't do. A Critique of Artificial Reason, 1992, with additions to the previous unchanged editions.)

Eckmiller, R. (1974). Hysteresis in the static characteristics of eye position coded neurons in the alert monkey. *Pflugers Archiv. European Journal of Physiology*, *350*(3), 249–258.

Eco, U. (1998). *The island of the day before.* London, UK: Vintage. (Translated by Harcourt Brace & Company)

Elman, J. L. (1990). Finding structure in time. *Cognitive Science*, *14*(2), 179–211.

Elman, J. L. (1998). Connectionism, artificial life, and dynamical systems: New approaches to old questions. In W. Bechtel & G. Graham (Eds.), *A companion to cognitive science.*

Oxford: Basil Blackwood.

Encarnação, J., Straßer, W., & Klein, R. (1996). *Graphische Datenverarbeitung I.* R. Oldenbourg Verlag.

Evolving brain structures for robot control. (2001). In J. Mira & A. Prieto (Eds.), *Bio-inspired applications of connectionism, proceedings IWANN 2001* (Vol. LNCS 2085, pp. 410–417). MPI-MIS-Preprint 20/2001: Springer, Berlin. (MPI-MIS-Preprint 20/2001)

Farmer, J. D., & Belin, A. d'A. (1992). Artificial life: The coming evolution. In C. G. Langton, C. Taylor, J. D. Farmer, & S. Rasmussen (Eds.), (pp. 815–838). Redwood City, Calif.: Addison-Wesley. (Workshop held February, 1990 in Santa Fe, New Mexico)

Faure, P., & Korn, H. (2001, Sep). Is there chaos in the brain? I. concepts of nonlinear dynamics and methods of investigation. *324*(9), 773–793.

Felleman, D. J., & Van Essen, D. C. (1991). Distributed hierarchical processing in the primate cerebral cortex. *Cerebral cortex*, *1*(1), 1-a-47.

Fiala, B. A., Joyce, J. N., & Greenough, W. T. (1978, May). Environmental complexity modulates growth of granule cell dendrites in developing but not adult hippocampus of rats. *Experimental Neurology*, *59*(3), 372–383.

Fischer, J. (2003). *A modulatory learning rule for neural learning and metalearning in real world robots with many degrees of freedom.* Unpublished doctoral dissertation, Westfälische Wilhelms-Universität Münster.

Floreano, D. (2000). Evolutionary robotics in behavior engineering and artificial life. In T. Gomi (Ed.), *ER - evolutionary robotics symposiums 1997, 1998, 2000.* Applied AI Systems.

Floreano, D., & Mondada, F. (1996). Evolution of plastic neurocontrollers for situated agents. In P. Maes, M. Mataric, J.-A. Meyer, J. Pollack, & S. Wilson (Eds.), *4th International Conference on Simulation of Adaptive Behavior (SAB'1996).* MA: MIT Press. (P. Maes, M. Mataric, J-A. Meyer, J. Pollack, and S. Wilson (eds.))

Fogel, G. B. (2008, Feb). *Scholarpedia – evolutionary algorithms.* `http://www.scholarpedia.org/article/Evolutionary_algorithms`.

Förster, H. von. (1993). *Wissen und Gewissen : Versuch einer Brücke* (1. Aufl. ed.; S. J. Schmidt, Ed.). Frankfurt am Main, D: Suhrkamp.

Förster, H. von. (2003). *Understanding understanding - essays on cybernetics and cognition.* Springer Verlag, New York.

Friedberg, R. (1958). A learning machine: Part 1. *IBM Journal of Research and Development*, *2*(1), 2–13.

Friedman, G. (1956). *Selective feedback computers for engineering synthesis and nervous system analogy.* Unpublished master's thesis, University of California (UCLA).

Froemke, R. C., Poo, M.-M., & Dan, Y. (2005, Mar). Spike-timing-dependent synaptic plasticity depends on dendritic location. *Nature*, *434*(7030), 221–5.

Gamma, E., Helm, R., Johnson, R., & Vlissides, J. (1995). *Design patterns: Elements of reusable object-oriented software.* Addison-Wesley, Reading, MA, USA.

Gelder, T. van. (1998, October). The dynamical hypothesis in cognitive science. *The Behavioral and brain sciences*, *21*(5), 615–665.

Gelder, T. van. (1999). Dynamic approaches to cognition. In R. Wilson & F. Keil (Eds.), *The MIT encyclopedia of cognitive science* (pp. 244–6). Cambridge MA:: MIT Press.

Geng, T., Porr, B., & Worgotter, F. (2006). Fast biped walking with a sensor-driven neuronal

controller and real-time online learning. *The International Journal of Robotics Research*, *25*(3), 243–259.

Germanteam2004 (`http://www.germanteam.org/GT2004.pdf`). (2004).

Gerstner, W., & Kistler, W. M. (2002a). Mathematical formulations of hebbian learning. *Biological Cybernetics*, *87*, 404–415.

Gerstner, W., & Kistler, W. M. (2002b). *Spiking neuron models: Single neurons, populations, plasticity*. Cambridge University Press.

Geva, S., & Sitte, J. (1993). The cart pole experiment as a benchmark for trainable controllers. *IEEE Control Systems Magazine*, *13*(5), 40–51.

Ghazi-Zahedi, K. M. (2001). *Analysis and visualistion of social networks*. Diplom thesis, Universitiy of Tübingen.

Gibson, J. J. (1977). *The theory of affordances* (Vols. Perceiving, Acting, and Knowing.; R. Shaw & J. Bransford, Eds.). Hillsdale, N.J.: Elrbaum.

Goldberg, D. E. (1989). *Genetic algorithms in search, optimization, and machine learning*. Addison-Wesley Professional.

Group, K. (2007, October). *Opengl*. `http://www.opengl.org/`. (OpenGL is a registered trademark of SGI)

Hanneman, R. A., & Riddle, M. (2005). *Introduction to social network methods*. Riverside, CA (published in digital form at `http://faculty.ucr.edu/~hanneman/`): University of California, Riverside.

Harand, S. (2001). What's wrong and right about searle's chinese romm argument? In M. Bishop & J. Preston (Eds.), *Essays on searle's chinese room argument*. Oxford Univeristy Press.

Harnad, S. (1991). The symbol grounding problem. 335–346.

Harnad, S. (2005). Searle's chinese room experiment. In *Encyclopedia of philosophy*. Macmillan.

Harvey, I., Husbands, P., Cliff, D., Thompson, A., & Jakobi, N. (1997). Evolutionary robotics: the sussex approach. *Robotics and Autonomous Systems*, *20*(2–4), 205–224.

Harvey, I., Paolo, E. D., Wood, R., Quinn, M., Tuci, E., & Iridia, E. T. (2005). Evolutionary robotics: A new scientific tool for studying cognition. *Artificial Life*, *11*(1), 79–98.

Haugeland, J. (1985). *Artificial intelligence: The very idea*. Cambridge, MA: MIT Press.

Havok.com Inc. (2006, July). *Havok*. `http://www.havok.com/`.

Hebb, D. O. (1949). *The organization of behavior*. New York: John Wiley.

Helmholtz, H. v. (1850). On the rate of transmission of the nerve impulse. *Monatsber. Preuss. Akad. Wiss. Berl.*, 14–15.

Heylighen, F., & Joslyn, C. (2001). Cybernetics and second-order cybernetics. In R. Meyers (Ed.), *Encyclopedia of physical science & technology*. Academic Press, New York.

Himsolt, M. (1997). *GML: a portable graph file format* (Tech. Rep.). 94030 Passau, Germany.

Holland, J. H. (1974). Genetic algorithms and the optimal allocation of trials. *SIAM Journal of Computing*, *3*(4), 326.

Hopfield, J. (1982). Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the National Academy of Sciences of the United States of America*, *79*, 2554–2558.

Hülse, M. (2007). *Multifunktionalität rekurrenter neuronaler Netze: Synthese und Analyse nichtlinearer Kontrolle autonomer Roboter* (W. Bibel, Ed.). Berlin: Akademische Verlagsgesellschaft Aka GmbH. (PhD Thesis, University of Osnabrueck)

Hülse, M., Lara, B., Pasemann, F., & Steinmetz, U. (2001). Evolving neural behavior control for autonomous robots. In G. Dorffner, H. Bischof, & K. Hornik (Eds.), *Icann 2001* (Vol. LNCS 2130, pp. 957–962).

Hülse, M., & Pasemann, F. (2002, August). Dynamical neural schmitt trigger for robot control. In J. Dorronsoro (Ed.), *Icann '02: Proceedings of the international conference on artificial neural networks* (Vol. LNCS 2415, pp. 783–788). London, UK: Springer-Verlag.

Hülse, M., Wischmann, S., & Pasemann, F. (2004). Structure and function of evolved neuro-controllers for autonomous robots. *Connection Science*, *16*(4), 294–266.

I Support Learning. (2007, July). *ASM webpresence.* `http://www.isupportlearning.com/noflash/ASM.html`.

Izhikevich, E. M. (2007, September). *Encyclopedia of dynamical systems.* `http://www.scholarpedia.org/article/Encyclopedia_of_Dynamical_Systems`.

Jacobi, N., Husbands, P., & Harvey, I. (1995). Noise and the reality gap: The use of simulation in evolutionary robotics. In *Proceedings of the third european conference on advances in artificial life* (pp. 704–720). London, UK: Springer-Verlag.

Jäger, H., & Christaller, T. (1997). Dual dynamics: designing behavior systems for autonomous robots. In *Second international symposium on artificial life and robotics.*

Jin, Y., & Branke, J. (2005). Evolutionary optimization in uncertain environments – a survey. *IEEE Trans. Evolutionary Computation*, *9*(3), 303–317.

Jordan, M. I. (1986). *Serial order: A parallel distributed processing approach* (ICS report No. 8608). La Jolla: Institute for Cognitive Science, UCSD.

Juraska, J. M., Greenough, W. T., Elliott, C., Mack, K. J., & Berkowitz, R. (1980, June). Plasticity in adult rat visual cortex: An examination of several cell populations after differential rearing. *Behavioral and Neural Biology*, *29*(2), 157–167.

Kandel, E. R., Schwartz, J. H., & Jessell, T. M. (2000). *Principles of neural science.* (4 ed.). McGraw-Hill.

Katok, A., & Hasselblatt, B. (1999). *Introduction to the modern theory of dynamical systems* (54 ed.; G.-C. Rota, Ed.). Cambridge Press.

Kaufman, H. (1967). An experimental investigation of process identification by competitive evolution. *IEEE Trans. Systems Science and Cybernetics*, *SSC3-1*, 11–16.

Kelso, S. J. A. (1995). *Dynamic patterns: The self-organization of brain and behavior (complex adaptive systems).* The MIT Press.

Klaassen, B., Zahedi, K., & Pasemann, F. (2004). A modular approach to construction and control of walking robots. In *Robotik 2004* (pp. 633–640).

Kleinschmidt, A., Buchel, C., Hutton, C., Friston, K. J., & Frackowiak, R. S. J. (2002, May). The neural structures expressing perceptual hysteresis in visual letter recognition. *Neuron*, *34*(4), 659–666.

Klinke, R., & Silbernagl, S. (2005). *Lehrbuch der Physiologie* (Auflage: 4 ed.). Stuttgart: Thieme.

Koza, J. R. (1992). *Genetic programming: On the programming of computers by means of natural selection (complex adaptive systems).* The MIT Press.

K-Team. (1999, March). *Khepera user manual.* Ch. de Vuasset, CP 111 1028 Préverenges Switzerland.

K-Team. (2005, July). *Khepera documentation.*

Khepera Modules: `http://www.k-team.com/robots/khepera/oldbase.html`
Khepera II: `http://www.k-team.com/robots/khepera/index.html`
Khepera II Modules: `http://www.k-team.com/robots/khepera/base.html`.

Lakshmanan, M., & Rajaseekar, S. (2003). *Nonlinear dynamics – integrability, chaos and patterns.* Beriln: Springer.

Langton, C. G., Taylor, C., Farmer, J. D., & Rasmussen, S. (Eds.). (1992). *Artificial life ii: Proceedings of the workshop on artificial life.* Redwood City, Calif.: Addison-Wesley. (Workshop held February, 1990 in Santa Fe, New Mexico)

Lara, B., Hülse, M., & Pasemann, F. (2001). Evolving different neuro-modules and their interfaces to control autonomous robots. In *World multiconference on systemics, cybernetics and informatics 2001, proceedings* (Vol. IX, pp. 259–264).

Law, C. C., & Cooper, L. N. (1994, Aug). Formation of receptive fields in realistic visual environments according to the Bienenstock, Cooper, and Munro (BCM) theory. *Proceedings of the National Academy of Sciences of the United States of America*, *91*(16), 7797–7801.

Leger, C. (2006, July). *Darwin2k.* `http://www.darwin2k.com/`.

Levy, S. (1992). *Artificial life: the quest for a new creation.* New York, NY, USA: Random House Inc.

Levy, W. B., & Steward, O. (1983, Apr). Temporal contiguity requirements for long-term associative potentiation/depression in the hippocampus. *Neuroscience*, *8*(4), 791–797.

Lipson, H. (2005). Evolutionary design and evolutionary robotics. In B. Cohen (Ed.), *Biomimetics: Biologically inspired technologies* (pp. 129–155). CRC Press.

Lipson, H., & Pollack, J. B. (2000, 31 August). Automatic design and manufacture of robotic lifeforms. *Nature*(406), 974–978.

Lucas, J., & Hazes, P. (Eds.). (1982). *Proceesdings of the cognitive curriculum conference.*

Mahn, B. (2003). *Entwicklung von neurocontrollern fÃ¼r eine holonome roboterplattform.* Diploma thesis, Fachbereich Technik, Fachhochschule Oldenburg/Ostfriesland/Wilhelmshaven, Emden.

Malaka, R., & Spitzer, M. (Eds.). (2006). *Interdisciplinary College. Focus Theme: Learning.*

Malenka, R. C., & Siegelbaum, S. A. (2001). Synaptic plasticity. diverse targets and mechnisms for regulating synaptic efficancy. In W. Cowan, T. Südhof, & C. Stevens (Eds.), (pp. 393–453). Baltimore, MD: Johns Hopkins Univ. Press.

Malsburg, C. von der. (1981). *The correlation theory of brain function* (Tech. Rep. Nos. 81–2). Göttingen, Germany: Dept. Neurobiology, Max-Plank-Insititute for Biophysical Chemistry. (Also in Domany (Eds.), Models of Neural Networks II (pp. 95–119), London: Springer Verlag)

Manoonpong, P. (2007). *Neural preprocessing and control of reactive walking machines towards versatile artificial perception-action systems.* Heidelberg: Springer.

Manoonpong, P., Pasemann, F., & Fischer, J. (2004). Neural processing of auditory-tactile sensor data to perform reactive behavior of walking machines. In *Proceedings of the IEEE MechRob 2004* (pp. 189–194).

Maris, M., & Boekhorst, I. J. A. R. te. (1996). Exploiting physical constraints: Heap formation through behavioral error in a group of robots. In *IROS'96, IEEE/RSJ international conference on intelligent robots and systems, November 4-8, 1996, Senri Life Science Center, Osaka, Japan.*

Markelić, I. (2005). *Evolving a neurocontroller for a fast quadrupedal walking behavior.* Diplom thesis, University of Koblenz.

Markelić, I., & Zahedi, K. (2007). An evolved neural network for fast quadrupedal locomotion. In M. Xie & S. Dubowsky (Eds.), *Advances in climbing and walking robots, proceedings of 10th international conference (CLAWAR 2007)* (pp. 65–72). World Scientific Publishing Company. (Singapore 16-18 July 2007)

Martone, M., Gupta, A., Ellisman, M., Sargis, J., Tran, J., Wong, W., et al. (2007, March). *Cell centered database.* `http://ccdb.ucsd.edu/index.html`. (University of California)

McAllister, A. K. (2007). Dynamic aspects of cns synapse formation. *Annual Review of Neuroscience*, *30*(1), 425–450.

McCulloch, W., & Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. *Bulletin of Mathematical Biophysics*, *5/115*, 15–27.

McGeer, T. (1990). Passive dynamic walking. *International Journal of Robotic Research*, *9*(2), 62–82.

Michel, O. (2005, July). *Khepera simulator 2.0.* `http://diwww.epfl.ch/lami/team/michel/khep-sim/`.

Michel, O. (2006, July). *Webots.* `http://www.cyberbotics.com/products/webots/`.

MICRO-EPSILON. (2006, July). *Iconnect.* `http://iconnect.micro-epsilon.de/`.

Miglino, O., Lund, H. H., & Nolfi, S. (1995). Evolving mobile robots in simulated and real environments. *Artificial Life*, *2*(4), 417–434.

Milholland, J. E., Copi, I. M., & Garner, W. R. (1954, June). Book reviews. *Psychometrika*, *19*(2), 165–172.

Miller, K. D. (1996). Synaptic economics: competition and cooperation in synaptic plasticity. *Neuron*, *17*, 371–374.

Mills, D. L., Plunkett, K., Prat, C., & Schafer, G. (2005). Watching the infant brain learn words: effects of vocabulary size and experience. *Cognitive Development*, *20*(1), 19–31.

Minsky, M., & Papert, S. (1969). *Perceptrons: An introduction to computational geometry.* Cambridge, Mass.: MIT Press.

Mondada, F., Franzi, E., & Ienne, P. (1993). Mobile robot miniaturization: A tool for investigation in control algorithms. In *Proceedings of the third international symposium on experimental robotics* (pp. 501–513). Berlin: Springer Verlag.

Morris, L. G., & Hooper, S. L. (1998). Muscle response to changing neuronal input in the lobster (panulirus interruptus) stomatogastric system: Slow muscle properties can transform rhythmic input into tonic output. *The Journal of Neuroscience*, *18*(9), 3433–3442.

Neumann, J. von. (1951). The general and logical theory of automata. In A. H. Taub (Ed.), *John von Neumann, Collected Works* (pp. 288–328). Oxford: Pergamon Press. (First published 1951 as *pages 1–41 of:* L. Jeffress, A. (ed), *Cerebral Mechanisms in Behavior—The Hixon Symposium*, New York: John Wiley)

Neumann, J. von. (1966). *Theory of self-reproducing automata.* Urbana: University of Illinois Press. (Edited and completed by Arthur W. Burks)

Newell, A., & Simon, H. A. (1963). GPS, a program that simulates human thought. In E. Feigenbaum & J. Feldman (Eds.), *Computers and thought* (pp. 279–296). McGraw-Hill.

Nolfi, S., & Floreano, D. (1998). Co-evolving predator and prey robots: Do 'arm races' arise in artificial evolution? *Artificial Life*, *4*(4), 311–335.

Nolfi, S., & Floreano, D. (2000). *Evolutionary robotics.* MIT Press.

Nolfi, S., Floreano, D., Miglino, O., & Mondada, F. (1994). How to Evolve Autonomous Robots: Different Approaches in Evolutionary Robotics. In R. A. Brooks & P. Maes (Eds.), *4th International Workshop on Artificial Life.* MA: MIT Press. (R. A. Brooks and P. Maes (eds.))

Ott, E. (1993). *Chaos in dynamical systems.* Cambridge: Cambridge University Press.

Pasemann, F. (1993). Dynamics of a single model neuron. *International Journal of Bifurcation and Chaos*, *2*, 271–278.

Pasemann, F. (1995). Characteristics of periodic attractors in neural ring networks. *Neural Networks*, *8*, 421–429.

Pasemann, F. (1996). Repräsentation ohne Repräsentation - Überlegungen zu einer Neurodynamik modularer kognitiver Systeme. In G. Rush, S. J. Schmidt, & O. Breidbach (Eds.), *Interne repräsentationen – neuro konzepte der hinforschung* (pp. 42–91). Frankfurt: Suhrkamp (stw 1277).

Pasemann, F. (1997a, October). Pole-balancing with different evolved neurocontrollers. In W. Gerstner, A. Germond, M. Hasler, & J.-D. Nicoud (Eds.), *Artificial neural networks - icann'97* (Vol. LNCS 1327, pp. 823–829). Lausanne, Switzerland: Springer, Berlin.

Pasemann, F. (1997b). A simple chaotic neuron. *Physica D. Nonlinear phenomena*, *104*, 205–211.

Pasemann, F. (1998). Evolving neurocontrollers for balancing an inverted pendulum. *Network: Computation in Neural Systems*, *9*, 495–511.

Pasemann, F. (2002). Complex dynamics and the structure of small neural networks. *Network: Computation in Neural Systems*, *13*(2), 195–216.

Pasemann, F., & Dieckmann, U. (1997a). *Balancing rotators with evolved neurocontrollers* (Tech. Rep.). Max Planck Insitute for Mathematics in the Sciences. MIS-MPG-preprint 97/37.

Pasemann, F., & Dieckmann, U. (1997b, June). Evolved neurocontrollers for pole-balancing. In J. C. J. Mira R. Moreno-Diaz (Ed.), *Biological and artificial computation: From neuroscience to technology, proceedings iwann'97* (pp. 1279–1287). Lanzarote, Canary Islands, Spain: Springer, Berlin.

Pasemann, F., Hild, M., & Zahedi, K. (2003). So(2)-networks as neural oscillators. In J. Mira & J. Alvarez (Eds.), *Computational methods in neural modeling, proceedings iwann 2003* (pp. 144–151). Berlin: Springer.

Pasemann, F., Hülse, M., & Zahedi, K. (2003). Evolved neurodynamics for robot control. In M. Verleysen (Ed.), *European symposium on artificial neural networks* (pp. 439–444). D-side publications.

Pasemann, F., Steinmetz, U., Hülse, M., & Lara, B. (2001). Robot control and the evolution of modular neurodynamics. *Theory in Biosciences*, *120*, 311–326.

Pasemann, F., Zahedi, K., & Rohde, M. (2004). *Adaptive behaviour control by self-regulating neurons* (Preprint No. 55). MPI MiS.

Pfeifer, R. (2007, February). *Talking robots podcast.* `http://lis.epfl.ch/resources/podcast/2007/02/rolf-pfeifer-new-ai.html`, Producer: Markus Waibel, EPFL.

Pfeifer, R., & Bongard, J. C. (2006). *How the body shapes the way we think: A new view of intelligence.* The MIT Press (Bradford Books).

Pfeifer, R., & Scheier, C. (1999). *Understanding intelligence.* Cambridge, MA, USA: MIT Press.

Pinker, S. (1999). *How the mind works.* W. W. Norton & Company.

Pollack, J., & Lipson, H. (2000, 13–15 July). The golem project: Evolving hardware bodies and brains. In J. Lohn, A. Stoica, & D. Keymeulen (Eds.), *The second NASA/DoD workshop on evolvable hardware* (pp. 37–42). Palo Alto, California: IEEE Computer Society.

Pollack, J., Lipson, H., Funes, P., Ficici, S., & Hornby, G. (1999). Coevolutionary robotics. In *EH '99: Proceedings of the 1st nasa/dod workshop on evolvable hardware* (p. 208). Washington, DC, USA: IEEE Computer Society.

Popp, J. (2005, November). *sphericalrobots.* `http://www.sphericalrobots.org`.

Porr, B. (2003). *Sequence-learning in a self-referential closed-loop behavioural system.* Unpublished doctoral dissertation, Faculty of Human Sciences, Department of Psychology.

Porr, B., & Wörgötter, F. (2003). Isotropic sequence order learning. *Neural Computation*, *15*(4), 831–864.

Port, R. F., & Van Gelder, T. (Eds.). (1998). *Mind as motion: Explorations in the dynamics of cognition.* Cambridge, MA, USA: MIT Press.

Rabinovich, M. I., Varona, P., Selverston, A. I., & Abarbanel, H. D. I. (2006). Dynamical principles in neuroscience. *Reviews of Modern Physics*, *78*(4).

Rechenberg, I. (1965). Cybernetic solution path of an experimental problem. *Royal Aircraft Establishment, Library Translation*, *1122*.

Relais d'information sur les science la cognitin. (2007, July). *Swarm robotics - collective light seeking of 5 ASM II robots.* `http://www.risc.cnrs.fr/detail_films.php?ID=536`.

Riedmiller, M. (1996). Learning to control dynamic systems. In R. Trappl (Ed.), *Proceedings of the 13th. european meeting on cybernetics and systems research – EMCSR '96* (pp. 1055–1063). Vienna.

Rosemann, M. (2004). *Visualisierung der Aktivität von Neurokontrollern auf autonomen mobilen Robotern.* Master thesis, Fachbereich Technik, Fachhochschule Oldenburg/Ostfriesland/Wilhelmshaven, Emden, Germany.

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

Royer, S., & Paré, D. (2003, April). Conservation of total synaptic weight through balanced synaptic depression and potentiation. *Nature*, *422*(6931), 518–522.

Rumelhart, D., Hinton, G., & Williams, R. (G1986). Learning representation by back-propagating errors. *Nature*, *323*, 533–536.

Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning internal representations by error propagation. 318–362.

Rumelhart, D. E., McClelland, J. L., & the PDP Research Group. (1986). *Parallel distributed processing: Explorations in the microstructure of cognition, volumes 1 and 2.* Cambridge: MIT Press.

Sarbadhikari, S. N., & Chakrabarty, K. (2001, Sep). Chaos in the brain: a short review alluding to epilepsy, depression, exercise and lateralization. *Medical engineering & physics*, *23*(7), 445–455.

Schwefel, H.-P. (1965). *Kybernetische Evolution als Strategie der experimentellen Forschung in der Strömungstechnik*. Unpublished master's thesis, Technical Univ. Berlin.

Searle, J. R. (1980). Minds, brains, and programs. *The Behavioral and Brain Sciences*, *3*(3), 417–457.

Seung, S. (2007, September). *Introduction to neurons and classical neurodynamics.* `http://hebb.mit.edu/courses/9.641/lectures/lecture01.pdf`. (Lecture Notes)

Sims, K. (1994). Evolving 3d morphology and behavior by competition. *Artificial Life*, *1*(4), 353–372.

Skarda, C., & Freeman, W. (1987). How brains makes chaos in order to make sense of the world. *The Behavioral and brain sciences*, *10*, 161–195.

Smith, R. (2005, Nov). *ODE.* `www.ode.org`.

Smythies, J. R. (2002). *The dynamic neuron.* Cambridge, Massachusetts, London, England: MIT Press, Bradford Books.

Socolar, J. E. S. (2006). Nonlinear dynamical systems. In E. Micheli-Tzanakou (Ed.), (pp. 115–140). Springer US.

Software, M. (2006, July). *Adams.* `http://www.mscsoftware.com/products/adams.cfm`.

Song, S., Miller, K., & Abbott, L. (2000). Competitive hebbian learning through spiketime-dependent synaptic plasticity. *Nature neurosciences*, *3*, 919–926.

Spenneberg, D. (2008, April). *Scorpion project website.* `http://www.dfki-bremen.de/robotik/forschung/projekte/weltraumrobotik/scorpion/`.

Spong, M. (1998). Underactuated mechanical systems. In B. Siciliano & K. P. Valavanis (Eds.), *Control problems in robotics and automation* (Vol. 230, pp. 135–150). London: Springer Verlag.

Squire, L. R., & Kandel, E. R. (1998). *Memory: From mind to molecules.* New York: Scientific American Library.

Steels, L. (2007). The symbol grounding problem is solved, so what's next? In M. De Vega, G. Glennberg, & G. Graesser (Eds.), *Symbols, embodiment and meaning.* Academic Press, New Haven.

Stein, R. B. (1967, January). The frequency of nerve action potentials generated by applied currents. *Royal Society of London Proceedings Series B*, *167*, 64–86.

Stewart, I. (1999). *Does god play dice? the mathematics of chaos.* Massachusetts, USA: Blackwell Publishers Inc.

Streeting, S., Xie, J., Castaneda, P. J., Muldowney, T., Doyle, J., O'Sullivan, J., et al. (2006, July). *Ogre.* `http://www.ogre3d.org`.

Strogatz, S. H. (1994). *Nonlinear dynamics and chaos.* Reading, MA, USA: Addison-Wesley.

Stroustrup, B. (2000). *The c++ programming language: Special edition* (Third Edition ed.). Addison-Wesley.

Sun Microsystems. (2007). *Java.* `http://java.sun.com`.

Tani, J. (2007, March). *Talking robots podcast.* `http://lis.epfl.ch/resources/podcast/2007/02/rolf-pfeifer-new-ai.html`, Producer: Markus Waibel, EPFL.

Taub, A. H. (Ed.). (1961). *John von Neumann: Collected Works. Volume V: Design of Computers, Theory of Automata and Numerical Analysis.* Oxford: Pergamon Press.

The RoboCup Federation. (2007, Novembre). *Robocup.* `http://www.robocup.org`.

Thomas, V. (2008). *Evolution einer kamerabasierten Neurokontrolle autonomer Roboter unter Berücksichtigung der Eigenbewegung.* Unpublished master's thesis.

Thomson, J. M. T., & Stewart, B. (2002). *Nonlinear dynamics and chaos* (Second Edition ed.). New York, NY, USA: John Wiley and Sons, LTD.

Tsodyks, M., Kenet, T., Grinvald, A., & Arieli, A. (1999). Linking spontaneous activity of single cortical neurons and the underlying functional architecture. *Science*, *286*(5446), 1943–1946.

Turrigiano, G. G. (1999, May). Homeostatic plasticity in neuronal networks: the more things change, the more they stay the same. *Trends in Neuroscience*, *22*(5), 221–7.

Turrigiano, G. G., Leslie, K. R., Desai, N. S., Rutherford, L. C., & Nelson, S. B. (1998, Feb). Activity-dependent scaling of quantal amplitude in neocortical neurons. *Nature*, *391*(6670), 892–6.

Twickel, A. von, & Pasemann, F. (2006). Reflex-oscillations in evolved single leg neurocontrollers for walking machines. *Natural Computing*.

Uylings, H. B., Kuypers, K., Diamond, M. C., & Veltman, W. A. (1978, Dec). Effects of differential environments on plasticity of dendrites of cortical pyramidal neurons in adult rats. *Exp Neurol*, *62*(3), 658–677.

Valverde, F. (1971, Oct). Rate and extent of recovery from dark rearing in the visual cortex of the mouse. *Brain Res*, *33*(1), 1–11.

Verbeek, C. (2008). *Robotino.* `http://www.openrobotino.org`. (Follow-up of Robertino)

Viklund, A. (2006, July). *JFreeChart.* `http://www.jfree.org/jfreechart/`.

Volkmar, F. R., & Greenough, W. T. (1972). Rearing complexity affects branching of dendrites in the visual cortex of the rat. *Science*, *176*(4042), 1445–1447.

Walker, J., Garrett, S., & Wilson, M. (2003, September). Evolving controllers for real robots: A survey of the literature. *Adaptive Behavior*, *11*(3), 179–203.

Walker, J. F., & Oliver, J. H. (1997). *A survey of artificial life and evolutionary robotics.* `http://citeseer.comp.nus.edu.sg/walker97survey.html`.

Wallace, C. S., Kilman, V. L., Withers, G. S., & Greenough, W. T. (1992, Jul). Increases in dendritic length in occipital cortex after 4 days of differential housing in weanling rats. *Behavioral and neural biology*, *58*(1), 64–68.

Walter, W. G. (1950, May). An imitation of life. *Scientific American*, *182*(5), 42–45.

Walter, W. G. (1951, August). A machine that learns. *Scientific American*, *185*(2), 60–63.

web3D Consortium. (1997). *Vrml '97.* `http://www.web3d.org/x3d/specifications/vrml/ISO-IEC-14772-VRML97/`. New York, NY, USA: ACM Press.

web3D Consortium. (2006, July). *X3d.* `http://www.web3d.org/x3d/specifications/x3d\_specification.html`.

Welch, R. B. (1974). Research on adaptation to rearranged vision. *Perception*, *3:*, 367–392.

Wiener, N. (1948). *Cybernetics.* Cambridge, Massachusetts: The Technology Press; New York: John Wiley & Sons, Inc.

Wiener, N. (1954). *The human use of human beings - cybernetics and society.* New York: Avon Books.

Williams, H. (2004). Homeostatic plasticity in recurrent neural networks. In S. Schaal, A. Ijspeert, A. Billard, S. Vijayakumar, J. Hallam, & J.-A. Meyer (Eds.), *Proceedings*

of the eighth international conference on simulation of adaptive behavior. Cambridge, MA: MIT Press / Bradford Books.

Williams, T., & Kelley, C. (2007, July). *gnuplot 4.2.* `http://www.gnuplot.info`.

Wilson, C. J., Weyrick, A., Terman, D., Hallworth, N. E., & Bevan, M. D. (2004, May). A model of reverse spike frequency adaptation and repetitive firing of subthalamic nucleus neurons. *Journal of Neurophysiology*, *91*(5), 1963–1980.

Wischmann, S. (2007). *Neural dynamics of social behavior: An evolutionary and mechanistic perspective on communication, cooperation, and competition among situated agents.* Unpublished doctoral dissertation, University of Bonn.

Wischmann, S., Hülse, M., & Pasemann, F. (2005). (co)evolution of (de)centralized neural control for a gravitationally driven machine. *Advances in Artificial Life*, 179–188.

Wischmann, S., & Pasemann, F. (2004). From Passive to Active Dynamic 3D Bipedal Walking - An Evolutionary Approach. In M. Armada & P. Gonzalez de Santos (Eds.), *CLAWAR 2004* (pp. 737–744).

Wisse, M. (2004). *Essentials of dynamic walking: Analysis and design of two-legged robots.* Unpublished doctoral dissertation, Technische Universiteit Delft.

Wisspeintner, T., & Bose, A. (2005). The volksbot concept - rapid prototyping for real-life applications in mobile robotics. *it - Information Technology*, *5*, 274–281.

Wolbers, T., & Büchel, C. (2005, Mar). Dissociable retrosplenial and hippocampal contributions to successful formation of survey representations. *The Journal of Neuroscience*, *25*(13), 3333–3340.

Ye, B., Zhang, Y., Song, W., Younger, S. H., Jan, L. Y., & Jan, Y. N. (2007, Aug). Growing dendrites and axons differ in their reliance on the secretory pathway. *Cell*, *130*(4), 717–729.

Zahedi, K., & Hülse, M. (2008, January). *ISEE – integrated structure envolution environment.* `http://sourceforge.net/projects/isee/`.

Zahedi, K., Hüsle, M., & Pasemann, F. (2004). Evolving neurocontrollers in the robocup domain. In *Robotik 2004* (pp. 63–70).

Zahedi, K., Laue, T., Röfer, T., Schöll, P., & Spiess, K. (2005, Januar). *RoSiML – robot simulation markup language.* `http://www.tzi.de/spprobocup/RoSiML.html`.

Zahedi, K., & Pasemann, F. (2007). Adaptive behavior control with self-regulating neurons. In M. Lungarella, F. Iida, J. Bongard, & R. Pfeifer (Eds.), *50 years of ai* (Vol. 4850, pp. 196–205). Berlin Heidelberg: Springer.

Zahedi, K., Twickel, A. von, & Wischmann, S. (2007, October). *YARS – yet another robot simulator.* `http://sourceforge.net/projects/yars/`.

Zigmond, M. J., Bloom, F. E., Landis, S. C., Roberts, J. L., & Squire, L. R. (1999). *Fundamental neuroscience.* San Diego, California: Academic Press.