RUPRECHT-KARLS-UNIVERSITÄT HEIDELBERG

Mihai Alexandru Petrovici

Form vs. Function: Theory and Models for Neuronal Substrates

Dissertation

UHEI-KIP

KIRCHHOFF-INSTITUT FÜR PHYSIK

Faculty of Physics and Astronomy

University of Heidelberg

Dissertation

in Physics

submitted by

Mihai Alexandru Petrovici

born in Bucharest, Romania

UHEI-KIP

Form vs. Function:
Theory and Models for Neuronal Substrates

This dissertation has been carried out by

Mihai Alexandru Petrovici

at the

Kirchhoff Institute for Physics

Ruprecht-Karls-Universität Heidelberg

under the supervision of

Prof. Dr. Karlheinz Meier

Form vs. Function:
Theory and Models for Neuronal Substrates

The quest for endowing form with function represents the fundamental motivation behind all neural network modeling. In this thesis, we discuss various functional neuronal architectures and their implementation in silico, both on conventional computer systems and on neuromorpic devices. Necessarily, such casting to a particular substrate will constrain their form, either by requiring a simplified description of neuronal dynamics and interactions or by imposing physical limitations on important characteristics such as network connectivity or parameter precision. While our main focus lies on the computational properties of the studied models, we augment our discussion with rigorous mathematical formalism. We start by investigating the behavior of point neurons under synaptic bombardment and provide analytical predictions of single-unit and ensemble statistics. These considerations later become useful when moving to the functional network level, where we study the effects of an imperfect physical substrate on the computational properties of several cortical networks. Finally, we return to the single neuron level to discuss a novel interpretation of spiking activity in the context of probabilistic inference through sampling. We provide analytical derivations for the translation of this “neural sampling” framework to networks of biologically plausible and hardware-compatible neurons and later take this concept beyond the realm of brain science when we discuss applications in machine learning and analogies to solid-state systems.

Form vs. Funktion:
Theorie und Modelle für neuronale Substrate

Das Streben nach der Ausstattung von Form mit Funktion repräsentiert die fundamentale Motivation von neuronaler Netzwerkmodellierung. In dieser Arbeit diskutieren wir verschiedene neuronale Architekturen und ihre Implementierung in siliziumbasierten Substraten, sowohl auf konventionellen Computersystemen als auch auf neuromorpher Hardware. Notwendigerweise wird eine solche Abbildung auf ein bestimmtes Substrat ihre Form einschränken, entweder durch die Erfordernis einer vereinfachten Beschreibung neuronaler Dynamik und Wechselwirkung oder durch das Auferlegen physikalischer Einschränkungen auf wichtige Eigenschaften wie etwa die Netzwerkkonnektivität oder die Präzision einzelner Parameter. Während unser Hauptaugenmerk auf der Rechenfähigkeit der untersuchten Modelle liegt, ergänzen wir unsere Diskussion mit rigorosen mathematischen Formalismen. Wir beginnen mit einer Untersuchung des Verhaltens von Punktneuronen unter synaptischem Beschuss und liefern analytische Vorhersagen über statistische Eigenschaften einzelner Neuronen und neuronaler Ensembles. Diese Überlegungen werden später nützlich, wenn wir zu einer funktionalen Netzwerkebene übergehen, auf der wir die Effekte imperfekter Physikalischer Substrate auf die Recheneigenschaften einiger kortikaler Modelle untersuchen. Zum Schluss kehren wir auf die Ebene einzelner Neuronen zurück, um eine neue Interpretation von Feuermustern im Kontext von stichprobenbasierter probabilistischer Inferenz zu diskutieren. Wir liefern analytische Herleitungen für die Übersetzung dieses sogenannten “neural-sampling”-Konzeptes in biologisch plausible und hardwarekompatible neuronale Netzwerke. Später überqueren wir auch die Grenzen der reinen Neurowissenschaft indem wir Anwendungen auf maschinelles Lernen und Analogien zu physikalischen Festkörpern erörtern.

Contents

"For a while, almost everyone was nervous about switching from the brain to the jewel. In the early days, the jewel was a separate device that learned its task by mimicking the brain, and it had to be handed control of the body at one chosen moment. It took another fifty years before it could be engineered to replace the brain incrementally, neuron by neuron, in a seamless transition throughout adolescence."
So Grace had lived to see the jewel invented, but held back, and died before she could use it? Jamil kept himself from blurting out this conclusion; all his guesses had proved wrong so far.
Margit continued. "Some people weren't just nervous, though. You'd be amazed how vehemently Ndoli was denounced in certain quarters. And I don't just mean the fanatics who churned out paranoid tracts about 'the machines' taking over, with their evil inhuman agendas. Some people's antagonism had nothing to do with the specifics of the technology. They were opposed to immortality, in principle."
Jamil laughed. "Why?"
"Ten thousand years' worth of sophistry doesn't vanish overnight," Margit observed dryly. "Every human culture had expended vast amounts of intellectual effort on the problem of coming to terms with death. Most religions had constructed elaborate lies about it, making it out to be something other than it was - though a few were dishonest about life, instead. But even most secular philosophies were warped by the need to pretend that death was for the best.
"It was the naturalistic fallacy at its most extreme - and its most transparent, but that didn't stop anyone. Since any child could tell you that death was meaningless, contingent, unjust, and abhorrent beyond words, it was a hallmark of sophistication to believe otherwise. Writers had consoled themselves for centuries with smug puritanical fables about immortals who'd long for death - who'd beg for death. It would have been too much to expect all those who were suddenly faced with the reality of its banishment to confess that they'd been whistling in the dark. And would-be moral philosophers - mostly those who'd experienced no greater inconvenience in their lives than a late train or a surly waiter - began wailing about the destruction of the human spirit by this hideous blight. We needed death and suffering, to put steel into our souls! Not horrible, horrible freedom and safety!"

Greg Egan, Border Guards, 1999

1. Prologue

The result is that the philosophy of mind is unique among contemporary philosophical subjects, in that all of the most famous and influential theories are false.

John Searle, Mind, 2004

Searle is right, of course – and the reason is quite obvious. Compared to other natural sciences such as physics or astronomy, neuroscience is only a very young discipline. Surgeons have been repairing broken skulls since the Neolithic Era and even treating mental disorders with trepanation, but they had only naked-eye phenomenological evidence to work with. The lack of proper means of observation gave rise to some – from our modern perspective – rather ludicrous medical theories, which remained surprisingly dominant in Western medicine for several millennia.[1] Any attempt for a theory of mind formulated before the 20th century was therefore as doomed to fall short of physical reality as an explanation of ferromagnetism[2] or radioactivity without an understanding of quantum mechanics. To add insult to injury, the dominance of theology on metaphysics was not particularly helpful either, as, for example, Searle convincingly demonstrates in his critical analysis of Cartesian dualism (Searle, 2004).

It is only for little over a century that we have started gaining insight into the "quanta" of thought, which Golgi and Cajal have identified to be single cells – the neurons. It is even quite likely that an even more microscopic level of description is needed for a thorough understanding of thought, if we consider the complex interaction of genes, proteins and neurotransmitters. In any case, if we are ever to appease the philosophers' search (and, in fact, our deeply human desire) for a correct theory of mind, our understanding must reach down to at least the level of neuron and synapse dynamics.

However, an understanding of microscopic dynamics is clearly not all there is when it comes to explaining macroscopic phenomena. Even if microscopic interactions completely determine the macroscopic behavior of dynamical systems, it is often difficult, if not impossible to comprehend the behavior of large systems in terms of their components' behavior alone. The semiclassical Ising model is a prime example for the emergence of complex phenomena such as phase transitions from relatively simple interactions between pairs of particles. An understanding of high-level ensemble phenomena must therefore

[1] In an address delivered at the International Congress of Physiology, Richet (1910) comments on the theory of humorism as follows: *But what is truly extraordinary, what surpasses our wildest dreams, is the fact that for sixteen hundred years all physicians and all physiologists remained bound in the shackles of this incomprehensible error of the four cardinal humours. By what miracle was the spirit of conservatism or of routine able to hide the truth to such a degree? The men of science and the doctors of former times were no less intelligent than those of to-day. Nevertheless they accepted without a shadow of proof these childish theories; they could not see most simple facts, and they saw most complicated things which not only did not exist but which were not even probable.*

[2] See (Feynman et al., 2011) for an intuitive explanation of the Bohr – van Leeuwen theorem.

be based on a thorough description of microscopic dynamics, but must also operate at a higher level of abstraction, defining new (macroscopic) properties and variables which obey their own laws of motion.

Under these premises, we are compelled to argue in favor of a functionalist approach to neuroscience in general and theoretical and computational neuroscience in particular. Indeed, although not often explicitly stated, functionalism is fundamental to all natural sciences, but this notion comes loaded with etymological baggage due to its prominence in the philosophy of mind. We must therefore establish what we consider to be the fundamental tenets of functionalism in the context of theoretical and computational neuroscience.

Firstly, we argue that the mathematical equations that describe generalized laws of motion do not always have to be exact in order to provide a useful description of the behavior of a physical system. Although technically incorrect, Newtonian mechanics is, for all practical purposes, sufficient for describing phenomena from the smallest (particle trajectories in monoatomic gases) to the largest (planetary orbits[3]) of scales. A useful approximation is a fundamental requirement whenever a complex physical system is to be described with a tractable amount of reasonably complicated equations. While being arguably the most complex among known physical systems, biological neural networks must be amenable to a similar treatment. This is, for example, the main argument behind our use of point neurons, which are also ubiquitous in contemporary theoretical and computational neuroscience.[4]

Secondly, we argue that equations do not necessarily need to model fundamental quantities to be useful. In particular, "high-level" ensemble observables such as temperature and pressure or even more abstract quantities such as entropy are not only practical, but become even necessary to describe and understand particular aspects of physical reality. This applies not only to the description of neural ensembles, but also to the information that functional units in the brain (single neurons, cortical microcircuits) may exchange. While there are certainly situations in which individual spike timing is essential, information can also be encoded in, e.g., spatiotemporal firing rate patterns (Decharms and Zador, 2000). If then, for example, a neural population encodes information in its firing rate only, then a functional representation of this population by means of a single number – its firing rate – is justified (and individual membrane potentials may be neglected).

Thirdly, we must define the notion of faithful simulation. Consider a physical system – in our case, this is usually a neural network – that we wish to model using a different physical device, usually a silicon substrate. If we define a set of dynamic variables to be a full representation of all the relevant information about the system to be modeled, then any other physical system that, at some level of abstraction, encodes a representation of these variables, with sufficiently similar[5] dynamics to their original counterparts, is a faithful simulation of the original physical system. In other words, a simulation that

[3] Yes, even for Mercury, for which the amount of the perihelion precession caused by general relativity amounts to less than an arc-minute per century (Clemence, 1947).

[4] This does not mean that the spatial structure of the dendritic tree has no computational properties. We make this clear in Section 2.1.4. However, for the theoretical and computational models discussed here, we choose the point neuron model as an appropriate level of abstraction. Any statements about biology must, of course, be validated with appropriate data.

[5] The required precision can admittedly be the subject of dispute. However, just as the motion of individual molecules is usually not considered in air flow simulations, it is usually possible to find a general consensus of what can be considered irrelevant imperfections of the assumed model.

computes the evolution of another system with sufficient precision is a good simulation.[6] This is an essential assumption that we must make if we are to learn anything from computer simulations. When arguing for particular physical realizations of neuronal systems, statements along the lines of "it does not rain in a computer simulation" are sometimes heard.[7] Taken unequivocally, such a statement must be wrong. While no current computer can predict the position of individual raindrops, the usefulness of weather simulations lies beyond any reasonable doubt.

From this functional rationale, a certain level of dichotomy between form and function becomes almost unavoidable in computational neuroscience. By the very nature of their field of research, modelers are constantly confronted with questions that pit form and function against each other. What is the optimal level of abstraction for a particular neuron or network model? What are its target computational properties? How does it correlate with experimental data? With what degree of accuracy and efficiency can it perform particular tasks? How robust is it with respect to various types of noise? Or, in summary: Can I design a network model with the following list of functional properties that is constrained in form by the following list of theoretical and experimental considerations?

By virtue of sheer numbers and the inevitability of natural selection, nature has improved form to achieve function over the course of billions of years. It has managed to enclose the pinnacle of known computational power within less than two liters of the most complex form of matter known to man. The interesting question is whether we are able to do the same in much less time but also with a much more guided strategy. If the methodology of modern physics and astronomy has anything to teach us, it is that we should simultaneously attack this problem with data, mathematics and silicon. The tenets of functionalism outlined above lie at the heart of this approach.

It is with this mindset that we approach neural theory and computation throughout this thesis. We take inspiration from biology, formulate mathematical models of neural dynamics and discuss implementations in neuromorphic hardware.

As this work is going to cover a broad range of topics and we wish it to be a useful reference for the interested reader, we considered it necessary to provide an ample introduction of the conceptual and material tools required throughout the main body of this work. Chapters 2 and 3 are intended to provide this knowledge base and are designed to provide the reader with at least an elementary mindset of a neurophysiologist, modeler, theoretician and neuromorphic hardware engineer. As such an (over)ambitious scope can not be covered exhaustively within the narrow confines of this thesis, we will often point out additional literature – especially textbooks – that should provide an adequate amount of detail on the respective topic.

In Chapter 4 we investigate the behavior of neurons under Poisson bombardment - a popular assumption in many network models with good support in experimental data. We derive detailed equations for the stochastic properties of certain point neuron models in this regime, which we will later use when we discuss various functional network

[6] We shall later discern between simulation and emulation when referring to conventional (von Neumann) computing architectures and physical-model neuromorphic hardware, respectively.

[7] This has been used to argue in favor of the necessity of "biomorphic" hardware, which represents a detailed physical model of biological neural circuits.

models. Additionally, we derive expressions for pairwise shared-input correlations of such neurons, which become useful for a formal understanding of the effects of finite-size pools of uncorrelated noise sources.

In Chapter 5 we discuss the physical implementation of cortical network models in analog neuromorphic hardware. We focus our discussion on a particular hardware system, but we study generally relevant phenomena and design techniques that are expected to be useful in any analog device of limited size. This chapter is intended as a toolbox for modelers that are prepared to face the challenge of working with an imperfect computational substrate in order to reap the benefits of low power, massive parallelism and enormous speedup that can be gained by departing from conventional von Neumann architectures.

Finally, in Chapter 6, we discuss several models for Bayesian inference in neural networks. Inspired by the ability of the brain to perform such computations, we build our networks with biologically-inspired neuron models, but also discuss applications that go beyond the realm of brain science. In particular, we consider problems ranging from psychophysics to machine learning and also discuss some interesting parallels to solid-state phenomena.

Many of the results that are discussed in this manuscript, both theoretical and experimental, are the outcome of collaborative efforts and have already been published in various forms. In particular these include several publicly available reports (Brüderle et al., 2010; Jordan et al., 2014; Petrovici et al., 2011, 2012) and the following journal papers and conference contributions (co-first authorship is denoted by a *):

- D. Brüderle, M. A. Petrovici, B. Vogginger, M. Ehrlich, T. Pfeil, S. Millner, A. Grübl, K. Wendt, E. Müller, M.-O. Schwartz, D. de Oliveira, S. Jeltsch, J. Fieres, M. Schilling, P. Müller, O. Breitwieser, V. Petkov, L. Muller, A. Davison, P. Krishnamurthy, J. Kremkow, M. Lundqvist, E. Muller, J. Partzsch, S. Scholze, L. Zühl, C. Mayr, A. Destexhe, M. Diesmann, T. Potjans, A. Lansner, R. Schüffny, J. Schemmel and K. Meier. *A comprehensive workflow for general-purpose neural modeling with highly configurable neuromorphic hardware systems.* Biological Cybernetics, 104:263–296, 2011.

- T. Pfeil*, A. Grübl*, S. Jeltsch*, E. Müller*, P. Müller*, M. A. Petrovici*, M. Schmuker*, D. Brüderle, J. Schemmel and K. Meier. *Six networks on a universal neuromorphic computing substrate.* Frontiers in Neuroscience, 7:11, 2013.

- M. A. Petrovici*, J. Bill*, I. Bytschok, J. Schemmel and K. Meier. *Stochastic inference with deterministic spiking neurons.* arXiv preprint arXiv:1311.3211, 2013.

- M. A. Petrovici, B. Vogginger, P. Müller , O. Breitwieser, M. Lundqvist, L. Muller, M. Ehrlich, A. Destexhe, A. Lansner, R. Schüffny, J. Schemmel and K. Meier. *Characterization and compensation of network-level anomalies in mixed-signal neuromorphic modeling platforms.* PloS one, 9(10):e108590, 2014.

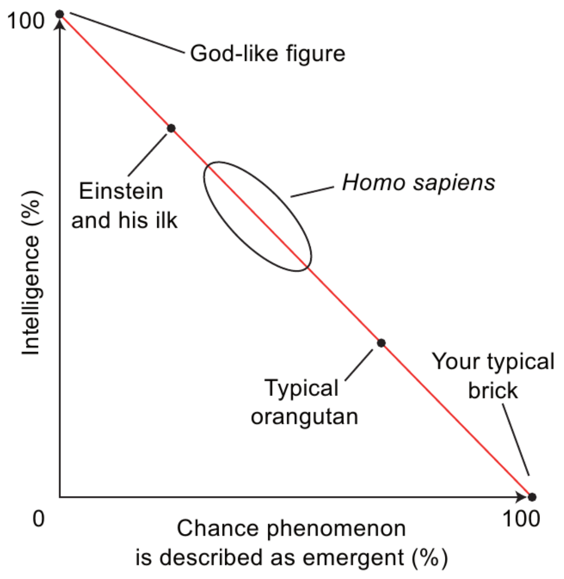
- D. Probst*, M. A. Petrovici*, I. Bytschok, J. Bill, D. Pecevski, J. Schemmel and K. Meier. *Probabilistic inference in discrete spaces can be implemented into networks of lif neurons.* Frontiers in computational neuroscience, 9, 2015.

- J. Jordan, T. Tetzlaff, M. A. Petrovici, O. Breitwieser, I. Bytschok, J. Bill, J. Schemmel, K. Meier and M. Diesmann. *Deterministic neural networks as sources of uncorrelated noise for probabilistic computations.* Accepted for presentation at the CNS 2015 conference. To appear in BMC neuroscience.

- M. A. Petrovici, I. Bytschok, J. Bill, J. Schemmel and K. Meier. *The high-conductance state enables neural sampling in networks of LIF neurons.* Selected for oral presentation at the CNS 2015 conference. To appear in BMC neuroscience.

Furthermore, some of the topics that we discuss here have also been addressed in several Bachelor, Master and Diploma theses that the author of this thesis has coordinated and co-supervised (Breitwieser, 2011, 2015; Bytschok, 2011; Korcsak-Gorzo, 2015; Leng, 2014; Müller, 2011; Petkov, 2012; Probst, 2014; Rivkin, 2014; Roth, 2014; Stöckel, 2015; Weilbach, 2015). In the beginning of each chapter, we therefore specifically point to other literature that has already been written on the respective subject and that has provided material for this manuscript. The sources of previously published material are listed again, in detail, in Section A.3.

2. Introduction: From Biological Experiments to Mathematical Models



Bruce C. Gibb, The emergence of emergence, 2011

At first glance, scientists seem to have needed a surprisingly long amount of time to find the physical correlate of thought. While the brain itself, as an organ, has long been considered the seat of the mind[1], it was only at the turn of the 20th century that Cajal and Golgi established the "neuron doctrine" - the hypothesis that the neuron is the fundamental functional unit of the brain. In the light of much earlier achievements in the physics of small scales, such as Bernoulli's kinetic theory of gases in the early 18th century, this comparatively slow development might appear paradoxical, especially given Hooke's and van Leeuwenhoek's discovery of the cell over two centuries before the discovery of the neuron. However, when considering the complexity of the "elementary" components involved, and moreover that of the emergent phenomena, a theory of mind has to appear much less intuitive – and probably much less mathematically tractable – than thermodynamics.

neuron doctrine

complexity and structure

Evidently, if we strive for more than just a phenomenological description of high-level information processing in the brain, we need to understand the functionality of its fundamental components. In this context, Section 2.1 represents as a brief introduction to the morphology and electrophysiology of neurons and synapses. In it, we review those

[1]Notable exceptions include Aristotle, who believed it to be a blood-cooling device (Gross, 1995).

processes in nerve cells that appear most relevant to information processing and communication. These "microscopic laws of motion" represent the fundamental building blocks for all theoretical and functional models that we discuss later on.

We should make clear that it happens with the mindset of a physicist when we follow such a reductionist approach. When we describe higher-level phenomena to "emerge" from low-level dynamics, there is no deus ex machina involved. When the whole becomes more than the sum of its parts, as is arguably the case for the brain, it is because a large ensemble of simple components exhibits features that are impossible to observe when only few such components interact. A useful description of such a system often involves the definition of new macroscopic variables, but they always represent a high-level view of a system that is entirely governed by fundamental interactions between its microscopic constituents.

emergence

Evolution is not equivalent to engineering and therefore its constructs, incrementally perfected over billions of years, always inherit vestiges of their predecessors, which might not necessarily serve a functional purpose. Additionally, all biological units need to perform metabolic and reproductive activities, for sustainment and growth, which massively add to their structural complexity. It is therefore quite likely that a rather high level of abstraction is sufficient (and probably even best suited) to understand information processing in neural networks. To which extent one can push this abstraction, however, remains an open question and subject to much debate among researchers in the field.

complexity and functionality

abstraction

The requirement of abstraction, as well as a precise description thereof, is well-established throughout all physical sciences. It is the search for a minimal, but complete set of laws required to describe a given system that gives rise to the notion of a model. For the most part, a "model" is understood as synonymous to a "mathematical model" of a system, that is, a set of rules and parameters, preferably expressed as equations, which completely describe the dynamics of variables associated to the relevant properties of the system. This makes it possible to apply a vast array of mathematical tools and formalisms to analyze and predict its behavior. The unparalleled success of this approach has established it as the centerpiece of all natural sciences.

model

Building on the aforegoing electrophysiological considerations, Section 2.2 outlines the construction of several abstract models for neurons and synapses. The presented neuron models exhibit various levels of abstraction, particularly concerning the spike generation mechanism of neurons, which are crucial for the investigations described in Chapters 4 to 6.

2.1. Morphology and Electrophysiology of Biological Neurons and Synapses

This section serves as a short overview of the biological mechanisms that underpin neuron and synapse dynamics. Understanding them is indispensable to formulating abstract models of neurons and synapses. For a much more in-depth view of the phenomena described here, we recommend Alberts et al. (1994) for the molecular biology topics and Gerstner and Kistler (2002) for the mathematical treatment.

2.1.1. Electrical Properties of the Cell Membrane

Just like every other cell, a neuron's interior is separated from the outside environment by the so-called plasma membrane. It consists mainly of a lipid bilayer, but also contains various (transmembrane) proteins which are essential for building up and dynamically changing a voltage across the membrane (Figure 2.1).

cell membrane

While highly permeable to small neutral molecules such as water, the lipid bilayer itself is practically impermeable to ions and polar molecules, which are the main charge carriers in organic media, thereby effectively insulating the interior of the cell from its exterior. Due to its narrow width of roughly 5 nm, the membrane effectively acts as a capacitor, with a typical capacitance of $C_\mathrm{m} = 1\ \mu\mathrm{F/cm}^2$. For a patch of membrane with an area of $1\,\mu\mathrm{m}^2$, membrane voltage changes of several tens of mV can therefore be achieved by moving only several thousand monovalent ions across the membrane, about 3 orders of magnitude below the total number of ions in the cytosol. It is important to note that only those ions lying very close (<1 nm) to the membrane influence its voltage, therefore any changes in the membrane potential need not affect ion concentrations in the rest of the cytosol.[2] This property is quintessential for enabling the high speed of neuron membrane potential dynamics required by real-world interaction and information processing – an aspect we shall return to later.

membrane capacitance

Cells make good use of their capability of establishing a gradient across their membrane, both chemical and electrical. Indeed, transmembrane gradients are the driving force of essentially all metabolic processes in the cell, especially solute transport and ATP synthesis.[3] However, the capability of the plasma membrane to maintain an electrochemical gradient is necessary, but not sufficient, to explain the existence of the gradient itself. The structure responsible for the (trans)membrane potential is the Na^+-K^+ pump.[4]

electrochemical gradient

[2]Let us assume a neocortical pyramidal cell has a total area of $S = 3 \cdot 10^4 \mu\mathrm{m}^2$ and a volume of $V = 10^4 \mu\mathrm{m}^3$, with a specific membrane capacitance of $c = 1 \mu\mathrm{F/cm}^2$ and a Na^+ concentration of $[\mathrm{Na}^+] = 3 \cdot 10^7 \mathrm{ions}/\mu\mathrm{m}^3$. The number of ions required for an increase in membrane potential by 10 mV is then $N = cSU/e \approx 2 \cdot 10^7$, as compared to the total number of ions in the cell $N_\mathrm{tot} = [\mathrm{Na}^+]V = 3 \cdot 10^{11}$. This is only an approximate calculation, as estimates of cell sizes vary considerably, see e.g. Ambros-Ingerson and Holmes (2005).

[3]The quintessential question in the search for the origin of life is how this gradient could have appeared in early cell-like structures. Furthermore, while in prokaryotic cells, ATP is synthesized at the plasma membrane, in eukaryotes this process is taken over by specialized organelles - mitochondria and plastids. It has been argued that it was the acquisition of these organelles by early prokaryotes that enabled the evolution of complex life. For an excellent discussion on these topics, we refer to Lane and Martin (2010, 2012).

[4]Actually, an electrochemical gradient would exist even without the Na^+-K^+ pump, due to the high concentration of organic compounds inside the cell. These, in turn, cause a high intracellular osmolarity, which would force water to move into the cell by osmosis. The Na^+-K^+ pump counters this

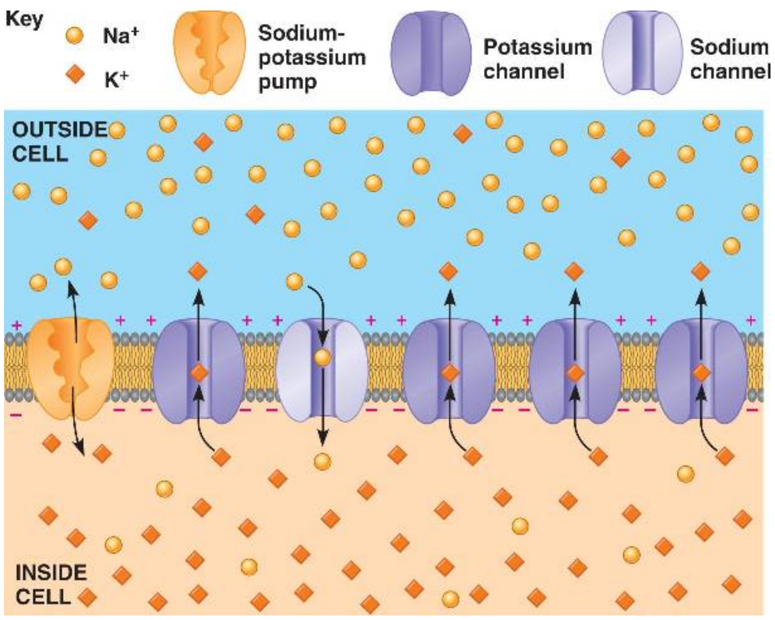


Figure 2.1.: Sketch of the plasma membrane, with embedded active pumps and leakage channels. See text for details. Image taken from expertsmind.com.

Na^+-K^+ *pump*

The Na^+-K^+ pump is an ATPase, a carrier protein embedded in the plasma membrane that hydrolyzes ATP to provide the energy for its operation. Within a full pump cycle, it transports 3 Na^+ ions from the cell plasma to the extracellular medium and 2 K^+ ions from the extracellular medium into the plasma. This leads to an excess of negative charge (or rather, a net total deficit of positive charge) inside the cell, which in turn translates to a negative membrane potential. As the electrochemical gradient increases, the pumping process becomes less efficient and the membrane potential reaches some equilibrium value.

membrane potential

The activity of Na^+-K^+ pumps alone, however, would only account for about a tenth of the measured equilibrium value of some -70 mV. A second type of transmembrane protein plays an essential role here: the K^+ leakage channel. This channel protein is permanently "open" and only allows the passage of K^+ ions. Because of the high concentration of potassium inside the cell, due to the Na^+-K^+ pump, an outflow of K^+ ions is established, that pulls down the membrane potential even further. This happens until an equilibrium value is reached, the so-called Nernst potential, the value at which diffusive and electrical forces counterbalance:

K^+ *leakage channel*

Nernst equation

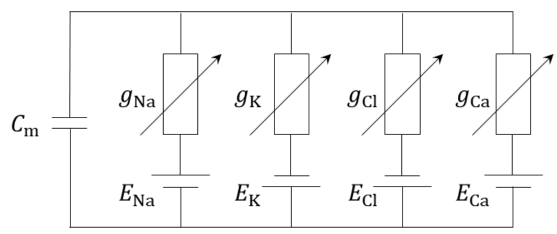
$$E_X = \frac{RT}{zF} \ln \frac{[X]_{\text{ext}}}{[X]_{\text{int}}}, \tag{2.1}$$

where R is the universal gas constant, F the Faraday constant, T the absolute temperature, z the charge of the ion in question X and $[X]_{\text{ext/int}}$ its extracellular and intracellular concentrations, respectively. However, various ions tend to have different Nernst potentials, due to their different charges and concentrations, hence the resting membrane potential becomes something like a weighted mean thereof.

resting potential

Figure 2.2 shows the equivalent circuit that determines the resting potential V_{rest} of the

potentially destructive effect by increasing the net extracellular concentration of inorganic ions.



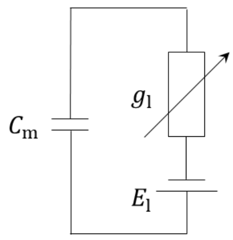


Figure 2.2.: **Left:** Full equivalent circuit of a cell membrane, with different ion concentrations acting as batteries. The Na^+ and K^+ batteries are governed by the Na^+-K^+ pump and the K^+ leakage channel, as described in the text above. Na^+ leakage channels also exist (see Figure 2.1), but are much fewer in number (with a ratio of about 1:100 with respect to K^+ leakage channels). The two other ions that have significant contributions to the membrane potential are Ca^{++} and Cl^-, which have their concentrations regulated by similar pumping and leakage mechanisms. **Right:** Reduced equivalent circuit, with all ion concentrations condensed into a single battery and conductance.

membrane. Taking into account only the contributions from monovalent ions[5], V_rest can then be calculated from the Goldman-Hodgkin-Katz (GHK) equation[6] :

Goldman-Hodgkin-Katz equation

$$V_\mathrm{rest} = \frac{RT}{F} \ln \frac{\sum_i P_{X_i^+}[X_i^+]_\mathrm{ext} + \sum_i P_{Y_i^-}[Y_i^-]_\mathrm{int}}{\sum_i P_{X_i^+}[X_i^+]_\mathrm{int} + \sum_i P_{Y_i^-}[Y_i^-]_\mathrm{ext}}, \tag{2.2}$$

with P_X denoting the permeability of the membrane to the ion type X. For a single ion type, it can be easily seen how the Nernst equation is merely a special case of the more general GHK equation. Because the membrane permeability to K^+ is usually at least an order of magnitude above the permeabilities to all other ion types, V_rest lies rather close to the K^+ reversal potential, which usually lies around -80 mV.[7]

equivalent circuit *leak potential*

At this point, we can reduce the equivalent circuit from Figure 2.2 to a simple RC circuit, with a single battery defining the rest or leak potential (here, as well as in most abstract neuron models, V_rest and E_l can be used interchangeably) as calculated from the GHK equation (Figure 2.2). The term "leak potential" stems from the intuitive picture that following any temporary external electric stimulation, the capacitor leaks charges into the battery and reverts to its rest potential along an exponential curve. Indeed, the associated ODE of the membrane potential can be easily derived from Ohm's law in the

membrane potential ODE

[5] For ions of higher valence, such as Ca^{++}, extensions to the GHK equation exist - see, e.g., Pickard (1976). However, since during resting conditions, both the permeability and the concentration of Ca^{++} ions is comparatively low, calcium does not play a significant role in defining V_rest.

[6] Permeability and conductance are closely related, but not equivalent. Channel conductance is, in particular, strongly voltage dependent. For a detailed discussion, see, e.g., Schultz et al. (1996).

[7] To get a feeling of the relevant variables, a numerical example is in order. For that, we consider measurements of the squid giant axon from Hodgkin (1958). The values given by Hodgkin read: $P_{Na^+} = 1$, $P_{K^+} = 100$, $P_{Cl^-} = 10$ (permeabilities are given relative to P_{Na^+}), $[K^+]_\mathrm{ext} = 20$, $[K^+]_\mathrm{int} = 200$, $[Na^+]_\mathrm{ext} = 440$, $[Na^+]_\mathrm{int} = 50$, $[Cl^-]_\mathrm{ext} = 540$, $[Cl^-]_\mathrm{int} = 40$ (ion concentrations given in mmol/l). At a temperature of 37 °C, the Nernst potentials then read $E_{Na^+} = 58.1$, $E_{K^+} = -61.5$ and $E_{Cl^-} = -69.6$, with the equilibrium membrane potential lying at $E_\mathrm{l} = -58.6$ (potentials given in mV).

reduced equivalent circuit:

$$C_\mathrm{m}\frac{du}{dt} = g_\mathrm{l}(E_\mathrm{l} - u) + I, \tag{2.3}$$

where I stands for any external current source. The variable $\tau_\mathrm{m} = C_\mathrm{m}/g_\mathrm{l}$ is called the membrane time constant and quantifies the speed at which the membrane potential reacts to external stimuli. This characteristic variable can be found in virtually any abstract neuron model. See Sections 2.2 and 4 for a much more detailed discussion of membrane potential dynamics.

membrane time constant

We now have a good mathematical model for the membrane potential of a cell in its resting state. Note that until this point, we have not yet addressed excitable cells (in particular, neurons), so the above considerations are, in principle, valid for any biological cell.

The above model also accounts for membrane dynamics under external current stimulation, albeit while implicitly neglecting the spatial extent, i.e., the 3D structure of the cell. It is extremely important to keep in mind that point models invariably limit the computational power of the units modeled as such. Also, for all point models, any claims of biological plausibility need to be reviewed carefully, especially since many neuron types have a very distinct branching structure. While most models and methods considered throughout this thesis do not take the spatial structure of the cells involved into consideration, we will address this issue briefly both from a theoretical perspective (Section 2.1.4), as well as in the case of a concrete cortical network model (Section 5.3).

point neuron model

2.1.2. Action Potentials and the Hodgkin-Huxley Neuron Model

Just like all other cells, excitable cells establish an electrochemical transmembrane gradient for metabolic reasons. In addition to that, however, evolution has provided them with ways of manipulating their membrane potential, thereby allowing much faster communication and computation than would normally be possible through chemical diffusion processes. While many types of excitable cells exist, which also play an essential role in information processing (such as receptor cells or myocytes), the particular class of excitable cells we are interested in here are the neurons in the central nervous system.

Neurons do not exchange information permanently.[8] Their communication is mediated by all-or-nothing events, so-called action potentials (APs), or simply spikes. These are large pulses in the membrane potential with an amplitude of about 100 mV and a typical duration of several ms.

action potentials, spikes

As a rough approximation, it can be said that action potentials occur when the membrane potential of a neuron increases beyond a certain value (usually around -50 mV). The membrane then spontaneously depolarizes, usually exceeding 0 mV, after which it quickly hyperpolarizes, even dropping below its resting potential for several ms. During this hyperpolarized state, also called the refractory phase, even strong stimuli can not initiate a second AP.

refractory phase

These phenomenological findings can be explained by another class of transmembrane proteins: voltage-gated ion channels. This hypothesis (later confirmed by Erwin Neher

voltage-gated ion channels

[8] This is, of course, not absolutely true, since mechanisms such as local ion depletion, neurotransmitter diffusion or electrical crosstalk do enable additional communication pathways between neurons. Furthermore, electrical synapses (see Section 2.1.3) can also create a continuous link between membrane potentials.

and Bert Sakmann, Nobel Prize 1991), along with a stochastic description of their dynamics, has earned Alan Lloyd Hodgkin and Andrew Huxley the 1963 Nobel Prize. The Hodgkin-Huxley model consists of a set of four differential equations and remains the most accurate model of neuronal membrane dynamics to date.[9]

molecular gates

Similar to leakage channels, voltage-gated ion channels only allow the passage of specific ion types. However, their conformation changes as a function of the membrane potential, thereby affecting their permeability. In a simplified picture[10], these transmembrane proteins can be thought of as having "gates" which open and close stochastically, depending on the membrane potential. Figure 2.5 sketches several possible states for the Na^+ and K^+ voltage-gated ion channels. Both channel proteins have four gates, with the Na^+ channel possessing two different types of gate.[11] If we denote the probabilities of the three gate types being open by m, h (for Na^+) and n (for K^+), respectively, and the maximum conductance of the ion channels (in a fully open state) by g_{Na^+} and g_{K^+}, then the average currents flowing through the two ion channels read

$$\langle I_{\mathrm{Na}^+} \rangle = g_{\mathrm{Na}^+} m^3 h (u - E_{\mathrm{Na}^+}) \quad \text{and} \tag{2.4}$$

$$\langle I_{\mathrm{K}^+} \rangle = g_{\mathrm{K}^+} n^4 (u - E_{\mathrm{K}^+}). \tag{2.5}$$

The voltage dependence of the gating variables m, h and n is given by

$$\dot{x} = -\frac{1}{\tau_x(u)}[x - x_0(u)], \quad x \in \{m, h, n\}, \tag{2.6}$$

with specific time constants $\tau_x(u)$ and equilibrium values $x_0(u)$. Alternatively, some authors prefer to use a somewhat different form of the above ODE:

$$\dot{x} = \alpha_x(u)(1 - x) - \beta_x(u)x, \quad x \in \{m, h, n\}, \tag{2.7}$$

with $x_0(u) = \alpha_x(u)/[\alpha_x(u) + \beta_x(u)]$ and $\tau_x(u) = 1/[\alpha_x(u) + \beta_x(u)]$. Figure 2.4 shows fits for $\tau_x(u)$ and $x_0(u)$ with the original parameters from Hodgkin and Huxley (1952).

[9] Modern extensions of the original model mainly include the addition of other types of ion channels and the morphology of neural cells.

[10] We have to stress that the Hodgkin-Huxley model is purely phenomenological and that the “gates” referenced multiple time in the text are only a mechanistic interpretation of the integer exponents in the gating variable equations. This is, however, quite close to reality: voltage-sensitive transmembrane proteins have, indeed, multiple identical compartments that change their conformation depending on the membrane potential. The voltage dependence of channel protein conformations is still the subject of intensive research, see, e.g., Long et al. (2007) for recent results on the structure of voltage-gated K^+ channels.

[11] More recent studies from the 1990s have shown that the K^+ channel also features several inactivation mechanisms of its own, one of which is similar to the ball-and-chain model of Na^+ inactivation from Figure 2.3. See Kurata and Fedida (2006) for a review.

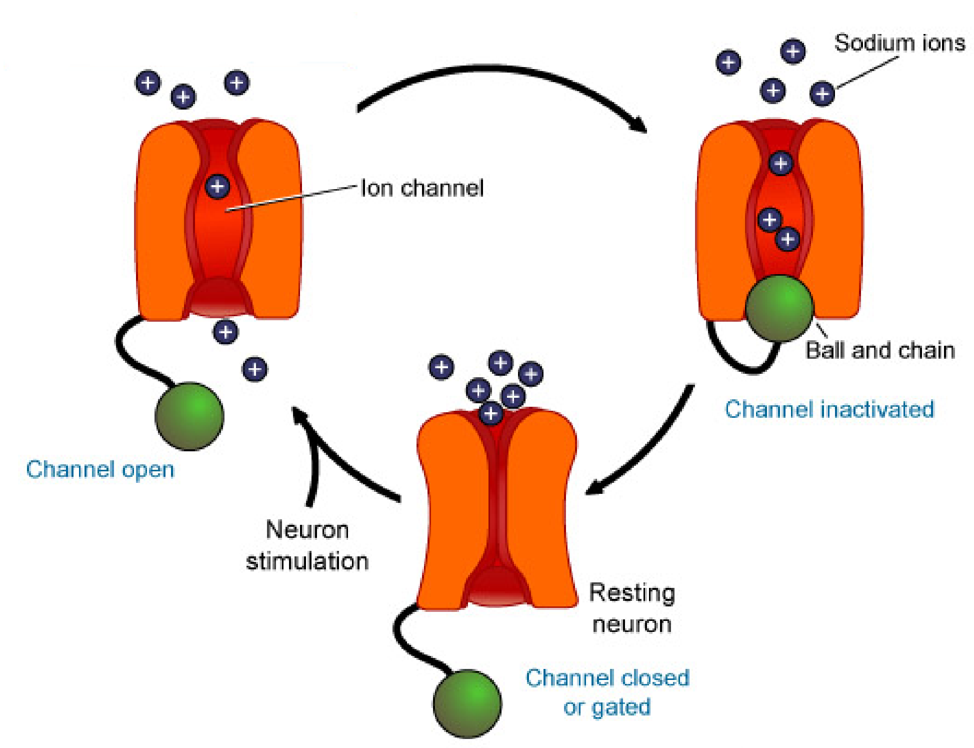


Figure 2.3.: Ball-and-chain model of a voltage-gated Na^+ channel. Upon excitatory stimulation of the neuron, its membrane potential increases, thereby (stochastically) triggering an opening, or activation, of the channel protein. The resulting influx of Na^+ ions increases the membrane potential even further, which causes even more Na^+ channels to open. This feedback loop continues until the "ball-and-chain" components of the molecule, which also react to high membrane voltages, but on a slower timescale, block the channel, thereby inactivating it. Note the difference between inactivation, which is an active self-blocking of ion channels, and deactivation, which is the process by which, when the membrane potential shifts outside the range that caused the channels to open in the first place, they simply close again. The latter is exactly what happens with voltage-gated K^+ channels in the Hodgkin-Huxley model, following a spike (see text for details). Image courtesy of Penn State Department of Biology.

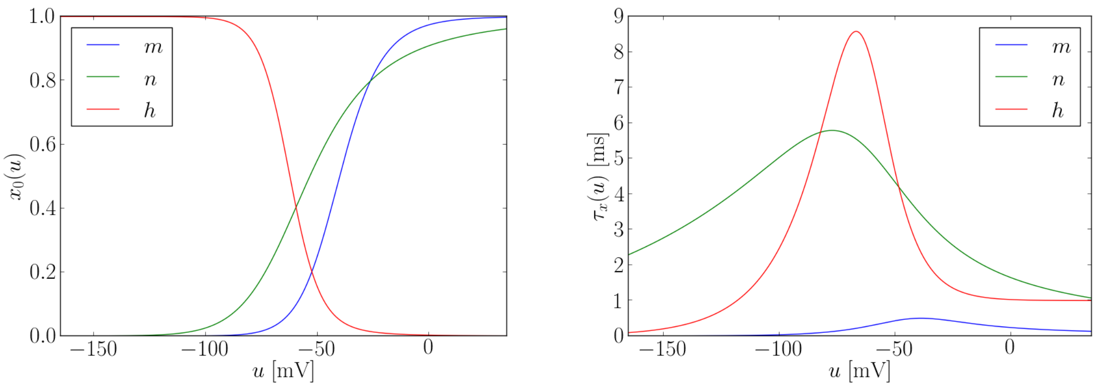
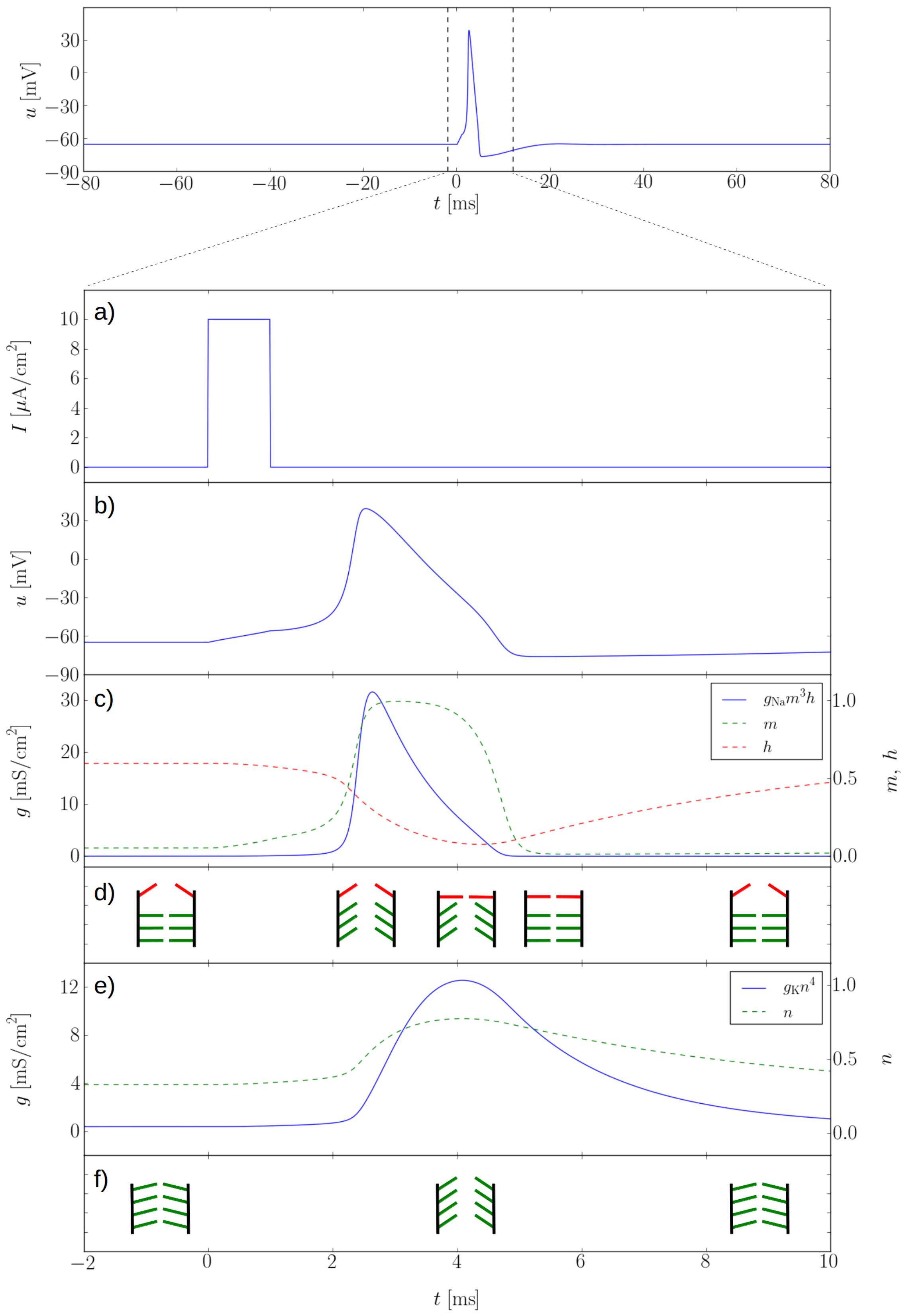


Figure 2.4.: Gating variables of the voltage-gated Na^+ and K^+ ion channels in the Hodgkin-Huxley model. **Left:** Equilibrium values as a function of the membrane potential u. Note how the m and n gates open at higher values of u, thereby activating the Na^+ and K^+ channels, respectively. Conversely, the h gates close for high u, thereby inactivating (i.e., actively closing) the Na^+ channels. **Right:** Time constants as a function of membrane potential u. Due to their fast dynamics (low time constant), the m channels open quickly, allowing the sharp onset of the action potential. Both the inactivation of the Na^+ channels via the h gates and the activation of the K^+ channels via the n gates occur on a slower timescale, jointly causing the falling flank of the action potential.



u [mV]
t [ms]
a)
I [μA/cm^2]
b)
u [mV]
c)
g [mS/cm^2]
$g_{\mathrm{Na}}m^3h$
m
h
m, h
d)
e)
g [mS/cm^2]
$g_{\mathrm{K}}n^4$
n
f)
t [ms]

Figure 2.5.: Spiking dynamics in the HH neuron model. **Top:** membrane potential during a single spiking event caused by a short step current stimulus at $t = 0$ ms. **Bottom:** zoom-in on the time axis during the AP showing the evolution in time of all the relevant dynamic variables. **(a)** Current stimulus. **(b)** Membrane potential. The initially small change of $\approx 10\,\mathrm{mV}$ caused by the current stimulus results in a fast opening of the voltage-gated Na^+ channels which is strong enough to trigger a cascade effect that ultimately results in the rising flank of the AP. After the peak voltage is reached, inactivation of the Na^+ channels and activation of the K^+ channels causes the membrane potential to drop below the leak potential and slowly return to the resting state. **(c)** Evolution of Na^+ channel conductance (solid curve) and gating variables (dashed curves). The fast activation due to the m gates (in green) is followed by a slower inactivation due to the h gates (in red). This difference in time constants allows the sharp onset of the AP. Together with the activation of the K^+ channels (n gates, see panel (e)), the inactivation of the Na^+ channels is responsible for the refractory period that follows the AP. **(d)** Gate configuration of Na^+ channels. This is only a symbolic representation, since the conformational changes of the proteic subunits that build up individual gates are stochastic processes. **(e)** Evolution of the K^+ channel conductance (solid curve) and gating variable (dashed curve). The slower dynamics of the n gates as compared to the m gates allows the buildup of the AP before the decay towards the K^+ reversal potential. Note the long tail of the K^+ conductance compared to the Na^+ conductance. **(f)** Gate configuration of K^+ channels. As in panel (d), this is only a symbolic representation. For the full set of parameters used for this simulation, see Tables A.1 and A.2.

Now we can return to the equivalent circuits from Figure 2.2. For the sake of clarity, let us further assume that the leak and pump conductances - and thereby also g_l - are constant. We can now extend the reduced equivalent circuit by the voltage-gated Na^+ and K^+ channels described above. With equations 2.4 and 2.5, the ODE for the membrane potential then becomes

$$C_\mathrm{m}\dot{u} = -g_{\mathrm{Na}^+}m^3h(u - E_{\mathrm{Na}^+}) - g_{\mathrm{K}^+}n^4(u - E_K) - g_\mathrm{l}(u - E_\mathrm{l}) + I \tag{2.8}$$

HH model The four ODEs given by Equations 2.6 and 2.8 fully define the Hodgkin-Huxley neuron model.

Let us take a look at the dynamics of this model, with Figure 2.5 serving as graphical guidance. The parameters used in this simulation are given in Tables A.2 and A.1. Without external stimulus, the membrane potential is at rest and does not change in time. Note that in the HH model, the rest potential is *not* equal to the leak potential E_l, but lies significantly lower, due to the K^+ channels always being open with nonzero probability. *deactivation* The Na^+ and K^+ voltage-gated channels are predominantly closed, or deactivated, due to the m and n gates, respectively. Upon stimulation with a strong enough step current, *activation* the m-gates start opening, activating the Na^+ channels and allowing an influx of Na^+, thereby increasing the membrane potential even further and leading to a cascade effect that pulls the membrane potential close to E_{Na^+}. This represents the steeply rising flank of the action potential. Due to their slower dynamics ($\tau_h \gg \tau_m$, see Figure 2.4), the *inactivation* inactivation of the Na^+ channels via the m gates becomes dominant only with a certain delay, thereby not interfering with the sharp action potential onset.

If only Na^+ channels were present, the membrane potential would now slowly ($\tau_\mathrm{m} = C_\mathrm{m}/g_\mathrm{l} \approx 10$ ms) decay towards E_l. However, on roughly the same timescale as the inactivation of the Na^+ channels, the activation of the K^+ channels occurs.[12] These quickly pull the membrane potential back down towards E_{K^+}, thereby generating the steep falling flank of the action potential. Again, due to their slow dynamics, the h gates remain closed and the n gates remain open for some time, leading to the relatively long characteristic "undershoot" of the membrane potential following the sharp spike. *refractory period* During this so-called refractory period, another spike is difficult to elicit, due to both the Na^+ channels being inactivated and the K^+ channels being open.

Apart from spiking when receiving strong and fast enough stimulation, HH neurons exhibit some other very interesting and maybe even surprising dynamics, which will be of particular interest later on, when we formulate more abstract neuron models. Whether these features are of any computational relevance is an important topic of ongoing debate.

Let us first turn our attention towards the "spiking threshold". Does there exist a value of the membrane potential which, once reached, guarantees that the neuron will spike? As we can see from Equation 2.8, one can control the equilibrium value of the membrane potential by varying the external input current I, thus allowing to define a "threshold current" that is analogous to a threshold potential, should one exist. Medical dictionaries *rheobase* define the so-called rheobase as the minimal electric current required to excite a tissue given an indefinitely long time during which the current is applied.

[12] Note that already in the deactivated state, individual n gates have a significant probability ($p \approx 0.3$) of being open. However, for a channel to be permeable, all gates have to be open at the same time, the probability of which scales with p^4 and is therefore much lower.

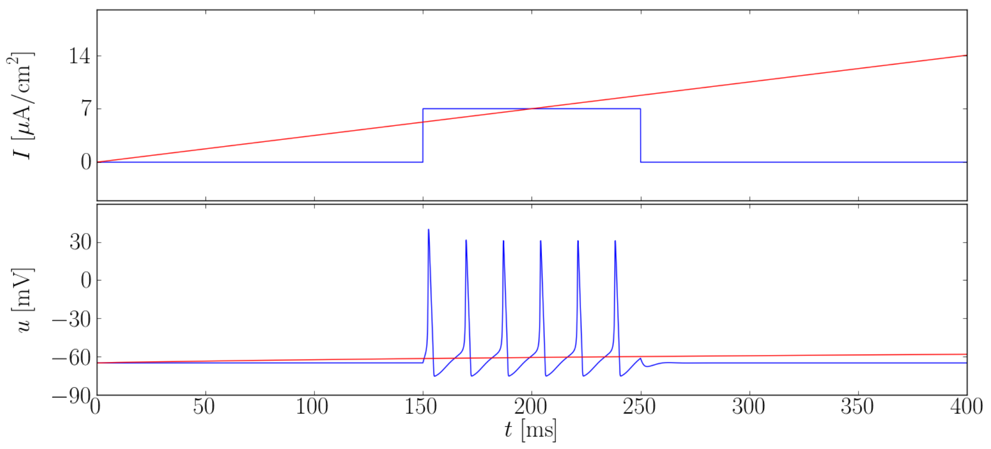


Figure 2.6.: Searching for the rheobase of an HH neuron (and not finding it). When stimulated with a step current of $7\,\mathrm{pA/cm^2}$, the neuron goes into a regular spiking mode (blue curves). If one increases the current slowly enough, one can reach double that value (and, in principle, any value) without triggering a spike response (red curves).

Figure 2.6 shows an HH neuron stimulated with a step current which is strong enough to trigger persistent spiking while it remains on. Following its definition, one must assume that the rheobase is smaller or equal to the applied stimulus. However, if one increases the input current slowly, even at double the value of the previous stimulus, no spike is triggered. Indeed, if the input current is increased slowly enough, one can converge smoothly to any membrane potential value. Even for step currents, there is no clear threshold for which spiking is initiated. When decreasing the step value with fine enough granularity, the neuron responds with an increasing delay, with a spike of decreasing amplitude (not shown here).

This goes to show that HH neurons do not have a firing threshold in the precise mathematical sense. While the threshold assumption may be a very practical one[13], as we shall discuss later in Section 2.2.1, it is important to keep in mind that it is a mathematical abstraction of an otherwise different physical phenomenon. In compliance with common terminology, we will nevertheless use the terms "suprathreshold" and "subthreshold" when describing regimes where a neuron fires or does not fire, respectively.

supra- and subthreshold regimes

inhibitory rebound

Another interesting phenomenon is the so-called (post-)inhibitory rebound. When stimulated (inhibited) by a negative current, the membrane potential naturally drops below E_l. If the stimulation ends abruptly and the inhibitory current was long and strong enough, a single spike can be elicited (Figure 2.7). The explanation for this behavior lies again in the difference between the time constants of gate dynamics. The temporarily low membrane potential results in a stronger deactivation of the $\mathrm{K^+}$ channels (n gates) and a weaker inactivation of the $\mathrm{Na^+}$ channels (h gates). Upon the abrupt termination of the inhibitory stimulus, the membrane is pulled back towards E_l, but the fast m gates

[13] See also Kistler et al. (1997) for a similar discussion related to a different type of simplified neuron model (the spike-response model).

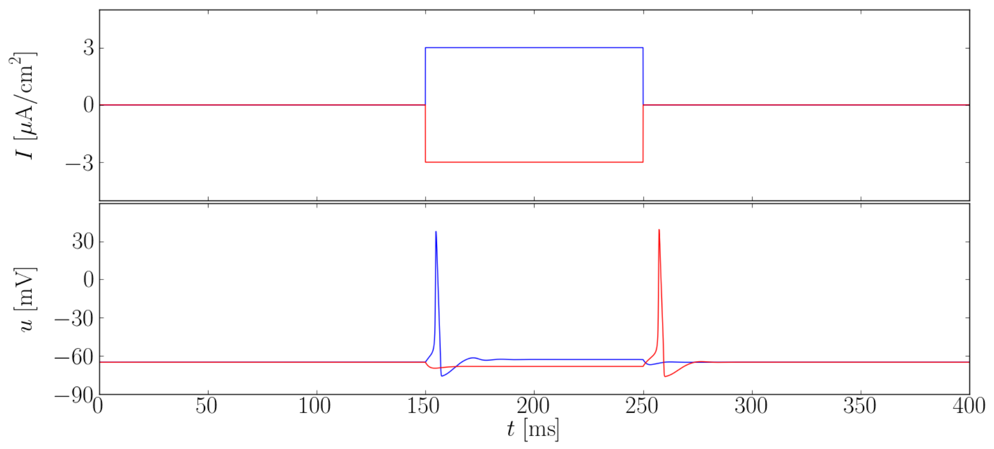


Figure 2.7.: Inhibitory rebound of an HH neuron. Given a high enough amplitude, both excitatory (blue curves) and inhibitory (red curves) stimuli can cause a spike in the HH model, if their onset (in the excitatory case) and end (in the inhibitory case) are quick enough.

open quicker than the other two gate types can react to the change in voltage, causing an overshoot, which, when large enough, leads to spiking.

This effect is impossible to account for in neuron models governed by a single, first-order ODE, which possess a resting potential (stable fixed point of u). The extremely popular LIF model, discussed in detail in Section 2.2.1.1, is one such example. This is one of the reasons why neuron models of intermediate complexity have been developed and are being used for modeling cortical function. We point to Sections 2.2.1.2, 3.3.1 and 5.5 for further elaboration on this topic.

resonance

The third and final phenomenon we shall address here is resonance. It is similar to the inhibitory rebound in that it is also caused by the different timescales on which gate dynamics evolve and it can also not be reproduced with first-order, single-ODE models. When stimulated periodically with a current of an amplitude that would otherwise not elicit a spike, an HH neuron can be provoked to fire, as shown in Figure 2.8. This happens only if the pulse frequency of the stimulus is close to a specific value for the given neuron, hence the denomination of the effect. As shown in e.g. Izhikevich (2007), in vitro recordings of cortical neurons also show rebound and resonant spiking.

2.1.3. Synapses

After having discussed the individual dynamics of the fundamental building blocks of neural networks, we now turn to the other key ingredient in neural information processing: the interneuron interaction.

When the membranes of two neurons lie in close proximity to each other[14], a so-called

[14] More precisely, where the axon of the presynaptic cell touches a dendrite of the postsynaptic cell. See Section 2.1.4 for more details on neuron morphology and its functional consequences.

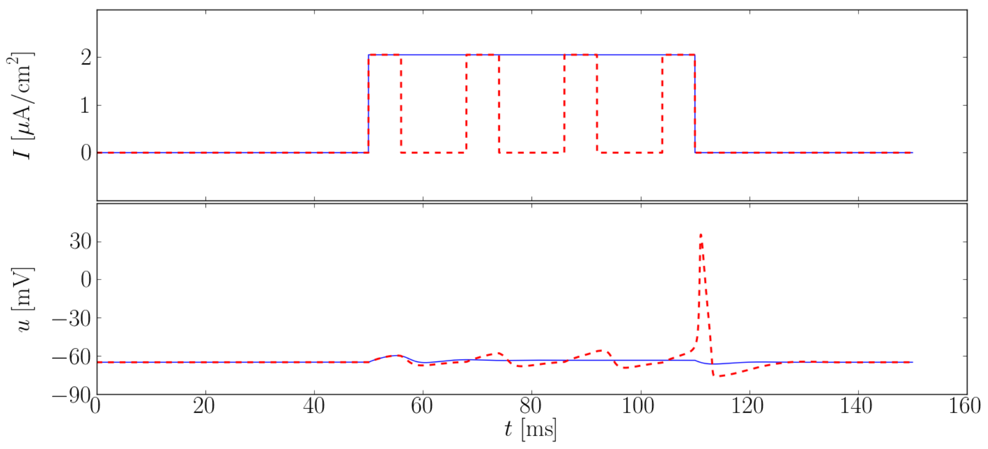


Figure 2.8.: Resonance phenomenon in the HH model. If an excitatory stimulus is too weak, it may not trigger a spike (blue curves). However, a pulsed stimulus with the same amplitude and correct frequency (which depends on the model parameters, especially the gating variable time constants), can cause a resonance phenomenon where the driven oscillations of the membrane lead to an AP.

synapse can form, enabling the transmission of electrical signals between the two cells. Through a synapse, a spike of the presynaptic neuron can elicit a temporary change in the membrane potential of the postsynaptic cell called a PSP[15]. Depending on whether the PSP has a positive (excitatory) or negative (inhibitory) influence on the membrane potential, it is also called an EPSP or an IPSP, respectively. Two fundamentally different types of synapses exist: electrical and chemical ones.

synapse

PSP

EPSP, IPSP

Electrical synapses are very simple in their structure. In an electrical synapse, the neuron membranes are separated by a narrow space called a gap junction or synaptic cleft, which is several nm wide. At the site of the gap junction, the membranes contain numerous junction channels called connexons, which can be thought of simply as pores that connect the cytoplasm of the two cells. These channels are wide enough to allow the passage of all relevant charged ion types (among others), thereby enabling the passive flow of ionic currents and thus the transmission of electrical signals. Electrical synapses are therefore very fast, with synaptic delays (time lag between the arrival of the presynaptic spike and the onset of the PSP) on the order of 0.2 ms. They also allow signal transmission in both directions. However, despite their distinct speed advantage, they only represent a distinct minority of synapses in the neocortex. One reason might be that due to their simple structure, they lack the versatility and plasticity of chemical synapses.

electrical synapse, gap junction, synaptic cleft, connexons

synaptic delay

Chemical synapses, on the other hand, have a distinctly asymmetric structure. The presynaptic terminal, or synaptic bouton, contains specialized transmission molecules called neurotransmitters. These are enclosed in so-called vesicles, spherical formations about

chemical synapse, bouton, neurotransmitters

[15] Which stands for "postsynaptic potential" and is therefore a rather unfortunate acronym for something representing a temporary change in the latter.

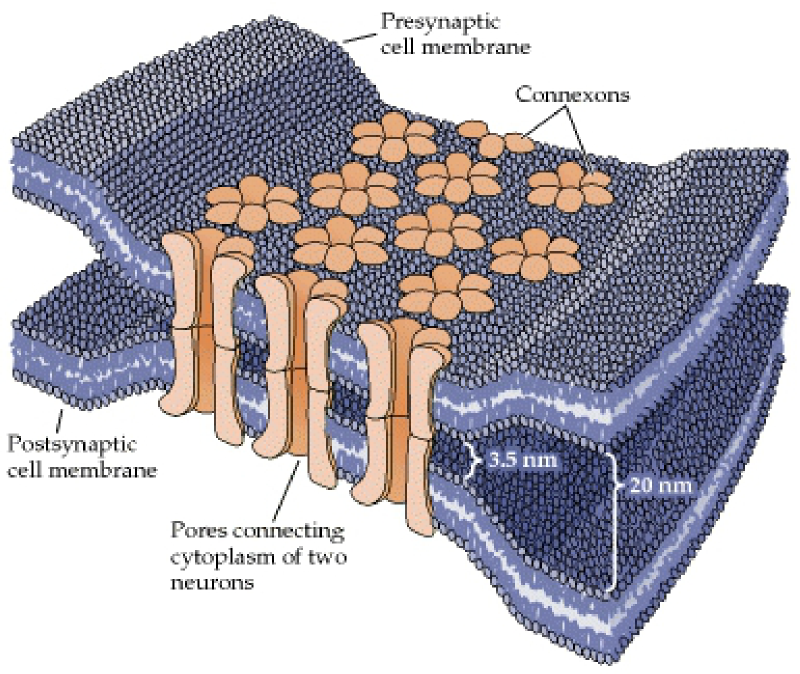

Presynaptic cell membrane
Connexons
Postsynaptic cell membrane
Pores connecting cytoplasm of two neurons
3.5 nm
20 nm

Figure 2.9.: Electrical synapse. In an electrical synapse, cell membranes are separated by an extremely narrow synaptic cleft. Special transmembrane proteins called connexons bridge the synaptic cleft and allow the passage of, among others, charged ions. Electrical excitations of the presynaptic membrane can thereby propagate passively to the postsynaptic membrane. Image taken from Purves et al. (2001).

vesicle 40 nm in diameter surrounded by a plasma membrane. Vesicle membranes contain a particular type of protein which, when activated by Ca^{++} ions, causes the fusion of the vesicle to the cell membrane. When the presynaptic cell fires, the AP causes the opening of voltage-gated Ca^{++} channels, thereby creating an influx of Ca^{++} ions into the synaptic bouton. The resulting high Ca^{++} concentration causes the vesicles lying in the proximity of the cell membrane to fuse with it, releasing their contents into the synaptic cleft. The released neurotransmitters can now freely diffuse throughout the synaptic cleft, which is about 20 nm wide, reaching the postsynaptic terminal within several ms (and making chemical synapses an order of magnitude slower than electrical ones). At the postsynaptic terminal, the target neuron membrane contains a high density of receptor proteins called *ligand-gated ion channels* ligand-gated ion channels, which change their conformation in the presence of particular molecules. The neurotransmitter molecules cause the opening of these ion channels, creating an influx of charged ions into the postsynaptic cell. Formally, this amounts to an increase in (postsynaptic) membrane conductance for a specific ion type and is consequently dubbed a *PSC* PSC[16]. This, in turn, elicits a PSP on the postsynaptic membrane. Due to Brownian motion and enzymatic metabolization, the transmitter molecules eventually

[16] Coincidentally, the abbreviation "PSC" can stand for either postsynaptic current (generated by the influx of ions through the ligand-gated ion channels) or postsynaptic conductance. The reader is therefore encouraged to pay particular attention to the context in which this acronym appears.

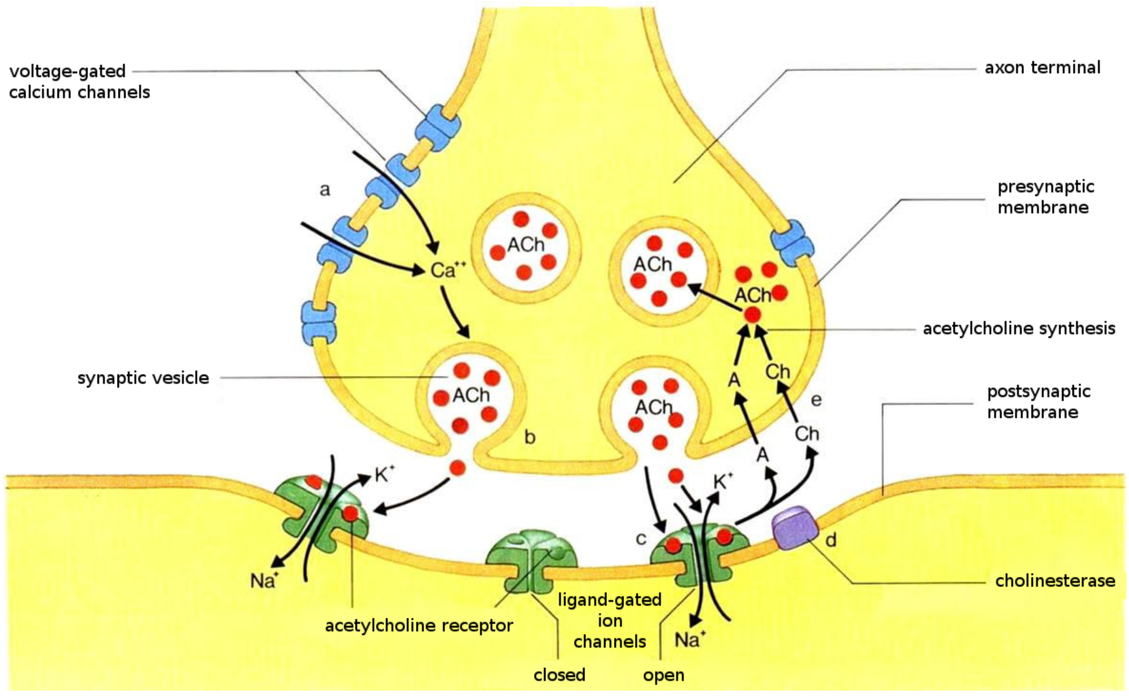


Figure 2.10.: Chemical synapse. When the presynaptic neuron fires, the elevated membrane potential at the presynaptic terminal causes voltage-gated Ca^{++} channels to open, creating an influx of Ca^{++} ions. These bind to vesicles containing neurotransmitter molecules, causing them to fuse with the cell membrane and spill their contents into the synaptic cleft. The diffusing neurotransmitters eventually reach the postsynaptic terminal, where they activate specific receptors in the postsynaptic cell membrane, which in turn allow the passage of charged ions, thereby eliciting a PSP. Eventually, the neurotransmitter molecules break loose from the receptors and are reabsorbed by the presynaptic cell for re-release. Image modified from Knodel (1998).

break loose from the receptors, returning them to a closed state and ending the PSC. The freed neurotransmitter molecules or their metabolites can then be reabsorbed and, if necessary, reconstituted by the presynaptic terminal and enclosed in new vesicles, thereby becoming available for future release.

temporal and spatial summation of synaptic inputs

If multiple spikes arrive in close succession, their effects on the membrane conductance/current are summed up (aside from short-term saturation/depletion mechanisms, which will be discussed in Section 2.2.2.2). The same is true for PSCs arriving from different synapses. Since the neural membrane integrates over its inputs, PSPs sum up as well, both temporally (over different spikes) and spatially (over different synapses).

The various steps and components of this complex chain of events have many profound functional consequences.

The use of neurotransmitters as intermediates requires their metabolization by the presynaptic neuron. Therefore, it appears reasonable that every neuron releases the same set of neurotransmitters at each of its efferent synaptic sites. This principle, coined by

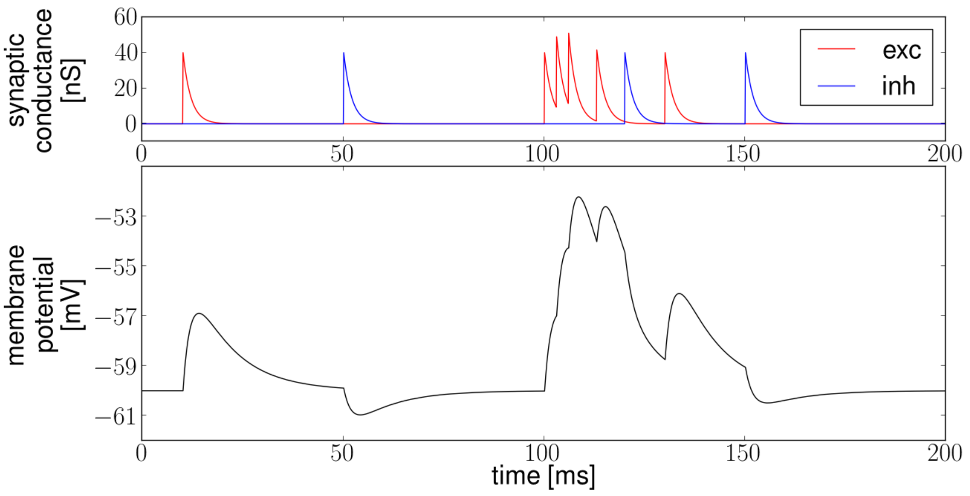


Figure 2.11.: Simulation of synaptic events. A single neuron is stimulated by an excitatory and an inhibitory presynaptic partner. Their spikes cause changes in the membrane conductance: red for excitatory (towards E_{Na^+}) and blue for inhibitory (towards E_{K^+}). Despite the two synapses having the same weight (as can be seen from the equal height of their PSCs), the inhibitory PSPs are significantly smaller than the excitatory ones due to the membrane potential lying much closer to E_{Na^+} than to E_{K^+}. When synaptic events arrive in quick succession, both temporal and spatial summation of their effects can be observed. Even when PSCs are too far apart to superpose, PSPs may still do so, due to the relatively long membrane time constant. A more detailed understanding of these phenomena, we point to the section on mathematical models of synaptic interactions (Section 2.2.2) and the analytical solution of the membrane potential equation (Sections 4.2.1 – 4.2.4).

Dale's law

John Eccles in 1954[17], is known as Dale's law, and remains until today an important rule of thumb with only few known exceptions.

The receptors activated by particular neurotransmitters are only permeable to specific ion types. Depending on whether channeled ions increase or decrease the neuron membrane potential, receptors can be classified as either excitatory or inhibitory. The two major neurotransmitters in the mammalian CNS are glutamate and GABA, which preferentially target excitatory and inhibitory receptors, respectively. Therefore, and as a corollary of Dale's law, depending on whether a neuron is glutamatergic or GABAergic, it can be either excitatory or inhibitory, but usually not both at the same time.

Given that communication at chemical synapses happens through diffusion, a fraction of the released neurotransmitter molecules can escape the synaptic cleft and diffuse freely

[17] There has been quite some historical controversy surrounding the precise wording of Dale's law. It concerns the ambiguity of the original statement from 1954 about whether one neuron may release multiple types of neurotransmitters at its terminals (Eccles et al., 1954). A revised version that is in compliance with today's knowledge has been formulated by Eccles in 1976: "I proposed that Dale's Principle be defined as stating that at all the axonal branches of a neurone, there was liberation of the same transmitter substance or substances." (Eccles, 1976)

into the intercellular medium. These can then affect neighboring neurons non-synaptically in various ways, triggering complex metabolic pathways that may, in turn, cause profound changes in the dynamics of the neural network. This capability of neuromodulation gives chemical synapses far-reaching functional control over a wide range of temporal and spatial scales, much in contrast to their electrical counterparts. *neuromodulation*

Due to the complexity of the neurotransmitter release-reuptake-cycle, synaptic transmission can vary over time in various ways. It can either be subject to intrinsic dynamics, such as a gradual weakening of the synapse due to vesicle depletion, or be influenced externally by e.g. the firing of the postsynaptic neuron. This phenomenon, called synaptic plasticity, can obviously carry deep functional consequences for any network of spiking neurons. As such, it plays an essential role in learning, adaptation, memory formation etc., and is therefore a key component in neural modeling. *synaptic plasticity*

Because of their complexity, synaptic dynamics are rarely modeled in full detail. Especially when it comes to plasticity, models become increasingly phenomenological and less mechanistic. Many famous models of synaptic plasticity, such as the Hebb and BCM rules, have been originally formulated as firing-rate dependent and are therefore not easily implementable in spiking neural network models. Section 2.2.2 will address the modeling of synaptic transmission and plasticity in more detail.

2.1.4. Spatial Structure of Neurons

Until now, we have only considered the dynamics of structureless, pointlike neurons. The structure of neural cells is, however, important for two reasons. First and foremost, the structure of a neuron can have a profound impact on the way it processes inputs from other neurons. Secondly, the interplay of membrane morphology and electrochemistry leads to a preferred directionality in the transmission and processing of information.

A sketch of the functionally most relevant structural components of a neuron can be found in Figure 2.12. A characteristic feature of a neuron is the branching tree of dendrites that grow out of the cell body or soma. Synaptic stimuli generate electrical excitations of the cell membrane that travel across the dendrites towards the soma. Projecting out of the soma is a single so-called axon, which is usually longer and thicker than the dendrites. At the junction between soma and axon, also called the axon hillock, the ion channel density is particularly high, making it the area most likely to trigger an AP. The AP then travels along the axon towards terminals connecting it, via synapses, to other neurons, through which it can impinge on their respective membranes. *soma* *axon* *axon hillock*

An AP that has been initiated at the axon hillock can only move away from the soma, which we shall henceforth label as the forward direction. The reason for this lies within the very mechanisms that generate the AP described in the previous section. While the leading edge of the spike moves forward due to the activation of Na^+ channels, the trailing edge does so due to the inactivation of the Na^+ channels and activation of the K^+ channels. Thus, at any point in time, the membrane excitation can not move backwards due to the Na^+ channels towards the soma having already been inactivated. *forward propagation*

Let us now consider a simple model for the propagation of electrical signals along the neural membrane. The so-called cable theory dates back to the work by Thomson *cable theory*

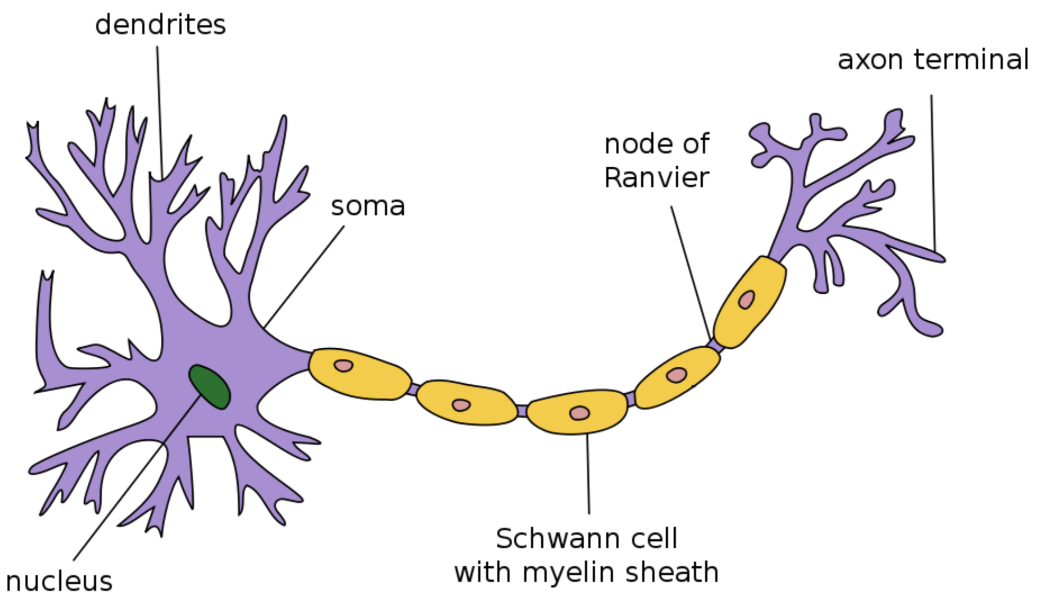


Figure 2.12.: Sketch of a neural cell. The cell nucleus resides in the cell body or soma. Thin, branching projections called dendrites gather information from afferent neurons. Spikes generated by this neuron are transmitted to efferent neurons via the axon, which is surrounded by a myelin sheath for faster signal transmission via saltatory propagation, from one node of Ranvier to the next. Dendrites of target neurons dock via synapses at the axon terminals. Modified from http://en.wikipedia.org/wiki/File:Neuron_Hand-tuned.svg.

(Lord Kelvin) from the 1850s, and was initially developed to model signal transmission in submarine telegraphic cables. We will not provide a full mathematical derivation here, since it is only of peripheral interest to the present work (see Section 5.3), and recommend Chapter 2.5 of Gerstner and Kistler (2002) instead, which features an in-depth discussion of the cable equations.

passive signal transmission

Neural cable theory assumes, in a first approximation, that all components of the neuron (dendrites, soma, axon) transmit electrical signals passively. The radius of the "neural cable" is modeled as constant and, at any point on the membrane, incident currents are assumed to sum up linearly. These are indeed very crude simplifications of neural electrophysiology[18], but the mathematical tractability gained from sacrificing biological fidelity yields important insights into the effects of spatial structure on the propagation of electrical signals, which obviously has functional implications for neural information processing. We consider it crucial to accentuate this aspect, especially given the fact that, as we shall see later, network models very often do not take into account the full consequences of neural morphology.

[18] As we have seen in Section 2.1.2, active (voltage-gated) channels on the membrane are a major component of membrane dynamics, even in the subthreshold regime. Dendrites also become thicker as they join and approach the soma, with the soma diameter being many times larger than that of distal dendrites. Finally, transmembrane currents do not sum up linearly, as the transmembrane proteins are not ideal resistors.

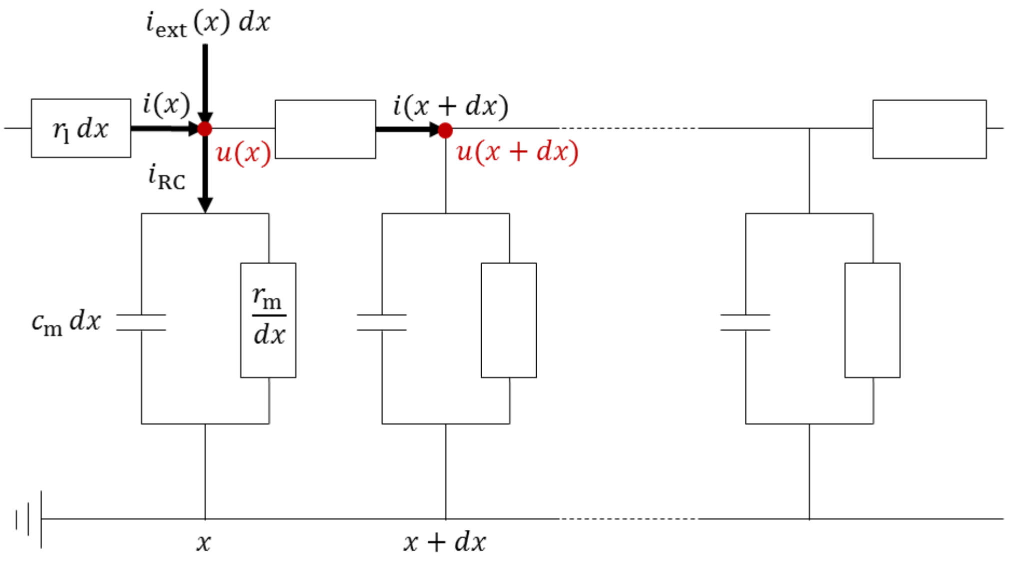


Figure 2.13.: The neuron membrane can be viewed as being composed of infinitesimal segments dx, each of which contains a longitudinal resistor $r_l dx$ and a leak circuit, which consists of a capacitor $c_{\text{m}}dx$ and a resistor r_{m}/dx connected in parallel. Due to gauge freedom one can set the voltage of the cell exterior, represented by the bottom horizontal wire, to zero (ground). The cable equation can be found by applying Ohm's law over the longitudinal resistive element and Kirchhoff's current law at a node along the inner cell membrane (top horizontal wire).

infinitesimal circuits

In classical cable theory, the membrane acts as a sequence of infinitesimal RC circuits connected in parallel (Figure 2.13). Here, $r_l dx$ represents the longitudinal resistive element and r_{m}/dx and $c_{\text{m}}dx$ the transversal resistive and capacitive elements, respectively. The latter are equivalent to $1/g_l$ and C_{m}, which have been discussed in Section 2.1.1. External currents that excite the membrane are subsumed under the notation $i^{\text{ext}}dx$. Note how the additive rules for resistors, capacitances and currents lead to the factors "$\cdot\, dx$" and "$/dx$", depending on whether they are connected in parallel or in series. The infinitesimal elements we use here are measured per membrane-length unit (and hence $[r_{\text{m}}] = \Omega\text{m}$, $[r_l] = \Omega/\text{m}$, $[c_{\text{m}}] = \text{F/m}$ and $[i^{\text{ext}}] = \text{A/m}$).

Without loss of generality, we can assume the resting potential of the membrane to be 0. The longitudinal current through the membrane $i(t,x)$ causes a voltage drop over the longitudinal resistance $r_l dx$ of

$$u(t,x) - u(t,x+dx) = i(t,x) r_l dx. \tag{2.9}$$

The current flowing through the infinitesimal RC circuit at point x is the sum of the current that charges the capacitance $c_{\text{m}}dx$ and the one flowing through the resistor r_{m}/dx:

$$i^{\text{RC}}(t,x) = c_{\text{m}}dx \frac{\partial}{\partial t} u(t,x) + \frac{u(t,x)}{r_{\text{m}}/dx}. \tag{2.10}$$

Conservation of current at point x demands that

$$i(t,x+dx) - i(t,x) - i^{\text{ext}}(t,x)dx + i^{\text{RC}} = 0. \tag{2.11}$$

In the limit of $dx \to 0$, we can rearrange equations such that we can substitute $\frac{u(t,x+dx)-u(t,x)}{dx}$ and $\frac{i(t,x+dx)-i(t,x)}{dx}$ by $\frac{\partial u(t,x)}{\partial x}$ and $\frac{\partial i(t,x)}{\partial x}$, respectively. As a final step, we can now substitute Equations 2.9 and 2.10 into Equation 2.11 (and drop the arguments x and t for clarity). The cable equation describing this system then reads

cable equation

$$\frac{1}{r_\mathrm{l}}\frac{\partial^2 u}{\partial x^2} = c_\mathrm{m}\frac{\partial u}{\partial t} + \frac{u}{r_\mathrm{m}} - i^\mathrm{ext}, \tag{2.12}$$

electrotonic length scale

By setting $\lambda_\mathrm{m} = \sqrt{r_\mathrm{m}/r_\mathrm{l}}$ (the so-called electrotonic length scale) and $\tau_\mathrm{m} = r_\mathrm{m}c_\mathrm{m}$ (the membrane time constant, see also 2.3), we can rewrite equation 2.12 as

$$\tau_\mathrm{m}\frac{\partial u}{\partial t} - {\lambda_\mathrm{m}}^2\frac{\partial^2 u}{\partial x^2} + u = r_\mathrm{m}i^\mathrm{ext}, \tag{2.13}$$

Each of the two specific constants in this equation describes an intuitive property of the membrane. Assume, for instance, that we inject a constant current $i^\mathrm{ext}(t,x) = \delta(x)/r_\mathrm{m}$ at $x = 0$ and wait until the membrane potential does not change anymore in time. Equation 2.13 then becomes

$${\lambda_\mathrm{m}}^2\frac{\partial^2 u}{\partial x^2} = u - \delta(x) \tag{2.14}$$

and we can find the stationary solution

$$u(x) = \frac{1}{2}\exp\left(-\frac{|x|}{\lambda_\mathrm{m}}\right). \tag{2.15}$$

It now becomes apparent that the electrotonic length scale λ_m is a measure of the attenuation of an input signal along the membrane. More precisely, it represents the length after which a stationary signal becomes weaker by a factor of $1/e$.

Similarly, if one injects a current homogeneously along the entire membrane, the spatial derivative in Equation 2.13 vanishes and we are left with

$$\tau_\mathrm{m}\frac{\partial u}{\partial t} = -V + r_\mathrm{m}i^\mathrm{ext}, \tag{2.16}$$

which is exactly equivalent to the pointlike leaky neuron model by Equation 2.3. For a constant current $i^\mathrm{ext}(t) = 1/r_\mathrm{m}\Theta(t)$, we can easily write down the solution

$$u(t) = \Theta(t)\left[1 - \exp\left(-\frac{t}{\tau_\mathrm{m}}\right)\right] \tag{2.17}$$

and see that the membrane time constant τ_m is a measure of the reaction speed of the membrane to changes in stimulus.[19] Similar to λ_m, it represents the time after which the membrane gets closer to its new equilibrium value by a factor of $1/e$.

[19] For a dendrite with a radius of 1 μm, we can find typical values of $r_\mathrm{l} = 3 \cdot 10^5\ \Omega/\mu\mathrm{m}$, $r_\mathrm{m} = 5 \cdot 10^{11}\ \Omega\mu\mathrm{m}$ and $c_\mathrm{m} = 5 \cdot 10^{-14}\ \mathrm{F}/\mu\mathrm{m}$. The corresponding electrotonic length scale and membrane time constant are $\lambda_\mathrm{m} = 1.2\,\mathrm{mm}$ and $\tau_\mathrm{m} = 25\,\mathrm{ms}$.

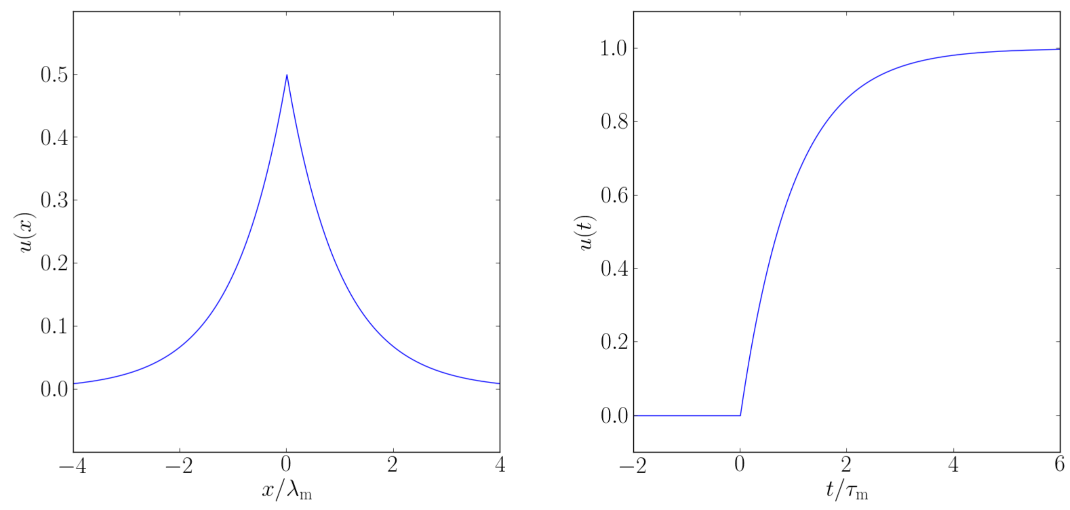


Figure 2.14.: Special solutions to the cable equation. The membrane potential is given in arbitrary units, as we are only interested in the shape of the curves. **Left:** stationary solution with respect to time. A constant current is applied at a single point $x = 0$ along the membrane. The membrane potential decays exponentially as a function of distance to the point of injection, with a decay constant equal to the electrotonic length scale λ_m. **Right:** stationary solution with respect to space. All points along the membrane receive the same current, which in this case is a step function at $t = 0$. The membrane potential decays exponentially towards the new equilibrium value, with a decay constant equal to the membrane time constant τ_m.

We can now search for analytical solutions to the cable Equation 2.13. In order to promote an intuitive understanding, we note how, although it appears here in the context of electrodynamics, this type of PDE is found in many areas of physics, including thermodynamics and quantum mechanics. It is closely related to e.g. the Fokker-Planck (a.k.a., depending on context, Smoluchowski) and Schrödinger equations. This already gives us a strong hint that dispersion (i.e., spreading of wave packets) plays an important role in the time evolution of membrane excitations. We need to stress, however, that the cable equation does not describe a diffusion process and is only formally similar to one. A detailed discussion of the Fokker-Planck equation as a formalization of a true diffusion process is, however, of central importance to the behavior of neurons driven by stochastic stimuli and shall be discussed in detail in the context of stochastic neural computation (Chapter 6).

dispersion

The general approach to solving linear PDEs such as the cable equation is by using the Green's function formalism. For a generic linear PDE

$$\mathcal{L}u(t,x) = f(t,x) \tag{2.18}$$

with an arbitrary linear differential operator $\mathcal{L} = \mathcal{L}(x,t)$, the Green's function $G(t,t',x,x')$

Green's function

is defined as the solution to

$$\mathcal{L}G(t,t',x,x') = \delta(t-t')\delta(x-x'). \tag{2.19}$$

It can be easily checked by substitution that the general solution of the PDE is given by

$$u(t,x) = \int\limits_{-\infty}^{t} \int\limits_{-\infty}^{\infty} G(t,t',x,x')f(t',x')dt'dx'. \tag{2.20}$$

In particular, if $\mathcal{L}$ is translation invariant (as it is in our case), G serves as a convolution operator for Equation 2.20:

$$G(x,x',t,t') = G(x-x',t-t') \tag{2.21}$$

In the following, we will drop all parameters from Equation 2.13 for clarity. This can be done by rescaling time and space to unit-free coordinates

$$x \to x/\lambda_{\mathrm{m}} \tag{2.22}$$

$$t \to t/\tau_{\mathrm{m}} \tag{2.23}$$

and by additional rescaling of the external current

$$i^{\mathrm{ext}} \to r_{\mathrm{m}} i^{\mathrm{ext}} \tag{2.24}$$

The cable equation then becomes

$$\frac{\partial u}{\partial t} - \frac{\partial^2 u}{\partial x^2} + u = i^{\mathrm{ext}} \tag{2.25}$$

and the Green's function can be given in closed form:

$$G(t,t',x,x') = \frac{\Theta(t-t')}{\sqrt{4\pi(t-t')}} \exp\left[-(t-t') - \frac{(x-x')^2}{4(t-t')}\right]. \tag{2.26}$$

Let us consider a synapse lying at a location that we define as $x = 0$ along a dendrite. Upon arrival of an afferent spike, the current that flows through the synapse and excites the membrane can be modeled as

$$i(t,x) = w^{\mathrm{syn}}\delta(x)\Theta(t)\exp\left[-\frac{t}{\tau^{\mathrm{syn}}}\right]. \tag{2.27}$$

If we neglect boundary effects, we can plug Equations 2.27 and 2.26 into Equation 2.20 to find out how this signal propagates along the dendrite (Figure 2.15).

As expected, we observe the formation of the characteristic PSP shape at the point of injection (blue curve). The steepness of the rising flank is partly determined by the synaptic time constant τ^{syn}, while the falling flank is largely governed by the membrane time constant τ_{m}. At points lying further away on the dendrite, however, the PSP

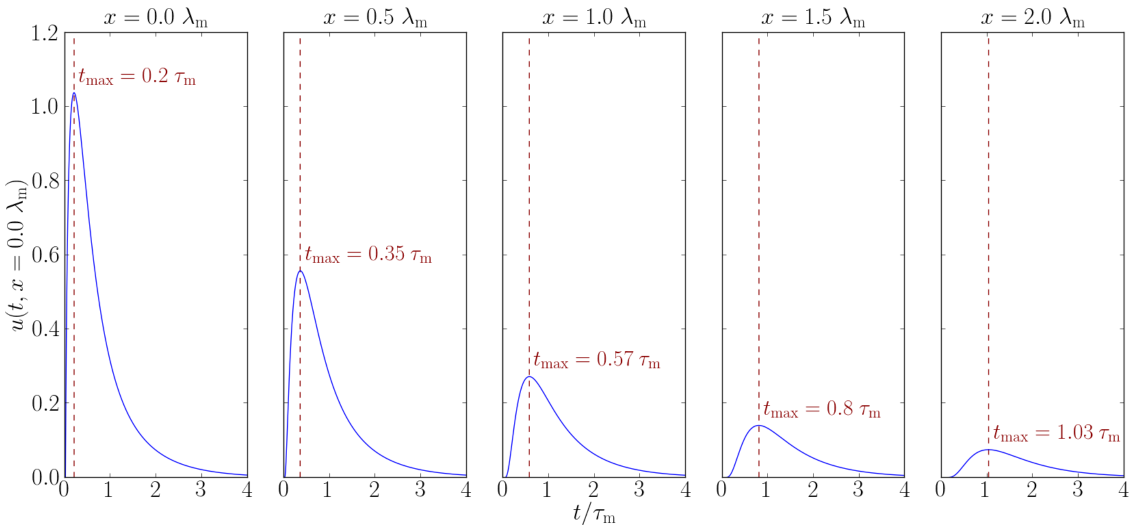


Figure 2.15.: Propagation of membrane potential excitations. The neural cable is excited by a simulated synaptic current with a sudden onset at $t = 0$ and an exponential decay with time constant $\tau^\mathrm{syn} = \tau_\mathrm{m}/5$ at the point $x = 0$ along the membrane. The time course of the membrane potential (PSP) is measured at integer multiples of half the electrotonic length scale $\lambda_\mathrm{m}/2$. Two characteristic dispersive effects become apparent. The amplitude of the PSP decreases and it becomes broader as it propagates away from the current injection site. In addition, the rising flank becomes less steep and the peak voltage is reached later, amounting to an effective dendritic delay.

becomes distorted - the peak voltage is reached later, due to the finite propagation speed of electrical signals along the membrane. Loosely defined as the speed at which the voltage peaks advance, the signal propagation speed is, quite intuitively, a monotonically decreasing function of τ_m and a monotonically increasing function of λ_m. Additionally, the further away one goes from the point of injection, the smaller the peak voltages become, reflecting the previously discussed effect of the electrotonic length scale.

signal propagation speed

We now turn our attention to the propagation of action potentials. By their very nature, it is no longer possible to neglect the active mechanisms that govern their time course. Their propagation is therefore determined by a combination of the passive cable theory from above and the voltage-gated channels from the Hodgkin-Huxley model (Section 2.1.2). However, the results we have gained from studying the behavior of purely passive dendrites can give us some important insights.

In order for an AP to propagate from position x further along the axon, it must elicit a high enough voltage change at the position $x + \Delta x$ for the active mechanisms to take over and cause an AP at this new position. The reaction speed to stimuli at any point along the membrane is governed, as we have seen (in e.g. Equation 2.17 or 2.14), by the membrane time constant τ_m and the electrotonic length scale λ_m. A fast propagation of action potentials towards their target neurons (as would e.g. be required by a quick reaction of the animal to a sensory stimulus) can be aided by two morphological features of axons, both of which have the effect of increasing λ_m.

axon diameter

One straightforward possibility is to increase the diameter of the axon. The total surface area of the axon would grow, allowing more charge to flow through per unit of time, but also requiring more charge per unit of voltage. This would lead to a decrease in r_m but also to a simultaneous increase in c_m, thereby leaving τ_m unaffected. However, the longitudinal resistance r_l would also be reduced, leading to an increase in λ_m and thereby increasing the passive signal propagation speed. Evolution has put this effect to use, as famously illustrated by the millimeter-thick giant squid axons on which Hodgkin and Huxley performed their Nobel-earning experimental work.

However, particularly in younger taxa[20], evolution has found a more efficient solution. Especially those neurons which require long axonal projections, such as motor neurons which project from the spine to the extremities, have their axons surrounded by a so-called myelin sheath. The myelin is produced by specialized glia called Schwann cells that wrap themselves around the axon in multiple layers, effectively increasing the thickness of the membrane. This decreases the membrane capacitance, but more importantly, it greatly increases the resistance across the membrane. This, in turn, leads to an increase in λ_m and thereby to a faster signal transmission. Between the Schwann cells there remain some unmyelinated axon surface patches, called the nodes of Ranvier. At each such node, the voltage-gated proteins have access to the intercellular medium and can therefore actively "refresh" the AP by the Hodgkin-Huxley mechanism. Due to the "jumping" nature of the APs, their propagation along a myelinated axon is called "saltatory".

myelin sheath, Schwann cells

nodes of Ranvier

saltatory propagation

The interplay between active and passive transmission also determines the directionality of the AP propagation. Having reached some point along the axon, the AP can not propagate backwards, because those patches of axon membrane lying behind have already been excited and then remain refractory for a significant period of time.

AP back-propagation

What about the propagation of action potentials from the axon hillock backwards through the dendritic tree? Even without voltage-gated ion channels, APs can still backpropagate by passive transmission only. Moreover, if dendrites are equipped with a high enough density of such channels, the AP propagation should be, in principle, quite similar to the one in unmyelinated axons. Indeed, both varieties of backpropagating APs have been found to occur in nature (Waters et al. (2005)). They have been hypothesised to play a significant functional role as well, allowing a sort of feedback mechanism for synaptic plasticity phenomena that require the "knowledge" of both afferent and efferent spiking activity. One such mechanism, called STDP, will be briefly addressed in Sections 2.2.2.2, where we discuss synaptic plasticity.

We conclude this section with several important remarks. Since spike initiation is quite narrowly localized in space - at the axon hillock - it is both the shape and the timing of the PSPs at that precise site that determines whether the neuron spikes or not. We have seen how these PSP features strongly depend on the position of synaptic current injection, as well as the membrane properties of the dendritic tree (in particular, λ_m and τ_m). We

[20] It is noteworthy that myelination can also be found in some older taxa as a result of convergent evolution. While not morphologically identical, invertebrate myelin sheaths serve the same functional purpose as in vertebrates.

therefore conclude that the morphology of a neural cell is essential to the information processing that it performs. The position of a synapse can, for example, influence both delay-based computation, which is essential in e.g. synfire chain models (see Section 5.4), as well as PSP size and shape, which determines functionality in virtually all network models.

As we shall see in the section on simulation software (3.1), dendritic delays can be taken into account for point neuron models, but not morphology-dependent PSP shapes. Modeling the full structure of the dendritic tree would be computationally costly, while also severely limiting the analytical tractability of network dynamics. It is precisely due to their analytic and computational tractability that the remainder of this work is largely dedicated to point neuron models. We therefore need to point out that while the implementation of algorithms in neural networks from a machine learning perspective does not require biological fidelity, any claims about biology coming from single-compartment modeling must be treated with appropriate care and rigor.

tractability vs. biological fidelity

2.2. Abstract Models

At this point, we have formulated a mathematical model of biological neuron dynamics and have given an approximate description of synaptic interaction. In its full complexity, our neuron model (Hodgkin-Huxley equations plus the cable equation) is computationally extremely costly, with 4 ODEs for the soma and 1 ODE for each dendritic compartment that is modeled as a single cable (assuming a spike can only be triggered at the soma). Synapse dynamics are even more complicated; modeled in full detail, each synapse would require multiple ODEs for neurotransmitter release at the presynaptic site, diffusion in the synaptic cleft, ligand-gated channeling at the postsynaptic site, neurotransmitter metabolization etc.

If we are to simulate neural networks at the level of individual neurons and synapses, a small set of simple, linear ODEs would be highly advantageous. Simplification, of course, comes at the price of accuracy – or worse, at the expense of functionality, as we have already discussed for the HH model (Section 2.1.2). The formulation of abstract models therefore always requires a careful consideration of the tradeoff between functionality (and/or faithfulness to biology) and computational complexity.

In the following, we will elaborate the neuron and synapse models that have been used as building blocks for all the network models discussed in later chapters. Furthermore, these models define the target dynamics of the circuits implemented in our neuromorphic hardware. In Section 2.2.1, we will address simplifications of the HH equations and formulate two abstract neuron models: the simple leaky integrate-and-fire model as a stripped-down version of the HH neuron and the more complex adaptive exponential integrate-and-fire model, which even includes dynamics beyond the HH equations. In Section 2.2.2, we will formalize synaptic dynamics and discuss two simple models of synaptic plasticity.

2.2.1. Neurons

point neuron models

As already addressed in Section 2.1.4, the first simplification we make is to utterly neglect signal propagation in the dendritic tree. We assume our neurons to be pointlike, so all input currents have an immediate effect on the membrane potential of the (pointlike) soma. However, the distance of a synapse from the soma, i.e., the dendritic delay, can still be taken into account. Here, we model the delayed arrival of PSPs at the soma by delaying the arrival of spikes. In both cases, the result is a temporal shift of the PSP incidence at the soma; however, in this model, we lose the (potentially computationally relevant) reshaping of the PSP as a function of the distance it travelled.

In a second step, we reconsider the equations of the HH model, which had been motivated by the desire to have a mechanistic model of the biological dynamics in excitable cells. In particular, most of the model complexity serves almost exclusively for modeling action potentials: the voltage-gated channel dynamics effectively make up three (Equation 2.6) of the four ODEs – and most of the fourth (Equation 2.8) as well. However, spikes are generally assumed to be stereotyped events, i.e., with nearly identical shape (Gerstner and Kistler (2002), Dayan and Abbott (2001)). Under this assumption, it is only the timing of the individual spikes that matters, thereby rendering its detailed modeling redundant.[21] In many simplified neuron models, the spiking mechanism is

[21] It is important to remember that this, too, is a simplification and does not hold without exception.

therefore replaced by a threshold condition: when a neuron's membrane potential reaches the said threshold at time t_s, it is instantaneously pulled back to a reset value and the neuron "sends a spike" to its efferent neurons. These spikes are typically modeled as a delta function of time. We discuss such a model in Section 2.2.1.1.

threshold models

One needs to remember, however, that the reduction in the number of equations of motion (and the accompanying reduction of the system's phase space spanned by its dynamic variables) invariably leads to a loss of "dynamic richness".[22] In particular, it is impossible to reproduce the driven-oscillator-like subthreshold dynamics of the HH model (see the end of Section 2.1.2) in a one-dimensional space. To what extent this results in a loss of computational functionality depends on the network model. In any case, networks based on one-dimensional neuron models can still exhibit extremely interesting dynamics (Chapter 4) and perform complex computational tasks (Chapter 6).

Evidently, the simplified one-dimensional model class outlined above does not need to be the end of the path of abstraction. Indeed, the basic[23] HH model has its own shortcomings and can not capture the entire observed spectrum of single-neuron behavior. This can, however, be remedied by the inclusion of additional dynamic variables which, for example, influence subthreshold dynamics on longer timescales. We shall discuss in detail two network models that rely on 2D neuron dynamics in Chapter 5. Their underlying neuron model is described in detail in Section 2.2.1.2.

adaptive models

2.2.1.1. The Leaky Integrator

LIF model

The leaky integrate-and-fire (LIF) model is one of the simplest neuron models that can claim biological relevance. It is almost as old as modern neuroscience itself (Lapicque, 1907), although its name was introduced only about half a century later (Brunel and Van Rossum, 2007). The name basically says it all, and we have already discussed the differential equation of this model in the section on passive membrane properties (Equation 2.3). A ($\dot{u} \propto -u$)-term represents the leak, whereas the integration of the input current is embedded via $\dot{u} \propto I$:

$$C_\mathrm{m}\frac{du}{dt} = g_\mathrm{l}(E_\mathrm{l} - u) + I^\mathrm{syn} + I^\mathrm{ext} \quad . \tag{2.28}$$

LIF equation

Here, we have subdivided the total input current I into a synaptic component I^syn and a generic external one I^ext. The latter gives additional control over the model and is equivalent to a modulation of E_l. The firing is taken care of by a simple threshold rule:

While most neocortical neurons appear to conform to this assumption, networks exist – e.g., in the elephantnose fish – in which different shapes of action potentials have been measured (Sugawara et al., 1999) and hypothesized to play a functional role (Mohr et al., 2003a,b).

[22] As an example, we mention the Poincaré-Bendixson theorem, which makes a statement about the periodicity of orbits (limit cycles) for two-dimensional dynamical systems. In particular, it forbids the existence of chaotic behavior such as strange attractors. This clearly does not hold for higher-dimensional phase spaces, such as the Lorenz system with its well-known "butterfly attractor".

[23] I.e., with additional dynamics that only cover the generation of action potentials - as described in Section 2.1.2 and defined by Equations 2.6 and 2.8 with the parameters from Section A.2.1

if the membrane potential crosses a threshold ϑ from below[24], a spike is emitted:

$$\text{neuron spikes at } t = t_{\text{spike}} \Longleftrightarrow u(t_{\text{spike}}) = \vartheta \quad . \tag{2.29}$$

spike train

The outgoing spikes are defined only by the time of their occurrence and form a so-called spike train, which is modeled as

$$\rho(t) = \sum_{\text{spikes } s} \delta(t - t_s) \quad . \tag{2.30}$$

reset potential *absolute refractory time*

Whenever the neuron spikes, its membrane is reset to a potential ϱ. In order to model the refractoriness of biological neurons, the membrane is clamped to the reset potential for a duration τ_{ref} called the (absolute) refractory time:

$$u(t_{\text{spike}} < t \leq t_{\text{spike}} + \tau_{\text{ref}}) = \varrho \quad . \tag{2.31}$$

1-D model

The Equations 2.28, 2.29 and 2.31 fully define the LIF model. Despite formally being described by three equations, it is important to note that the model itself only has a single dynamic variable, namely the membrane potential u, thus being one-dimensional. It might seem that the synaptic interactions condensed into I^{syn} offer additional degrees of freedom, but this is not the case, since I^{syn} is fully determined by the spike trains, and therefore by the membrane potentials, of other neurons in the network - as we shall see in Section 2.2.2.

To gain some intuition for this model, we shall briefly describe several simple single-neuron experiments. In particular, this means that there is no synaptic stimulus, so $I^{\text{syn}} \overset{!}{=} 0$. The general solution of the LIF equation 2.28 can be easily found:

$$u(t) = u_0 e^{-\frac{t}{\tau_{\text{m}}}} + \frac{e^{-\frac{t}{\tau_{\text{m}}}}}{\tau_{\text{m}}} \int_0^t dt' \left(E_{\text{l}} + \frac{I^{\text{ext}}(t')}{g_{\text{l}}} \right) e^{\frac{t'}{\tau_{\text{m}}}} \quad . \tag{2.32}$$

If the current remains constant in time, the equation takes an even simpler and more intuitive form:

$$u(t) = E_{\text{l}} + \frac{I^{\text{ext}}}{g_{\text{l}}} + \left(u_o - E_{\text{l}} - \frac{I^{\text{ext}}}{g_{\text{l}}} \right) e^{-\frac{t}{\tau_{\text{m}}}} \quad , \tag{2.33}$$

membrane time constant

where $u_0 := u(t = 0)$ and $\tau_{\text{m}} = \frac{C_{\text{m}}}{g_{\text{l}}}$ represents the membrane time constant (see also Equation 2.13), which quantifies the relaxation speed of the membrane potential towards the equilibrium value $E_{\text{l}} + I^{\text{ext}}/g_{\text{l}}$.

subthreshold stimulus *suprathreshold stimulus*

If the equilibrium value lies below the spiking threshold ($E_{\text{l}} + I^{\text{ext}}/g_{\text{l}} < \vartheta$), the current stimulus is called subthreshold. The time course of the membrane potential then corresponds to the charging/discharging of a capacitor. If the equilibrium value lies above the spiking threshold ($E_{\text{l}} + I^{\text{ext}}/g_{\text{l}} > \vartheta$), the current stimulus is called suprathreshold. The trajectory of the membrane potential then remains piecewise exponential, but becomes discontinuous, due to the reset when crossing the threshold. Consequently, the LIF

[24] This is the standard textbook definition, but in this formulation of the model, the "from below" can be omitted, since the equations prevent the membrane potential from ever lying above the threshold.

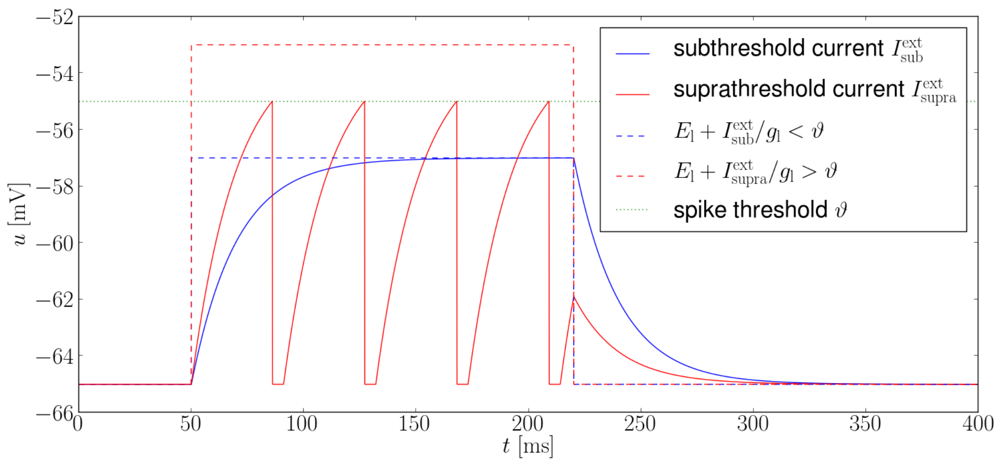


Figure 2.16.: Membrane potential of an LIF neuron with step current stimulus. The membrane potential always converges exponentially towards the equilibrium value $E_{\mathrm{l}} + I^{\mathrm{ext}}/g_{\mathrm{l}}$ with a time constant $\tau_{\mathrm{m}} = \frac{C_{\mathrm{m}}}{g_{\mathrm{l}}}$. During subthreshold stimulation (blue), the membrane follows the charge/discharge curve of a capacitor. During suprathreshold stimulation (red), the neuron spikes regularly with a rate given by Equation 2.36.

neuron fires periodically, with a firing rate that can be computed by setting appropriate boundary conditions for Equation 2.33: *periodic firing*

$$u_0 = \varrho \tag{2.34}$$

$$u(T) = \vartheta \tag{2.35}$$

$$\Rightarrow \nu = (\tau_{\mathrm{ref}} + T)^{-1} = \left(\tau_{\mathrm{ref}} + \tau_{\mathrm{m}} \ln \frac{\varrho - E_{\mathrm{l}} - \frac{I^{\mathrm{ext}}}{g_{\mathrm{l}}}}{\vartheta - E_{\mathrm{l}} - \frac{I^{\mathrm{ext}}}{g_{\mathrm{l}}}} \right)^{-1} \tag{2.36}$$

The firing rate of a neuron as a function of its input - in this case, of the input current I^{ext} - is appropriately called an f-I curve. The terms gain function and activation function are often used synonymously in literature. Figure 2.17 shows the f-I curve of two (nearly) identically parametrized LIF neurons, one with and the other without a refractory period. *f-I curve, gain function, activation function*

Without refractoriness, the firing rate diverges for large input currents, as the argument of the logarithm in Equation 2.36 approaches unity. The asymptotic behavior can be easily found with an appropriate Taylor expansion of Equation 2.36 in $(I^{\mathrm{ext}})^{-1}$:

$$\nu(I^{\mathrm{ext}}) \quad \overset{I^{\mathrm{ext}}\to\infty}{\approx} \quad \frac{1}{\tau_{\mathrm{m}}(\vartheta - \varrho)} \left(E_{\mathrm{l}} - \frac{\varrho + \vartheta}{2} + \frac{I^{\mathrm{ext}}}{g_{\mathrm{l}}} \right) \quad . \tag{2.37}$$

The inclusion of a nonzero refractory period enables a more biologically plausible, convergent behavior towards a finite firing rate which, again, follows directly from Equation 2.36:

$$\nu(I^{\mathrm{ext}}) \quad \xrightarrow{I^{\mathrm{ext}}\to\infty} \frac{1}{\tau_{\mathrm{ref}}} \quad . \tag{2.38}$$

We shall see the activation function reappear prominently in Chapter 6.

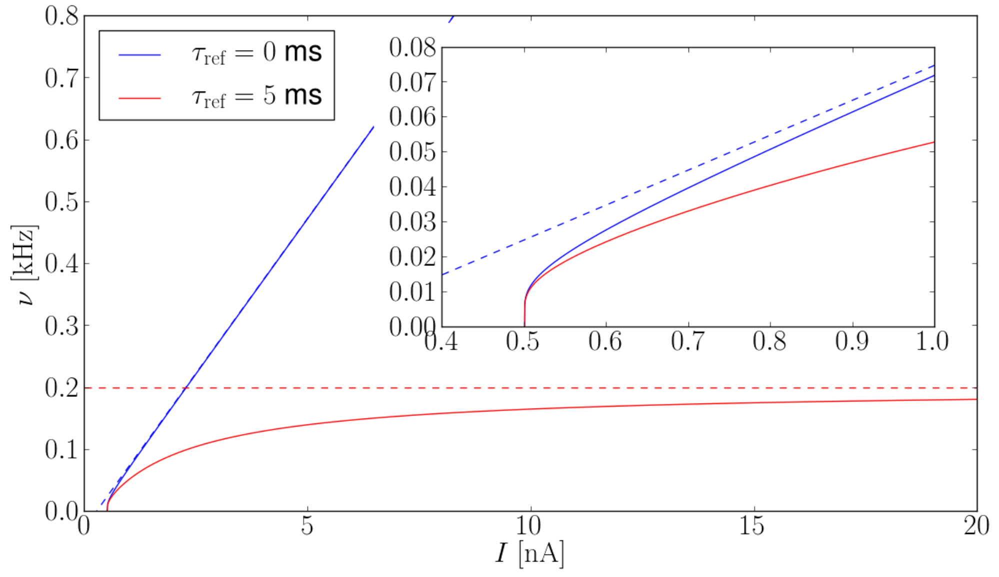


Figure 2.17.: f-I curve of an LIF neuron. Without refractoriness, the firing rate diverges with a linear asymptotic behavior. If the refractory time is nonzero, the firing rate converges towards a maximum value of $1/\tau_{\rm ref}$. The inset represents a zoom onto the point of firing initiation. We can see that the firing rate is a continuous function of time - there is no jump at $I^{\rm ext} = 0.5$ nA. This feature is characteristic of type I models.

As can be seen in Figure 2.17, the transition from a non-firing (subthreshold excitation) to a firing state (suprathreshold excitation) is a continuous function of the input current. This is a characteristic feature of so-called type I models, which display a sharp voltage threshold and a zero-frequency onset of stable oscillations (regular spiking). In contrast, so-called type II models do not have a sharp threshold and oscillations start with nonzero frequency. Two examples, both derived as simplifications of the HH model (which is itself of type II), are the Connor model for type I (Connor et al., 1977) and the Fitzhugh-Nagumo model for type II (FitzHugh, 1961); depending on their parameters, some models can be either (Morris and Lecar, 1981).

type I models
type II models

It should be noted here that the formal definition of model types varies throughout literature (see, e.g., Ermentrout, 1996) and there is no clear-cut criterion for classification. Some authors define the type of the excitability directly from the f-I curve, in which case the LIF model is classified as type I excitable. However, most authors discuss this classification in terms of bifurcation theory (in particular, Hopf vs. saddle-node bifurcations), therefore making it difficult to formally classify the linear LIF model as either type according to these criteria. For a comprehensive discussion of phase plane analysis, we refer to Chapter 3 of Gerstner and Kistler (2002).

2.2.1.2. The Adaptive Exponential Integrate-and-Fire Model

Up to here, we have argued from the perspective that action potentials are stereotyped events with no information contained in their shape but only in their timing. Furthermore, we have implied that the LIF model captures the most important aspects of subthreshold dynamics, rendering any additional terms and equations of the HH model essentially perturbative. We now reconsider these hypotheses by raising two issues.

Let us first consider the spike initiation of an LIF neuron. Due to the firing condition (Equation 2.29), the spike timing depends critically on the choice of the threshold. Since biological neurons do not have such a threshold, the extraction of this parameter from electrophysiological data is conceivably difficult and prone to error. Even if we loosely define the "biological threshold" as the initiation point of the membrane potential upswing during a spike, then clearly the nonlinear terms in the HH equation play an important role for the membrane dynamics close to this threshold. Such effects can not be captured by a purely linear model such as LIF.

The second argument comes directly from electrophysiological data. A regular oscillatory behavior, as described above for LIF neurons, is only one of many possible responses of different neuron types in the cortex to a constant current stimulus (see, e.g., Markram et al., 2004b). In contrast, both the LIF model and the basic HH model can only produce constant-frequency oscillations. The adaptive exponential integrate-and-fire (short: AdEx) model covers both of the above issues - expectably at the price of added complexity. *AdEx model*

The issue of spike generation can be addressed by adding nonlinear (current) terms to the equation that governs the membrane potential. In case of the AdEx model, the added term is an exponential function of the membrane potential: *exponential term*

$$I^{\mathrm{exp}} = g_{\mathrm{l}} \Delta_{\mathrm{T}} \exp\left(\frac{u - E_{\mathrm{T}}}{\Delta_{\mathrm{T}}}\right) \tag{2.39}$$

This term offers two degrees of freedom. The threshold voltage E_{T} plays a similar role to the "hard" threshold ϑ in the LIF model. Loosely speaking, when the membrane potential crosses E_{T} from below, I^{exp} begins to dominate the membrane dynamics, pushing the membrane potential even further "upwards". In contrast to the LIF model however, this is not an all-or-none condition: at any point in time, the positive contribution of the exponential term can be countered by an appropriate negative contribution from external stimuli (e.g., inhibitory afferents). The relative strength of I^{exp} is modulated by the so-called slope factor Δ_{T}, which is required to be positive. *threshold voltage* *slope factor*

The more important issue is the one related to spike pattern complexity. This is addressed by including an additional dynamic variable, the adaptation variable w. It enters the membrane potential ODE linearly and is itself described by a first-order linear ODE with jumps upon spiking. Before we address the resulting dynamics, we first write down the full mathematical description of the AdEx model: *adaptation variable*

$$C_{\mathrm{m}} \frac{du}{dt} = g_{\mathrm{l}}(E_{\mathrm{l}} - u) + g_{\mathrm{l}} \Delta_{\mathrm{T}} \exp\left(\frac{u - E_{\mathrm{T}}}{\Delta_{\mathrm{T}}}\right) - w + I^{\mathrm{syn}} + I^{\mathrm{ext}} \quad , \tag{2.40}$$

$$\tau_w \frac{dw}{dt} = a(u - E_{\mathrm{l}}) + b\, \tau_w\, \rho - w \quad , \tag{2.41}$$

where $\rho(t)$ represents the neuron's own spike train (Equation 2.30), and a, b, and τ_w are adaptation parameters discussed below. As these equations still lack a mechanism for *adaptation parameters* pulling down the membrane potential following a spike, the reset mechanism of the LIF neuron is kept in place, albeit with a different, much higher spiking threshold V_{spike}:

$$t_{\text{spike}} \Leftrightarrow u(t_{\text{spike}}) = V_{\text{spike}} \tag{2.42}$$

$$u(t_{\text{spike}} < t \leq t_{\text{spike}} + \tau_{\text{ref}}) = \varrho \tag{2.43}$$

Another advantage of the exponential term becomes apparent here. Barring all constants that can be removed by an appropriate linear transformation of u, the solution to an ODE of type

$$\frac{du}{dt} = \exp(u) \tag{2.44}$$

is

$$u(t) = -\log(c - t) \quad . \tag{2.45}$$

asymptotic divergence

Once the exponential term becomes dominant, the membrane potential diverges asymptotically, reaching infinity in finite time (i.e., at $t = c$). Therefore, as long as the cutoff V_{spike} is high enough, the spike timing does not critically depend on the precise choice of V_{spike}.

As mentioned above, the inclusion of a second dynamic variable is the most important departure from the simple LIF model. The time constant τ_w governs the rate at which the adaptation variable w decays back to 0 and is usually on the order of hundreds of ms. The parameter a determines the influence of the membrane potential on the adaptation; it is used to, for example, model variations in the ion concentration caused by sustaining either a high or a low membrane potential. The parameter b represents the quantal increase of the adaptation variable following a spike, sharing a similar electrophysiological motivation as a. Since it only comes into play when the neuron spikes, b is used to emulate so-called spike frequency adaptation (SFA).

spike frequency adaptation

Depending on the choice of the parameters a and b, the adaptation variable w can have both an excitatory and an inhibitory effect on the membrane potential. Very often, a and b are chosen to be positive, thereby causing a negative adaptation current and therefore a firing rate that tends to decrease over time. In this case, the adaptation acts as a homeostatic mechanism on both the membrane potential and the spike frequency of the neuron. Figure 2.18 shows an example of adapting AdEx dynamics.

neuronal firing patterns

More important, however, is the fact that w enables the AdEx model to emulate a vast array of complex firing patterns, including, but not limited to, the driven oscillations and the inhibitory rebound spiking we discussed earlier for the HH model (Section 2.1.2). Figure 2.20 shows an array of typical cortical neuron firing patterns reproduced by an AdEx neuron with appropriate parameter settings. For a much more in-depth discussion of AdEx dynamics, we refer to Gerstner and Brette (2009) and Naud et al. (2008).

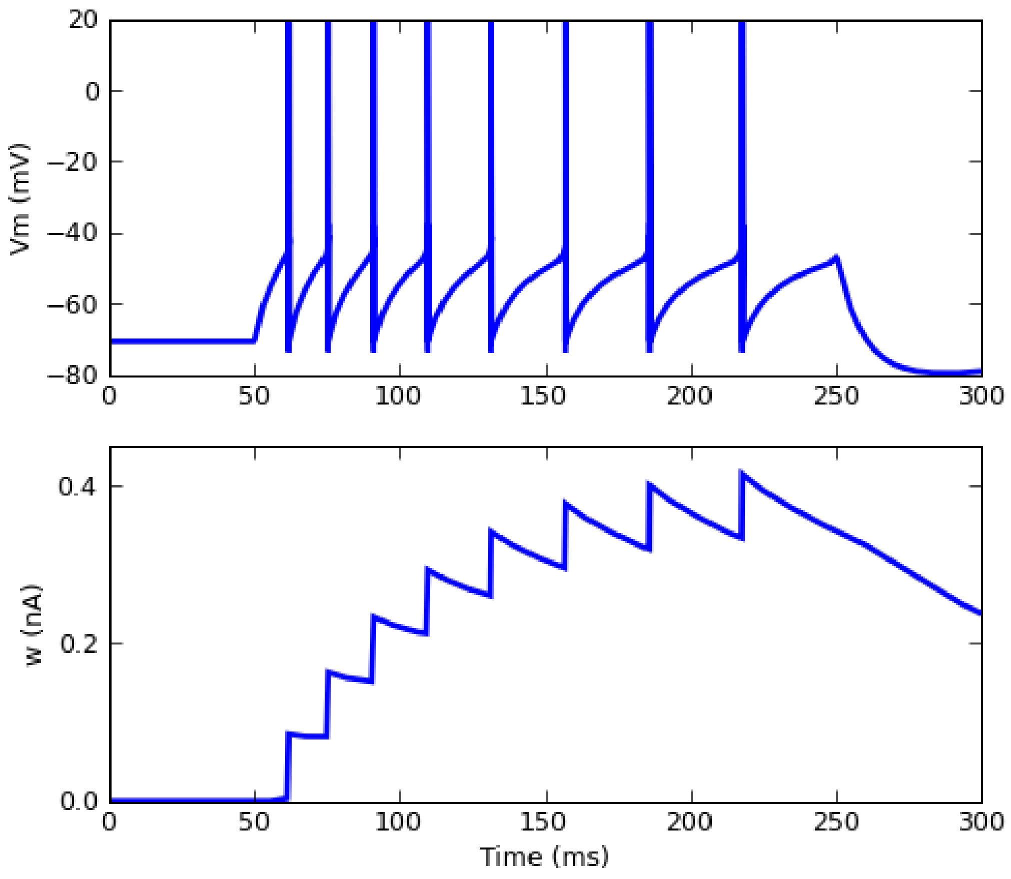


Figure 2.18.: Exemplary dynamics of the AdEx model with a step current stimulus. The parameters were chosen to simulate a slowly adapting behavior (i.e., a slowly decaying firing rate). **Top:** temporal evolution of the membrane potential u. The main component of the membrane potential dynamics is the same as in the LIF model: an exponential decay towards an equilibrium potential, as seen in the trace segments that precede the spikes. This equilibrium potential becomes lower over time due to an increasing (negative) adaptation current $-w$. When it exceeds the spike threshold (-50 mV), the membrane potential diverges asymptotically. **Bottom:** temporal evolution of the adaptation variable w. The leak component of the adaptation ODE can be easily seen in the exponentially decaying segments between the spikes. The quantal increase following each spike causes the jumps in the trace. The dependence of w on u is a bit more subtle, but it can be seen in the slight increase of w around 50 ms and the inflection point of the curve around 250 ms. Figure taken from Gerstner and Brette (2009).

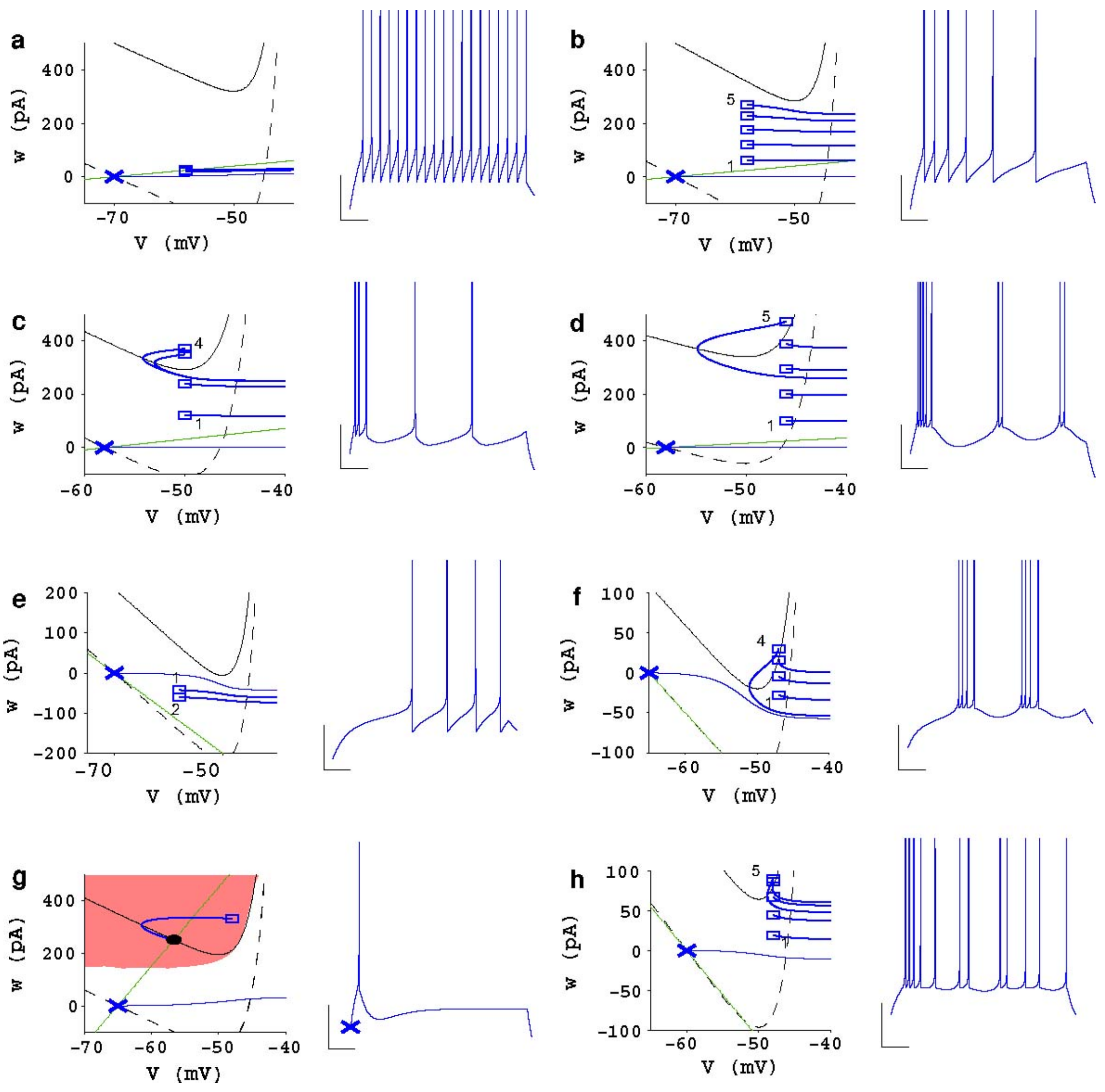

a
b
c
d
e
f
g
h
w (pA)
V (mV)
400
200
0
-70
-50
-60
-40
100
50
-50
-100
-200
1
2
4
5

Figure 2.20.: Eight firing patterns generated by AdEx neurons stimulated by a constant current. The voltage traces are shown with scale bars that correspond to 100 ms and 20 mV, respectively. Each of the voltage plots is accompanied by a depiction of the model's trajectories in the phase plane spanned by u and w. (The membrane potential u is denoted by V in the plots.)

The locus of states where the temporal derivative of a dynamic variable is zero is called a nullcline. (Therefore, trajectories always cross nullclines either vertically or horizontally.) Intersections between nullclines are called fixed points. As becomes evident from Equations 2.41 and 2.40, the w-nullcline (green) is a straight line, whereas the u-nullclines (black) are a superposition of a linear and an exponential component. The u-nullclines with and without current stimulation are represented as solid and dashed lines, respectively.

The neurons are always initialized at their resting state, i.e. at the (stable) fixed point of the system without current stimulus, which is denoted by a blue cross. The reset condition causes the trajectories to be discontinuous: upon spiking, the membrane potential is reset to u_{reset} and the adaptation variable is incremented by b. The points where trajectories reenter the phase plane following a spike are marked by blue rectangles. If this happens more than once, the first and last point of reentry are accompanied by the index of the preceding spike.

(a) Tonic spiking.
(b) Adaptation.
(c) Initial burst.
(d) Regular bursting.
(e) Delayed accelerating.
(f) Delayed regular bursting.
(g) Transient spiking. The stable fixed point is indicated with a black, filled circle.
(h) Irregular spiking.

Figure taken from Naud et al. (2008). The AdEx parameters that were used for the different firing patterns are given in Table 1 of the paper.

LIF from AdEx

On a final note, it should be mentioned that the LIF model is a special case of the AdEx model and can be emulated by an appropriate setting of the parameters Δ_T, a and b. Setting $a = 0$ and $b = 0$ effectively removes adaptation. The exponential term is removed by setting $\Delta_\mathrm{T} = 0$, since

$$\lim_{x\to 0, x>0} x \exp(a/x) \overset{y=1/x}{=} \lim_{y\to\infty, y>0} \frac{\exp(ay)}{y} = \begin{cases} 0 & \text{for} \quad a < 0 \\ \infty & \text{for} \quad a > 0 \end{cases} \quad . \tag{2.46}$$

The exponential current then becomes an all-or-none firing condition, rendering E_T formally (and numerically) equivalent to the hard threshold ϑ in the LIF model.[25]

2.2.2. Synapses

synaptic interaction kernel

As outlined in Section 2.1.3, synaptic interaction is a complex phenomenon, arguably even more complex than the neuron dynamics themselves. For this reason, few, if any, detailed[26] biophysical synaptic models exist. The most widely used synapse models in computational neuroscience are purely phenomenological: synaptic interactions are modeled by stereotypical functions of time called interaction kernels which sum up linearly over space (i.e., over different synapses) and time. The total impact of all synapses can then be written as

$$f^\mathrm{syn}(t) = \sum_{\text{synapses } k} \sum_{\text{spikes } s} w_k \epsilon_k(t - t_s) \quad , \tag{2.47}$$

where w_k denotes the weight or strength of the kth synapse and ϵ_k its synaptic interaction kernel. The interaction kernel can, in principle, assume an arbitrary shape, but in most models it is constrained by the biophysics of synaptic interaction. Before addressing the exact nature of the synaptic input f^syn, we shall first discuss the shape of the synaptic kernels $\epsilon(t)$.

difference-of-exponentials kernel

As described in Section 2.1.3, the synaptic release of neurotransmitters happens very quickly, as does their diffusion towards the postsynaptic terminal, due to the narrow width of the synaptic cleft. The removal of neurotransmitters from the postsynaptic terminal, however, may occur on a wide range of time scales, depending on the nature of the transmitter and receptor molecules. Therefore, a useful phenomonological model is the difference-of-exponentials function[27]:

$$\epsilon(t) = A\Theta(t) \frac{1}{\tau_\mathrm{rise} - \tau_\mathrm{fall}} \left[\exp\left(-\frac{t}{\tau_\mathrm{rise}}\right) - \exp\left(-\frac{t}{\tau_\mathrm{fall}}\right) \right] \quad , \tag{2.48}$$

[25] What works well in theory may not be equally unproblematic in practice. Expressions which may converge to finite values, but contain terms that diverge in the required limit, are notoriously problematic in software implementations. Neural simulation software handles such problems with varying degrees of success: NEST 2.1.1, for example, returns an error, while Neuron 7.1 returns a warning. Therefore, when such a limit is required (such as for the L23 model fit in Section 5.3), particular care needs to be taken. For hardware implementations, such terms become even more problematic, due to limited parameter precision (see Section A.2.2.2). On the HICANN chip, this issue is solved by having the exponential term implemented in a separate circuit that can be effectively decoupled from the cell membrane (see Section 3.3.1).

[26] In the spirit of the Hodgkin-Huxley model of neuron/membrane dynamics.

[27] Incidentally, this function is identical to the PSP shapes derived in Section 4.2 (Equations 4.39 and 4.59). In order to avoid any confusion, we note explicitly that Equation 2.48 represents a phenomenological model of *PSCs*, whereas Equations 4.39 and 4.59 represents the shape of a *PSP*, i.e., the analytical solution of the LIF equation driven by a single, exponentially shaped PSC.

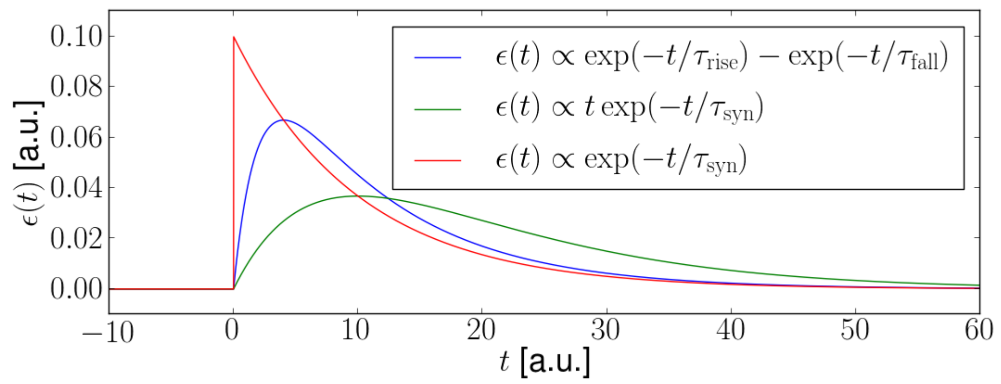


Figure 2.21.: The three different synaptic interaction kernels described in the text. All three kernels have been scaled to unit area. The time constants were set to $\tau_{\text{rise}} = 2$ and $\tau_{\text{fall}} = \tau^{\text{syn}} = 10$ (in arbitrary units of time).

where the two exponential functions model the (stochastic) arrival and removal of neurotransmitters at the postsynaptic site, governed by their respective time constants τ_{rise} and τ_{fall}. For now, A simply represents a constant factor that transforms ϵ_k to an amplitude and units of choice.

Very often, even simpler kernels are used in both theoretical and computational approaches. In the limit of identical time constants τ_{rise} and τ_{fall}, the interaction kernel becomes a so-called α-function, as can be easily derived via l'Hôpital's rule:

α-kernel

$$\lim_{\tau_{\text{rise}} \to \tau_{\text{fall}} = \tau^{\text{syn}}} \epsilon(t) \propto \Theta(t) t \exp\left(-\frac{t}{\tau^{\text{syn}}}\right) \quad . \tag{2.49}$$

The probably most popular synaptic interaction kernel results from the assumption that the diffusion of neurotransmitters happens much faster than their removal. If one therefore neglects τ_{rise}, the synaptic interaction kernel becomes a simple exponential function:

exponential kernel

$$\lim_{\tau_{\text{rise}} << \tau_{\text{fall}} = \tau^{\text{syn}}} \epsilon(t) \propto \Theta(t) \exp\left(-\frac{t}{\tau^{\text{syn}}}\right) \quad . \tag{2.50}$$

The three kernels discussed above are depicted in Figure 2.21. Equation 2.50 represents the synaptic interaction model used from here on throughout this work.

Now that we have established the functional shape of the synaptic interaction, we need to discuss its nature. The neuron model equations discussed in earlier sections (Equations 2.28 and 2.40) may suggest that synaptic transmission is equivalent to *current* injection into the membrane. On the other hand, we have explicitly discussed in Section 2.1.3 how (chemical) synapses cause an increase in the membrane *conductance* for specific ion types. While the latter is certainly true, arguments can be made for modeling synaptic interactions as currents. Indeed, both current and conductance-based synaptic models are widely used in theoretical and computational neuroscience.[28] Below, we explain the

[28] We point out again that, depending on the context, the abbreviation PSC may refer to either a postsynaptic current or a postsynaptic conductance.

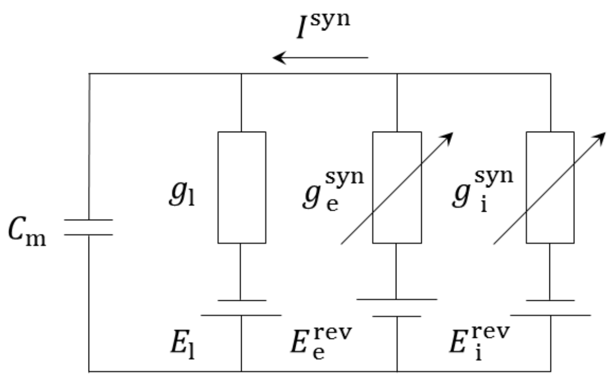

I^{syn}
g_{l}
$g_{\mathrm{e}}^{\mathrm{syn}}$
$g_{\mathrm{i}}^{\mathrm{syn}}$
C_{m}
E_{l}
$E_{\mathrm{e}}^{\mathrm{rev}}$
$E_{\mathrm{i}}^{\mathrm{rev}}$

Figure 2.22.: Circuit diagram of an LIF neuron with COBA synapses. Incoming spikes trigger changes in the synaptic conductances towards their respective reversal potential, generating an input current that depends on the momentary value of the membrane potential.

reasoning behind these models and briefly outline their differences. A much more detailed discussion is provided in Section 4.2.

2.2.2.1. Current-Based and Conductance-Based Models

Conductance-Based Synaptic Interaction

As outlined in Section 2.1.3, an incoming spike causes a synapse to locally change the conductance of the neural membrane towards the reversal potential of the ion type its ligand-gated ion channels are permeable for. Consequently, in this scenario, f^{syn} represents a conductance and shall therefore be renamed g^{syn}. Figure 2.22 shows a schematic of the corresponding circuit.

Here, we need to explicitly differentiate between excitatory ($g_{\mathrm{e}}^{\mathrm{syn}}$) and inhibitory ($g_{\mathrm{i}}^{\mathrm{syn}}$) conductances, since they "connect" the membrane to different reversal potentials $E_{\mathrm{e}}^{\mathrm{rev}}$ and $E_{\mathrm{i}}^{\mathrm{rev}}$, respectively. Since membrane dynamics are primarily determined by Na$^+$ and K$^+$ flows, the reversal potentials are usually chosen as

$$E_{\mathrm{e}}^{\mathrm{rev}} = E_{\mathrm{Na}^+} \quad \text{and} \tag{2.51}$$

$$E_{\mathrm{i}}^{\mathrm{rev}} = E_{\mathrm{K}^+} \quad . \tag{2.52}$$

We can now apply Ohm's law to the synaptic conductances in Figure 2.22 to obtain the total synaptic current

$$I^{\mathrm{syn}} = g_{\mathrm{e}}^{\mathrm{syn}}(E_{\mathrm{e}}^{\mathrm{rev}} - u) + g_{\mathrm{i}}^{\mathrm{syn}}(E_{\mathrm{i}}^{\mathrm{rev}} - u) \quad . \tag{2.53}$$

COBA model

This equation underpins the conductance-based (COBA) synaptic model. Note how the synaptic current explicitly depends on the membrane potential. By plugging the synaptic current into the LIF equation (Equation 2.28), we can now obtain the COBA LIF equation:

COBA LIF equation

$$C_{\mathrm{m}}\frac{du}{dt} = g_{\mathrm{l}}(E_{\mathrm{l}} - u) + g_{\mathrm{e}}^{\mathrm{syn}}(E_{\mathrm{e}}^{\mathrm{rev}} - u) + g_{\mathrm{i}}^{\mathrm{syn}}(E_{\mathrm{i}}^{\mathrm{rev}} - u) + I^{\mathrm{ext}} \quad . \tag{2.54}$$

Despite saving a more thorough discussion for later (Section 4.2), we can already point out several important characteristics of the COBA LIF equation. Firstly, the relationship

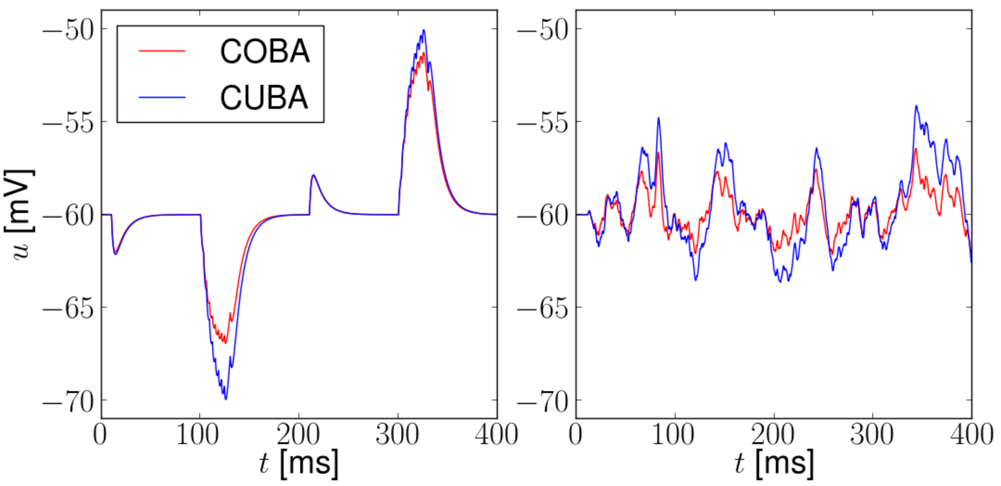


Figure 2.23.: COBA vs. CUBA synapses. **Left:** PSP saturation as the membrane approaches the inhibitory reversal potential. The effect is visible for excitatory PSPs as well, but is much weaker due to the larger distance towards the excitatory reversal potential. **Right:** effect of an increased total conductance. Both the COBA and the CUBA neuron are stimulated with identical excitatory and inhibitory Poisson spike trains. Despite the dynamic range of the membrane potential being smaller than in the left plot, the membrane of the CUBA neuron fluctuates significantly stronger than the one of its COBA counterpart. This happens because due to the increased total conductance of the COBA neuron, which leads to a faster membrane and thereby smaller PSPs.

between the membrane potential and its derivative is no longer determined only by a constant coupling g_l (as it was in the simple LIF equation), but also by explicit functions of time $g_\mathrm{e}^\mathrm{syn}$ and $g_\mathrm{i}^\mathrm{syn}$. This makes the task of finding a closed-form solution for the temporal evolution of the membrane potential much more difficult (see, in particular, Section 4.2.4). Secondly, due to the dependence on the distance towards the reversal potentials, summation of PSPs is no longer linear. This is particularly visible for inhibitory PSPs, where saturation effects can easily appear due to the close proximity of the inhibitory reversal potential to the dynamic range of the membrane potential. This effect is also present, albeit less visible, for excitatory PSPs, in particular since the spike threshold prohibits large depolarizations of the membrane. Thirdly, judging just by the formal equivalence of the three conductance-regulated additive terms in the COBA LIF equation, the total membrane conductance $g^\mathrm{tot} = g_\mathrm{l} + g_\mathrm{e}^\mathrm{syn} + g_\mathrm{i}^\mathrm{syn}$ becomes itself a function of time – and, in particular, larger than g_l alone (since conductances can only be positive by definition). Therefore, the "reaction speed" of the membrane given by its time constant $\tau_\mathrm{eff} = \frac{C_\mathrm{m}}{g^\mathrm{tot}}$ increases and becomes itself time-dependent. This, in turn, causes the effect of an incoming spike on the membrane potential to depend on all other spikes received from all presynaptic partners. Figure 2.23 shows an example of these effects.

By assuming a particular synaptic interaction kernel, we can now provide a closed-form expression for the COBA synapse dynamics. With the exponential kernel from Equation 2.50, the excitatory and inhibitory synaptic conductances can be written as

$$g_x^{\text{syn}}(t) = \sum_{\text{syn } k} \sum_{\text{spk } s} w_k \Theta(t - t_s) \exp(-\frac{t - t_s}{\tau^{\text{syn}}}), \quad x \in \{\text{e}, \text{i}\} \quad . \tag{2.55}$$

The total synaptic current I^{syn} then becomes

$$I^{\text{syn}}(t, u) = \sum_{x \in \{\text{e,i}\}} \sum_{\text{syn } k} \sum_{\text{spk } s} w_k \Theta(t - t_s)(E_x^{\text{rev}} - u) \exp(-\frac{t - t_s}{\tau^{\text{syn}}}) \quad . \tag{2.56}$$

Current-Based Synaptic Interaction

While the conductance-based nature of chemical synapses is an empirical fact, it does not necessarily imply that synaptic interaction models must be conductance-based themselves. The reason for this non sequitur lies within the spatial structure of neurons (Section 2.1.4). An incoming spike may cause a *local* change in the membrane conductance, but the elicited PSP propagates *passively* towards the soma. The soma therefore only experiences an incoming current and is not affected by distal conductance dynamics. If one wishes to use a point neuron model, it is somewhat natural to consider the "point" to represent the soma, since it is there where the afferent inputs are summed up to generate action potentials. It can therefore be argued that a current-based (CUBA) synaptic interaction model is more natural when combined with a point neuron model.

CUBA model

In this scenario, the membrane potential equation remains identical to Equation 2.28:

CUBA LIF equation

$$C_\text{m} \frac{du}{dt} = g_\text{l}(E_\text{l} - u) + I^{\text{syn}} + I^{\text{ext}} \quad . \tag{2.57}$$

With the exponential kernel from Equation 2.50, the total synaptic current can be written as

$$I^{\text{syn}}(t) = \sum_{\text{syn } k} \sum_{\text{spk } s} w_k \Theta(t - t_s) \exp(-\frac{t - t_s}{\tau^{\text{syn}}}) \tag{2.58}$$

These equations express the fact that, in contrast to the COBA scenario, CUBA PSPs are summed up linearly and do not otherwise interact with each other. As we shall see in Section 4.2, this greatly simplifies the analytical treatment of membrane potential dynamics.

neuron/synapse model nomenclature

We end this section with a brief explanation of commonly used nomenclature for the synapse dynamics discussed above. Network models in computational neuroscience rarely use different types of neuron or synapse models simultaneously. Furthermore, any single neuron usually has the same dynamics for all of its afferent synapses. As a consequence, synaptic attributes are often allocated to the used neuron model. Therefore, it is common to speak of, e.g., "COBA EXP LIF neurons"[29], despite the fact that COBA EXP characterizes the synapse model and LIF the neuron membrane dynamics.

[29] In PyNN, for example, neuron models implicitly characterize their synapse dynamics. This is accounted for by the typical naming of neuron models, such as, e.g., `IF_cond_alpha` or `aEIF_curr_exp`.

2.2.2.2. Synaptic Plasticity

In living tissue, the coupling strength between neurons is not a fixed quantity, but may change over time. These variations can be tracked back to morphological changes in their synaptic connections. which can be broadly classified based on the time scales on which they occur. In this section, we will only give a brief overview of theoretical models of synaptic plasticity. In particular, short-term plasticity will play an important role in the dynamics of the spiking network models discussed in Sections 5.3 and 6.5. For a more detailed discussion of synaptic plasticity, we recommend "Part three" (Chapters 10-12) of the textbook by Gerstner and Kistler (2002) and "Part III" (Chapters 8-10) of the textbook by Dayan and Abbott (2001).

Structural Plasticity

On very long time scales in the order of days to years, neurons in the brain rewire as a consequence of, e.g., cognitive (learning, memory formation), genetic (aging) or environmental (injury, disease) factors. The formation of new connections and pruning of old ones is called structural plasticity and is mediated by highly complex biochemistry: it involves not only electrical interactions, but also multiple molecular signalling pathways (neurotransmitters, genetic factors etc.). While providing a fertile ground for modern experimental techniques (Caroni et al., 2012), the complex nature of structural plasticity has so far forestalled the formulation of a unified theory, making it a rare sight in computational and theoretical neuroscience models.

structural plasticity

Long-Term Plasticity: Rate-Based Models

On intermediate time scales in the order of minutes to hours, existing synapses may also change their weight. Depending on whether synapses are strengthened or weakened, one speaks of long-term potentiation (LTP) and long-term depression (LTD), respectively.[30] A well-known rule of thumb for LTP was coined by Donald O. Hebb (Hebb, 2002): "When an axon of cell A is near enough to excite 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." - which is often paraphrased as "What fires together wires together." This is commonly referred to as Hebb's law and has found its way in many models of synaptic plasticity.

LTP/LTD

Hebb's law

Most mathematical formulations of Hebbian learning rules are rate-based and can generally be written as

$$\frac{dw_{ij}}{dt} = F(w_{ij}, \nu_i, \nu_j) \quad , \tag{2.59}$$

Hebbian learning rule

where w_{ij} denotes the synaptic weight between the presynaptic neuron j and the postsynaptic neuron i and ν_j and ν_i their respective firing rates. In order to account for Hebb's law, the function F usually features a positive dependence on the product of the neurons' firing rates. In the rare cases where the dependence is chosen to be negative, the plasticity rule is called anti-Hebbian.

anti-Hebbian learning rule

[30] Sometimes long-term and short-term plasticity are abbreviated as LTP and STP, respectively. This can be easily confused with long-term and short-term potentiation and must be inferred from the context, if necessary. Here, we use "P" as an abbreviation for potentiation and do not abbreviate short-term- and long-term plasticity.

The simplest possible rate-based Hebbian learning rule is given by

$$\frac{dw_{ij}}{dt} = c\nu_i\nu_j \quad . \tag{2.60}$$

For positive c - which conforms to Hebb's law - synapses are strengthened when both the pre- and postsynaptic neuron fire. This is also an obvious drawback of this simple model: synaptic weights increase indefinitely for nonzero firing rates. However, this can be easily fixed by introducing an upper bound to the learning rate c:

$$c \to c(w_{ij}) = \gamma(w^{\max} - w_{ij})^{\beta} \quad . \tag{2.61}$$

Still, this model has the limitation that it can not simultaneously account for LTP and LTD (and neither does Hebb's original rule). This can, however, be included, by allowing more complex functions F, along with more complex behavior with various functional consequences. Below, we list three such learning rules that have enjoyed relative popularity in theoretical studies.

The so-called covariance rule proposed in Sejnowski (1977) strengthens the synapse if neural activity is positively correlated and weakens it otherwise:

covariance rule

$$\frac{dw_{ij}}{dt} = \gamma(\nu_i - \langle\nu_i\rangle)(\nu_j - \langle\nu_j\rangle) \quad . \tag{2.62}$$

By adding a quadratic term to the plasticity equation, Oja's rule

Oja's rule

$$\frac{dw_{ij}}{dt} = \gamma(\nu_i\nu_j - w_{ij}\nu_i^2) \tag{2.63}$$

enables a homeostatic mechanism of sorts: under certain conditions, it can be shown that the afferent weights of a neuron converge asymptotically to a configuration where $\sum_j w_{ij}^2 = 1$ (Oja, 1982). Finally, the plasticity rule proposed by Bienenstock, Cooper and Munro (BCM rule, see Figure 2.24)

BCM rule

$$\frac{dw_{ij}}{dt} = c_1\nu_i^2\nu_j - c_2\nu_i\nu_j \tag{2.64}$$

allows neurons to become selective to particular input patterns and has been successfully used to model the development of receptive fields (Bienenstock et al., 1982).

Long-Term Plasticity: Spike-Based Models

The above considerations are based on empirical observation and sucessfully reproduce some experimentally verified phenomena, but remain at a rather abstract level. In particular, they make no statement about how firing rates are encoded at the site of a particular synapse and offer no mechanistic model of how an individual synapse performs the required computation of $F(w_{ij}, \nu_i, \nu_j)$. For a better understanding of the microscopic phenomena that enable long-term plasticity, having a spike-based plasticity rule is more convenient.

STDP

The arguably most popular spike-based model is STDP, which is short for spike-timing-dependent plasticity. It is based on the observation that the timing of pre- and post-synaptic spikes is critical to the evolution of synaptic weights (Bi and Poo, 1998; Markram et al., 1997). While by now a lot of experimental evidence for STDP exists, Figure 2.25

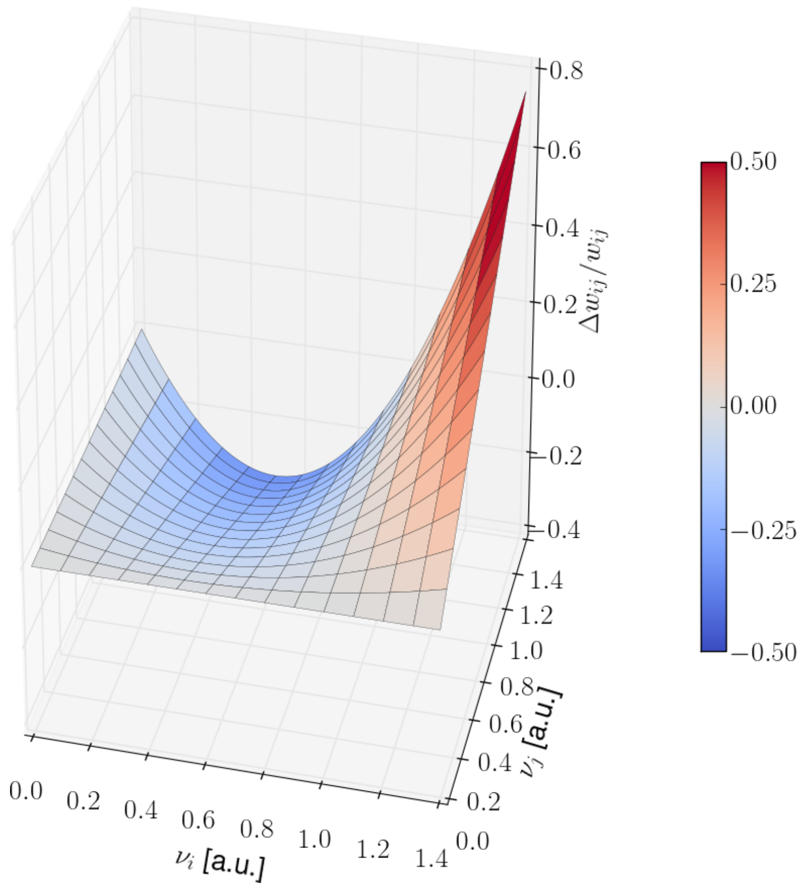


Figure 2.24.: BCM plasticity rule. The parameters c_1 and c_2 in Equation 2.64 were both set to 1. Depending on the postsynaptic firing rate ν_i, the synapse is either weakened (LTD, blue hue) or strengthened (LTP, red hue). The fixed point defined by $\frac{\Delta w_{ij}}{w_{ij}} \overset{!}{=} 0$ is unstable. The presynaptic firing rate ν_j serves as a linear modulator of the change in synaptic strength.

probably remains the most recognizable result of an STDP measurement protocol. Given these measurements, it is quite straightforward to formulate a phenomenological STDP model.

For any pair of pre- and postsynaptic spikes, we can define the synaptic weight change to be some function W of the difference in spike timing, as well as of the current synaptic weight itself. For any pre- and postsynaptic spike trains ρ_j and ρ_i (as defined in Equation 2.30), the total synaptic weight change can be written as a sum over all weight changes induced by all pre- and postsynaptic spike pairings:

$$\Delta w_{ij} = \sum_{\text{postsynaptic spikes } k} \; \sum_{\text{presynaptic spikes } l} W(w_{ij}, t_i^k - t_j^l) \quad . \tag{2.65}$$

causal and acausal branches

The function W is generally split, depending on the relative timing of the pre- and postsynaptic neuron $\Delta t := t_i^k - t_j^l$, into a causal and an acausal branch:

$$W(w_{ij}, \Delta t) = \begin{cases} A_+(w_{ij}) \exp\left(-\frac{\Delta t}{\tau_+}\right) & \text{for} \quad \Delta t > 0 \quad \text{(causal branch)} \\ -A_-(w_{ij}) \exp\left(\frac{\Delta t}{\tau_-}\right) & \text{for} \quad \Delta t < 0 \quad \text{(acausal branch)} \quad . \end{cases} \tag{2.66}$$

The causal and acausal branches are also often called Hebbian and anti-Hebbian, respectively. In most models, the function W factorizes into a pure weight-dependent term A and a pure spike-timing-dependent term. For simplicity and analytical tractability, the latter is often chosen as a decaying exponential function of the pre- and postsynaptic spike interval, which is also in reasonable agreement with the experimental data shown in Figure 2.25.

In this formulation, the weight-dependent term can be chosen individually for each branch ("+" encodes the causal and "−" the acausal branch):

$$A_+(w_{ij}) = f(w^{\max} - w_{ij})\eta_+ \quad \text{and} \tag{2.67}$$

$$A_-(w_{ij}) = f(w_{ij} - w_{\min})\eta_- \quad . \tag{2.68}$$

The parameters η_+ and η_- represent learning rates. For reasons of physical plausibility, the synaptic weights are bounded from above and below by $w^{\max}$ and $w_{\min}$, respectively. The function f controls the shape of the weight dependence. Popular choices include the soft-bounded multiplicative update rule

multiplicative rule

$$f(x) = x \tag{2.69}$$

additive rule

and the additive rule with hard bounds

$$f(x) = \Theta(x) \quad , \tag{2.70}$$

where Θ denotes the Heaviside step function.

In order to avoid a switch from excitation to inhibition, it is usually assumed that $w_{\min} \geq 0$. Apart from being theoretically problematic, such a switch would cause a violation of Dale's law (see Section 2.1.3); in particular, it would be biologically implausible, since excitation and inhibition are usually mediated by different neurotransmitters. Indeed, the existence of "inhibitory STDP" has not yet been properly studied.

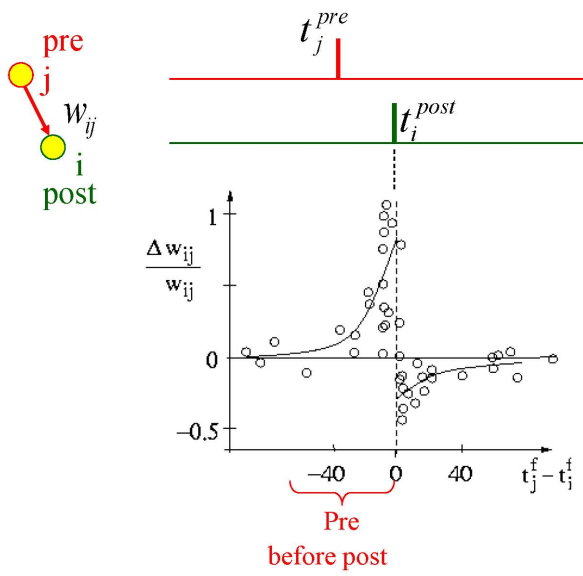


Figure 2.25.: In-vitro measurement of STDP for an excitatory synapse. The black circles denote individual measurements of the EPSP amplitude, which reflects the synaptic weight. When the presynaptic neuron spikes before the postsynaptic one (causal relation), the synapse is strengthened. When the timing of pre- and postsynaptic spikes is reversed (acausal relation), the synapse is weakened. The relative weight change is largest when the pre- and postsynaptic spikes appear in quick succession, regardless of their order. The overlayed solid line represents a schematic timing-dependent learning rule $W(t_i, t_j)$. Figure taken from Sjöström and Gerstner (2010), which is itself modified from Bi and Poo (1998).

As already mentioned, the STDP model described here, while very popular with computational and theoretical neuroscientists, remains purely phenomenological. However, it is significantly closer to an electrophysiological explanation of long-term plasticity than the rate-based models described in the previous section. In this model, the synapse only needs to "know" the timing of pre- and postsynaptic spikes. While the former is trivial, the latter has been shown to be possible by action potentials that propagate back from the soma throughout the dendritic tree (Markram and Sakmann, 1995). While the biological basis of STDP is not yet completely understood, the standard STDP model described above has evolved into more complex STDP models based on experimentally established synaptic molecular dynamics (see, e.g., Shouval et al., 2010). Moreover, these models also typically strive to explain the vast array of different STDP shapes observed across different species and brain regions (Abbott and Nelson, 2000).

back-propagating action potentials

Short-Term Plasticity: The Tsodyks-Markram Mechanism

Changes in synaptic efficacies also occur on shorter timescales, on the order of milliseconds to seconds. Such changes are usually due to physiological reasons and do not necessarily encode learning processes, since they are only transient. However, since they are practically ubiquitous in neural tissue (see, e.g., Thomson and Deuchars, 1994), they do play an important role in the way the brain encodes and processes information.

Many experiments have demonstrated that synaptic efficacy is influenced by the firing rate of the presynaptic neuron. Both weakening and strengthening of synapses have been observed, sometimes even simultaneously (see, e.g. Markram et al., 1998a, for an extensive collection of experimental papers). Analogously to their long-term counterparts, these short-term synaptic weight changes are abbreviated as STD (short-term depression) and STP (short-term potentiation). Here, we discuss a phenomenological model of short-term synaptic plasticity: the Tsodyks-Markram-Model (short: TSO, see Tsodyks and Markram, 1997b). For a comprehensive list of biophysical models, we again refer to the review by Markram et al. (1998a).

short-term depression and potentiation

TSO model

Initially, the TSO model was designed for representing STD only and was extended later to encompass STP as well. We shall therefore also start with a discussion of STD in the TSO model. The TSO model assumes that the total amount of synaptic resources, which could, for example, model the total number of vesicles within a synapse, is naturally limited and subdivided into three partitions: recovered, effective and inactive, which assume the fractions R, E and I of the total resource amount, respectively. The first model equation must therefore read

recovered, effective and inactive partitions

$$R + E + I = 1 \quad . \tag{2.71}$$

When the synapse is not activated by presynaptic spikes, all resources are collected into the recovered partition, such that the resting state is $(R, E, I) = (1, 0, 0)$. Upon arrival of a presynaptic spike, a portion U (utilization of synaptic efficacy) of the recovered partition is instantaneously transferred to the effective partition, which can be interpreted as neurotransmitter release into the synaptic cleft. The net synaptic effect (postsynaptic current/conductance) is therefore considered proportional to the amount of resources in the effective partition. The effective partition inactivates exponentially with a time constant τ_{inact}, which models the removal of neurotransmitters from the postsynaptic site and is therefore equivalent to the synaptic time constant τ^{syn} from Equation 2.50. The resources removed from the effective partition are transferred to the inactive partition, which then decays exponentially back into the recovered partition with a recovery time constant τ_{rec}, thereby modeling neurotransmitter reuptake by the presynaptic terminal. With this, the TSO model of STD is fully defined and we can cast its dynamics into equations:

utilization of synaptic efficacy

inactivation time constant

recovery time constant

$$\frac{dR}{dt} = \frac{I}{\tau_{\text{rec}}} - \sum_{\text{spikes } s} U R\, \delta(t - t_s) \tag{2.72}$$

$$\frac{dE}{dt} = -\frac{E}{\tau_{\text{inact}}} + \sum_{\text{spikes } s} U R\, \delta(t - t_s) \tag{2.73}$$

Note that the dynamics of I must not be explicitly given, since they follow directly from Equations 2.71 – 2.73.

In order to model STP, the original TSO model from Tsodyks and Markram (1997b) was extended in Markram et al. (1998b) by making U itself a dynamic variable. The first incoming spike of a train triggers a resource transfer of amplitude U_0, but with each incoming spike, U is increased by a certain amount ΔU. By setting $\Delta U = U_0(1 - U)$, one can ensure that the synapse may never use more resources than it has available. In between spikes, U decays back towards its resting state U_0 with a facilitation time constant τ_{facil}. The final equation of the complete TSO model thereby reads:

facilitation time constant

$$\frac{dU}{dt} = \frac{U_0 - U}{\tau_{\mathrm{facil}}} + \sum_{\text{spikes } s} U_0(1-U)\,\delta(t-t_s) \tag{2.74}$$

Typically, synaptic time constants are much shorter than those of short-term plasticity (both depression and facilitation). By integrating Equations 2.72 – 2.74 under this assumption, we can calculate the change in the TSO variables as a function of the interspike interval Δt:

$$R_{n+1} = R_n(1-U_{n+1})\exp\left(\frac{-\Delta t}{\tau_{\mathrm{rec}}}\right) + 1 - \exp\left(\frac{-\Delta t}{\tau_{\mathrm{rec}}}\right) \tag{2.75}$$

$$E_{n+1} = E_n \exp\left(\frac{-\Delta t}{\tau_{\mathrm{inact}}}\right) + R_n U_{n+1}\exp\left(\frac{-\Delta t}{\tau_{\mathrm{inact}}}\right) \tag{2.76}$$

$$U_{n+1} = U_n \exp\left(\frac{-\Delta t}{\tau_{\mathrm{facil}}}\right) + U_0\left[1 - U_n\exp\left(\frac{-\Delta t}{\tau_{\mathrm{facil}}}\right)\right] \quad . \tag{2.77}$$

For a constant presynaptic firing frequency ν, we can now easily derive steady-state expressions for the TSO variables by setting $X_{n+1} \overset{!}{=} X_n =: \tilde{X}$ (with $X \in R, E, U$):

steady-state expressions

$$\tilde{U} = \frac{U_0}{1-(1-U_0)\exp\left(-\frac{1}{\nu\tau_{\mathrm{facil}}}\right)} \tag{2.78}$$

$$\tilde{R} = \frac{1-\exp\left(-\frac{1}{\nu\tau_{\mathrm{rec}}}\right)}{1-(1-\tilde{U})\exp\left(-\frac{1}{\nu\tau_{\mathrm{rec}}}\right)} \tag{2.79}$$

$$\tilde{E} = \frac{\tilde{R}\tilde{U}\exp\left(-\frac{1}{\nu\tau_{\mathrm{inact}}}\right)}{1-\exp\left(-\frac{1}{\nu\tau_{\mathrm{inact}}}\right)} \quad . \tag{2.80}$$

Figure 2.26 shows several examples of STD, STP and a combination of both using the TSO mechanism.

The TSO mechanism of short-term plasticity plays an important role in later sections of this manuscript. In the cortical attractor memory model discussed in Section 5.3, it takes part in controlling the duration of particular activity patterns. In the LIF-based sampling networks from Section 6.5, it is used to constrain interneuron coupling strengths by simulating renewing synapses.

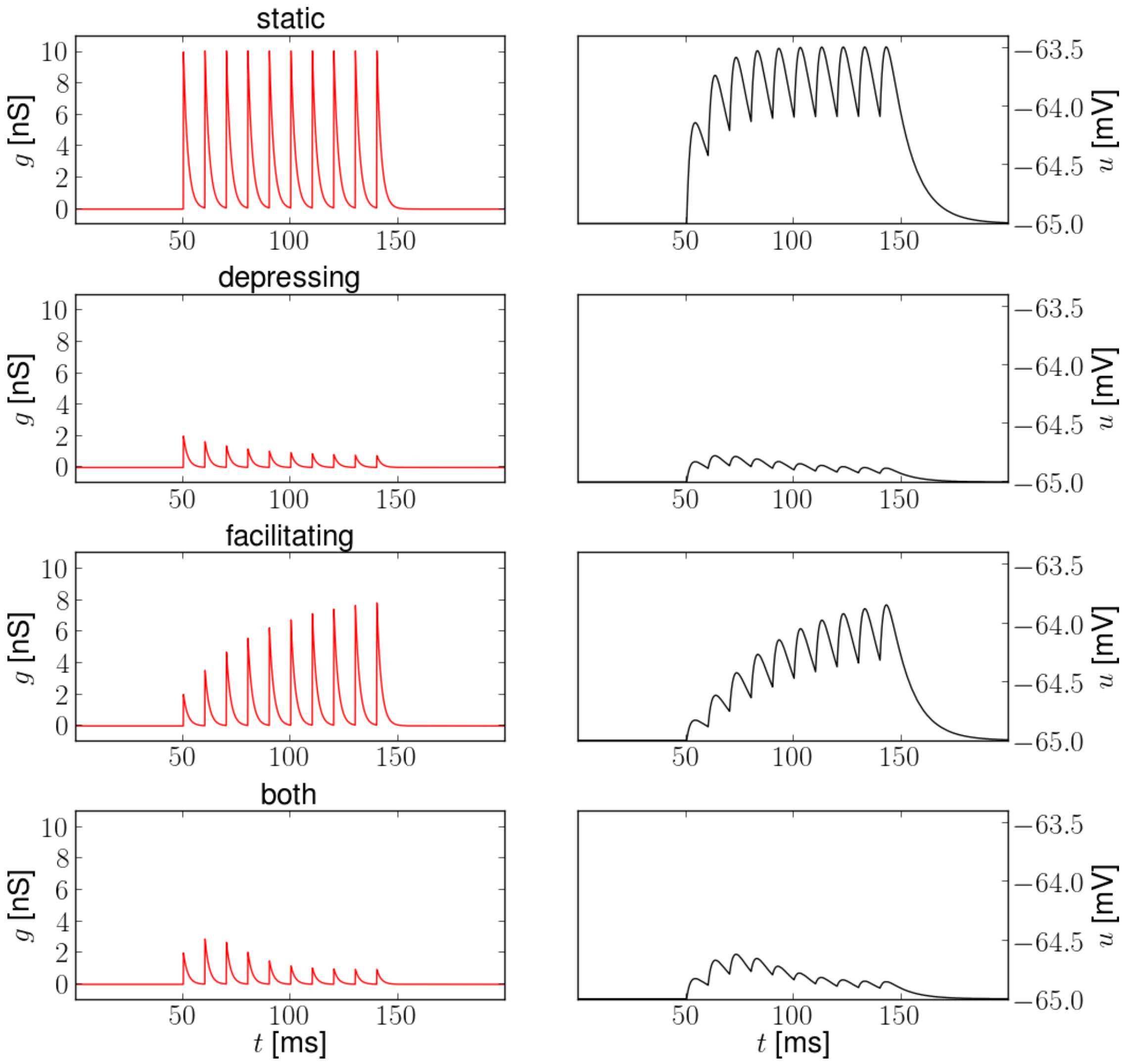


Figure 2.26.: Simulation of synaptic plasticity using the TSO mechanism. The cell is stimulated by a 100 Hz regular spike train for 100 ms. Synaptic efficacy is plotted on the left hand side and the resulting membrane potential trace on the right hand side. U_0 was set to 0.2, so the first PSC/PSP in the topmost example (no TSO) is five times as high as the first PSCs/PSPs of the other three examples. The synaptic time constant (equivalent to τ_{inact}) was set to 2 ms. For the purely depressing mode, we have set $\tau_{\mathrm{rec}} = 100$ ms and $\tau_{\mathrm{facil}} = 0$ ms. For the purely facilitating mode, we have set $\tau_{\mathrm{rec}} = 0$ ms and $\tau_{\mathrm{facil}} = 200$ ms. The third mode is a combination of the previous two, with $\tau_{\mathrm{rec}} = 100$ ms and $\tau_{\mathrm{facil}} = 200$ ms. Note how in this regime, where STD and STP are happening simultaneously, the synaptic efficacy first rises before dropping off.

3. Artificial Brains: Simulation and Emulation of Neural Networks

TECHNICALLY TRUE: EVERY OBJECT IS AN ANALOG COMPUTER OF ITSELF
THIS MACHINE IS RUNNING A PERFECT SIMULATION OF CHEESE.

Comic by Zach Weiner, SMBC 3054

When describing increasingly complex[1] systems, the required array of equations equivalently grows in size and complexity. In many (usually simple) cases, statistical methods can be applied to distill macroscopic equations from those governing the microscopic components of a system, with thermodynamics offering a paradigmatic example. More often though – and this is usually the case for neural networks – complexity rises beyond mathematical tractability. In such cases, it is nowadays possible to fall back onto *simulating* these systems.

complexity vs. mathematical tractability

Owing to recent advances in general-purpose computing architectures, very large systems of coupled equations can be numerically evaluated in reasonable time on parallel multiprocessor machines. Especially in neuroscience, the advent of modern-day computers and algorithms has had an enormous impact, with computational neuroscience now dominating the theoretical research landscape. Simulating networks of tens of thousands of spiking neurons has, by now, become routine (see, e.g., Brette et al., 2007) and plans are even underway to simulate the entire human brain with very large scale parallel machines (Markram, 2012).

simulation

It is clear that any hardware back-end is ultimately only as powerful and versatile as the software controlling it allows it to be. Therefore, neural network simulators represent the backbone of computational neuroscience – with the present work making no exception. Section 3.1 gives a brief overview of the simulation software used for the various neural network models presented later on, with a particular focus on the abstraction offered by

software

[1]Here, complexity can be understood both as number of constituent components, as well as concerning the nature of the equations describing their dynamics and interactions.

the simulator-independent language PyNN and the integration with the Python programming language provided by this common interface.

versatility vs. scalability

It might now appear that the combination of powerful software and fast general-purpose hardware is the best possible instrument for modeling neural networks. While this approach is, indeed, hard to surpass in terms of versatility, it is rather questionable whether it is optimal in terms of power efficiency and scalability. Obviously, these potential drawbacks result from the hardware architecture. Von-Neumann-style machines require a huge structural overhead in order to enable their use as general-purpose computers.

speed and power efficiency

A machine specifically designed for neural network simulations has no need for such structures and could therefore outperform a conventional machines by orders of magnitude in terms of speed and power efficiency, while using essentially the same VLSI technology. Tailoring hardware to the specific needs of neural network modeling also has the potential to overcome scalability problems, which appear on conventional architectures due to communication bandwidth bottlenecks between the processing cores.

physical implementation emulation

One particular class of such machines are aptly named “neuromorphic” devices. They break away from the classical description-abstraction-simulation paradigm by realizing a physical implementation of the system to be studied (Mead, 1989, 1990; Mead and Mahowald, 1988). In this context, it is more intuitive to say that they emulate the system rather than simulating it. This particular approach is not without limitations of its own – but the motivation behind neuromorphics is that whatever such systems might lose in versatility through their choice of a physical model, they more than make up for in efficiency, speed and scalability (Furber et al., 2012; Indiveri et al., 2006; Rocke et al., 2008; Schemmel et al., 2010; Vogelstein et al., 2007).

The vision of a universal neuromorphic emulator lies at the very heart of the present work. While the models and methods developed here strive for generality, they have been tested and studied on several particular neuromorphic systems. In Sections 3.2 and 3.3, we describe these systems in detail, as these details will be essential in understanding the particular strategies chosen for the implementation of the models that we shall later elaborate on.

software stack workflow

Just as with conventional hardware, a complex stack of software modules is required for operating neuromorphic systems. The interplay between hardware, software and the users themselves can be systematized into a particular workflow, which is to a large extent characteristic for the employed hardware platform (Brüderle et al., 2011). The most relevant aspects of the software and workflows built around the neuromorphic devices used for this work are addressed subsequently to their respective hardware description.

In the hardware-related sections 3.2 and 3.3, we only offer a high-level modeler's view of the neuromorphic systems. For a more detailed description of the constituent circuits, we point to the relevant publications in the respective sections. The material (text and figures) of Sections 3.2 and 3.3 is taken entirely from publications that were co-authored by the author of this thesis, in particular Pfeil et al. (2013) and Petrovici et al. (2014).

3.1. Simulation of Neural Networks

Computer simulations are an irreplaceable tool for computational neuroscience. While significant efforts are being made towards simulating large-scale, highly detailed cortical models (see, e.g., Markram, 2006), most simulation software is designed to handle more abstract neuron models (Brette et al., 2007), such as the ones we are using in our present work. Depending on the particular requirements of an individual network model, such as the required neuron model or the network size, or on the specific investigated question, which might require large amounts of simulation runs, one may choose to favor one particular simulation engine that is exceptionally adept for the task at hand. Unfortunately, switching from one simulation software to another is rarely straightforward and usually involves a complete rewrite of the entire simulation code.

PyNN This represents the motivation behind PyNN, a simulator-independent API for the high-level definition of point neuron networks (Davison et al., 2008). PyNN abstracts away the details of the software back-end and unifies the interface for instantiating neurons and synapses, controlling their parameters and recording relevant dynamic variables. Individual back-end-specific modules that remain hidden from the user then take care of the translation of the networks defined in PyNN to the chosen simulation engine. A particularly useful feature of PyNN in the context of our work is that it also supports both the Spikey chip (Section 3.2) and the waferscale system (Section 3.3) as emulation back-ends.

All the network models that we discuss later on are defined via PyNN. For our software simulations, we use either NEST (Diesmann and Gewaltig, 2002; Gewaltig and Diesmann, 2007; Website, 2009) or NEURON (Hines and Carnevale, 2003; Hines et al., 2008; Hines and Carnevale, 2006) as back-ends. For the hardware emulations, we use the Spikey chip for the small network models and the ESS (Section 3.3.4) of the waferscale system for the networks with a larger number of neurons.

NEST

NEURON

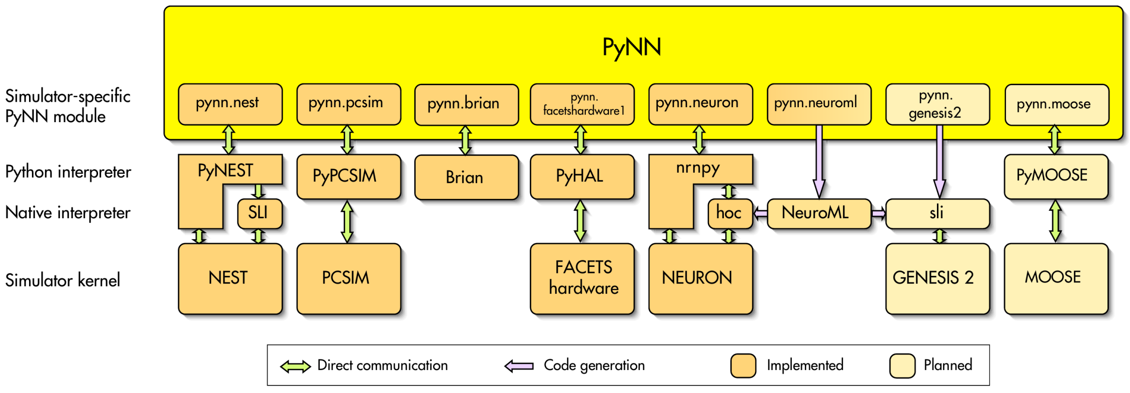


Figure 3.1.: Schematic of the PyNN architecture and its interaction with several simulation/emulation back-ends. Figure taken from Davison et al. (2008).

3.2. The Spikey Chip

Spikey VLSI physical model

The central component of the single-chip neuromorphic setup we will use in Chapter 5 is the "Spikey" neuromorphic microchip. It contains analog very-large-scale integration (VLSI) circuits modeling the electrical behavior of neurons and synapses.

In such a physical model, measurable quantities in the neuromorphic circuitry have corresponding biological equivalents. For example, the membrane potential u of a neuron is modeled by the voltage over a capacitor C_m that, in turn, can be seen as a model of the capacitance of the cell membrane.

speedup

hardware vs. biological domain

In contrast to numerical approaches, dynamics of physical quantities like u evolve continuously in time. We designed our hardware systems to have time constants approximately 10^4 times faster than their biological counterparts allowing for high-throughput computing. This is achieved by reducing the size and hence the time constant of electrical components, which also allows having more neurons and synapses on a single chip with fixed dimensions. To avoid confusion between hardware and biological domains of time, voltages and currents, all parameters are specified in the biological domain.

3.2.1. The Neuromorphic Chip

COBA LIF neurons

On the Spikey chip (Figure 3.2), a VLSI version of the LIF neuron model (Section 2.2.1.1) with COBA synapses (Section 2.2.2.1) is implemented:

$$C_\mathrm{m}\frac{du}{dt} = -g_\mathrm{l}(u - E_\mathrm{l}) - \sum_i g_i^\mathrm{syn}(u - E_i) \quad . \tag{3.1}$$

The time course of the synaptic activation is modeled by

$$g_i(t) = p_i(t) \cdot w_i \cdot g_i^\mathrm{max} \tag{3.2}$$

exponential PSCs

where g_i^max are the maximum conductances and w_i the weights for each synapse, respectively. The time course $p_i(t)$ of synaptic conductances is a linear transformation of the current pulses shown in Figure 3.2B (in green), and hence an exponentially decaying function of time. For a more detailed layout of the relevant circuits, we refer to Schemmel et al. (2006) and Indiveri et al. (2011).

STDP

The implementation of STDP is described in Schemmel et al. (2006) and Pfeil et al. (2012a). Correlation measurements between pre- and post-synaptic action potentials are carried out in each synapse, and the 4-bit weight is updated by an on-chip controller located in the digital part of the Spikey chip. As STDP is not relevant for our studies, we do not discuss it in more detail.

STP

Short-term plasticity (STP) modulates g_i^max (Schemmel et al., 2007) similar to the model by Tsodyks and Markram (1997a). On hardware, STP can be configured individually for each synapse line driver that corresponds to an axonal connection in biological terms. It can either be facilitating or depressing, but, in contrast to the original model, not both at the same time.

spike propagation

The propagation of spikes within the Spikey chip is illustrated in Figure 3.2 and described in detail in Schemmel et al. (2006). Spikes enter the chip as time-stamped events using standard digital signaling techniques that facilitate long-range communication, e.g., to the host computer or other chips. Such digital packets are processed in discrete time

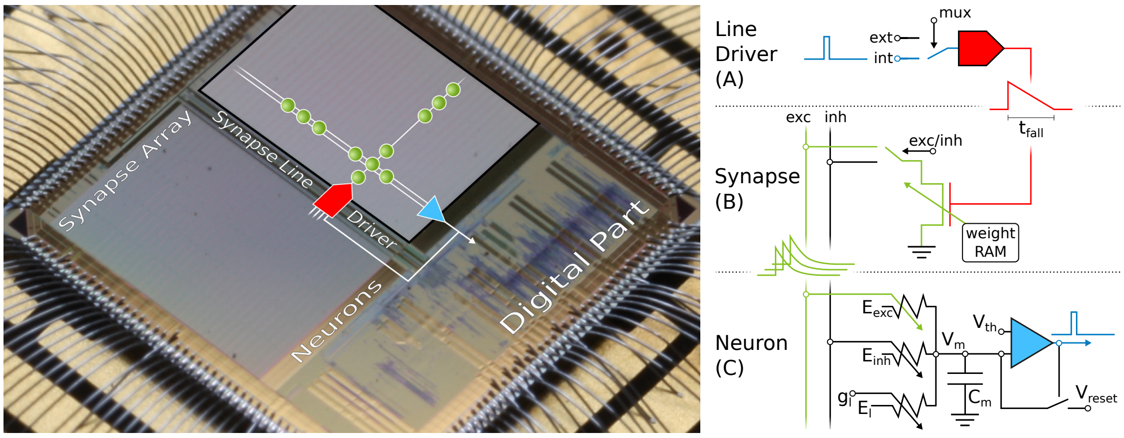


Figure 3.2.: The Spikey neuromorphic chip. **Left:** Microphotograph of the chip (fabricated in a 180 nm CMOS process with a die size of $5 \times 5\,\mathrm{mm}^2$). Each of its 384 neurons can be connected to any other neuron on the chip. In the following, we give a short overview of the technical implementation of neural networks on the Spikey chip. **(A)** Within each synapse array, 256 synapse line drivers convert incoming digital spikes (blue) into a linear voltage ramp (red) with a falling slew rate t_{fall}. For simplicity, the slew rate of the rising edge is not illustrated here, as it is, in general, chosen to be comparatively small. Each of these synapse line drivers are individually driven by either another on-chip neuron (int) or an external spike source (ext). **(B)** Within each synapse, depending on its individually configurable weight w_i, the linear voltage ramp (red) is then translated into a current pulse (green) with exponential decay. These postsynaptic pulses are sent to the neuron via excitatory (exc) and inhibitory (inh) input lines, which are shared by all synapses belonging to the same column. **(C)** Upon reaching the neuron circuit, the total current on both input lines is converted into conductances. If the membrane potential u crosses the firing threshold ϑ, a digital pulse (blue) is generated, which can be recorded and fed back into the synapse array. After each spike, u is set to ϱ for a refractory period τ_{ref}. Neuron and synapse line driver parameters can be configured as summarized in Table 3.1. Figure taken from Pfeil et al. (2013).

in the digital part of the chip, where they are transformed into digital pulses entering the synapse line driver (marked red in Figure 3.2A). These pulses propagate in continuous time between on-chip neurons, and are optionally transformed back into digital spike packets for off-chip communication.

3.2.2. System Environment

The Spikey chip is mounted on a network module (see the schematic in Figure 3.3. Digital spike and configuration data is transferred via direct connections between a field-programmable gate array (FPGA) and the Spikey chip. Onboard digital-to-analog converter (DAC) and analog-to-digital converter (ADC) components supply external parameter voltages to the Spikey chip and digitize selected voltages generated by the chip for calibration purposes. Furthermore, up to eight selected membrane voltages can be recorded in parallel by an oscilloscope. Because communication between a host computer and the FPGA has a limited bandwidth that does not satisfy real-time operation requirements of the Spikey chip, experiment execution is controlled by the FPGA while operating the Spikey chip in continuous time. To this end, all experiment data is stored in the local random access memory (RAM) of the network module. Once the experiment data is transferred to the local RAM, emulations run with an acceleration factor of 10^4 compared to biological real-time, independently of the emulated network size.

FPGA

DAC, ADC

RAM

control software

Execution of an experiment is split into three steps. First, the control software within the memory of the host computer generates configuration data (such as synaptic weights, network connectivity, etc., see Table 3.1), as well as input stimuli to the network. All data is stored as a sequence of commands and is transferred to the memory on the network module. In the second step, a playback sequencer in the FPGA logic interprets this data and sends it to the Spikey chip, after which it triggers the emulation. Data produced by the chip (essentially, spike times) is recorded in parallel. In the third and final step, this recorded data stored in the memory on the network module is retrieved and transmitted to the host computer, where they are processed by the control software.

PyNN API

Having a control software that abstracts away the hardware details greatly increases the accessibility for a diverse community of users. However, modelers are already struggling with mutually incompatible interfaces to various software simulators. That is why the Spikey system supports PyNN, a widely used application programming interface (API) that strives for a coherent user interface, allowing portability of neural network models between different software simulation frameworks (such as NEST or Neuron) and hardware systems (such as the Spikey system or the wafer-scale device in Section 3.3).

3.2.3. Configurability

In order to facilitate the emulation of a large variety of network models (with a declared focus on biologically-inspired structures), it is essential to support the implementation of different neuron and synapse types. This can be achieved by varying the appropriate parameters of the implemented neurons and synapses. We assume, implicitly, that the implemented COBA LIF dynamics are sufficient for a good enough approximation of the neuron/synapse dynamics that are to be modeled.

The Spikey chip provides 2969 different analog parameters (Table 3.1) stored on current memory cells that are continuously refreshed from a digital on-chip memory. Most of

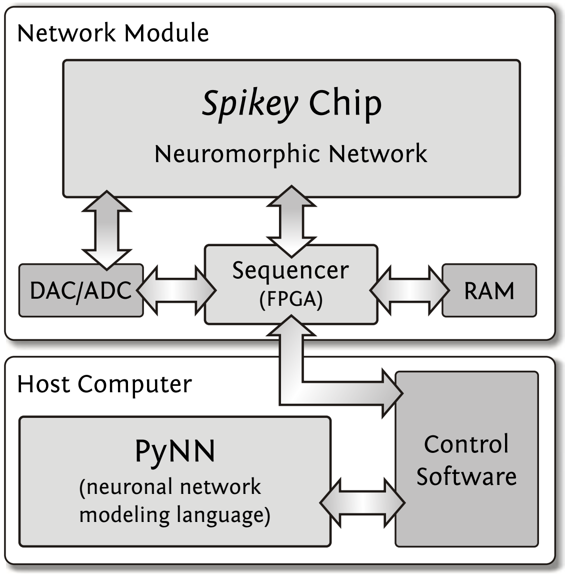


Figure 3.3.: Integrated development environment. User access to the Spikey chip is provided using the PyNN neural network modeling language. The control software controls and interacts with the network module which is operating the Spikey chip. The RAM size (512 MB) limits the total number of spikes for stimulus and spike recordings to approx. $2 \cdot 10^8$ spikes. The required data for a full configuration of the Spikey chip has a size of approximately 100 kB. Figure taken from Pfeil et al. (2013).

Scope	Name	Type	Description
Neuron circuits	n/a	$\mathsf{i_n}$	Two digital configuration bits activating the neuron and readout of its membrane voltage
	g_l	$\mathsf{i_n}$	Bias current for neuron leakage circuit
	τ_ref	$\mathsf{i_n}$	Bias current controlling neuron refractory time
	E_l	$\mathsf{s_n}$	Leakage reversal potential
	$E_\mathrm{i}^\mathrm{rev}$	$\mathsf{s_n}$	Inhibitory reversal potential
	$E_\mathrm{e}^\mathrm{rev}$	$\mathsf{s_n}$	Excitatory reversal potential
	ϑ	$\mathsf{s_n}$	Firing threshold voltage
	ϱ	$\mathsf{s_n}$	Reset potential
Synapse line drivers	n/a	$\mathsf{i_l}$	Two digital configuration bits selecting input of line driver
	n/a	$\mathsf{i_l}$	Two digital configuration bits setting line excitatory or inhibitory
	$t_\mathrm{rise}, t_\mathrm{fall}$	$\mathsf{i_l}$	Two bias currents for rising and falling slew rate of presynaptic voltage ramp
	g_i^max	$\mathsf{i_l}$	Bias current controlling maximum voltage of presynaptic voltage ramp
Synapses	w	$\mathsf{i_s}$	4-bit weight of each individual synapse
short-term-plasticity-related	n/a	$\mathsf{i_l}$	Two digital configuration bits selecting short-term depression or facilitation
	U_SE	$\mathsf{i_l}$	Two digital configuration bits tuning synaptic efficacy for STP
	n/a	$\mathsf{s_l}$	Bias voltage controlling spike driver pulse length
	τ_rec, τ_facil	$\mathsf{s_l}$	Voltage controlling STP time constant
	I	$\mathsf{s_l}$	Short-term facilitation reference voltage
	R	$\mathsf{s_l}$	Short-term capacitor high potential
STDP-related	n/a	$\mathsf{i_l}$	Bias current controlling delay for presynaptic correlation pulse (for calibration purposes)
	$A_\pm$	$\mathsf{s_l}$	Two voltages dimensioning charge accumulation per (anti)causal correlation measurement
	n/a	$\mathsf{s_l}$	Two threshold voltages for detection of relevant (anti)causal correlation
	τ^STDP	g	Voltage controlling STDP time constants

Table 3.1.: List of analog current and voltage parameters as well as digital configuration bits, each with corresponding model parameter names (excluding technical parameters that are only relevant for correctly biasing analog support circuitry or controlling digital chip functionality). Electronic parameters that have no direct translation to model parameters are denoted n/a. The membrane capacitance is fixed and identical for all neuron circuits at $C_\mathrm{m} = 0.2\,\mathrm{nF}$. Parameter types: (i) controllable for each corresponding circuit: 192 for neuron circuits (denoted with subscript n), 256 for synapse line drivers (denoted with subscript l), 49152 for synapses (denoted with subscript s); (s) two values, shared for all even/odd neuron circuits or synapse line drivers, respectively; (g) global, one value for all corresponding circuits on the chip. All numbers refer to circuits associated to one synapse array (the other one can be controlled in the same way). For technical reasons, the revision of the chip used in this work only allowed usage of one the two synapse arrays. Therefore, all experiments on Spikey were limited to a maximum of 192 neurons.

these cells deliver individual parameters for each neuron or synapse line driver. However, due to the size of the current-voltage conversion circuitry, this was not achievable for all parameters. In particular, this concerns the reversal potentials E_l, $E_\mathrm{e}^\mathrm{rev}$ and $E_\mathrm{i}^\mathrm{rev}$, for each neuron.[2] As a consequence, groups of 96 neurons share most of these voltage parameters. Parameters that can not be controlled individually are delivered by global current memory cells.

individual parameters

shared parameters

In addition to the possibility of controlling analog parameters, the Spikey chip also offers an almost arbitrary configurability of the network topology. As illustrated in Figure 3.2, the fully configurable synapse array allows connections from synapse line drivers (located alongside the array) to arbitrary neurons (located below the array) via synapses whose weights can be set individually with a 4-bit resolution. This limits the maximum fan-in to 256 synapses per neuron, which can be composed of up to 192 synapses from on-chip neurons, and up to 256 synapses from external spike sources. Because the total number of neurons exceeds the number of inputs per neuron, an all-to-all connectivity is not possible. However, it is rather sensible to assume that network models only rarely require all-to-all connectivity. For all Spikey experiments that we discuss here, the connection density is completely unproblematic.

4-bit weights

3.2.4. Calibration

Device mismatch that arises from the inevitable variability in the manufacturing process causes fixed-pattern noise, which manifests itself as parameter variability from neuron to neuron as well as from synapse to synapse. Electronic noise (including thermal noise) also affects dynamic variables such as the membrane potential u. Consequently, experiments will exhibit some amount of both neuron-to-neuron and trial-to-trial variability given the same input stimulus.

fixed-pattern noise

variability

To facilitate modeling and provide sufficient repeatability of experiments on Spikey chips, it is essential to minimize these effects by calibration routines. Many calibration routines target parameters with a direct correspondence to biology, such as membrane time constants (described in the following), firing thresholds, synaptic efficacies or PSP shapes. Others have no biological equivalents, such as compensations for shared parameters or workarounds of defects (Bill et al., 2010; Kaplan et al., 2009; Pfeil et al., 2012b). In general, calibration results are used to improve the mapping between biological input parameters and the corresponding target hardware voltages and currents, as well as to determine the dynamic range of all model parameters (see, e.g., Brüderle et al., 2009).

calibration

While the calibration of most parameters is rather technical, but straightforward (e.g., all neuron voltage parameters), some require more elaborate techniques. These include the calibration of τ_m, short-term synaptic plasticity, as well as synapse line drivers.

The membrane time constant $\tau_\mathrm{m} = C_\mathrm{m}/g_\mathrm{l}$ differs from neuron to neuron mostly due to variations in the leakage conductance g_l. However, g_l is independently adjustable for every neuron. Because this conductance is not directly measurable, an indirect calibration method is employed. To this end, the threshold potential is set below the resting potential. Following each spike, the membrane potential is clamped to ϱ for an absolute refractory time τ_ref, after which it evolves exponentially towards the resting potential E_l until the

τ_m calibration

[2] Strictly speaking, E_l is of course not a reversal potential in the same sense as $E_\mathrm{e}^\mathrm{rev}$ and $E_\mathrm{i}^\mathrm{rev}$ (see Section 2.1.1). However, from a purely mathematical perspective, these three parameters have equivalent contributions to the LIF equation 2.54.

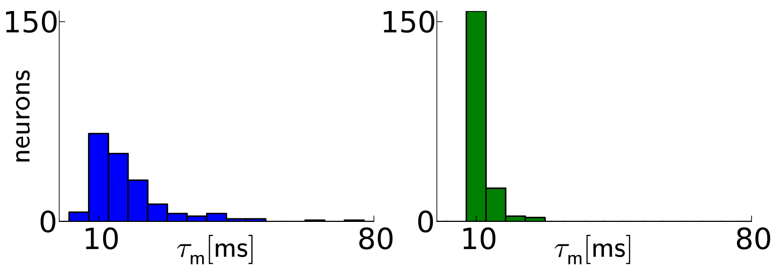


Figure 3.4.: Calibration results for membrane time constants. **Left:** Before calibration, the distribution of τ_{m} values has a median of 15.1 ms with 20th and 80th percentiles of ${\tau_{\mathrm{m}}}^{20} = 10.3\,\mathrm{ms}$ and ${\tau_{\mathrm{m}}}^{80} = 22.1\,\mathrm{ms}$, respectively. **Right:** After calibration, the distribution median lies closer to the target value and narrows significantly: the median is 11.2 ms, with ${\tau_{\mathrm{m}}}^{20} = 10.6\,\mathrm{ms}$ and ${\tau_{\mathrm{m}}}^{80} = 12.0\,\mathrm{ms}$. Two neurons were discarded, because the automated calibration algorithm did not converge.

threshold voltage triggers a spike and the next cycle begins (see Figure 2.16 and Equation 2.36). If the threshold voltage is set to $\vartheta = E_{\mathrm{l}} - 1/e \cdot (E_{\mathrm{l}} - \varrho)$, the spike frequency equals $1/(\tau_{\mathrm{m}} + \tau_{\mathrm{ref}})$, thereby allowing an indirect measurement and calibration of g_{l} and therefore τ_{m}. The effect of calibration on a typical chip can best be exemplified for a typical target value of $\tau_{\mathrm{m}} = 10\,\mathrm{ms}$. Figure 3.4 depicts the distribution of τ_{m} of a typical chip before and after calibration.

The short-term plasticity hardware parameters have no direct translation to model equivalents. In fact, the implemented transconductance amplifier tends to easily saturate within the available hardware parameter ranges. These non-linear saturation effects can be hard to handle in an automated fashion on an individual circuit basis. Consequently, the translation of these parameters is based on short-term plasticity courses averaged over several circuits.

3.3. Wafer-Scale Integration

Going from single chips to multi-chip systems can be achieved in various ways. For example, multiple Spikey modules can be interconnected on a backplane to form larger neural networks (see, e.g., Jeltsch, 2010). However, this appears to be quite inefficient considering the large communication overhead imposed by the non-neuromorphic hardware parts. Considering the fact that individual chips are actually cut down from a wafer, previously to which they lie in close physical proximity to each other, a potential solution is conceptually almost apparent, although technically far from trivial: employ an entire wafer as a single neuromorphic substrate. This the core idea behind the concept of wafer-scale integration.

Figure 3.5 shows a 3D-rendered image of the BrainScaleS wafer-scale hardware system. The 8 inch silicon wafer contains 196 608 neurons and 44 million plastic synapses implemented in mixed-signal VLSI circuitry. As for the Spikey chip (Section 3.2), due to the high integration density of the circuits, the intrinsic time constants of their dynamics are small, fostering a speedup of approx. 10^4 compared to biological real time. The principal building block of the wafer is the so-called HICANN (High Input Count Analog Neural Network) chip (Schemmel et al., 2008, 2010). During chip fabrication, only a limited area called a reticle can be simultaneously exposed during photolithography, which is one of the reasons why such a wafer is cut into individual chips after production. For the BrainScaleS system, however, the wafer is left intact, and additional structures are grown onto the wafer surface in a post-processing step. This process establishes connections between all 384 HICANN blocks that allow a very high bandwidth for on-wafer pulse-event communication (Schemmel et al., 2008). The neuromorphic wafer is accompanied by a stack of digital communication modules for the connection of the wafer to the host PC and to other wafers (see Section 3.3.2 and Figure 3.6).

HICANN

reticle

3.3.1. HICANN Building Block

On the HICANN chip (lower left of Figure 3.6), one can recognize two symmetric blocks which hold the analog core modules. The upper block is depicted in detail in Figure 3.7. Most of the area is occupied by the synapse array with 224 rows and 256 columns. All synapses in a column are connected to one of the 256 neuron circuits located at the center of the chip. For each two adjacent synapse rows, there is one synapse driver that forms the input for pre-synaptic pulses to the synapse array. Synapse drivers are evenly distributed to the left and right side of one synapse array (56 per side). A grid of horizontal and vertical buses enables the routing of spikes from neuron circuits to synapse drivers.

synapse array, neurons

buses

Up to 64 neuron circuits (a.k.a. DenMems, short for dendritic membranes) can be interconnected to form neurons with up to 14336 synapses (see Table 3.2). The neurons emulate the dynamics of the AdEx model in analog circuitry, defined by equations for the membrane voltage u, the adaption current w and a reset condition that applies when a

DenMem

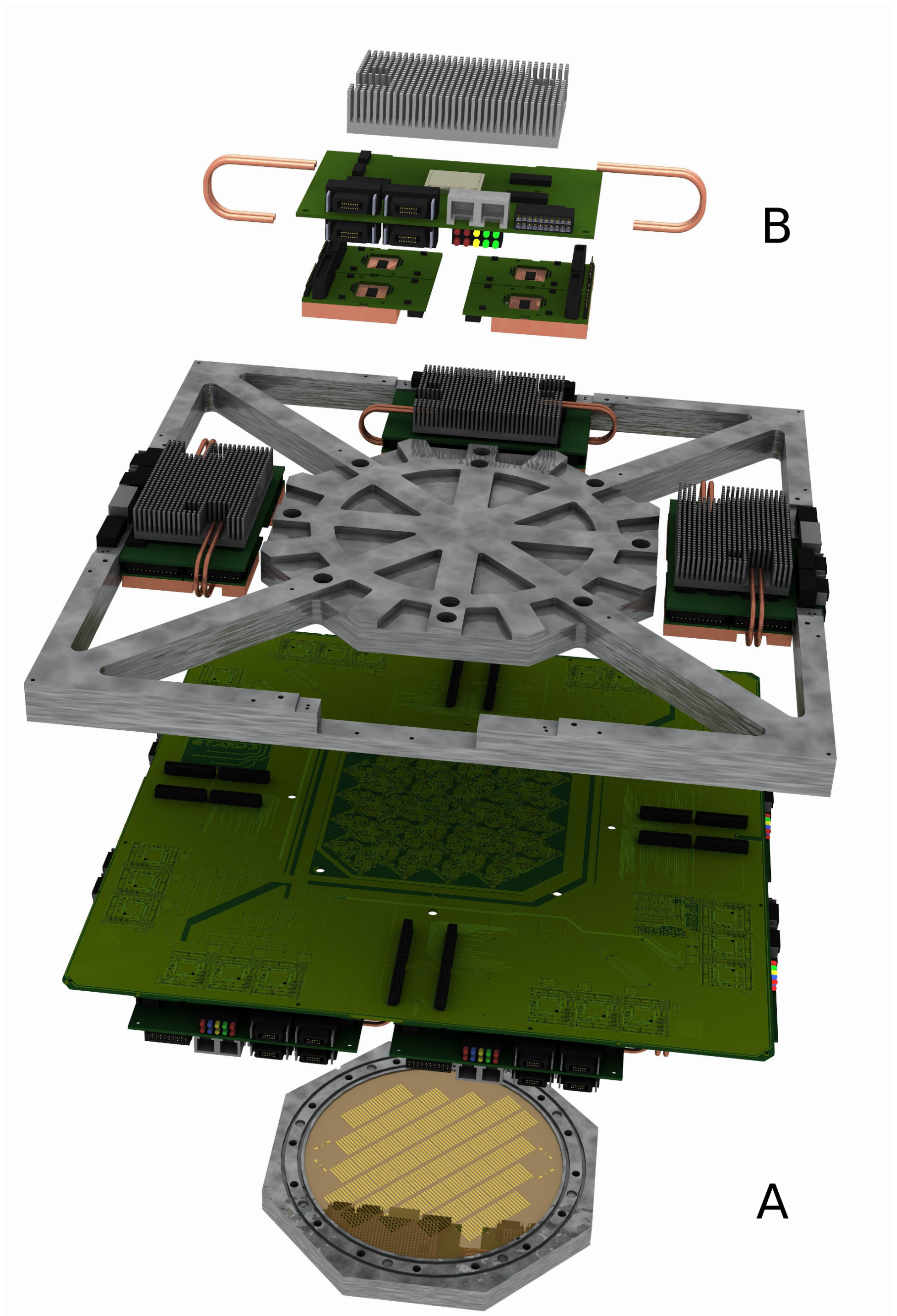


Figure 3.5.: The BrainScaleS wafer-scale hardware system. **(A)** Wafer comprising HICANN building blocks and on-wafer communication infrastructure covered by an aluminum plate. **(B)** Digital inter-wafer and wafer-host communication modules. Also visible: mechanical and electrical support.

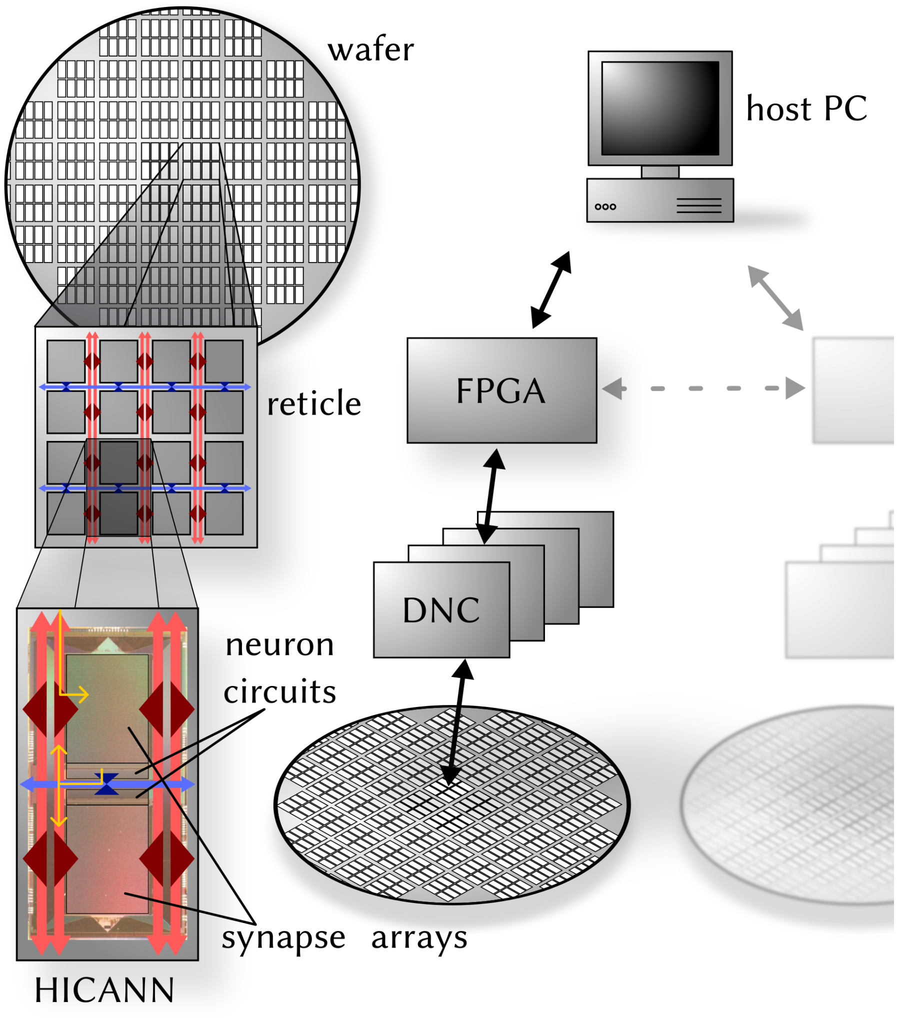


Figure 3.6.: Architecture of the BrainScaleS wafer-scale hardware system. **Left:** The HICANN building block has two symmetric halves with synapse arrays and neuron circuits. Neural activity is transported horizontally (blue) and vertically (red) via asynchronous buses that span over the entire wafer. Exemplary spike paths are shown in yellow on the HICANN. Incoming spike packets are routed to the synapse drivers. In the event that a neuron spikes, it emits a spike packet back into the routing network. **Right:** Off-wafer connectivity is established by a hierarchical packed-based network via DNCs and FPGAs. It interfaces the on-wafer routing buses on the HICANN building blocks. Several wafer modules can be interconnected using routing functionality between the FPGAs.

spike is triggered:

$$C_\mathrm{m}\frac{du}{dt} = -\,g_\mathrm{l}(u-E_\mathrm{l}) + g_\mathrm{l}\Delta_\mathrm{T}\exp\left(\frac{u-\vartheta}{\Delta_\mathrm{T}}\right) - w + I^\mathrm{syn} \quad , \tag{3.3}$$

$$\tau_\mathrm{w}\frac{dw}{dt} = a(u-E_\mathrm{l}) - w \quad , \tag{3.4}$$

$$\text{if } u \geq V_\mathrm{spike}: \quad \begin{cases} u \to \varrho \\ w \to w + b \end{cases} \quad . \tag{3.5}$$

An absolute refractory mechanism is supported by clamping u to its reset value for the refractory time τ_ref. We refer to Section 2.2.1.2 for a more detailed discussion of this neuron model.

synapse drivers, synapses

The generated spikes are transmitted digitally to synapse drivers (which effectively implement analog multipliers), synapses (additional digital multipliers) and finally other neurons, where postsynaptic conductance courses are generated and summed up linearly, resulting in the synaptic current I^syn:

$$I^\mathrm{syn} = \sum_{\text{synapses}\, i} g_i(E_i^\mathrm{rev} - u) \quad , \tag{3.6}$$

$$\tau^\mathrm{syn}\frac{dg_i}{dt} = -g_i + w_i^\mathrm{syn}\sum_{\text{spikes}\, s}\delta(t-t_s) \quad , \tag{3.7}$$

with the same notations as in Section 2.2.2.1. In the hardware implementation (Millner et al., 2010), each neuron features two of such synaptic input circuits, which are typically used for excitatory and inhibitory input. Nearly all parameters of the neuron model and the synaptic input circuits are individually adjustable by means of analog storage banks based on floating gate technology (Lande et al., 1996).

floating gates

In the hardware neuron, both the circuit for the adaption mechanism and the exponential term circuit can be effectively disconnected from the membrane capacitance, such that a simple LIF model can also be emulated. The hardware membrane capacitance is fixed to one of two possible values. As the parameters controlling the temporal dynamics of the neuron such as g_l and the time constants are configurable within a wide range, the hardware is able to run at a variable speedup factor ($10^3 - 10^5$) compared to biological real time. In particular, the translation of the membrane capacitance between the hardware and the biological domain can be chosen freely due to the independent configurability of both membrane and synaptic conductances, thereby effectively allowing the emulation of point neurons of arbitrary size - within the limits imposed by the hardware parameter ranges.

variable speedup

synaptic weights

In contrast to neurons, where each parameter is fully configurable within the specified ranges (see Table 3.3), the synaptic weights are adjustable by a combination of analog and digital memories. The synaptic weight w^syn is proportional to a row-wise adjustable analog parameter g_max and to a 4-bit digital weight specific to each synapse. The g_max of two adjacent rows can be configured to be a fixed multiple of each other. This way, two synapses of adjacent rows can be combined to offer a weight resolution of 8 bits, at the cost of halving the number of synapses for this synapse driver.

STDP

Long-term learning is incorporated in every synapse through STDP (see Section 2.2.2.2). The implemented STDP mechanism follows a pairwise update rule with programmable

Nr of Neurons	Synapses/ Neuron	DenMems/ Neuron	Neurons/ HICANN
196 608	224	1	512
98 304	448	2	256
49 152	896	4	128
24 576	1792	8	64
12 288	3584	16	32
6144	7168	32	16
3072	14 336	64	8

Table 3.2.: Some typical usage scenarios of the wafer-scale hardware system. The number of synapses per neuron can be increased by interconnecting DenMems to form larger point neurons.

update functions (Morrison et al., 2008). As the models we discuss here (Chapter 5) do not incorporate STDP, we refer to Brüderle et al. (2011); Schemmel et al. (2006, 2007) for details on the hardware implementation and to Pfeil et al. (2012a) for an applicability study of these circuits.

short-term plasticity

In contrast to the long-term learning, the implemented short-term plasticity mechanism is not permanent, i.e., all effects decay over periods of up to several hundred ms. It is motivated by the phenomenological model by Markram et al. (1998a) (see also Section 2.2.2.2, but note that the dynamics are not identical!) and depends only on the presynaptic activity, therefore being implemented in the synapse driver. For every incoming spike, a synapse only has access to a portion U of the recovered partition R of its total synaptic weight $w^{\mathrm{syn}}_{\mathrm{max}}$, which then instantly decreases by a factor $1-U$ and recovers slowly along an exponential with the time constant τ_{rec}, thus emulating synaptic depression. Facilitation is implemented by replacing the fixed U with a running variable U, which increases with every incoming spike by an amount $U(1-U)$ and then decays exponentially back to U with the time constant τ_{facil}:

$$w^{\mathrm{syn}}_{n+1} = w^{\mathrm{syn}}_{\mathrm{max}} R_{n+1} U_{n+1} \tag{3.8}$$

$$R_{n+1} = 1 - [1 - R_n(1-U_n)] \exp\left(-\frac{\Delta t}{\tau_{\mathrm{rec}}}\right) \tag{3.9}$$

$$U_{n+1} = U + U_n(1-U) \exp\left(-\frac{\Delta t}{\tau_{\mathrm{facil}}}\right) \tag{3.10}$$

with Δt being the time interval between the nth and $(n+1)$st afferent spike. In contrast to the original TSO mechanism, the hardware implementation does not allow simultaneous depression and facilitation (Bill et al., 2010; Schemmel et al., 2008). See Section A.2.2.1 for details about the hardware implementation and the translation of the original model to the hardware STP.

fixed-pattern noise

trial-to-trial variability

All of the neuron and synapse parameters mentioned above are affected by fixed-pattern noise due to transistor-level mismatch in the manufacturing process. Additionally, the floating gate analog parameter storage reproduces the programmed voltage with a limited precision on each re-write. This leads to trial-to-trial variability for each experiment (see Section A.2.2.2 for exemplary measurements). Limited configurability, such as the

Description	Name	Min	Max	Unit	Comment
Neuron (Adaptive Exponential Integrate&Fire)					
Absolute refractory period	τ_{ref}	0.16	10.0	ms	
Spike detection potential	V_{spike}	-125.0	45.0	mV	
Reset potential	E_{r}	-125.0	45.0	mV	
Leakage reversal potential	V_{rest}	-125.0	45.0	mV	
Membrane time constant	τ_{m}	9	105	ms	
Adaptation coupling param	a	0	10.0	nS	adaptation can be fully disabled
Spike triggered adapt. param	b	0	86	pA	
Adaptation time constant	τ_{w}	20.0	780.0	ms	
Threshold slope factor	Δ_{T}	0.4	3.0	mV	exponential spike
Spike initiation threshold	E_{T}	-125.0	45.0	mV	generation can be fully
Excitatory reversal potential	$E_{\text{e}}^{\text{rev}}$	-125.0	45.0	mV	disabled
Inhibitory reversal potential	$E_{\text{i}}^{\text{rev}}$	-125.0	45.0	mV	
Exc. synaptic time constant	$\tau_{\text{e}}^{\text{syn}}$	1.0	100.0	ms	
Inh. synaptic time constant	$\tau_{\text{i}}^{\text{syn}}$	1.0	100.0	ms	
Synapses					
Weight	w^{syn}	0	0.300	µS	4-bit resolution
Axonal delay (on-wafer)	delay	1.2	2.2	ms	not configurable
Short Term Plasticity					
Utilization of synaptic efficacy	U	0.11	0.47		possible values: $[\frac{1}{9}, \frac{3}{11}, \frac{5}{13}, \frac{7}{15}]$
Recovery time constant	τ_{rec}	40.0	900.0	ms	One of the two time constants has to be set to 0.0. Available range depends on U (maximum range given).
Facilitation time constant	τ_{facil}	35.0	200.0	ms	
Stimulus					
External spike sources	ν	0.0	4000	Hz	cf. Scholze et al. (2011b)

Table 3.3.: Parameter ranges of the BrainScaleS wafer-scale hardware. All ranges correspond to a membrane capacitance of $C_{\text{m}} = 0.2\,\text{nF}$ and a hardware speedup of 10^4 compared to real time. It is possible to choose an arbitrary value for C_{m}, but then the ranges of parameters a, b and of the synaptic weights are multiplied by $\frac{C_{\text{m}}}{0.2\,\text{nF}}$.

discretization of available synaptic weights, is another source for discrepancy between targeted and realized configuration. The trial-to-trial variability, which cannot be reduced by calibration (Section 3.3.3), is assumed to be less than 30 % (standard-deviation-to-mean ratio) for synaptic weights. Other neuron parameters are assumed to have a much smaller variability: E_l, ϑ, E^rev have a standard deviation of less than 1 mV in the biological domain.

calibration

For technical details about the HICANN chip and its components, we refer to Schemmel et al. (2008, 2010).

3.3.2. Communication Infrastructure

The infrastructure for pulse communication in the wafer-scale system is supplied by a two-layer approach. While the on-wafer network routes pulses between neurons on the same wafer, the off-wafer network connects the wafer to the outside world, i.e., to the host PC or to other wafers.

The backbone of the on-wafer communication consists of a grid of horizontal and vertical buses enabling the transport of action potentials by a mixture of time division and space division multiplexing. Each HICANN building block contains 64 horizontal buses at its center and 128 vertical buses located on each side of the synapse blocks, as can be seen in Figure 3.7. A bus can carry the spikes of up to 64 source neurons by transmitting a serial 6-bit signal encoding the currently sending neuron (with an ID from 0 to 63).

buses

When a neuron fires, its pulse is first processed by one of eight priority encoders and finally injected into a horizontal bus after passing a merger stage. By enabling a static switch of a sparse crossbar between horizontal and vertical buses, the injected serial signal can be made available to a vertical bus next to the synapse array. Another sparse switch matrix allows to feed the signals from the vertical buses into the synapse array, more precisely into the synapse drivers which represent the data sinks of the routing network. Synapse drivers can be connected in a chain, forwarding their input to their top or bottom neighbors, thereby allowing to increase the number of synapse rows fed by the same routing bus.

priority encoders, merger tree

crossbar switches

The bus lanes do not end at the HICANN border but run over the whole wafer by edge-connecting the HICANN building blocks (Figure 3.6). Both the sparseness of the switches and the limited number of horizontal and vertical buses represent a possible restriction for the connectivity of network models. If an emulated network requires a connectivity that exceeds the on-wafer bus capacity, some synapses will be impossible to map to the wafer and will therefore be lost.

Pulse propagation delays in the routing network are small, distance-dependent and not configurable. The time between spike detection and the onset of a PSP has been measured as 120 ns for a recurrent connection on a HICANN. The additional time needed to transmit a pulse across the entire wafer is typically less than 100 ns (Schemmel et al., 2008), hence the overall delay sums up to 1.2 - 2.2 ms in the biological time domain, assuming a speedup factor of 10^4.

delays

Also, in case of synchronous bursting of the neurons feeding one bus, some pulses are delayed with respect to others, as they are processed successively. A priority encoder handles the spikes of 64 hardware neurons with priority fixed by design. If several

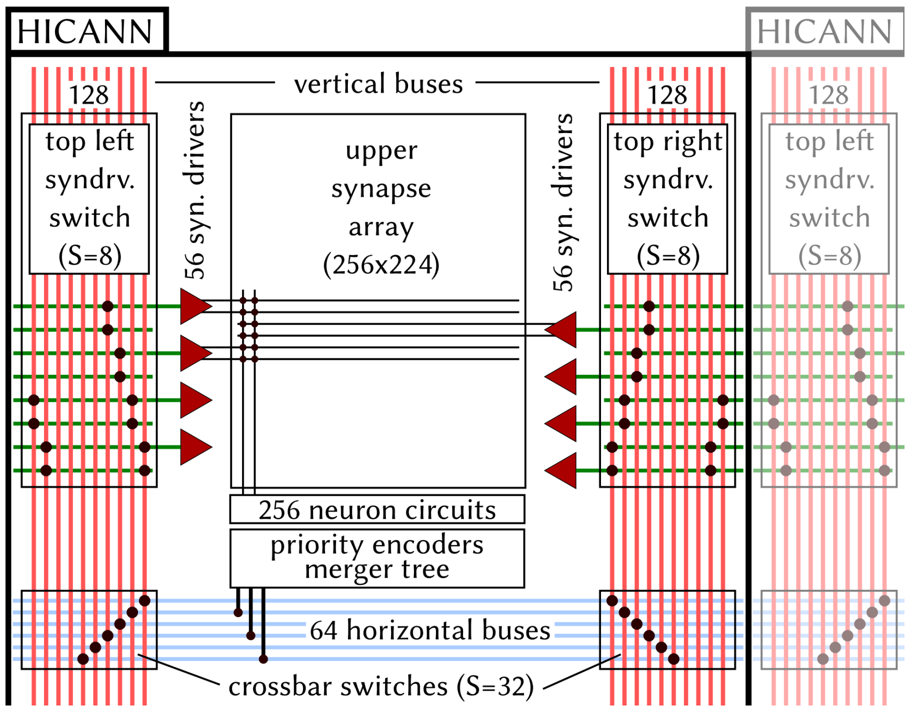

HICANN
HICANN
vertical buses
128
128
128
top left syndrv. switch (S=8)
56 syn. drivers
upper synapse array (256x224)
56 syn. drivers
top right syndrv. switch (S=8)
top left syndrv. switch (S=8)
256 neuron circuits
priority encoders merger tree
64 horizontal buses
crossbar switches (S=32)

Figure 3.7.: Components and connectivity of the HICANN building block. The figure shows the upper block of the HICANN chip. Most of the area is occupied by the synapse array with 256 columns and 224 rows. Each synapse column is connected to one of 256 neuron circuits, from which up to 64 can be interconnected to form larger neurons with up to 14336 input synapses. When a neuron fires, a neuron-specific 6-bit address is generated and injected into one of eight accessible horizontal buses after passing a merger stage. Via two statically configurable switches (crossbar and synapse driver switch) these pulses are routed to the synapse drivers, each of which controls two synapse rows. Every synapse is configured to a specific 6-bit address, so that, when a presynaptic pulse with a matching address arrives, a post-synaptic conductance course is generated at the associated neuron circuit. Both switch matrices are sparse, i.e. configurable switches do not exist at all crossings of horizontal and vertical lines, but e.g. only at every 8th crossing (sparseness S=8). On the wafer, the horizontal and vertical buses, as well as the horizontal lines connected to the synapse drivers do not end at the HICANN borders, but go beyond them.

neurons have fired, the pulse of the neuron with highest priority is transmitted first to the connected horizontal bus. The priority encoder can process one pulse every two clock cycles (2×5 ns), leading to an additional delay for the pulses with lower priority. In rare cases some pulses may be completely discarded, e.g., when the total rate of all 64 neurons feeding one bus exceeds 10 kHz for longer than 6.4 ms (in biological real-time).

off-wafer communication, PCS DNC

A hierarchical packet-based network provides the infrastructure for off- and inter-wafer communication. All HICANNs on the wafer are connected to the surrounding system and to other wafers via 12 pulse communication subgroups (PCS). Each PCS consists of one FPGA and 4 ASICs (Application Specific Integrated Circuits) that were designed for high-bandwidth pulse-event communication (so-called Digital Network Chips or DNCs). Being the only communication link to/from the wafer, the off-wafer network also transports the configuration and control information for all the circuits on the wafer. As depicted in Figure 3.6, the network is hierarchically organized: one FPGA is connected to four DNCs, each of which is connected to 8 HICANNs of a reticle. Each FPGA is also connected to the host PC and potentially to up to 4 other FPGAs.

When used for pulse-event communication, an FPGA-DNC-HICANN connection supports a throughput of 40 Mevents/s (Scholze et al., 2011b) with a timing precision of 4 ns. In the biological time domain, this corresponds to monitoring the spikes of all 512 neurons on a HICANN firing with a mean rate of 8 Hz each with a resolution of 0.04 ms. The same bandwidth is available simultaneously in the opposite direction, allowing a flexible network stimulation with user-defined spiketrains.

For further technical details about the PCS, the FPGA design and the DNC, we refer to Scholze et al. (2010, 2011a) and Hartmann et al. (2010).

3.3.3. Software Framework

The utilized software stack (Brüderle et al., 2011) allows the user to define a network description and maps it to a hardware configuration. As for the Spikey chip, the network definition can be done in PyNN, thus abstracting away the hardware details. In principle, one could use only the PyNN level for completely hardware-agnostic modeling. As it turns out, however, more challenging emulation scenarios (large networks that are sensitive to parameter distortions) do require at least a high-level understanding of the hardware itself, which constitutes the subject of the entire Chapter 5.

hardware-agnostic modeling

mapping

The mapping process (Brüderle et al., 2011; Ehrlich et al., 2010) translates the PyNN description of the neural network structure, as well as its neuron and synapse models and parameters, in several steps into a neuromorphic device configuration. This translation is constrained by the architecture of the device and its available resources.

placement routing parameter transformation

The first step of the mapping process is to allocate static structural neural network elements to particular neuromorphic components during the so-called placement . Subsequently, a routing step is executed for establishing connections in between the placed components. During the final parameter transformation step, all parameters of the network components (neurons and synapses) are translated into hardware parameters. First, the model parameters are transformed to the voltage and time domain of the hardware, taking into account the acceleration and the voltage range of 0 V to 1.8 V (Millner et al., 2010). Afterwards, previously obtained calibration data is used to reduce mismatches

calibration

between the target behavior and real behavior of the analog components.

The objective of the mapping process is to find a configuration of the hardware that best reproduces the experiment specified in PyNN. The most relevant constraints are sketched in the following.

Each hardware neuron circuit has a limited number of 224 incoming synapses. By interconnecting several neuron circuits one can form larger neurons with more incoming synapses (see Section 3.3.1), with the trade-off that the overall number of neurons is reduced. Still, each hardware synapse can not be used to implement a connection from an arbitrary neuron but only from a subset of neurons, namely the 64 source neurons whose pulses arrive at the corresponding synapse driver. For networks larger than 10 000 neurons it is the limited number of inputs to one HICANN that becomes even more restricting, as there are only 224 synapse drivers (see Figure 3.7), yielding a maximum of 14366 different source neurons for all neurons that are placed to the same HICANN. Hence, one objective of the mapping process is to reduce this number of source neurons per HICANN, thus increasing the number of realized synapses on the hardware. In general, this criterion is met when neurons with common pre-synaptic partners are placed onto the same HICANN and neurons with common targets inject their pulses into the same on-wafer routing bus.

All of the above, as well as the limited number of on-wafer routing resources (see Section 3.3.2) make the mapping optimization an NP-hard problem. The used placement and routing algorithms, which improve upon the ones described in Brüderle et al. (2011) and Fieres et al. (2008) but are far from being optimal, can minimize the effect of these constraints only to a certain degree. Thus, depending on the network model size, its connectivity, and the choice of the mapping algorithms, synapses are lost during the mapping process; in other words, some synapses of a network defined in PyNN will simply not exist in the corresponding network that is emulated on the hardware.

synapse loss

3.3.4. Executable System Specification (ESS)

ESS

The ESS is a detailed simulation of the hardware platform (Brüderle et al., 2011; Ehrlich et al., 2007) that replicates the topology and dynamics of the communication infrastructure as well as the analog synaptic and neuronal components.

The ESS encompasses a numerical solver of the equations that govern the hardware neuron and synapse dynamics, as well a detailed reproduction of the digital communication infrastructure at the level of individual spike transmission in logical hardware modules. The ESS is a specification of the hardware in the sense that its configuration space faithfully maps the possible interconnection topologies, parameter limits, parameter discretization and shared parameters. Being executable, the ESS also covers dynamic constraints, such as the consecutive processing of spikes which can lead to spike time jitter or spike loss.

Variations in the analog circuits that are inherent to analog VLSI are not simulated at transistor level but are rather artificially imposed on the ideal hardware parameters. All of this allows to simultaneously capture the complex dynamic behavior of the hardware and comply with local bandwidth limitations, while allowing relatively quick simulations due to the high level of abstraction. Analogously to the neuromorphic hardware itself, simulations on the ESS can be controlled using PyNN, with only a few additional hardware-specific commands. Both for the real hardware and for the ESS, the mapping process translates a PyNN network into a device configuration, which is then used as an input for the respective back-end. One particular advantage of the ESS is that it allows access to state variables which can otherwise not be read out from the real hardware, such as the logging of lost or jittered time events.

4. Dynamics and Statistics of Poisson-Driven LIF Neurons

The miracle of the appropriateness of the language of mathematics for the formulation of the laws of physics is a wonderful gift which we neither understand nor deserve. We should be grateful for it and hope that it will remain valid in future research and that it will extend, for better or for worse, to our pleasure, even though perhaps also to our bafflement, to wide branches of learning.

Eugene Wigner, The Unreasonable Effectiveness of Mathematics in the Natural Sciences, 1960

Eugene Wigner wrote a famous essay on the unreasonable effectiveness of mathematics in natural sciences. He meant physics, of course. There is only one thing which is more unreasonable than the unreasonable effectiveness of mathematics in physics, and this is the unreasonable ineffectiveness of mathematics in biology.

Israel Gelfand, alleged quote

In Chapter 2, we have described how point neuron and exponential synapse models can be obtained as an abstraction of the complex electrochemical dynamics of neurons and synapses observed in vivo. Throughout the remainder of this thesis, we shall continue working with such abstract models, in particular point LIF and AdEx neurons with either current or conductance-based synapses with an exponential interaction kernel.

Despite the relatively simple form of their governing differential equations, it turns out that closed-form solutions for their phase space trajectories are not always easily found. Furthermore, when subjected to background synaptic bombardment, as is usually the case in vivo, even these simple models turn out to have a surprising behavior. In particular, important differences appear between current- and conductance-based synaptic interaction. As the investigated questions are of rather fundamental nature, we are not the first to discuss these problems. Related work, although treated from a different perspective, can be found in, e.g., Richardson and Gerstner (2006).

In Section 4.1, we start with a short recapitulation of some fundamental statistical concepts and use the opportunity to define important notations. We then study closed-form solutions and approximations thereof for COBA and CUBA LIF neurons in Section 4.2. In particular, we show how the high-conductance state enables us to find a good approximation for COBA LIF neurons. Based on these findings, we move on to the description of statistical properties of LIF dynamic variables under balanced Poisson stimulation in Section 4.3. We derive general expressions for the first two moments of the distribution of a Poisson-driven dynamic variable and apply these findings to the synaptic current and the membrane potential of LIF neurons. This enables a theoretical understanding of the qualitatively different behavior of COBA and CUBA neurons at high input rates. In Section 4.4, we then use our previous results to derive quantitative expressions for shared-input correlations of LIF neurons, both for the free membrane potential and at the level of spike trains.

A significant part of this work has been done in collaboration with Ilja Bytschok and Johannes Bill and has been the subject of several technical internal reports dating back to 2009-2011. Furthermore, the investigated subjects have also been the driving questions behind the Diploma thesis by Bytschok (2011), from which we especially use material in Section 4.4.

4.1. Probability Theory: Essentials

As a short introduction to the following sections, we review several fundamental concepts, definitions and notations from probability theory to which we will return repeatedly throughout this work.

4.1.1. Random Variables and Probability Distributions

Consider an experiment that can produce several outcomes to which we attach some uncertainty. We denote the (multidimensional) outcome of the experiment by capitals – $\boldsymbol{Z}$ represents a set of random variables (RVs) – while a specific outcome $\boldsymbol{z} = (z_1, ..., z_n)$ is denoted by minuscules. The complete set of possible outcomes $\boldsymbol{z}$ is called the sample space

random variable

sample space

$$\Omega = \prod_{i=1}^{n} \Omega_i, \text{ with } z_i \in \Omega_i \quad , \tag{4.1}$$

where $\prod$ denotes a Cartesian product. Each outcome z can occur with a probability $f(\boldsymbol{z})$:

$$P(\boldsymbol{Z} = \boldsymbol{z}) = f(\boldsymbol{z}) \ . \tag{4.2}$$

$f(\boldsymbol{Z})$ is called the joint probability distribution of the variables $Z_1, \ldots, Z_n$. Similarly to using $f(\boldsymbol{z})$, when the interpretation becomes clear from the context, we shall use the shorthand notation of $P(\boldsymbol{z})$ in lieu of the more cumbersome $P(\boldsymbol{Z} = \boldsymbol{z})$.

joint multivariate probability distribution

If Ω is discrete, in order for f to represent a probability distribution, we require

$$f(\boldsymbol{z}) \leq 1, \forall \boldsymbol{z} \in \Omega \tag{4.3}$$

and

discrete probability distribution

$$\sum_{\boldsymbol{z} \in \Omega} f(\boldsymbol{z}) = 1 \ . \tag{4.4}$$

If either Ω or $\sum\limits_{\boldsymbol{z} \in \Omega} \tilde{f}(\boldsymbol{z})$ is finite, any mapping $\tilde{f} : \Omega \to \mathbb{R}$ can be transformed into a valid probability distribution by normalization:

normalization

$$f(\boldsymbol{z}) := \frac{\tilde{f}(\boldsymbol{z})}{\sum\limits_{\boldsymbol{z} \in \Omega} \tilde{f}(\boldsymbol{z})} \ . \tag{4.5}$$

4.1.2. Joint and Conditional Distributions

If one is only interested in the joint probability distribution over a subset of RVs, the total PDF $f(\boldsymbol{z})$ needs to be marginalized over all other RVs:

marginalization

$$f(z_1, \ldots, z_m) = \sum_{z_{m+1} \in \Omega_{m+1}} \sum_{z_{m+2} \in \Omega_{m+2}} \cdots \sum_{z_n \in \Omega_n} f(\boldsymbol{z}) \ . \tag{4.6}$$

It is quite apparent that this operation is very demanding from a computational point of view. The total number of operations required to perform marginalization scales exponentially with the number of variables one needs to marginalize over. This can occur at its most extreme in normalization scenarios, where the denominator in Equation 4.5 represents a marginalization of the distribution over all RVs. The problem of marginalization and normalization is a key issue in computational statistics and will represent a focal point of Chapter 6.

conditional distribution

Bayes' rule

If a subset $\boldsymbol{X}$ of the RVs are fixed to some value $\boldsymbol{x}$, such as when they represent inputs to the stochastic system, then the resulting distribution $P(\boldsymbol{Y}=\boldsymbol{y}|\boldsymbol{X}=\boldsymbol{x})$ over all other RVs $\boldsymbol{Y}$ is called a conditional distribution. Conditional, marginal and joint distributions are intimately linked through Bayes' rule:

$$P(\boldsymbol{Y}=\boldsymbol{y}|\boldsymbol{X}=\boldsymbol{x}) = \frac{P(\boldsymbol{Y}=\boldsymbol{y},\boldsymbol{X}=\boldsymbol{x})}{P(\boldsymbol{X}=\boldsymbol{x})} \quad . \tag{4.7}$$

posterior distribution

inference

Such conditional distributions are often called posterior distributions, in which context the fixed or observed variables are referred to as evidence. The process of calculating posterior distributions given evidence is called probabilistic inference.

indepen-dence

A set of RVs $\{\mathbf{X_i}\}$ is called mutually independent if and only if their joint probability factorizes:

$$p(\boldsymbol{X}_1=\boldsymbol{x}_1,\ldots,\boldsymbol{X}_n=\boldsymbol{x}_n) = \prod_i p(\boldsymbol{X}_i=\boldsymbol{x}_i) \tag{4.8}$$

A set of RVs is called pairwise independent if every pair of RVs in the set is independent.

4.1.3. Moments of Probability Distributions

Given any function $g:\Omega\to\mathbb{R}$, the quantity $E[g(\boldsymbol{Z})]$ is called the expected value of g over $\boldsymbol{Z}$ and is defined as

$$E[g(\boldsymbol{Z})] = \sum_{\boldsymbol{z}\in\Omega} g(\boldsymbol{z})f(\boldsymbol{z}) \quad . \tag{4.9}$$

raw moment

central moment

mean, variance

Depending on the context, we shall use $E[\cdot]$ and $\langle\cdot\rangle$ interchangeably, in order to improve the readability of some equations. The n^{th} raw moments μ'_n of a probability distribution over a scalar RV Z are defined as the expected values of Z^n. The central moments are the moments about its mean $E[Z]$. The first raw and second central moment are called the mean and variance of Z, respectively, and are often denoted by μ_Z and σ_Z^2 (or simply $\mathrm{Var}[Z]$):

$$\mu_Z := \mu'_1(Z) = E[Z] \tag{4.10}$$
$$\sigma_Z^2 \equiv \mathrm{Var}[Z] := \mu'_2(Z-E[Z]) = E\left[(Z-\mu_Z)^2\right] \quad . \tag{4.11}$$

covariance

σ_Z is also called the standard deviation of Z. Similar to the variance, but for pairs of RVs, the covariance is defined as

$$\mathrm{Cov}[X,Y] := E\left[(X-\mu_X)(Y-\mu_Y)\right] \quad . \tag{4.12}$$

correlation

A pair of RVs is called uncorrelated if and only if their covariance vanishes. In particular, independent RVs are always uncorrelated. The converse is, of course, not necessarily true, as can be easily exemplified by considering the case where $p(Y=x|X=x)=p(Y=-x|X=x)=1/2$.

In case of multivariate distributions, the mean is represented by a vector $\boldsymbol{\mu}$ and the variance becomes a covariance matrix[1] $\boldsymbol{\Sigma}$:

covariance matrix

$$\boldsymbol{\mu}_{\boldsymbol{Z}} := E[\boldsymbol{Z}] \tag{4.13}$$

$$\boldsymbol{\Sigma}_{\boldsymbol{Z}} \equiv \mathrm{Cov}[\boldsymbol{Z},\boldsymbol{Z}] \equiv \mathrm{Var}[\boldsymbol{Z}] := E\left[(\boldsymbol{Z}-\boldsymbol{\mu})(\boldsymbol{Z}-\boldsymbol{\mu})^T\right] \quad , \tag{4.14}$$

with the covariance of two multidimensional RVs defined as

$$\mathrm{Cov}[\boldsymbol{X},\boldsymbol{Y}] := E\left[(\boldsymbol{X}-\boldsymbol{\mu}_{\boldsymbol{X}})(\boldsymbol{Y}-\boldsymbol{\mu}_{\boldsymbol{Y}})^T\right] \tag{4.15}$$

The covariance can be transformed to a sometimes more convenient representation

$$\mathrm{Cov}[\boldsymbol{X},\boldsymbol{Y}] = E[\boldsymbol{X}\boldsymbol{Y}^T] - \boldsymbol{\mu}_{\boldsymbol{X}}\boldsymbol{\mu}_{\boldsymbol{Y}}^T \quad . \tag{4.16}$$

For a set of random variables $\{\boldsymbol{Z}_1,\ldots,\boldsymbol{Z}_n\}$ of equal dimensionality, the following equations hold:

$$E\left[\sum_{i=1}^{n}\boldsymbol{Z}_i\right] = \sum_{i=1}^{n} E\left[\boldsymbol{Z}_i\right] \tag{4.17}$$

$$\mathrm{Cov}\left[\sum_{i=1}^{k}\boldsymbol{Z}_i, \sum_{j=k}^{n}\boldsymbol{Z}_j\right] = \sum_{i=1}^{k}\sum_{j=k}^{n}\mathrm{Cov}\left[\boldsymbol{Z}_i,\boldsymbol{Z}_j\right] \quad . \tag{4.18}$$

4.1.4. Continuous Random Variables

If Ω is an infinite subset of $\mathbb{R}^n$ (i.e, one has n constinuous scalar RVs), the equations above remain, in general, unchanged, except that summations over (subsets of) Ω are replaced by integrals. One additional concept is required, which is the cumulative (multivariate) distribution function (CDF), defined as

cumulative distribution function

$$F(\boldsymbol{z}) = P(Z_1 \le z_1, Z_2 \le z_2, \ldots, Z_n \le z_n) \ , \tag{4.19}$$

which must satisfy the following conditions:
1. F is monotonically increasing and right-continuous
2. $\lim_{\boldsymbol{z}\to-\vec{\infty}} F(\boldsymbol{z}) = 0$
3. $\lim_{\boldsymbol{z}\to+\vec{\infty}} F(\boldsymbol{z}) = 1$

If F is absolutely continuous, one can define a probability density function (PDF) $f(\boldsymbol{z})$ which must satisfy

probability density function

$$F(\boldsymbol{z}) = \int_{-\infty}^{z_1}\cdots\int_{-\infty}^{z_n} f(\boldsymbol{z})d\boldsymbol{z} \quad . \tag{4.20}$$

[1] Sometimes, the covariance matrix is simply called a variance, regardless of the dimensionality of the RV. This denomination is meant to show how the variance matrix of multidimensional RVs is a natural extension of the scalar variance of scalar RVs. The term "covariance matrix", on the other hand, points towards the fact that individual matrix elements of $\boldsymbol{\Sigma}_{\boldsymbol{Z}}$ are, indeed, covariances between the scalar components of $\boldsymbol{Z}$.

In particular, for any hyperrectangle $\mathcal{I}^n = I_1 \times \cdots \times I_n$, with $I_i = [a_i, b_i] \in \mathbb{R}$, it holds that

$$P(\boldsymbol{z} \in \mathcal{I}^n) = \int_{a_1}^{b_1} \dots \int_{a_n}^{b_n} f(\boldsymbol{z}) d\boldsymbol{z} \quad . \tag{4.21}$$

In the limit of $\mathcal{I}^n$ becoming infinitesimally small around some value $\boldsymbol{z}_0$, the probability for $P(\boldsymbol{z} \in \mathcal{I}^n) \to f(\boldsymbol{z}_0) d\boldsymbol{z}$ of a continuous RV is the closest equivalent to $P(\boldsymbol{Z} = \boldsymbol{z}_0) = f(\boldsymbol{z}_0)$ for a discrete RV $\boldsymbol{Z}$ (which is why we have chosen an identical notation $f(\cdot)$ for discrete probability distributions and continuous PDFs).

4.2. Closed-Form Solutions for the LIF Equations

In the following, we discuss closed-form solutions for the LIF membrane equations. We shall see that, for the CUBA case, such solutions are more easily found. Despite the fact that an exact solution for the COBA case can not be given, we shall discuss how the high-conductance state enables us to find a quantitatively good approximation. Furthermore, we discuss some counterintuitive properties of membrane statistics in the high-conductance state that are observed in computer simulations, which we then treat analytically in Section 4.3.

4.2.1. Reformulation of the LIF Equation with an Effective Membrane Potential

We have discussed in the introduction how neurons "integrate" their input, as long as they are in the subthreshold regime. While this is, indeed, an essential property of subthreshold membrane potential dynamics, it is often useful to adopt a somewhat different view. In this section, we will introduce a new variable u_eff, which will call an "effective membrane potential". This variable will represent a (linear) function of the input, while the true membrane potential will become a low pass filter thereof. Up to a certain point, this formalism applies to both COBA and CUBA synapses; the remaining – and important – differences between the two models (as already mentioned in Section 2.2.2.1) will be addressed in detail throughout the following sections.

u as a low-pass filter of u_eff

We start by restating the set of equations that govern the LIF neuron model. For the membrane potential, we have

$$C_\mathrm{m} \frac{du}{dt} = g_\mathrm{l}(E_\mathrm{l} - u) + I \quad , \tag{4.22}$$

where the input current I can be partitioned into an external background I^ext (injected directly into the neuron, be it in tissue or in software simulations) and a synaptic stimulus current I^syn:

$$I = I^\mathrm{ext} + I^\mathrm{syn} \quad . \tag{4.23}$$

The total synaptic current I^syn obeys the equation

total synaptic current

CUBA

$$I^\mathrm{syn} = \sum_{\text{synapses}\,k} i_k^\mathrm{syn} \tag{4.24}$$

COBA

$$I^\mathrm{syn} = \sum_{\text{synapses}\,k} g_k^\mathrm{syn}\,(E_k^\mathrm{rev} - u) \ , \tag{4.25}$$

where i_k^syn and g_k^syn denote the current/conductance arriving from the kth synapse (which, themselves, are sums of PSCs). We can now define a total conductance g^tot, which represents the sum of all the conductances that affect the membrane:

total conductance

CUBA

$$g^\mathrm{tot} = g_\mathrm{l} \tag{4.26}$$

COBA

$$g^\mathrm{tot} = g_\mathrm{l} + \sum_k g_k^\mathrm{syn} \quad , \tag{4.27}$$

effective membrane time constant

an effective membrane time constant, which depends on g^{tot}:

$$\tau_{\text{eff}} = \frac{C_{\text{m}}}{g^{\text{tot}}} \quad , \tag{4.28}$$

effective membrane potential

and an effective membrane potential, which can be expressed as a weighted sum of all the mechanisms that affect the membrane (leak, external current, synaptic stimulus):

CUBA

$$u_{\text{eff}} = \frac{g_{\text{l}} E_{\text{l}} + I^{\text{ext}} + \sum_k i_k^{\text{syn}}}{g^{\text{tot}}} \tag{4.29}$$

COBA

$$u_{\text{eff}} = \frac{g_{\text{l}} E_{\text{l}} + I^{\text{ext}} + \sum_k g_k^{\text{syn}} E_k^{\text{rev}}}{g^{\text{tot}}} \quad . \tag{4.30}$$

By dividing the RHS of Equation 4.22 by g^{tot} and rearranging the terms, we finally obtain

$$\tau_{\text{eff}} \frac{du}{dt} = u_{\text{eff}} - u \quad . \tag{4.31}$$

The appeal of this formulation lies in its simplicity. By abstracting away the complex interaction of external stimuli into an effective membrane potential, one can view the membrane potential simply as a variable that decays exponentially ("leaks") towards some target value $u_{\text{eff}}(t)$. This effective potential encompasses the neuron's own leak potential, as well as all external stimuli – synapses in particular.

In this formulation, a hallmark effect of conductance-based synapses becomes immediately apparent: as synaptic bombardment increases, so does g^{tot} (Equation 4.27), resulting in a smaller τ_{eff} and therefore in a faster membrane and smaller PSPs (see also Figure 2.23). The existence of this so-called high-conductance state (see Section 4.2.5) with its accelerated membrane dynamics has important computational consequences, and has been covered by vast amounts of literature[2]. In particular, it plays an essential role in the LIF sampling framework discussed in Section 6.5. For now, we will use Equation 4.31 for a formal derivation of the PSP shapes produced by exponential synapses.

4.2.2. Analytical Solutions for the LIF Equations: CUBA Synapses

Having unified the mathematical description of COBA and CUBA LIF neurons into a common ODE (Equation 4.31), we can now attempt to find a unified solution for it. It quickly becomes apparent that this is not possible in the general case.

Assuming the solution of the ODE

$$\frac{d}{dt} f(t) = b(t) - a(t) \cdot f(t) \tag{4.32}$$

exists and is unique[3], it takes on the form

$$f(t) = \exp\left(\int^t -a(x)\, dx \right) \left[\int^t b(x) \exp\left(\int^x a(y)\, dy \right) dx + C \right] \quad , \tag{4.33}$$

[2]For a good review with a comprehensive bibliography, see Destexhe (2007).

[3]The Picard-Lindelöf theorem guarantees existence and uniqueness for our particular case, but it only applies between two consecutive spikes.

where $\int^t g(x)\,dx$ denotes the antiderivative of a function $g(t)$. For clarity, we now write out the time dependencies explicitly. By setting

$$a(t) = \frac{1}{\tau_{\text{eff}}(t)} \tag{4.34}$$

and

$$b(t) = \frac{u_{\text{eff}}(t)}{\tau_{\text{eff}}(t)} \quad , \tag{4.35}$$

we can now obtain the solution to Equation 4.31:

$$u(t) = \exp\left(-\int^t \frac{dx}{\tau_{\text{eff}}(x)}\right)\left[\int^t \frac{u_{\text{eff}}(x)}{\tau_{\text{eff}}(x)} \exp\left(\int^x \frac{dy}{\tau_{\text{eff}}(y)}\right) dx + C\right] \quad . \tag{4.36}$$

In LIF neurons with COBA synapses, g^{tot} depends explicitly on time. In the particular case of exponential conductances (Equation 2.58), τ_{eff} is proportional to a sum of inverse exponentials. Finding an analytical solution for $u(t)$ therefore requires a closed-form expression for $\int^t \exp[\exp(x)]\,dx$. Such an expression does not exist and therefore neither does a closed-form solution for the membrane potential of LIF neurons with exponential COBA synapses.

However, in particular situations, it is possible to effectively reduce COBA LIF neurons to a CUBA-like description. Such a case will be made in Section (4.2.4) for the so-called high-conductance state. We will therefore proceed to solving Equation 4.33 in the CUBA case and return to this result later on.

We can start by plugging g^{tot}, τ_{eff} and u_{eff} from 4.26, 4.28 and 4.29, respectively, into 4.36 to obtain

$$u(t) = \exp\left(-\int^t \frac{dx}{\tau_{\text{m}}}\right)\left[\int^t \frac{g_{\text{l}}E_{\text{l}} + I^{\text{ext}} + \sum_k i_k^{\text{syn}}}{g_{\text{l}}\tau_{\text{m}}} \exp\left(\int^x \frac{dy}{\tau_{\text{m}}}\right) dx + C\right] \quad . \tag{4.37}$$

We further assume a constant external current stimulus I^{ext}. After explicitly writing out the synaptic currents (Equation 2.58), we can arrive at a closed-form solution for the membrane potential of an LIF neuron with exponentially decaying CUBA synapses under synaptic bombardment:

membrane potential of an LIF neuron with exponential synaptic currents

$$\begin{aligned} u(t) &= \frac{\exp\left(-\frac{t}{\tau_{\text{m}}}\right)}{g_{\text{l}}\tau_{\text{m}}}\left[\left(g_{\text{l}}E_{\text{l}} + I^{\text{ext}}\right)\int^t \exp\left(\frac{x}{\tau_{\text{m}}}\right) dx \right. \\ &\qquad \left. + \int^t \sum_{\text{syn }k}\sum_{\text{spk }s} w_k \Theta(t-t_s)\exp\left(-\frac{x-x_s}{\tau_k^{\text{syn}}} + \frac{x}{\tau_{\text{m}}}\right) dx\right] \\ &= \dots \\ &= E_{\text{l}} + \frac{I^{\text{ext}}}{g_{\text{l}}} + \sum_{\text{syn }k}\sum_{\text{spk }s} \frac{\tau_k^{\text{syn}} w_k}{g_{\text{l}}\left(\tau_k^{\text{syn}} - \tau_{\text{m}}\right)}\Theta(t-t_s)\left[\exp\left(-\frac{t-t_s}{\tau_k^{\text{syn}}}\right) - \exp\left(-\frac{t-t_s}{\tau_{\text{m}}}\right)\right] \quad . \end{aligned} \tag{4.38}$$

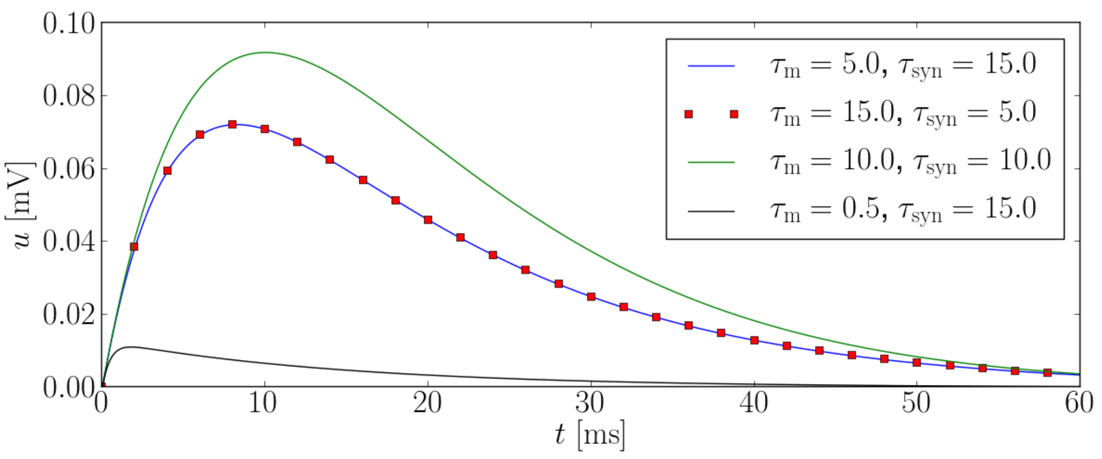


Figure 4.1.: Characteristic PSP shapes. PSPs are symmetric in τ_m and τ^syn (blue line and red dots). When $\tau_\mathrm{m} = \tau^\mathrm{syn}$, the PSP can be described by an α-function (green line), which is similar in shape to generic PSPs (other lines). When either time constant becomes very small, PSPs take on a "more exponential" shape (black line).

Despite being somewhat lengthy, this equation paints a very intuitive picture. The first two terms represent a constant offset given by the leak potential E_l and the external current I^ext. The last term represents a linear superposition of PSPs which take the shape of a difference of exponentials (DOE):

difference of exponentials

CUBA PSP

$$\mathrm{PSP}(t) = \frac{\tau_k^\mathrm{syn} \tau_\mathrm{m} w_k}{C_\mathrm{m} \left(\tau_k^\mathrm{syn} - \tau_\mathrm{m}\right)} \Theta(t - t_s) \left[\exp\left(-\frac{t - t_s}{\tau_k^\mathrm{syn}}\right) - \exp\left(-\frac{t - t_s}{\tau_\mathrm{m}}\right) \right] \quad , \tag{4.39}$$

reflecting the linearity of the ODE for the effective membrane potential (Equation 4.31). The shape of a single PSP follows trivially from calculating $u'(t) - u(t)$, where u' represents the value of the membrane potential when a single spike, i.e., a single term in the double sum, is added to the others.

A short discussion of the DOE PSP is in order.

PSP symmetry in τ_m and τ^syn

synaptic memory

The decaying nature of the membrane potential and synaptic conductance become apparent in the exponential terms depending on τ_m and τ^syn, respectively. Most notably, the PSP is symmetric in τ_m and τ^syn, which is central to the analytical derivation of the noisy LIF activation function in Section 6.5.3. However, there is one essential asymmetry: due to the reset following a spike, the membrane potential itself has no memory of its past. It may, however, react to stimuli that have preceded the spike, due to the fact that the membrane conductance is not reset at the time of spiking. Therefore, independently of plasticity and adaptation, a single neuron has a "synaptic memory" with a specific time constant τ^syn that may stretch beyond its last spike, a property that is essential for the LIF sampling framework described in Section 6.5.

In the limit of $\tau^{\rm syn} \to \tau_{\rm m}$, the shape of the PSP becomes a function of the type $t \exp(-t)$:

PSP shape for $\tau^{\rm syn} = \tau_{\rm m}$

$$\lim_{\tau^{\rm syn} \to \tau_{\rm m}} \mathrm{PSP}(t) = \frac{w_k}{C_{\rm m}} t \exp\left(-\frac{t - t_s}{\tau_{\rm m}}\right) \quad . \tag{4.40}$$

In literature, this is often called an α-function. As can be seen in Figure 4.1, the shape of an α-PSP is qualitatively similar to that of a DOE PSP.

α-function

When either time constant ($\tau_{\rm m}$ or $\tau^{\rm syn}$) is close to zero, the PSP shape becomes “more exponential” and the neuron reacts “faster” to afferent spikes - i.e., the maximum of the PSP is reached sooner, although the amplitude decreases. This limit appears naturally in the high-conductance state (see Section 4.2.5) and also plays an important role in the discussion of probabilistic inference in networks of LIF neurons (Section 6.5).

On a final note, we need to repeat that the PSP shapes discussed above are not to be confused with the PSC shapes discussed in Section 2.2.2. All neuron/network models discussed in this manuscript are based on synapses which generate exponential PSCs, which lead to DOE PSPs as an analytical solution of the LIF equation. The formal similarity to DOE-PSCs is purely coincidental.

4.2.3. The High-Conductance State I: First Observations

As already stated, it is not possible to find a general solution for the LIF neuron with exponential COBA synapses as we just did in the case of CUBA synapses. However, a good approximation for such a solution can be found in the high-conductance state (HCS). We will set out by qualitatively describing the effects of the HCS on the membrane and conductance dynamics and use these to find the abovementioned approximate solution to the COBA LIF equation. Later on (Section 4.3), we will derive exact expressions for the statistics of the synaptic input (be it current or conductance) and membrane potential.

high-conductance state

As implied by its name, the HCS is characterized by a high total membrane conductance

$$g^{\rm tot} = g_{\rm l} + \sum_k g_k^{\rm syn} \quad . \tag{4.41}$$

While “high” is not precisely defined in literature, it is usually assumed that in the HCS the synaptic conductance is the dominant term in the total membrane conductance:

$$\sum_k g_k^{\rm syn} =: g^{\rm syn} \gg g_{\rm l} \quad . \tag{4.42}$$

dominant synaptic conductance

A high synaptic conductance can be achieved by either increasing the weight of single synapses or by increasing the total number of incoming spike events. The former is unrealistic in a biological regime due to the small size (and thereby limited number of ligand-gated ion channels) of single synapses. The latter, however, is quite common in cortex, where individual neurons can have on the order of 10^4 presynaptic partners, each of them firing, on average, at several Hz.

high total input firing rates

In a first approximation, we can model this scenario by stimulating a neuron with two spike sources, an excitatory and an inhibitory one, each of them representing the combined firing of all the excitatory and inhibitory presynaptic partners, respectively (Figure 4.2). For simplicity, their firing rates are set as equal and their weights are scaled with $|E_x^{\rm rev} - E_{\rm l}|$ in order to keep their PSPs approximately equal. As we ramp up the

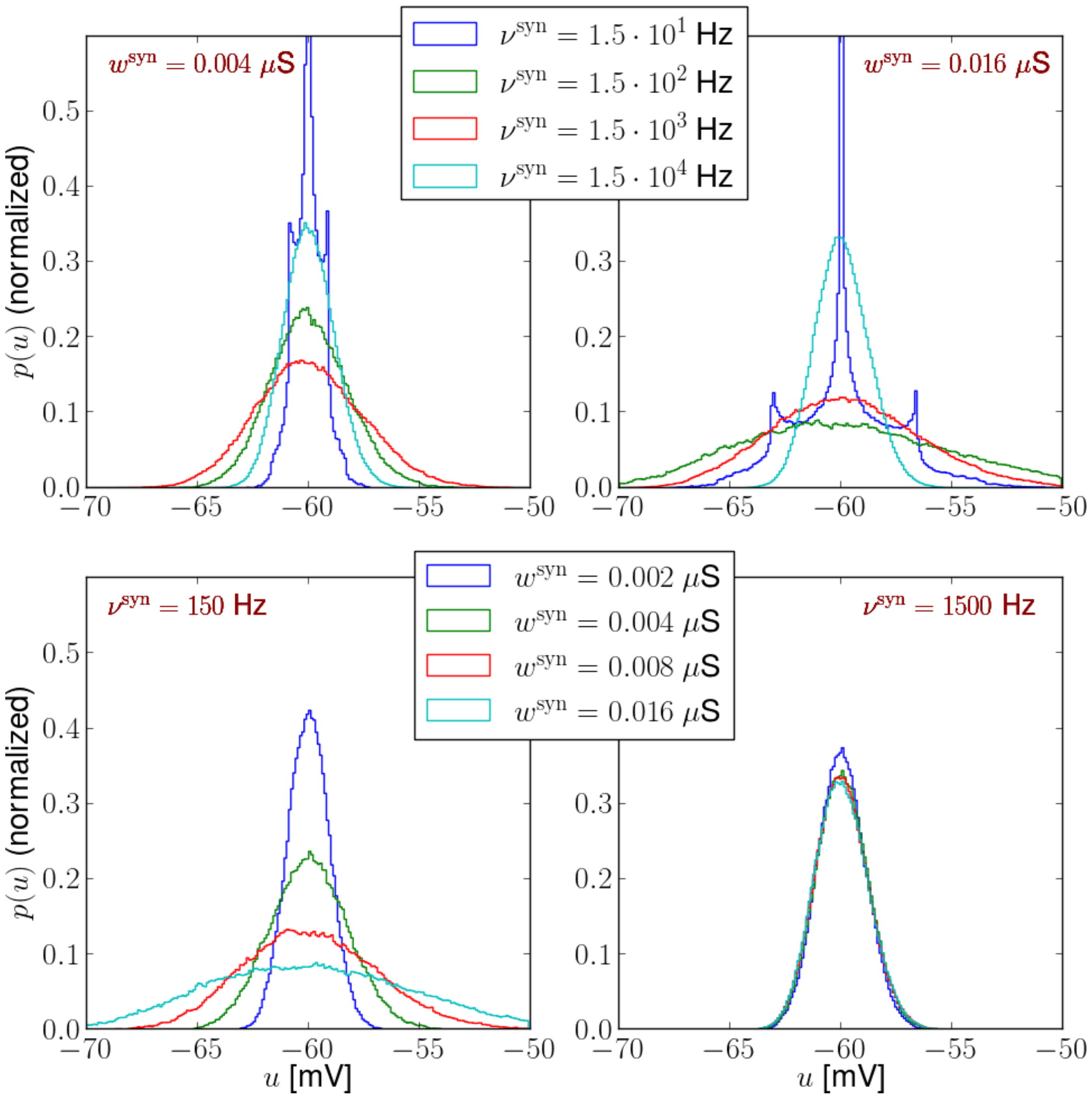


Figure 4.2.: Membrane potential statistics for different synaptic conductance regimes (simulation). A single COBA LIF neuron is stimulated by an excitatory and an inhibitory Poisson spike source with identical firing rates. Their synaptic weights are normalized to their respective reversal potentials in order to keep the process symmetric. The upper two plots show how the membrane potential distribution first broadens and then narrows down again as the input firing rates are increased. The lower two plots show the dependence of the membrane potential distribution on the synaptic weight of the inputs: at lower rates (bottom left), the distribution broadens, as expected, while at higher rates, the synaptic weight has nearly no effect on the width of the distribution. One can already observe that for all but the lowest input rates, the membrane potential distribution can be well approximated by a Gaussian, which will be explained in Section 4.3.

total synaptic conductance by increasing the firing rates of the two sources, an interesting and somewhat surprising effect occurs. At first, the width of the membrane potential distribution broadens, as one would expect from a stronger stimulation. However, as the firing rate increases further, the distribution becomes narrower once again. On the other hand, at low firing rates, an increase of the synaptic weight results in a broadening of the membrane potential distribution, again as expected. For high firing rates however, the synaptic weight appears to have no effect on the distribution.

We shall see later how these effects can be understood in the framework of an analytical description of the synaptic conductance and membrane potential statistics. For now, we can restrict ourselves to the phenomenological observation of two properties of the HCS that appear in the regime of high firing rates:

1) the membrane potential distribution becomes much narrower than the distance towards the reversal potentials and

narrow u-distribution

2) the membrane potential distribution is not (or only weakly) affected by the synaptic weights.

weak w^{syn}-dependence of u-distribution

From the narrowness of the membrane potential distribution, we can deduce that

3) the relative fluctuations of the total membrane conductance are very small,

small relative fluctuations of g^{tot}

since large conductance fluctuations would naturally result in large membrane potential fluctuations.

We can now use these three observations to derive an approximative closed-form solution for the COBA LIF equation which holds in the HCS in the same way we were able to do for CUBA LIF neurons in general.

4.2.4. Analytical Solutions for the LIF Equations: COBA Synapses

Intuitively speaking, the difficulty of finding a closed-form solution to the COBA LIF equation has two reasons: the dependence of the PSP amplitude on the momentary value of the membrane potential (e.g., saturation as u approaches E_x^{rev}) and the fluctuating nature of the total conductance which leads to a non-constant effective (membrane) time constant. In the HCS, these problems can be circumvented due to the narrow membrane potential distribution, which leaves the distance $|E_x^{\text{rev}} - E_\text{l}|$ approximately constant, and the small relative fluctuations of the total conductance, which render τ_{eff} approximately constant as well. (In Section 4.2.2, we have given a more technical argument, which we can now resolve in the HCS as well.)

However, the substitutions are not quite trivial, since if we simply replaced u and g^{tot} with their average values, we would have no membrane dynamics left. We will therefore treat the problem as perturbative and start by considering the effect of a single additional single spike on u_{eff} (Equation 4.30) coming from the jth synapse:

perturbative approximation

$$\Delta_j u_{\text{eff}}(t) = u_{\text{eff}}'(t) - u_{\text{eff}}(t) \tag{4.43}$$

$$= \frac{g_\text{l} E_\text{l} + I^{\text{ext}} + \sum_k g_k^{\text{syn}}(t) E_k^{\text{rev}} + \epsilon_j(t) E_i^{\text{rev}}}{g^{\text{tot}}(t) + \epsilon_j(t)} \tag{4.44}$$

$$- \frac{g_\text{l} E_\text{l} + I^{\text{ext}} + \sum_k g_k^{\text{syn}}(t) E_k^{\text{rev}}}{g^{\text{tot}}(t)} \quad , \tag{4.45}$$

where ${u_{\text{eff}}}'(t)$ denotes the time course of the membrane potential with the additional spike and $\epsilon_j(t)$ represents the time course of the additional PSC. We have also made time dependencies explicit in order to preempt any ambiguity when we later make approximations of the sort $f(t) \approx \langle f \rangle$. Since we assume the amplitude of $\epsilon_j(t)$ to be small with respect to $g^{\text{tot}}(t)$, we can expand ${u_{\text{eff}}}'(t)$ in $\epsilon_j(t)$ up to first order:

$$
\begin{aligned}
{u_{\text{eff}}}'(t) &\approx \frac{g_{\text{l}} E_{\text{l}} + I^{\text{ext}} + \sum\limits_k g_k^{\text{syn}}(t) E_k^{\text{rev}}}{g^{\text{tot}}(t)} && (4.46)\\
&+ \epsilon_j(t) \frac{g^{\text{tot}}(t) E_j^{\text{rev}} - \left[g_{\text{l}} E_{\text{l}} + I^{\text{ext}} + \sum\limits_k g_k^{\text{syn}}(t) E_k^{\text{rev}} \right]}{\left[g^{\text{tot}}\right]^2(t)} && (4.47)\\
&= u_{\text{eff}}(t) + \frac{\epsilon_j(t) \left[E_j^{\text{rev}} - u_{\text{eff}}(t) \right]}{g^{\text{tot}}(t)} \quad , && (4.48)
\end{aligned}
$$

leaving us with the perturbative contribution of a single PSC to the total effective membrane potential:

$$
\Delta_j u_{\text{eff}}(t) \approx \frac{\epsilon_j(t) \left[E_j^{\text{rev}} - u_{\text{eff}}(t) \right]}{g^{\text{tot}}(t)} \quad . \tag{4.49}
$$

We can now write the effective membrane potential as an offset value plus the sum of all synaptic perturbations:

$$
\begin{aligned}
u_{\text{eff}}(t) &\approx {u_{\text{eff}}}^0 + \sum_{\text{syn } k} \Delta_k u_{\text{eff}}(t) && (4.50)\\
&= {u_{\text{eff}}}^0 + \frac{\sum\limits_k g_k^{\text{syn}}(t) \left[E_k^{\text{rev}} - u_{\text{eff}}(t) \right]}{g^{\text{tot}}(t)} \quad , && (4.51)
\end{aligned}
$$

where we have returned to our previous notation

$$
g_k^{\text{syn}}(t) = \sum_{\text{spk } s} \Theta(t - t_s) \epsilon_j(t - t_s) \quad . \tag{4.52}
$$

Finally, we can use our previous observations of the small fluctuations of u_{eff} and g^{tot} for replacing them with their average values, leaving us with

$$
u_{\text{eff}}(t) = {u_{\text{eff}}}^0 + \frac{\sum\limits_k g_k^{\text{syn}}(t) \left(E_k^{\text{rev}} - \langle u_{\text{eff}} \rangle \right)}{\langle g^{\text{tot}} \rangle} \quad . \tag{4.53}
$$

This approximation effectively solves the problem of finding a closed-form solution to Equation 4.36. Since the total conductance can be assumed as approximately constant in the HCS, the effective membrane time constant is no longer explicitly a function of time

$$
\tau_{\text{eff}} \approx \langle \tau_{\text{eff}} \rangle = \frac{C_{\text{m}}}{\langle g^{\text{tot}} \rangle} \quad , \tag{4.54}
$$

so the third integrand in Equation 4.36 can be treated just like in the CUBA case. By performing the following replacements in Equation 4.37:

$$g_\mathrm{l} \to \left\langle g^\mathrm{tot} \right\rangle \quad , \tag{4.55}$$

$$\tau_\mathrm{m} \to \langle \tau_\mathrm{eff} \rangle \quad \text{and} \tag{4.56}$$

$$\frac{g_\mathrm{l} E_\mathrm{l} + I^\mathrm{ext} + \sum_k i_k^\mathrm{syn}}{g_\mathrm{l}} \to {u_\mathrm{eff}}^0 + \frac{\sum_k g_k^\mathrm{syn} \left(E_k^\mathrm{rev} - \langle u_\mathrm{eff} \rangle \right)}{g^\mathrm{tot}} \text{ (from Eqn. 4.53) ,} \tag{4.57}$$

membrane potential of an LIF neuron with exponential synaptic conductances and PSP shape in the HCS

we can obtain a solution that is analogous to Equation 4.38 for COBA LIF neurons in the HCS:

$$u(t) = {u_\mathrm{eff}}^0 + \sum_{\mathrm{syn}\ k} \sum_{\mathrm{spk}\ s} \frac{\tau_k^\mathrm{syn} w_k \left(E_k^\mathrm{rev} - \langle u_\mathrm{eff} \rangle \right)}{\langle g^\mathrm{tot} \rangle \left(\tau_k^\mathrm{syn} - \langle \tau_\mathrm{eff} \rangle \right)} \Theta(t - t_s) \left[\exp\left(-\frac{t - t_s}{\tau_k^\mathrm{syn}} \right) - \exp\left(-\frac{t - t_s}{\langle \tau_\mathrm{eff} \rangle} \right) \right] , \tag{4.58}$$

with the shape of a single PSP being again a DOE:

$$\mathrm{PSP}(t) = \frac{\tau_k^\mathrm{syn} \langle \tau_\mathrm{eff} \rangle w_k \left(E_k^\mathrm{rev} - \langle u_\mathrm{eff} \rangle \right)}{C_\mathrm{m} \left(\tau_k^\mathrm{syn} - \langle \tau_\mathrm{eff} \rangle \right)} \Theta(t - t_s) \left[\exp\left(-\frac{t - t_s}{\tau_k^\mathrm{syn}} \right) - \exp\left(-\frac{t - t_s}{\langle \tau_\mathrm{eff} \rangle} \right) \right] . \tag{4.59}$$

For now, we are still missing closed-form expressions for ${u_\mathrm{eff}}^0$, $\langle u_\mathrm{eff} \rangle$ and $\langle \tau_\mathrm{eff} \rangle$ in Equations 4.58 and 4.59. We shall first derive these in Section 4.3, as well as in Section 6.5.2 within the framework of an entirely different formalism. Let us, however, anticipate Equations 4.104 and 4.105 in order to demonstrate the validity of our closed-form expressions for the CUBA and COBA LIF neurons.

In Figure 4.3, we compare the theoretical prediction of the membrane potential time course (Equations 4.38 and 4.58) with the results from NEST simulations in the limits of low- and high-frequency Poisson input. Both the CUBA and the COBA neurons were stimulated with identical Poisson trains, one excitatory and one inhibitory. As expected, the CUBA prediction is exact, since it is possible to integrate the membrane potential equation analytically. For small input rates and weights, the COBA neuron behaves almost identically to the CUBA one, since the total conductance, as well as the distance towards the reversal potentials does not change significantly. For the HCS, where we were able to formulate an analytical expression for the COBA membrane as well, the prediction lies neatly on top of the simulation data, thus validating our approach. Note how the COBA neuron behaves significantly differently from the CUBA one, despite them receiving identical input. Not only is the membrane potential distribution narrower, as already discussed in Section 4.2.3, but it also has a different time course (not being just a "vertically compressed" version of the CUBA membrane potential time course). This is a consequence of the CUBA neuron reacting to stimuli on much faster time scales ($\tau_\mathrm{eff} \ll \tau_\mathrm{m}$).

Before moving on, we need to point out what may at first appear as a contradiction of our premises. Note how the synaptic weight enters Equation 4.59 multiplicatively. This might look like a potential invalidation of our perturbative argument: if one would increase all synaptic weights, surely all PSPs would become larger and accumulate towards larger membrane potential fluctuations. This is, in fact, not necessarily the case in the HCS, as we have observed experimentally in the previous section: at high enough input

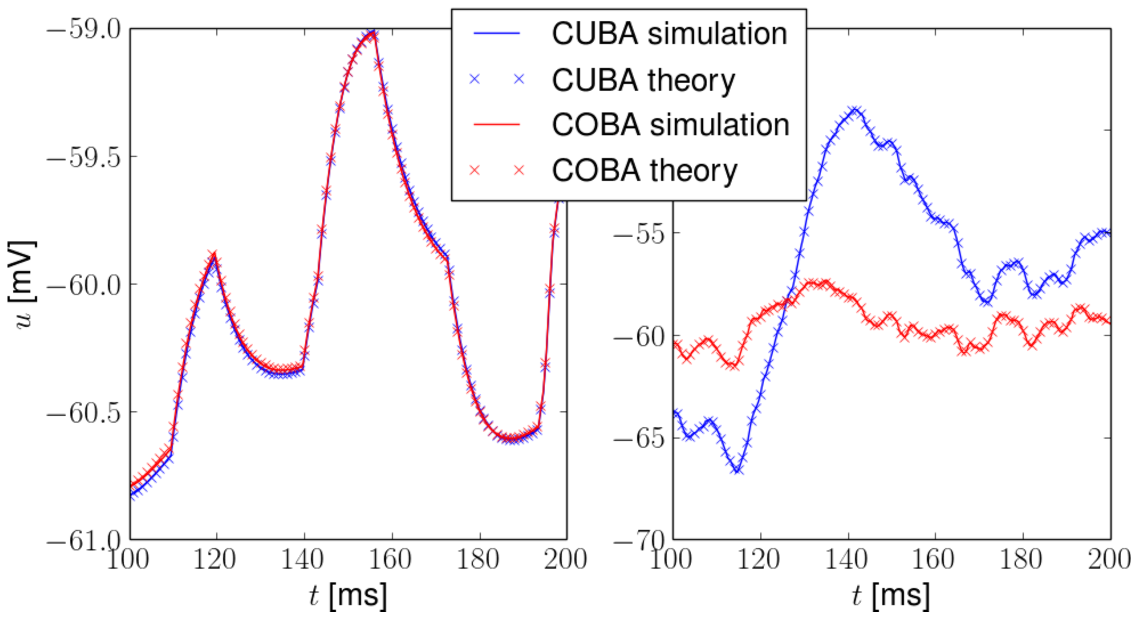


Figure 4.3.: CUBA and COBA membrane potentials: analytical prediction vs. simulation data (NEST). Both neurons received identical Poisson input of frequency ν from an excitatory and an inhibitory source, respectively. **Left:** low-frequency input ($\nu = 40$ Hz). The two neuron models behave nearly identically, since the weak input does not affect the total conductance and distance towards the reversal potentials significantly. **Right:** high-frequency input ($\nu = 5000$ Hz). The COBA membrane has a smaller dynamic range, due to the $\langle g^{\mathrm{tot}} \rangle$ in the denominator of Equation 4.58. Its time course is also different, due to a modified PSP shape ($\tau_{\mathrm{eff}} << \tau_{\mathrm{m}}$). In all of the depicted cases, the theoretical prediction agrees very well with the simulation results.

rates, the width of the membrane potential distribution becomes independent of the individual synaptic weights. Intuitively speaking, larger synaptic weights lead to a larger total conductance and thereby to a smaller τ_{eff}, which also enters the PSP amplitude multiplicatively. It turns out that these two effects cancel out at high firing rates, a formal proof of which will be given in Section 4.3.5.

In summary, at this point, we have a full analytical description of the membrane potential and PSPs for CUBA LIF neurons in general and COBA neurons in the HCS (under the assumption of exponential synaptic interaction kernels); for the latter however, we are still missing closed-form expressions for $\langle u_{\mathrm{eff}} \rangle$ and $\langle \tau_{\mathrm{eff}} \rangle$. We shall derive these in Section 4.3. We can, however, already understand several properties of the HCS from the form of Equations 4.58 and 4.59.

4.2.5. The High-Conductance State II: PSP Shapes

The most apparent property of the HCS was already mentioned earlier, but now follows directly from the equation describing individual PSPs. Similarly to the CUBA PSP, the COBA PSP is a DOE where the membrane time constant $\tau_{\mathrm{m}} = C_{\mathrm{m}}/g_{\mathrm{l}}$ has been replaced by an effective time constant $\tau_{\mathrm{eff}} = C_{\mathrm{m}}/g^{\mathrm{tot}}$. Since the HCS is defined by a large g^{tot}, the membrane time constant becomes very small, allowing the cell to react very quickly to incoming stimuli. This already becomes apparent when considering Equation 4.31 in the limit of vanishing τ_eff: the membrane potential becomes approximately equal to the effective membrane potential, which is a linear transformation of the total synaptic conductance (Equation 4.30).

fast membrane

For the PSP shape, this can entail a reversal of the role of the membrane and synaptic time constants. Note, again, the symmetry of Equations 4.39 and 4.59 in τ^syn and τ_m or τ_eff, respectively: it is always the smaller of the two time constants which determines the rising flank of the PSP. Typically, $\tau_\text{m} > \tau^\text{syn}$, so the rising flank of a (CUBA) PSP is determined by τ^syn and the falling flank by τ_m. In the HCS, these roles can be reversed, with the small τ_eff causing a nearly instantaneous rise of the PSP to its maximum value.

nearly exponential PSPs

In addition to the "more exponential" shape of the PSPs, the HCS also causes them to shrink, due to $\langle\tau_\text{eff}\rangle$ entering the PSP equation multiplicatively. This is a nice confirmation of the small relative fluctuations of the membrane potential distribution discussed in Section 4.2.3.

small PSPs

What we are still missing is a formal understanding of the somewhat counterintuitive relationships between the input firing rate, the synaptic weight and the membrane potential distribution (Figure 4.2). For this, we shall now investigate the statistical properties of additive Poisson processes.

4.3. Single-Neuron Statistics

In this section, we begin by deriving general closed-form expressions the distribution of Poisson-driven dynamic variables. Using the results from the previous section, we can then find quantitative expressions for the distributions of the synaptic input and the membrane potential of both CUBA and COBA LIF neurons. These will also allow us to explain the interesting "anomalies" found in the computer simulations of COBA neurons from the previous section.

4.3.1. Statistics of Additive Poisson-Driven Processes

We begin by addressing the general problem of a dynamic variable controlled by a Poisson point process. Later on, the equations we derive here can be applied to different intrinsic neuronal variables, such as membrane potentials, input currents, synaptic conductances etc.

Consider a random variable $Y(T) \in \mathbb{R}$. Let its evolution in time be determined by a Poisson process with rate λ that triggers events at times t_i, which cause changes in $Y(t)$, given by some kernel $\kappa(t - t_i)$, that sum up linearly. Our goal is to find the first two moments of the distribution of Y, i.e. its mean $E[Y]$ and variance $\mathrm{Var}[Y]$.

Note how we have explicitly dropped the time dependence in the characterization of the distribution of Y. This is neither an omission nor trivial, but rather a consequence of the ergodicity of the dynamics of Y. We will address the ergodicity of Markov processes in a more formal fashion in Section 4.1. For now, it is only important to know that the Poisson process is ergodic - and by extension also $Y(t)$, since it is fully determined by a Poisson process. The distribution of $Y(t)$ over time is therefore equivalent to the distribution of $Y(T)$ at a fixed time T over all its possible histories ("distribution over trials"). We will make use of this property in our formal derivation of $E[Y]$ and $\mathrm{Var}[Y]$, which is easier to perform in a trial-based setting.

Let us start by considering a time interval $t \in [0, T)$. From the linearity of changes in $Y(t)$, it follows that

$$Y(T) = \sum_{0 \leq t_i < T} \kappa(T - t_i) \quad . \tag{4.60}$$

Assuming that exactly n events have taken place in $[0, T)$, it follows from the definition of the Poisson process that each event has an equal probability of occurring at any time $t \in [0, T)$:

$$p_n(t_i) := p(t_i | n \text{ events}) = \frac{1}{T} \quad . \tag{4.61}$$

Since events generated by a Poisson process are, also by definition, independent, the times t_i follow a multivariate uniform distribution:

$$p_n(t_1, \ldots, t_n) := p(t_1, \ldots, t_n | n \text{ events}) = \prod_{i=1}^{n} p_n(t_i) = \frac{1}{T^n} \quad . \tag{4.62}$$

The probability for the Poisson process to produce n events in $[0, T)$ is

$$p_\lambda(n) = \frac{e^{-\lambda \cdot T} \cdot (\lambda \cdot T)^n}{n!} . \tag{4.63}$$

So we can now write, using Bayes' rule:

$$p(n, t_1, \ldots, t_n) = p_n(t_1, \ldots, t_n|n) p_\lambda(n) = \frac{\lambda^n}{n!} e^{-\lambda T} \tag{4.64}$$

With Equations 4.64 and 4.60, it is now possible to calculate the moments of $Y(T)$. In the limit of $T \to \infty$, these become equivalent to the moments of Y.

From the definition of the expectation value, it follows that

$$\begin{aligned} E[Y(T)] &= \sum_{n=0}^{\infty} \int_0^T dt_1 \ldots \int_0^T dt_n \; Y(T, n, t_1, \ldots, t_n) \; p(n, t_1, \ldots, t_n) \\ &= \sum_{n=0}^{\infty} \int_0^T dt_1 \ldots \int_0^T dt_n \; \sum_{i=1}^{n} \kappa(T - t_i) \frac{\lambda^n}{n!} e^{-\lambda T} \end{aligned} \tag{4.65}$$

We can rearrange the terms depending on their running variables and use the linearity of the integral operator ($\int \sum = \sum \int$) to obtain

$$E[Y(T)] = e^{-\lambda T} \sum_{n=0}^{\infty} \frac{\lambda^n}{n!} \sum_{i=1}^{n} \int_0^T dt_1 \ldots \int_0^T dt_n \; \kappa(T - t_i) \quad . \tag{4.66}$$

For any $i \in \{1, \ldots, n\}$, the integrand $\kappa(T - t_i)$ only depends on a single running variable t_i, so it factors out of all but one of the integrals:

$$\begin{aligned} \int_0^T dt_1 \ldots \int_0^T dt_n \; \kappa(T - t_i) &= \underbrace{\int_0^T dt_1 \ldots \int_0^T dt_{i-1} \int_0^T dt_{i+1} \ldots \int_0^T dt_n}_{n-1 \text{ independent variables}} \int_0^T dt_i \; \kappa(T - t_i) \\ &= T^{n-1} \int_0^T dt \; \kappa(T - t) \quad , \end{aligned} \tag{4.67}$$

leaving us with

$$\begin{aligned} E[Y(T)] &= e^{-\lambda T} \sum_{n=0}^{\infty} \frac{\lambda^n}{n!} \sum_{i=1}^{n} T^{n-1} \int_0^T dt \; \kappa(T - t) \\ &= \lambda e^{-\lambda T} \underbrace{\sum_{n=0}^{\infty} \frac{(\lambda T)^{n-1}}{(n-1)!}}_{e^{\lambda T}} \int_0^T dt \; \kappa(T - t) \\ &= \lambda \int_0^T dt \; \kappa(T - t) \\ &\overset{(\tau = T - t)}{=} \lambda \int_0^T d\tau \; \kappa(\tau) \end{aligned} \tag{4.68}$$

expectation value of a Poisson-driven linear process

In the limit of $T \to \infty$, we obtain the desired expectation value of Y:

$$E[Y] = \lambda \int_0^\infty d\tau \; \kappa(\tau) \quad . \tag{4.69}$$

For calculating the variance of Y, we can use the univariate case of relation 4.16:

$$Var[Y] = E[(Y - E[Y])^2] = E[Y^2] - E^2[Y] \quad . \tag{4.70}$$

With Equation 4.69, the second term can be easily written down as

$$E^2[Y] = \lambda^2 \left[\int_0^\infty d\tau \; \kappa(\tau) \right]^2 \tag{4.71}$$

For the first term, we can use the definition of the expectation value and find, equivalently to 4.65 and 4.66:

$$\begin{aligned} E[Y^2(T)] &= \sum_{n=0}^{\infty} \int_0^T dt_1 \ldots \int_0^T dt_n \; Y^2(T, n, t_1, \ldots, t_n) \; p(n, t_1, \ldots, t_n) \\ &= \sum_{n=0}^{\infty} \int_0^T dt_1 \ldots \int_0^T dt_n \; \left[\sum_{i=1}^{n} \kappa(T - t_i) \right]^2 \frac{\lambda^n}{n!} e^{-\lambda T} \\ &= e^{-\lambda T} \sum_{n=0}^{\infty} \frac{\lambda^n}{n!} \int_0^T dt_1 \ldots \int_0^T dt_n \; \left[\sum_{i=1}^{n} \kappa(T - t_i) \right]^2 \quad . \end{aligned} \tag{4.72}$$

By exlicitly squaring the sum we obtain

$$\left[\sum_{i=1}^{n} \kappa(T - t_i) \right]^2 = \sum_{i=1}^{n} \kappa^2(T - t_i) + \sum_{j=1}^{n} \sum_{k=1}^{n} \kappa(T - t_j) \kappa(T - t_k) \tag{4.73}$$

We can now plug this back into the multiple integral in the expression for $E[Y^2(T)]$ and, for any triple (i, j, k), perform a similar factorization as in Equation 4.67 to obtain

$$
\begin{aligned}
&\int_0^T dt_1 \ldots \int_0^T dt_n \left[\sum_{i=1}^{n} \kappa(T-t_i)\right]^2 \\
&= \int_0^T dt_1 \ldots \int_0^T dt_n \left[\sum_{i=1}^{n} \kappa^2(T-t_i) + \sum_{i=1}^{n}\sum_{j=1}^{n} \kappa(T-t_i)\kappa(T-t_j)\right] \\
&= \sum_{i=1}^{n} \underbrace{\int_0^T dt_1 \ldots \int_0^T dt_{i-1} \int_0^T dt_{i+1} \ldots \int_0^T dt_n}_{n-1 \text{ independent variables}} \int_0^T dt_i \, \kappa^2(T-t_i) \\
&\quad + \sum_{j=1}^{n}\sum_{k=1}^{n} \underbrace{\int_0^T dt_1 \ldots \int_0^T dt_{j-1} \int_0^T dt_{j+1} \ldots \int_0^T dt_{k-1} \int_0^T dt_{k+1} \ldots \int_0^T dt_n}_{n-2 \text{ independent variables}} \int_0^T dt_j \int_0^T dt_k \, \kappa(T-t_i)\kappa(T-t_j) \\
&= nT^{n-1} \int_0^T dt \, \kappa^2(T-t) + n^2 T^{n-2} \left[\int_0^T dt \, \kappa(T-t)\right]^2 \quad , \qquad (4.74)
\end{aligned}
$$

which , similarly to Equation 4.68, leaves us with

$$
\begin{aligned}
E[Y^2(T)] &= e^{-\lambda T} \sum_{n=0}^{\infty} \frac{\lambda^n}{n!} \left\{ nT^{n-1} \int_0^T dt \, \kappa^2(T-t) + n^2 T^{n-2} \left[\int_0^T dt \, \kappa(T-t)\right]^2 \right\} \\
&= e^{-\lambda T} \left\{ \lambda \underbrace{\sum_{n=0}^{\infty} \frac{(\lambda T)^{n-1}}{(n-1)!}}_{e^{\lambda T}} \int_0^T dt \, \kappa^2(T-t) + \lambda^2 \underbrace{\sum_{n=0}^{\infty} \frac{(\lambda T)^{n-2}}{(n-2)!}}_{e^{\lambda T}} \left[\int_0^T dt \, \kappa(T-t)\right]^2 \right\} \\
&= \lambda \int_0^T dt \, \kappa^2(T-t) + \lambda^2 \left[\int_0^T dt \, \kappa(T-t)\right]^2 \\
&\overset{(\tau=T-t)}{=} \lambda \int_0^T d\tau \, \kappa^2(\tau) + \lambda^2 \left[\int_0^T d\tau \, \kappa(\tau)\right]^2 \qquad (4.75)
\end{aligned}
$$

In the limit of $T \to \infty$ we therefore obtain

$$
E[Y^2] = \lambda \int_0^\infty d\tau \, \kappa^2(\tau) + \lambda^2 \left[\int_0^\infty d\tau \, \kappa(\tau)\right]^2 \quad . \qquad (4.76)
$$

This can now be plugged, together with the expression for $E^2[Y]$ (Equation 4.71) into *variance of a Poisson-driven linear process*

Equation 4.70 to obtain the sought variance of Y:

$$\mathrm{Var}[Y] = \lambda \int_0^\infty d\tau\ \kappa^2(\tau) \quad . \tag{4.77}$$

We complement the results of this formally rigorous, trial-based approach by a less formal but more intuitive picture, in which we consider the evolution of Y over time.

Consider a time window $[0, T)$ which is much larger than the duration[4] of a single kernel κ. Assume that Y is stimulated by a single event at some random time t_i. The time-average of Y over $[0, T)$ is given by

$$\begin{aligned} E[Y_i] &= E[\kappa(t - t_i)] \\ &= \frac{1}{T} \int_0^T dt\ \kappa(t - t_i) \quad . \end{aligned} \tag{4.78}$$

For N events, the expectation value becomes

$$\begin{aligned} E[Y_N] &= E\left[\sum_{i=1}^{N} Y_i\right] \\ &\overset{(4.17)}{=} \sum_{i=1}^{N} E\left[\kappa(t - t_i)\right] \\ &\overset{(4.78)}{=} \frac{N}{T} \int_0^T dt\ \kappa(t - t_i) \quad . \end{aligned} \tag{4.79}$$

On average, a Poisson process with rate λ produces $E[N] = \lambda T$ events in a time T. By plugging this into Equation 4.79, performing an appropriate change of variables and taking the limit $T \to \infty$, we recover the result from Equation 4.69:

$$E[Y] = \lambda \int_0^\infty d\tau\ \kappa(\tau) \quad . \tag{4.80}$$

Similarly, for the time-variance of a single kernel, we have

$$\begin{aligned} \mathrm{Var}[Y_i] &\overset{(4.16)}{=} E[Y_i^2] - E^2[Y_i] \\ &= \frac{1}{T} \int_0^T dt\ \kappa^2(t - t_i) - \frac{1}{T^2} \left[\int_0^T dt\ \kappa(t - t_i)\right]^2 \end{aligned} \tag{4.81}$$

[4] We loosely define the duration of a kernel as the time interval over which it is significantly different from 0. For example, we could choose the duration of $\kappa(t)$ as the length $\Delta t = t_2 - t_1$ of the shortest interval $[t_1, t_2)$ with the property that $\int_{t_1}^{t_2} \kappa(t)\, dt > (1 - \epsilon) \int_{-\infty}^{\infty} |\kappa(t)|\, dt$ for some predefined "allowed relative error" ϵ.

In a Poisson process, events arrive independently of each other, so $\mathrm{Cov}[Y_i, Y_j] = \delta_{ij}\mathrm{Var}[Y_i]$. Therefore,

$$\begin{aligned}\mathrm{Var}[Y_N] &= \mathrm{Var}\left[\sum_{i=1}^{N} Y_i\right] \\ &\overset{(4.18)}{=} \sum_{i=1}^{N} \mathrm{Var}[Y_i] \\ &\overset{(4.81)}{=} \frac{N}{T}\int_0^T dt\ \kappa^2(t-t_i) - \frac{N}{T^2}\left[\int_0^T dt\ \kappa(t-t_i)\right]^2 \end{aligned} \tag{4.82}$$

As above, we can now replace N with $E[N] = \lambda T$, perform a change of variables and let $T \to \infty$ (whereby the second term vanishes) to recover the result from Equation 4.77:

$$\mathrm{Var}[Y] = \lambda \int_0^\infty d\tau\ \kappa^2(\tau) \quad . \tag{4.83}$$

Note how we have arrived independently at the same results using first the distribution over trials and then the distribution over time - as expected given the ergodicity of Y. We can therefore use the derived moments of the distribution in both contexts, whether we discuss large populations of stochastically equivalent Y-type RVs or long measurements of an individual Y. Finally, due to the linearity of Y, Equations 4.69 and 4.77 can be easily generalized to stimulation by n independent Poisson processes:

mean and variance of a superposition of Poisson processes

$$E[Y] = \sum_{k=1}^{n} \lambda_k \int_0^\infty d\tau\ \kappa_i(\tau) \tag{4.84}$$

$$\mathrm{Var}[Y] = \sum_{k=1}^{n} \lambda_k \int_0^\infty d\tau\ \kappa_i^2(\tau) \quad . \tag{4.85}$$

4.3.2. The Gaussian Approximation

In the previous section, we have calculated the first two moments of the distribution of an additive Poisson process, but have made no statement about the shape of the distribution itself. This could be done by calculating all central moments up to some order, which would (almost) uniquely determine the distribution. However, we will provide a more intuitive argument here and follow up with a formal proof in Section 6.5.2.

Consider, again, the value of Y at the end of the time window $[0, T)$ from the last paragraph. Each event causes a change in $Y(T)$ that can be considered as an RV of its own:

$$Y_i := Y(T \mid \text{a single event has occurred at } t_i) \quad . \tag{4.86}$$

As the distribution of t_i is flat (due to the events being generated by a Poisson process) and the kernel κ does not diverge in either direction, we can assume that the PDF of Y_i

does not have any particularly nasty characteristics.[5] Due to the Poisson nature of the generating process, the Y_i-RVs are independent and identically distributed (IID). With our definition of Y_i, $Y(T)$ can now be written as a sum over our newly defined RVs

$$Y(T) = \sum_{i=1}^{n} Y_i \quad . \tag{4.89}$$

central limit theorem

The central limit theorem (CLT) of probability theory guarantees that a sum of n IID RVs, each with moments μ and σ, converges almost surely to a Gaussian in the limit of large enough n ($n \to \infty$). For a high Poisson event rate λ, the number of events $n = \lambda T$ can become arbitrarily large as well and therefore CLT applies to $Y(T)$, rendering $p(Y(T)|n)$ Gaussian:

$$p(Y(T) = y|n) \xrightarrow{n\to\infty} f_{\mathcal{N}}(y, n\mu, n\sigma^2) \quad . \tag{4.90}$$

Gaussian approximation

The number n of events itself follows a Poisson distribution (Equation 4.63), which in the same limit of large λT can be well approximated by a Gaussian distribution with mean and variance both equal to λT:

$$p_\lambda(n) \xrightarrow{n\to\infty} f_{\mathcal{N}}(n, \lambda T, \lambda T) \quad . \tag{4.91}$$

With Bayes' rule, we can now write

$$p(Y(T)) = \int_n p(Y(T), n)\, dn = \int_n p(Y(T)|n) p_\lambda(n)\, dn \tag{4.92}$$

Such an integral over a product of Gaussians can be shown to yield a Gaussian as well (Bishop, 2009). As in the previous section, we can now invoke the ergodicity of Y to argue that the distribution of $Y(T)$ over mutliple trials (which we have discussed above) is equivalent to its distribution over time. We can therefore conclude that if the Poisson rate is high enough, the distribution of any additive Poisson process is Gaussian, with a mean and variance given by Equations 4.84 and 4.85. As we now have a full statistical description of additive Poisson processes, we can move on to predict the distribution of input currents, conductances and membrane potentials of the previously discussed LIF neuron models.

4.3.3. Current and Conductance Statistics

Under the assumption of identically shaped PSP kernels, the Poisson-driven synaptic input of CUBA and COBA neurons looks the same, up to the synaptic weight - a multiplicative

[5] For a monotonically decreasing κ we can calculate the CDF of Y_i:

$$p(Y_i < y) = p\left(\kappa(T - t_i) \le y\right) = p(t_i \le T - \kappa^{-1}(y)) = 1 - \frac{\kappa^{-1}(y)}{T} \quad , \tag{4.87}$$

and from there its PDF:

$$p(Y_i = y) = \frac{\partial}{\partial y} p(Y_i < y) = \frac{1}{T}\frac{\partial}{\partial y}\kappa^{-1}(y) \quad . \tag{4.88}$$

For non-monotonic kernels, the support of κ can simply be partitioned into a set of intervals on each of which κ is monotonic and the CDF becomes a sum of terms that have the same form as the one in Equation 4.87.

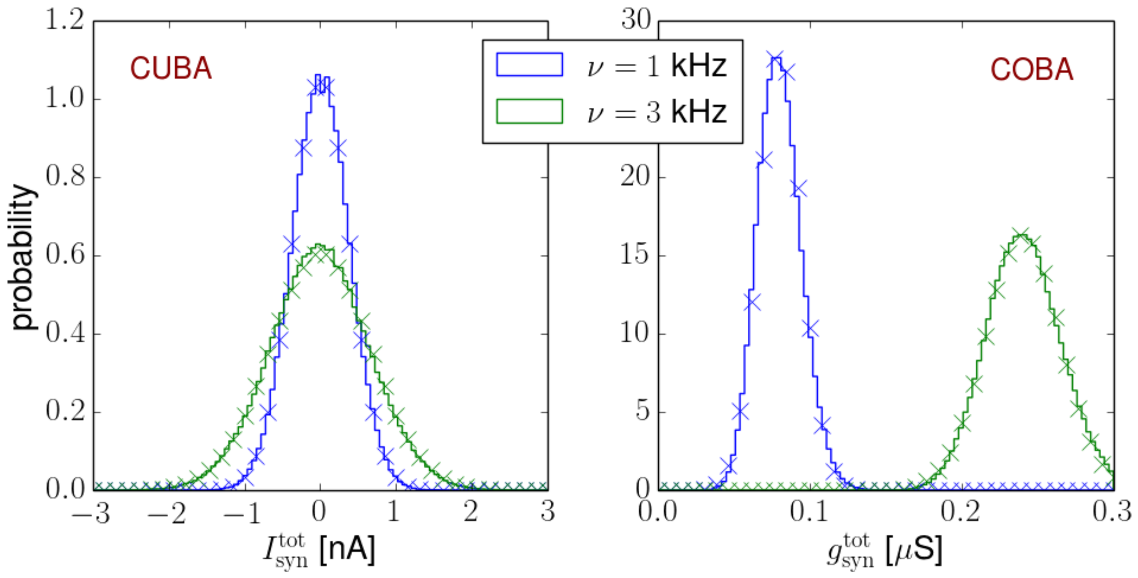


Figure 4.4.: Probability distributions of Poisson-driven input current and conductance: theoretical prediction vs. simulation results (NEST). Excitation and inhibition were chosen to be balanced (identical firing rates and identical-amplitude but opposite-sign PSPs; for the COBA neuron, this implied a rescaling of the synaptic weights with $E_x^{\mathrm{rev}} - E_{\mathrm{l}}$). The average total input is therefore zero for the CUBA neuron, but nonzero in the COBA case, as conductances are, by definition, non-negative. An increased firing rate (while maintaining the balance) only results in a broader distribution of the total synaptic input for the CUBA neuron. In the COBA case, both the mean and the variance of the distribution increase. The increase of the mean total synaptic input is essential for achieving a HCS. Altogether, the theoretical prediction is in excellent agreement with the simulation data.

factor typically expressed in nA or μS, respectively (compare Equations 2.55 and 2.58). For exponential synapses, the interaction kernel is given by

$$\kappa(t) = w_k \Theta(t) \exp(-\frac{t}{\tau_k^{\mathrm{syn}}}) \quad . \tag{4.93}$$

We can now use Equations 4.84 and 4.85 to obtain the mean and variance of the total (Poissonian) synaptic input:

mean and variance of the synaptic input

$$E[f^{\mathrm{syn}}] = \sum_{k=1}^{n} \nu_k \int_0^\infty w_k \Theta(t) \exp(-\frac{t}{\tau_k^{\mathrm{syn}}})\, dt = \sum_{k=1}^{n} w_k \nu_k \tau_k^{\mathrm{syn}} \quad , \tag{4.94}$$

$$\mathrm{Var}[f^{\mathrm{syn}}] = \sum_{k=1}^{n} \nu_k \int_0^\infty \left[w_k \Theta(t) \exp(-\frac{t}{\tau_k^{\mathrm{syn}}}) \right]^2 dt = \sum_{k=1}^{n} \frac{1}{2} w_k^2 \nu_k \tau_k^{\mathrm{syn}} \quad , \tag{4.95}$$

where f^{syn} represents the total synaptic current or conductance and ν_k the input frequency at the kth synapse. Figure 4.4 shows a comparison of the predicted distributions to simulation data. Note how opposing currents (excitatory and inhibitory) may cancel out in the CUBA model, as, e.g., reflected by the zero mean current in the figure. This can not happen in the COBA case, in which all conductances are (by definition) strictly non-negative. This "feature" of conductances is what enables the HCS in the first place.

Before moving on to membrane potential distributions, we need to point out a particular feature of the input distributions. As one would already expect from the invocation of the CLT in the previous section, the width and mean of the input distributions must increase as the number of PSC kernels in an interval $[0, T)$ increases, but their ratio (or relative width) must decrease. Indeed, as the abovementioned "kernel frequency" is actually the Poisson rate ν, both phenomena follow directly from Equations 4.94 and 4.95. In particular, for a single input source, the width-to-mean-ratio takes the form

width-to-mean ratio of synaptic input

$$\left[\frac{\sigma}{\mu}\right]_{f^{\mathrm{syn}}} = \frac{1}{\sqrt{2\nu_k \tau_k^{\mathrm{syn}}}} \quad , \tag{4.96}$$

which converges to 0 as ν_k increases. In particular, this validates the assertion we made in Section 4.2.3 about the relative fluctuations of the total membrane conductance being very small. We will make use of this hereby rigorously derived property of the HCS in the following section.

4.3.4. Free Membrane Potential Statistics

We now turn to the distribution of the membrane potential of Poisson-driven LIF neurons with exponential synapses. In particular, we are interested in the free membrane potential, i.e., the firing threshold θ is set high enough to prevent the neuron from spiking. Again, we can use the mean and variance Equations 4.84 and 4.85 and apply them to the CUBA (Equation 4.38/4.39) and COBA (Equation 4.58/4.59) equations/PSP kernels.

For CUBA neurons, the calculation is straightforward. For the expectation value of the membrane potential, we obtain

expectation value of the CUBA membrane potential

$$\begin{aligned} E[u] &= E_\mathrm{l} + \frac{I^{\mathrm{ext}}}{g_\mathrm{l}} + \sum_{k=1}^{n} \nu_k \int_0^\infty \frac{\tau_k^{\mathrm{syn}} \tau_\mathrm{m} w_k}{C_\mathrm{m}\left(\tau_k^{\mathrm{syn}} - \tau_\mathrm{m}\right)} \Theta(t) \left[\exp\left(-\frac{t}{\tau_k^{\mathrm{syn}}}\right) - \exp\left(-\frac{t}{\tau_\mathrm{m}}\right)\right] dt \\ &= \dots \\ &= E_\mathrm{l} + \frac{I^{\mathrm{ext}}}{g_\mathrm{l}} + \frac{\sum_{k=1}^{n} w_k \nu_k \tau_k^{\mathrm{syn}}}{g_\mathrm{l}} \quad , \end{aligned} \tag{4.97}$$

and for its variance

variance of the CUBA membrane potential

$$\begin{aligned} \mathrm{Var}[u] &= \sum_{k=1}^{n} \nu_k \int_0^\infty \left\{ \frac{\tau_k^{\mathrm{syn}} \tau_\mathrm{m} w_k}{C_\mathrm{m}\left(\tau_k^{\mathrm{syn}} - \tau_\mathrm{m}\right)} \Theta(t) \left[\exp\left(-\frac{t}{\tau_k^{\mathrm{syn}}}\right) - \exp\left(-\frac{t}{\tau_\mathrm{m}}\right)\right]\right\}^2 dt \\ &= \dots \\ &= \sum_{k=1}^{n} \left[\frac{\tau_\mathrm{m} \tau_k^{\mathrm{syn}}}{C_\mathrm{m}(\tau_\mathrm{m} - \tau_k^{\mathrm{syn}})}\right]^2 w_k^2 \nu_k \left(\frac{\tau_\mathrm{m}}{2} + \frac{\tau_k^{\mathrm{syn}}}{2} - 2\frac{\tau_\mathrm{m}\tau_k^{\mathrm{syn}}}{\tau_\mathrm{m} + \tau_k^{\mathrm{syn}}}\right) \quad . \end{aligned} \tag{4.98}$$

For COBA neurons, we have initially assumed that we are able to find expressions for ${u_\mathrm{eff}}^0$, $\langle u_\mathrm{eff} \rangle$ and $\langle \tau_\mathrm{eff} \rangle$ in order to calculate the PSP shape, so we need a slightly different approach. In Section 4.2.1, we have already derived an expression for u_eff in Equation 4.30. Now, we just need to calculate its expectation value:

$$E[u_\mathrm{eff}] = E\left[\frac{g_\mathrm{l} E_\mathrm{l} + I^\mathrm{ext} + \sum\limits_k g_k^\mathrm{syn} E_k^\mathrm{rev}}{g_\mathrm{l} + \sum\limits_k g_k^\mathrm{syn}}\right] \quad . \tag{4.99}$$

In the previous section, we have argued that in the high-frequency regime, the relative fluctuations of the synaptic conductance become very small. This permits a replacement of all g_k^syn by their expectation values (Equation 4.94), leaving us with

expectation value of the COBA effective membrane potential

$$E[u_\mathrm{eff}] = \frac{g_\mathrm{l} E_\mathrm{l} + I^\mathrm{ext} + \sum\limits_k w_k \nu_k \tau_k^\mathrm{syn} E_k^\mathrm{rev}}{g_\mathrm{l} + \sum\limits_k w_k \nu_k \tau_k^\mathrm{syn}} \quad . \tag{4.100}$$

Note the similarity with the equivalent equation for CUBA neurons (Equation 4.97), which could have been derived identically from the expression for u_eff in the CUBA case (Equation 4.29). In the same approximation, the average effective time constant can be written as

expectation value of the COBA effective time constant

$$E[\tau_\mathrm{eff}] = \frac{C_\mathrm{m}}{E[g^\mathrm{tot}]} = \frac{C_\mathrm{m}}{g_\mathrm{l} + \sum\limits_k w_k \nu_k \tau_k^\mathrm{syn}} \quad . \tag{4.101}$$

We can now derive an expression for ${u_\mathrm{eff}}^0$ from the assertion that calculating the expectation value of Equation 4.53 must be self-consistent:

$$E[u_\mathrm{eff}] \overset{!}{=} {u_\mathrm{eff}}^0 + E\left[\frac{\sum\limits_k g_k^\mathrm{syn}(t)\left(E_k^\mathrm{rev} - \langle u_\mathrm{eff} \rangle\right)}{\langle g^\mathrm{tot} \rangle}\right] \tag{4.102}$$

By using the newly derived expressions for $E[u_\mathrm{eff}]$ and $E[g^\mathrm{tot}]$, we obtain

$$\begin{aligned} {u_\mathrm{eff}}^0 &= E[u_\mathrm{eff}] - \frac{\sum\limits_k E\left[g_k^\mathrm{syn}(t)\right]\left(E_k^\mathrm{rev} - E[u_\mathrm{eff}]\right)}{\langle g^\mathrm{tot} \rangle} \\ &\overset{\text{(Eqns. 4.100,4.101)}}{=} \dots \\ &= \frac{g_\mathrm{l}}{E[g^\mathrm{tot}]}\left(E[u_\mathrm{eff}] - E_\mathrm{l} - \frac{I^\mathrm{ext}}{g_\mathrm{l}}\right) \end{aligned} \tag{4.103}$$

We can now proceed to calculating the moments of the membrane potential distribution analogously to the CUBA case. The expectation value $E[u]$ must be identical to $E[u_\mathrm{eff}]$, since u is merely a low-pass-filtered version of u_eff (Equation 4.31). A formal calculation

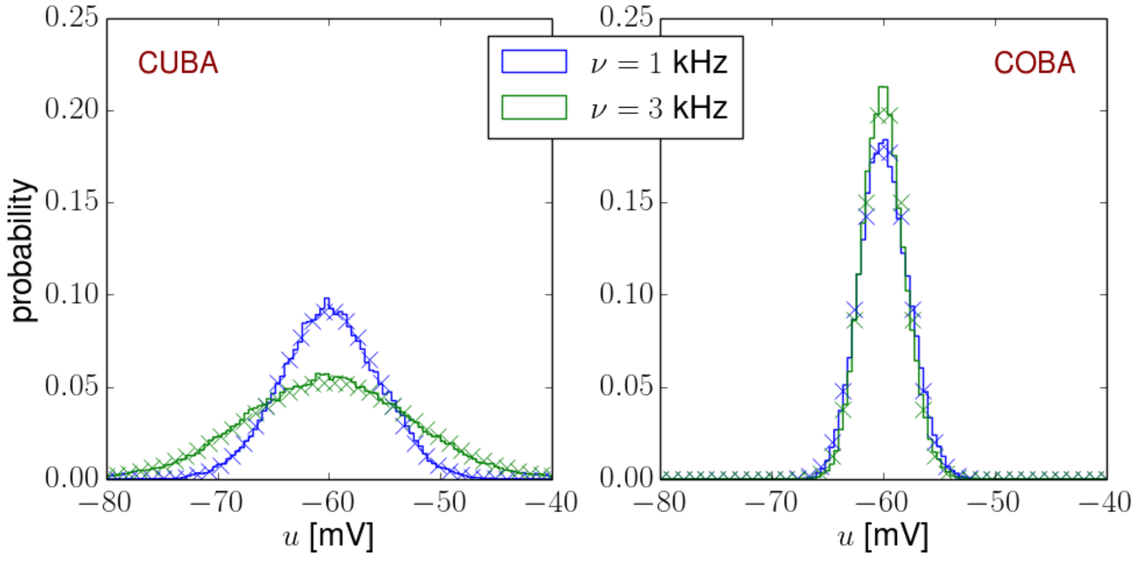


Figure 4.5.: Free ($\vartheta \to \infty$) membrane potential distributions of Poisson-driven CUBA and COBA neurons: theoretical prediction (crosses) vs. simulation results (continuous lines, NEST). The simulation parameters are identical to the ones from Figure 4.4. The balanced input regime (on average, excitation and inhibition have the same impact, i.e., identical rates and identical-amplitude PSPs) is reflected by a constant mean membrane potential, which lies at the resting potential of $E_\mathrm{l} = -60\,\mathrm{mV}$ for both neurons. In the CUBA case, an increased input rate manifests itself in an increased variance of the membrane potential (Equation 4.98), which is a direct consequence of the increased variance of the input current (see Figure 4.4). In the COBA case, the input conductance distribution becomes broader as well, but in the HCS a resulting broadening of the membrane potential distribution is countered by an increased total conductance (Equations 4.101 and 4.105), in the end leading to an opposite effect (compare also with Figure 4.2, top right). Again, the theoretical prediction is in excellent agreement with the simulation data.

using Equations 4.58 and 4.84 yields the same result:

expectation value of the COBA membrane potential

$$\begin{aligned}
E[u] &= {u_\mathrm{eff}}^0 + \sum_{k=1}^{n} \nu_k \int_0^\infty \frac{\tau_k^\mathrm{syn} w_k \left(E_k^\mathrm{rev} - \langle u_\mathrm{eff}\rangle\right)}{\langle g^\mathrm{tot}\rangle \left(\tau_k^\mathrm{syn} - \langle \tau_\mathrm{eff}\rangle\right)} \Theta(t) \left[\exp\left(\frac{-t}{\tau_k^\mathrm{syn}}\right) - \exp\left(\frac{-t}{\langle \tau_\mathrm{eff}\rangle}\right)\right] dt \\
&= {u_\mathrm{eff}}^0 + \sum_{k=1}^{n} \frac{w_k \nu_k \tau_k^\mathrm{syn} (E_k^\mathrm{rev} - \langle u_\mathrm{eff}\rangle)}{\langle g^\mathrm{tot}\rangle} \\
&\overset{\text{(Eqn. 4.103)}}{=} E\left[u_\mathrm{eff}\right] \\
&\overset{\text{(Eqn. 4.100)}}{=} \frac{g_\mathrm{l} E_\mathrm{l} + I^\mathrm{ext} + \sum\limits_k w_k \nu_k \tau_k^\mathrm{syn} E_k^\mathrm{rev}}{g_\mathrm{l} + \sum\limits_k w_k \nu_k \tau_k^\mathrm{syn}} \quad . \qquad (4.104)
\end{aligned}$$

For the variance, we obtain

variance of the COBA membrane potential

$$
\begin{aligned}
\mathrm{Var}[u] &= \sum_{k=1}^{n} \nu_k \int_0^\infty \left\{ \frac{\tau_k^{\mathrm{syn}} \langle \tau_{\mathrm{eff}} \rangle w_k \left(E_k^{\mathrm{rev}} - \langle u_{\mathrm{eff}} \rangle\right)}{C_\mathrm{m} \left(\tau_k^{\mathrm{syn}} - \langle \tau_{\mathrm{eff}} \rangle\right)} \Theta(t) \left[\exp\left(\frac{-t}{\tau_k^{\mathrm{syn}}}\right) - \exp\left(\frac{-t}{\langle \tau_{\mathrm{eff}} \rangle}\right) \right] \right\}^2 dt \\
&= \dots \\
&= \sum_{k=1}^{n} \left[\frac{\langle \tau_{\mathrm{eff}} \rangle \tau_k^{\mathrm{syn}} \left(E_k^{\mathrm{rev}} - \langle u_{\mathrm{eff}} \rangle\right)}{C_\mathrm{m}(\langle \tau_{\mathrm{eff}} \rangle - \tau_k^{\mathrm{syn}})} \right]^2 w_k^2 \nu_k \left(\frac{\langle \tau_{\mathrm{eff}} \rangle}{2} + \frac{\tau_k^{\mathrm{syn}}}{2} - 2 \frac{\langle \tau_{\mathrm{eff}} \rangle \tau_k^{\mathrm{syn}}}{\langle \tau_{\mathrm{eff}} \rangle + \tau_k^{\mathrm{syn}}} \right) \quad ,
\end{aligned}
\tag{4.105}
$$

where $\langle u_{\mathrm{eff}} \rangle$ and $\langle \tau_{\mathrm{eff}} \rangle$ are given by Equations 4.100 and 4.101, respectively.

Figure 4.5 shows the excellent agreement between the above theoretical predictions and simulation data. Excitatory and inhibitory inputs are balanced, so the average membrane potential lies at the resting potential $E_\mathrm{l} = -60\,\mathrm{mV}$. In the CUBA case, the frequency dependence of the variance (Equation 4.98) is reflected by the broadening of the distribution at a higher input rate. Interestingly , the opposite appears to happen in the COBA case: the variance drops slightly with increasing input rate - as evidenced by the higher peak of the normalized PDF. This mirrors our previous observations from Section 4.2.3 (Figure 4.2). With Equations 4.105, 4.100 and 4.101, we are now able to explain all of our experimental observations, which we do in the following section.

4.3.5. The High-Conductance State III: Theory vs. Experiment

We have previously defined the HCS as a state where the synaptic conductance dominates the total membrane conductance (Equation 4.42). In general, this is assumed to be the result of a high-frequent synaptic bombardment. We can now analyze how the variance of the membrane potential distribution behaves in this regime.

In the limit of high input rates (and non-vanishing synaptic weights), the effective time constant (Equation 4.101) goes to zero:

COBA τ_{eff} in the HCS

$$
E\left[\tau_{\mathrm{eff}}\right] \propto \left(\sum_k \nu_k\right)^{-1} \xrightarrow{\sum_k \nu_k \to \infty} 0 \quad . \tag{4.106}
$$

We can therefore neglect all additive $\langle \tau_{\mathrm{eff}} \rangle$-terms in the variance of the membrane potential, leaving us with

variance of the COBA membrane potential in the HCS

$$
\begin{aligned}
\mathrm{Var}[u] &\xrightarrow{\sum_k \nu_k \to \infty} \sum_{k=1}^{n} \left[\frac{\langle \tau_{\mathrm{eff}} \rangle \left(E_k^{\mathrm{rev}} - \langle u_{\mathrm{eff}} \rangle\right)}{C_\mathrm{m}} \right]^2 w_k^2 \nu_k \frac{\tau_k^{\mathrm{syn}}}{2} \\
&= \frac{1}{2} \frac{\sum_k w_k^2 \nu_k \tau_k^{\mathrm{syn}} \left(E_k^{\mathrm{rev}} - \langle u_{\mathrm{eff}} \rangle\right)}{\left(\sum_k w_k \nu_k \tau_k^{\mathrm{syn}}\right)^2} && (4.107) \\
&\propto \frac{1}{\sum_k \nu_k \tau_k^{\mathrm{syn}}} \quad . && (4.108)
\end{aligned}
$$

This result immediately implies two conclusions about the properties of the HCS:

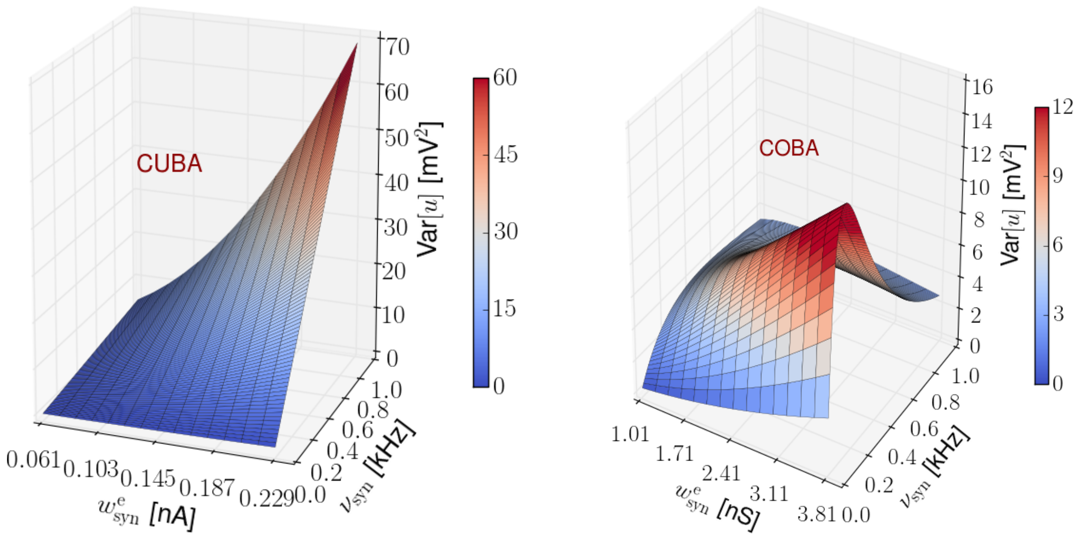


Figure 4.6.: Variance of the membrane potential of an LIF neuron when stimulated by Poisson inputs as a function of the input rate and synaptic input weight. For CUBA neurons (left), it increases linearly with the rate and quadratically with the weight, as follows directly from Equation 4.98. In the COBA case (right), the variance behaves similarly in the low-conductance regime (small input rates and relatively small weights, in the foreground of the figure), which is not surprising, as in this regime COBA and CUBA neurons are functionally identical. For increasing input rates however, the variance reaches a peak and then drops off again, going to zero in the limit of large input rates (background of the figure). In this regime, the variance effectively loses its dependence on the synaptic input weights. This theoretical prediction precisely mirrors the phenomena shown in Figure 4.2.

1) the variance of the membrane potential is (approximately) inversely proportional to the total input rate and

2) the variance of the membrane potential becomes largely independent of the synaptic weights,

which precisely mirror the experimental observations from Section 4.2.3. As argued in Section 4.2.5, our third assumption concerning the small relative fluctuations of the membrane potential now follows directly from the inverse input rate entering the PSP height multiplicatively (see Equations 4.59 and 4.101).

The full w^{syn} and ν^{syn} dependence of the membrane potential variance can be seen in Figure 4.6 for both CUBA and COBA neurons. In the CUBA case, the variance scales quadratically with w^{syn} and linearly with ν^{syn}, as evidenced directly by Equation 4.98. The COBA variance starts off similarly to the CUBA variance in the case of small input rates: the intersection of the variance surface with the w^{syn}-Var[u]-plane (left foreground) is quadratic, whereas its intersection with the ν^{syn}-Var[u] plane starts off as linear (right foreground). For high input rates, the trend is reversed: the variance reaches a maximum and then begins to drop, ultimately becoming nearly independent of w^{syn}, as evidenced by the surface becoming almost flat in the w^{syn}-direction.

The independence of the free membrane potential variance on the synaptic input weights in the HCS is not only an interesting phenomenon, but may play a significant role during learning. In many scenarios where neural networks learn to perform certain tasks, be they biologically plausible or more abstract, but useful for, e.g., machine learning, it turns out that homeostatic mechanisms can be a potent ingredient (see, e.g., Habenschuss et al., 2012). In broad terms, homeostasis ensures that the output of a constituent unit in a network remains unchanged despite changes in its input. If we assume that the connectivity of a neural network remains unchanged (no structural plasticity) – which is a simplifying, but common assumption in computational neuroscience and machine learning – then the only changes that a neuron experiences in its input are changes in its afferent synaptic weights. If we further assume that the output spike train of a neuron depends only on the dominant time constant in its dynamics (which, in the HCS, is the synaptic one) and on the moments of its membrane potential distribution (we shall provide a rigorous treatment of this conjecture in Section 6.5.3), and the ratio of excitation to inhibition remains balanced, then the w^{syn}-independence discussed above directly provides a neuronal homeostatic mechanism. Most interestingly, this feature is not added to the network by virtue of additional dynamic components, but rather a direct consequence of the LIF dynamics in the HCS. In other words, this form of homeostasis is, by default, built into the physics of COBA LIF neurons.

homeostasis

4.4. Shared-Input Correlations of Neuronal Dynamics

When an overlap between the presynaptic populations of two neurons exists, it induces correlations in their behavior, even if the two neurons have no direct synaptic connection. In this section, we discuss quantitative measures for these shared-input pairwise correlations. Building on the expressions derived in the previous sections, we can now make quantitative predictions of shared-input correlations both for the free membrane potential and for the spike trains of neurons with partly overlapping Poisson background pools.

shared-input correlations

4.4.1. Multivariate Distributions and the Correlation Coefficient

We shall start by extending our conclusions from Section 4.3.1 to multivariate distributions of additive Poisson processes. In Section 4.3.2, we have argued that a single additive Poisson-driven RV will follow a Gaussian (marginal) distribution, with a mean and variance given by Equations 4.84 and 4.85, respectively. A set $\boldsymbol{X}$ of such RVs will therefore follow a multivariate Gaussian (joint) distribution, which is defined by the PDF:

multivariate Gaussian

$$f(\boldsymbol{x}, \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \frac{1}{(2\pi)^{k/2}|\boldsymbol{\Sigma}|^{1/2}} \exp\left[-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu})\right] \quad , \tag{4.109}$$

where the mean vector $\boldsymbol{\mu}$ is defined as

$$\boldsymbol{\mu} := E\left[\boldsymbol{X}\right] \quad , \tag{4.110}$$

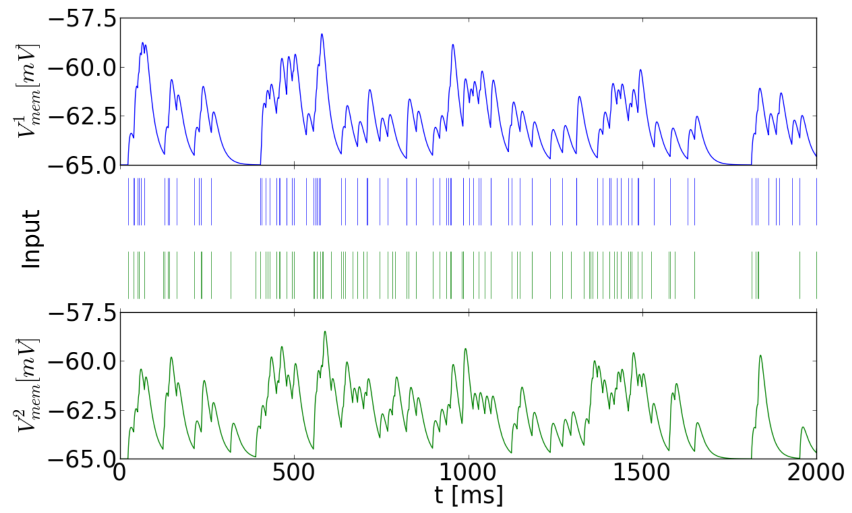


Figure 4.7.: Two conductance-based LIF neurons, each with a total amount of excitatory input channels of $Sy_{tot} = 7$. They consist of $Sy_{1p} = Sy_{2p} = 2$ “private“ input channels for each neuron and $Sy_s = 5$ input channels shared by both neurons. As expected, the membrane potential time course is very similar, due to the high proportion of shared inputs, but not identical. In this section, we discuss quantitative measures for their degree of similarity. Figure taken from Bytschok (2011).

$\boldsymbol{\Sigma}$ represents a covariance matrix with elements

$$\boldsymbol{\Sigma} = \mathrm{Cov}[X_i, X_j] \quad , \tag{4.111}$$

and $|\boldsymbol{\Sigma}|$ denotes its determinant.

In the following, since we will later investigate second-order (pairwise) correlations between neural state variables, we will restrict ourselves to bivariate distributions, but this framework can be straightforwardly extended to correlations of any order. For two RVs X_1 and X_2, the multivariate Gaussian from above reduces to

bivariate Gaussian

$$f(x_1, x_2) = \frac{1}{2\sigma_1\sigma_2\sqrt{1-\rho^2}} \exp\left[-\frac{1}{2\cdot(1-\rho^2)}\left(z_1^2 + z_2^2 - 2\rho z_1 z_2\right)\right] \quad , \tag{4.112}$$

where Z_1 and Z_2 (with their respective values z_1 and z_2) represent normalized versions of the original RVs:

$$z_1 := \frac{x_1 - \mu_1}{\sigma_1} \qquad z_2 := \frac{x_2 - \mu_2}{\sigma_2} \quad . \tag{4.113}$$

In this notation, the covariance has also been normalized to the Pearson product-moment correlation coefficient (or simply correlation coefficient CC)

correlation coefficient

$$\rho = \frac{\mathrm{Cov}[X_1, X_2]}{\sigma_1\sigma_2} = \mathrm{Cov}[Z_1, Z_2] \quad , \tag{4.114}$$

which lets us rewrite the covariance matrix as

$$\boldsymbol{\Sigma} = \begin{pmatrix} \sigma_1^2 & \rho\sigma_1\sigma_2 \\ \rho\sigma_1\sigma_2 & \sigma_2^2 \end{pmatrix} \quad . \tag{4.115}$$

This is a more convenient notation, since the CC is a bounded measure of the correlation between RVs[6] and is also insensitive to linear transformations of the sample space Ω, as follows directly from Equation 4.114. In particular,

$$\rho_{X_1,X_2} \in [-1, 1] \quad , \tag{4.118}$$

with the extreme values 1 and -1 representing a linear dependence of X_1 and X_2, i.e., perfect correlation and anticorrelation, respectively. For independent variables, $\rho_{X_1,X_2} = 0$. Due to these useful properties, we choose the CC as the measure of choice for correlations between continuous, real-valued RVs. In the following sections, we shall use the previously found equations for the membrane potential distribution in order to predict neuronal shared-input correlations in the subthreshold regime.

Following all of the above, Figure 4.8 shows two examples of how we expect the membrane potential distributions to look like for a pair of LIF neurons. Here, we only show the

[6] The relation

$$|\mathrm{Cov}[X, Y]| \leq \sigma_X\sigma_Y \tag{4.116}$$

follows directly from the Cauchy-Schwartz inequality for the inner product space of square-integrable functions (to which finite-variance Gaussian PDFs belong by definition)

$$\left|\int_{\mathbb{R}^n} f(x)\overline{g(x)}\,dx\right|^2 \leq \int_{\mathbb{R}^n} |f(x)|^2\,dx \cdot \int_{\mathbb{R}^n} |g(x)|^2\,dx. \quad . \tag{4.117}$$

special case of $\mu_1 = \mu_2$ and $\sigma_1 = \sigma_2$. In general, iso-probability loci are ellipses centered at $\boldsymbol{\mu}$, with the projections of their axes being proportional to σ_1 and σ_2, respectively. For some probability density p, the iso-probability locus follows directly from Equation 4.112:

iso-probability loci of bivariate Gaussians

$$\frac{1}{R}x^2 - \frac{2\rho}{R}xy + \frac{1}{R}y^2 = 1 \quad , \tag{4.119}$$

which is the canonical form of an ellipse, where

$$R = -2(1-\rho^2)\ln\left(2p\sigma_1\sigma_2\sqrt{1-\rho^2}\right) \quad . \tag{4.120}$$

For uncorrelated RVs ($\rho = 0$), the main axes of the ellipses are parallel to the RV axes. For correlated RVs ($|\rho| > 0$), the ellipses become tilted and squeezed towards the line

$$y(x) = \mathrm{sgn}(\rho)\frac{\sigma_y}{\sigma_x}(x - \mu_x) + \mu_y \quad . \tag{4.121}$$

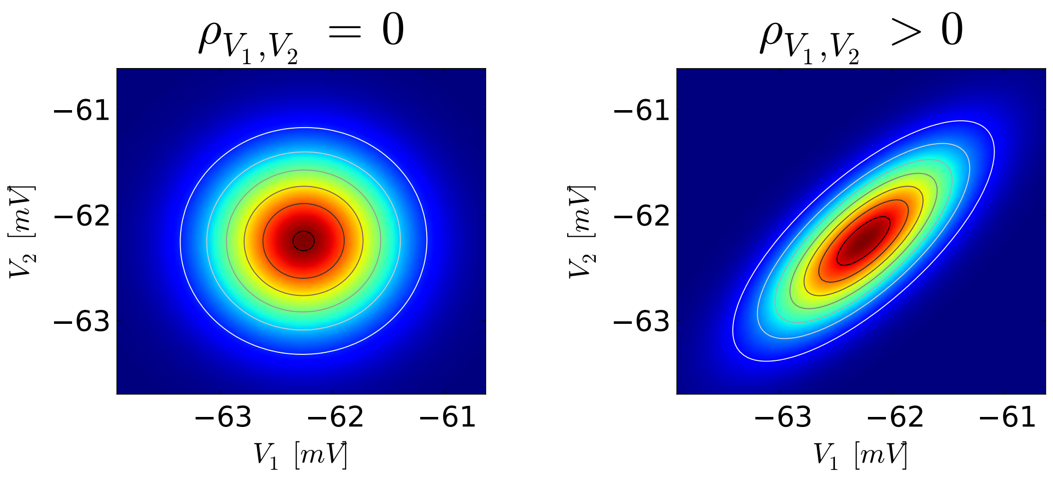


Figure 4.8.: Two examples of bivariate normal distributions of neuron membrane potentials with identical means and variances. Left: independent inputs. Right: some inputs are shared, causing the membrane potentials to have positive correlation. The colormap represents the height of the joint density function $f(u_1, u_2)$ (red is high, blue is low). The continuous lines represent iso-probability loci and have an elliptical shape. Figure taken from Bytschok (2011).

4.4.2. Derivation of the Free (Subthreshold) Membrane Potential CC

Let us start by considering the general case of two LIF neurons that are fed by multiple independent Poisson sources which we denote by their indices $i \in \{1, \ldots, n\}$. These sources are characterized by their rates $\boldsymbol{\nu} = (\nu_1, \ldots, \nu_n)$, by the synaptic weights via which they are connected to the two neurons $\boldsymbol{w}_1 = (w_{11}, \ldots, w_{1n})$ and $\boldsymbol{w}_2 = (w_{21}, \ldots, w_{2n})$, as well as by their respective time constants $\boldsymbol{\tau}_1^{\text{syn}} = (\tau_{11}^{\text{syn}}, \ldots, \tau_{1n}^{\text{syn}})$ and $\boldsymbol{\tau}_2^{\text{syn}} = (\tau_{21}^{\text{syn}}, \ldots, \tau_{2n}^{\text{syn}})$.[7] Furthermore, the neurons themselves may have different parameter sets, which we denote by $\boldsymbol{P}_1$ and $\boldsymbol{P}_2$. What we are looking for is an expression for

$$\rho_{u_1,u_2} = \rho(\boldsymbol{P}_1, \boldsymbol{P}_2, \boldsymbol{w}_1, \boldsymbol{w}_2, \boldsymbol{\tau}_1^{\text{syn}}, \boldsymbol{\tau}_2^{\text{syn}}, \boldsymbol{\nu}) \quad . \tag{4.122}$$

We have shown previously (Section 4.2) that the membrane potential of CUBA LIF neurons in general and COBA LIF neurons in the HCS can be written as a sum over PSPs from individual spikes (Equations 4.38 and 4.58). Due to this linearity, we can group the PSPs by their synaptic source and write the membrane potential as a sum over all input sources of the contribution of each individual source:

$$u_i = \sum_{k=1}^{n} u_{ik} \quad , \tag{4.123}$$

where each u_{ik} is a Poisson-driven stochastic process with mean and variance given by the equations derived in Section 4.3.4. We can exploit the bilinearity of the covariance to find

$$\begin{aligned} \text{Cov}[u_1, u_2] &= \text{Cov}\left[\sum_{k=1}^{n} u_{1k}, \sum_{j=1}^{n} u_{2j}\right] \\ &= \sum_k \sum_j \text{Cov}[u_{1k}, u_{2j}] \quad . \end{aligned} \tag{4.124}$$

Since we have defined the individual Poisson sources as independent, the membrane potential contributions that they generate are also independent and therefore uncorrelated:

$$k \neq j \quad \Longrightarrow \quad \text{Cov}[u_{1k}, u_{2j}] = 0 \quad , \tag{4.125}$$

which allows us to further simplify the membrane potential covariance equation:

covariance of membrane potentials driven by a (common) set of independent Poisson sources

$$\text{Cov}[u_1, u_2] = \sum_k \text{Cov}[u_{1k}, u_{2k}] \quad . \tag{4.126}$$

For calculating $\text{Cov}[u_{1k}, u_{2k}]$, we can use the same formalism as we did in Section 4.3.1 for the variance of Poisson-driven processes. All we need to do is replace the terms $E\left[Y^2\right]$ and $E^2[Y]$ by $E\left[XY\right]$ and $E\left[X\right]E\left[Y\right]$, respectively, where X and Y are both Poisson-driven processes that have the same rate λ but different kernels κ_X and κ_Y, respectively. In particular, Equation 4.76 transforms to

$$E\left[XY\right] = \lambda \int_0^\infty d\tau\ \kappa_X(\tau)\kappa_Y(\tau) + \lambda^2 \int_0^\infty d\tau\ \kappa_X(\tau) \int_0^\infty d\tau\ \kappa_Y(\tau) \tag{4.127}$$

[7] Here, we explicitly allow the violation of Dale's law, which would, in principle, not only require $\text{sgn}(w_{1j}) = \text{sgn}(w_{1j})$, but also $\tau_{1j}^{\text{syn}} = \tau_{2j}^{\text{syn}}$, since the synaptic transmission should be mediated by the same type of neurotransmitters (see Section 2.1.3). However, Dale's law does not necessarily apply to artificial neural networks in general. In particular, allowing it to be violated turns out to be quite useful, such as for various types of neuromorphic hardware or for the specific class of networks we discuss in Chapter 6.

and Equation 4.71 becomes

$$E\left[X\right]E\left[Y\right] = \lambda^2 \int_0^\infty d\tau \ \kappa_X(\tau) \int_0^\infty d\tau \ \kappa_Y(\tau) \quad . \tag{4.128}$$

covariance of two RVs driven by the same Poisson process

With this, we can now write

$$\mathrm{Cov}[X,Y] = \lambda \int_0^\infty d\tau \ \kappa_X(\tau)\kappa_Y(\tau) \quad . \tag{4.129}$$

We can now calculate the subthreshold membrane potential CC explicitly. The CUBA and HCS COBA kernels are given by Equations 4.39 and 4.59, respectively. In the CUBA case, the covariance becomes

membrane potential covariance and CC of two Poisson-driven CUBA neurons

$$\begin{aligned}\mathrm{Cov}[u_1,u_2] &\overset{\text{(Eqns 4.129, 4.39)}}{=} \dots \\ &= \sum_{k=1}^{n} \left[\nu_k \prod_{i=1}^{2} w_{ik} \frac{\tau_{ik}^{\mathrm{syn}} \tau_{\mathrm{m}i}}{C_{\mathrm{m}i}\left(\tau_{ik}^{\mathrm{syn}} - \tau_{\mathrm{m}i}\right)} \right. \\ &\times \left. \left(\frac{\tau_{1k}^{\mathrm{syn}}\tau_{2k}^{\mathrm{syn}}}{\tau_{1k}^{\mathrm{syn}} + \tau_{2k}^{\mathrm{syn}}} + \frac{\tau_{\mathrm{m}1}\tau_{\mathrm{m}2}}{\tau_{\mathrm{m}1} + \tau_{\mathrm{m}2}} - \frac{\tau_{1k}^{\mathrm{syn}}\tau_{\mathrm{m}1}}{\tau_{1k}^{\mathrm{syn}} + \tau_{\mathrm{m}1}} - \frac{\tau_{2k}^{\mathrm{syn}}\tau_{\mathrm{m}2}}{\tau_{2k}^{\mathrm{syn}} + \tau_{\mathrm{m}2}} \right) \right] \ . \end{aligned} \tag{4.130}$$

With Equation 4.98 for the membrane potential variance, we obtain for the CC:

$$\rho_{u_1,u_2} = \frac{\pi\left(\boldsymbol{\nu}, \boldsymbol{w}_1, \boldsymbol{w}_2, \boldsymbol{\tau}_1^{\mathrm{syn}}, \boldsymbol{\tau}_2^{\mathrm{syn}}, \boldsymbol{\phi}_{12}\right)}{\prod_{i=1}^{2} \sqrt{\pi\left(\boldsymbol{\nu}, \boldsymbol{w}_i, \boldsymbol{w}_i, \boldsymbol{\tau}_i^{\mathrm{syn}}, \boldsymbol{\psi}_i\right)}} \quad , \tag{4.131}$$

For a simplified representation, we have defined the kth elements of the vectors $\boldsymbol{\phi}_{ij}$ and $\boldsymbol{\psi}_i$ as

$$\phi_{ijk} = \frac{\frac{\tau_{ik}^{\mathrm{syn}}\tau_{jk}^{\mathrm{syn}}}{\tau_{ik}^{\mathrm{syn}}+\tau_{jk}^{\mathrm{syn}}} + \frac{\tau_{\mathrm{m}i}\tau_{\mathrm{m}j}}{\tau_{\mathrm{m}i}+\tau_{\mathrm{m}j}} - \frac{\tau_{ik}^{\mathrm{syn}}\tau_{\mathrm{m}i}}{\tau_{ik}^{\mathrm{syn}}+\tau_{\mathrm{m}i}} - \frac{\tau_{jk}^{\mathrm{syn}}\tau_{\mathrm{m}j}}{\tau_{jk}^{\mathrm{syn}}+\tau_{\mathrm{m}j}}}{\left(\tau_{ik}^{\mathrm{syn}} - \tau_{\mathrm{m}i}\right)\left(\tau_{jk}^{\mathrm{syn}} - \tau_{\mathrm{m}j}\right)} \quad \text{and} \tag{4.132}$$

$$\psi_{ik} = \frac{\tau_{\mathrm{m}i}}{2} + \frac{\tau_{ik}^{\mathrm{syn}}}{2} - 2\frac{\tau_{\mathrm{m}i}\tau_{ik}^{\mathrm{syn}}}{\tau_{\mathrm{m}i} + \tau_{k}^{\mathrm{syn}}ik} \quad , \tag{4.133}$$

respectively, and the function $\pi(\cdot)$ as

$$\pi(\boldsymbol{v}_1, \dots, \boldsymbol{v}_m) := \sum_{k=1}^{n} \prod_{i=1}^{m} v_{ik} \quad , \quad \text{with} \quad \dim(v_i) = n \quad . \tag{4.134}$$

It is interesting to note yet another very useful property of the CC: in the CUBA case, apart from the configuration of the stimuli given by $\{\boldsymbol{\nu}, \boldsymbol{w}_1, \boldsymbol{w}_2, \boldsymbol{\tau}_1^{\mathrm{syn}}, \boldsymbol{\tau}_2^{\mathrm{syn}}\}$, it does not depend on the full parameter sets $\boldsymbol{P}_1$ and $\boldsymbol{P}_2$ of the two neurons, but only on their membrane time constants $\tau_{\mathrm{m}1}$ and $\tau_{\mathrm{m}2}$.

The calculation for the HCS COBA case is analogous, essentially requiring the following replacements:

$$w_{ik} \to w_{ik} \left(E_{ik}^{\mathrm{rev}} - \langle u_{\mathrm{eff}} \rangle_i \right) \quad \text{and} \tag{4.135}$$

$$\tau_{\mathrm{m}i} \to \langle \tau_{\mathrm{eff}} \rangle_i \tag{4.136}$$

where $\langle \tau_{\mathrm{eff}} \rangle_i$ and $\langle u_{\mathrm{eff}} \rangle_i$ are given by Equations 4.101 and 4.100. The covariance then reads

membrane potential covariance and CC of two Poisson-driven COBA neurons

$$\begin{aligned} \mathrm{Cov}[u_1, u_2] &\overset{\text{(Eqns 4.129, 4.59)}}{=} \dots \\ &= \left[\sum_{k=1}^{n} \nu_k \prod_{i=1}^{2} w_{ik} \frac{\langle \tau_{\mathrm{eff}} \rangle_i \, \tau_{ik}^{\mathrm{syn}} \left(E_{ik}^{\mathrm{rev}} - \langle u_{\mathrm{eff}} \rangle_i \right)}{C_{\mathrm{m}} \left(\langle \tau_{\mathrm{eff}} \rangle_i - \tau_{ik}^{\mathrm{syn}} \right)} \right] \\ &\times \left[\frac{\tau_{1k}^{\mathrm{syn}} \tau_{2k}^{\mathrm{syn}}}{\tau_{1k}^{\mathrm{syn}} + \tau_{2k}^{\mathrm{syn}}} + \frac{\langle \tau_{\mathrm{eff}} \rangle_1 \langle \tau_{\mathrm{eff}} \rangle_2}{\langle \tau_{\mathrm{eff}} \rangle_1 + \langle \tau_{\mathrm{eff}} \rangle_2} - \frac{\tau_{1k}^{\mathrm{syn}} \langle \tau_{\mathrm{eff}} \rangle_1}{\tau_{1k}^{\mathrm{syn}} + \langle \tau_{\mathrm{eff}} \rangle_1} - \frac{\tau_{2k}^{\mathrm{syn}} \langle \tau_{\mathrm{eff}} \rangle_2}{\tau_{2k}^{\mathrm{syn}} + \langle \tau_{\mathrm{eff}} \rangle_2} \right] , \end{aligned} \tag{4.137}$$

and the CC is given by

$$\rho_{u_1,u_2} = \frac{\pi \left(\boldsymbol{\nu}, \boldsymbol{w}_1, \left(\boldsymbol{E}_1^{\mathrm{rev}} - \langle \boldsymbol{u}_{\mathrm{eff}} \rangle_1 \right), \boldsymbol{w}_2, \left(\boldsymbol{E}_2^{\mathrm{rev}} - \langle \boldsymbol{u}_{\mathrm{eff}} \rangle_2 \right), \boldsymbol{\tau}_1^{\mathrm{syn}}, \boldsymbol{\tau}_2^{\mathrm{syn}}, \boldsymbol{\phi}_{12} \right)}{\prod_{i=1}^{2} \sqrt{\pi \left(\boldsymbol{\nu}, \boldsymbol{w}_i, \left(\boldsymbol{E}_i^{\mathrm{rev}} - \langle \boldsymbol{u}_{\mathrm{eff}} \rangle_i \right), \boldsymbol{w}_i, \left(\boldsymbol{E}_i^{\mathrm{rev}} - \langle \boldsymbol{u}_{\mathrm{eff}} \rangle_i \right), \boldsymbol{\tau}_i^{\mathrm{syn}}, \boldsymbol{\psi}_i \right)}} , \tag{4.138}$$

where $\boldsymbol{E}_i^{\mathrm{rev}}$ denotes the reversal potential vector of the ith neuron (corresponding to the weight vector $\boldsymbol{w}_i$), $\langle \boldsymbol{u}_{\mathrm{eff}} \rangle_i$ is an n-dimensional vector with all components equal to $\langle u_{\mathrm{eff}} \rangle_i$, and all other notations are as defined above.

In their most general form, the CC Equations 4.131 and 4.138 are somewhat unwieldy. To gain a better intuition, we can assume several simplifications. First of all, we can assume the two neurons to share several parameters, in particular τ_{m} (in the CUBA case) and $\langle \tau_{\mathrm{eff}} \rangle$ and $\langle u_{\mathrm{eff}} \rangle$ in the COBA case. Furthermore, we can assume them to have the same synaptic time constant and reversal potential vectors ($\boldsymbol{\tau}^{\mathrm{syn}}$ and $\boldsymbol{E}^{\mathrm{rev}}$), which can be viewed as a direct consequence of Dale's law. Under these assumptions, a lengthy but straightforward calculation shows that all the factors that depend on the abovementioned parameters in Equations 4.131 and 4.138 cancel out, leaving us with a single equation for the CC that holds for both the CUBA and the COBA case:

membrane potential CC of two Poisson-driven neurons with identical parameters

$$\rho_{u_1,u_2} = \frac{\pi \left(\boldsymbol{\nu}, \boldsymbol{w}_1, \boldsymbol{w}_2 \right)}{\sqrt{\pi \left(\boldsymbol{\nu}, \boldsymbol{w}_1, \boldsymbol{w}_1 \right)} \sqrt{\pi \left(\boldsymbol{\nu}, \boldsymbol{w}_2, \boldsymbol{w}_2 \right)}} \quad . \tag{4.139}$$

This equation paints a more intuitive picture. The numerator represents the fact that the input channels that contribute most to the CC are those which have a large common impact on both neurons, i.e., a high firing rate ν_k and strong projections w_{1k} and w_{2k} towards both neurons. The denominator is simply a reduced version of the standard deviation of the two membrane potentials (compare with, e.g., Equation 4.98). As mentioned above, the τ_{m}-, τ^{syn}- and C_{m}-dependent terms cancel out with their equivalents in the numerator, leaving only a dependence on $\boldsymbol{\nu}$ and $\boldsymbol{w}$.

4.4.3. Subthreshold Shared-Input Correlations: Theory vs. Simulation

For the beginning, let us narrow down the problem even further. By assuming that some of the inputs are only connected to only one of the two neurons ($w_{ik} = 0$), we can subdivide the total set of inputs into two sets of "private" ones

$$P_1 = \{k | w_{1k} \neq 0, w_{2k} = 0\} \quad , \quad |P_1| = p \quad , \tag{4.140}$$

$$P_2 = \{k | w_{2k} \neq 0, w_{1k} = 0\} \quad , \quad |P_2| = p \tag{4.141}$$

and a set of "shared" ones

$$S = \{k | w_{1k} \neq 0, w_{2k} \neq 0\} \quad , \quad |S| = s \quad . \tag{4.142}$$

Here, we have required the respective number of presynaptic inputs (and therefore the respective number of private inputs) to be equal for the two neurons. Furthermore, we set all input rates and all nonzero weights as equal, such that the CC should now only depend on s and p. Indeed, a simple calculation yields

dependence of the membrane potential CC the number of shared and private inputs

$$\rho_{u_1,u_2} = \ldots = \frac{s}{s+p} \quad . \tag{4.143}$$

This result again confirms the usefulness of the CC as a correlation measure. Ideally, we require a measure that is invariant to identical transformations of the two RVs (membrane potentials) in question, such as modifications of the synaptic kernel (by changing, e.g. τ_m or τ^syn), modifications of the total stimulus strength (by changing, e.g., w or nu) or any other parameter changes. Indeed, under these simplified conditions, the CC of two neurons is given only by the ratio of shared inputs to the total number of inputs.

This intuitive result is depicted in Figure 4.9. The predicted simplified CC (Equation 4.143) is compared to results from simulations where the stimulus frequency and strength are varied. Apart from expected variations in the CC extracted from the simulations (due to their limited duration), the simulations confirm our previous assertion that, when the two neurons share a common parameter set, the CC only depends on the ratio of shared to total inputs.

An interesting observation is that the fluctuations due to insufficient sampling statistics appear to decrease with an increasing shared-to-total input ratio (if the simulation time is kept constant). This phenomenon is easily understood when taking into account the volume (in our case, area) of the sampled configuration space. If we assume some low value $\tilde{p} \to 0$, we can guarantee that all samples are almost surely within the iso-probability locus of $\tilde{p}$, which also has a finite area A, since it is a closed curve. The area of an ellipse written in canonical form

$$ax^2 + bxy + cy^2 = 1 \tag{4.144}$$

is given by

$$A = \frac{2\pi}{\sqrt{4ac - b^2}} \quad . \tag{4.145}$$

area of $\tilde{p}$-iso-probability loci

From the iso-probability locus Equation 4.119, we can therefore easily derive

$$A = -2\pi\sqrt{1-\rho^2} \ln\left(2\tilde{p}\sigma_1\sigma_2\sqrt{1-\rho^2}\right) \quad . \tag{4.146}$$

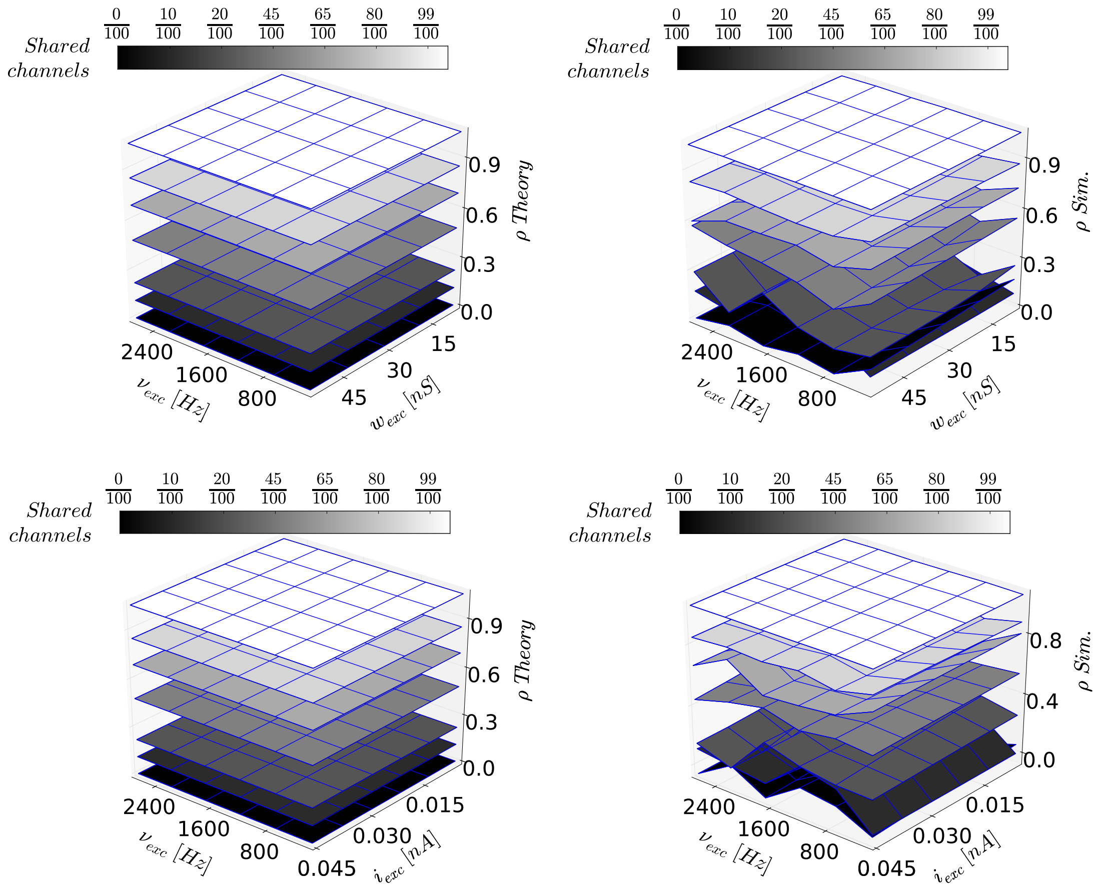
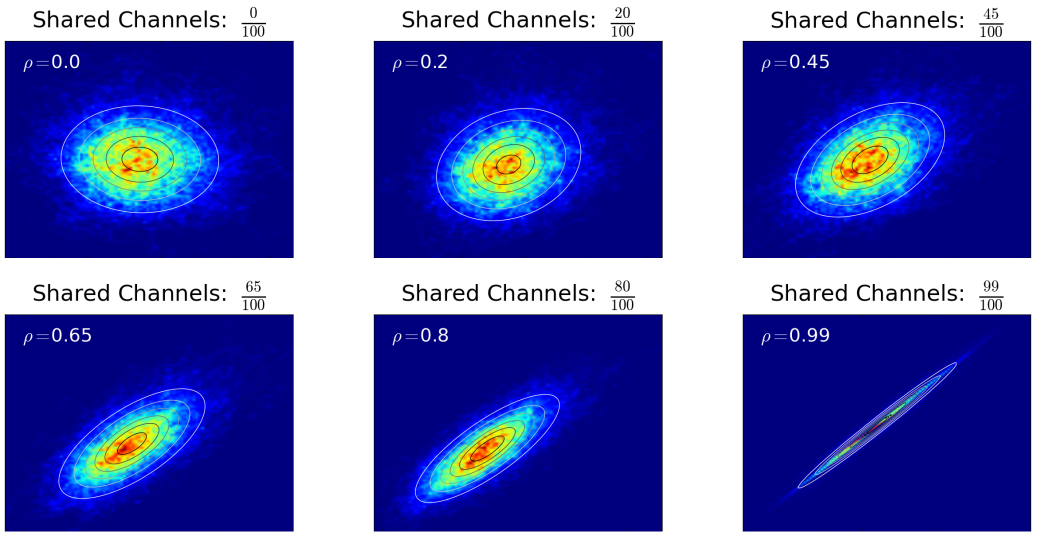


Figure 4.9.: Membrane potential CC of two identical neurons for different neuron models and parameters: theoretical prediction vs. simulation results. **Top row:** CUBA neurons. The CC is calculated and simulated for different values of the shared input ratio $s/(s+p)$, input rate ν and input weight w. The dependence on $s/(s+p)$ is linear, whereas the other two variables do not affect the CC. **Bottom row:** COBA neurons. The CC is not sensitive to the change from CUBA to COBA, as can be seen by the horizontal planes not changing their position. Apart from slight deviations due to insufficient statistics (too short simulation time), the theoretical prediction is confirmed by the simulation results. Figure taken from Bytschok (2011).



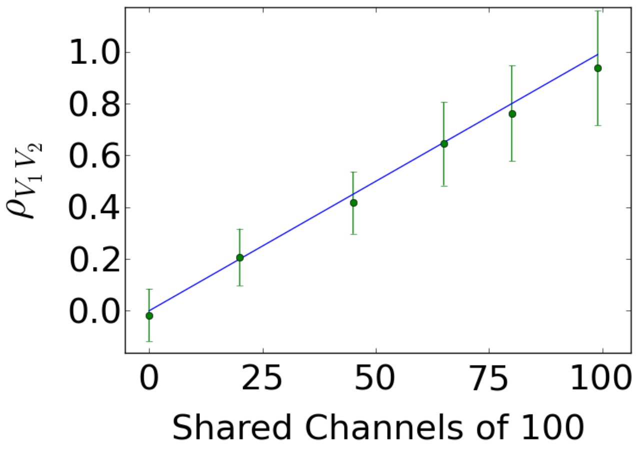


Figure 4.10.: Two neurons driven by 100 independent Poisson sources in the subthreshold regime that share an increasing number of inputs: theoretical prediction vs. simulation results. Top: membrane potential joint distributions. The colormap represents the height of the simulated joint density function $f(u_1, u_2)$ (red is high, blue is low). The CC and iso-probability loci are calculated from Equations 4.143 and 4.119. As the number of shared inputs increases, the CC increases as well and the iso-probability ellipses become narrower. Bottom: linear increase of the correlation coefficient with the increasing number of shared channels, as predicted by Equation 4.143. The theoretical predictions are in very good agreement with the simulation data. Figure taken from Bytschok (2011).

The only variable that appears in this equation and also changes its value in our simulations is the CC ρ. From a straightforward application of l'Hôpital's rule, we find that

$$\lim_{\rho \to 1} A = 0 \quad . \tag{4.147}$$

Since the sampled configuration space volume decreases, but the total number of samples remains constant due to the fixed simulation duration, the sample density increases, leading to a more precise sample-based estimate of ρ as it approaches 1.

With the CC from Equation 4.143 and $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}$ from Section 4.3.4, we can now predict the full membrane potential joint distribution for two Poisson-driven neurons with identical parameters (for the general case we would need to use Equations 4.131 or 4.138). The comparison to simulation results is shown in Figure 4.10. As the number of shared sources increases, the two membrane potentials become more correlated, as indicated by the narrowing-down of their joint distribution and by their increasing CC. The predicted CC is in very good agreement with the simulation data. This allows us to calculate the iso-probability loci (Equation 4.119) for the various simulated scenarios, which are, as expected, confirmed by the simulations. Figure 4.10 also offers a nice visual confirmation of the sample density argument given above.

4.4.4. The Symmetric Uncertainty as a Spike-Based Correlation Measure

We can now take our investigation one step further and construct a framework that allows us to predict spike-based correlations. In principle, this should turn out even more useful than the prediction of the free membrane potential correlations, since it is the spikes that mediate the information exchange between neurons.

The first problem we encounter is a rather trivial one and it concerns the mathematical description of a spike train. Until now, we have treated spike trains as point processes, i.e., defined as a sequence $(t_1 < \ldots < t_n)$ of points in time, or, alternatively, a sum of delta functions (Equation 2.29). In order to be able to apply typical correlation measures, it would be useful to first transform spike trains to finite functions of time that are almost everywhere[8] continuous. The simplest way to do this is by convolution with a box (a.k.a. boxcar) function of duration τ_on:

$$\begin{aligned} \tilde{s}(t) :&= \rho(t) * \mathrm{box}(t|0, \tau_\mathrm{on}, 1) \\ &= \int_{-\infty}^{\infty} d\tau \sum_{\text{spikes } s} \delta(\tau - t_s) \mathrm{box}(t - \tau | 0, \tau_\mathrm{on}, 1) \end{aligned} \tag{4.148}$$

where we use the following definition of the box function:

$$\mathrm{box}(t|t_1, t_2, a) := a \cdot [\Theta(t - t_1) - \Theta(t - t_2)] = \begin{cases} a & \text{if } t \in (t_1, t_2] \ , \\ 0 & \text{otherwise} \ , \end{cases} \tag{4.149}$$

[8] A property holds "almost everywhere" if, the set for which the property does not hold has measure zero. In other words, the set for which the property holds takes up almost the entire configuration space. Its analog in probability theory is "almost surely" (see, e.g., the statement of the CLT from Section 4.3.2).

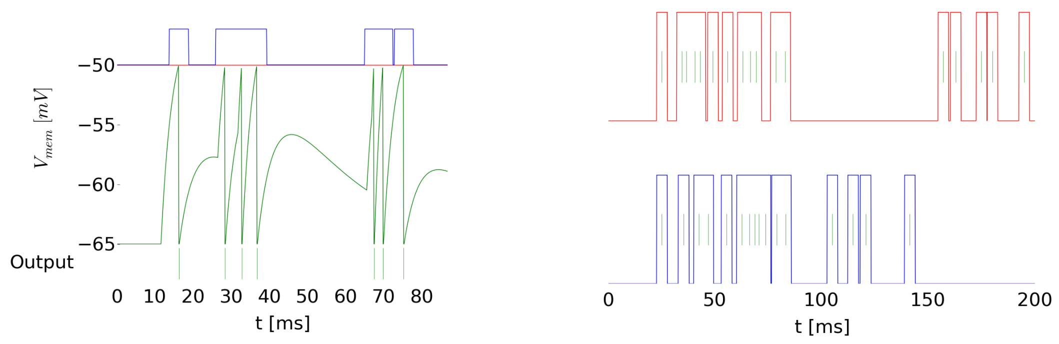


Figure 4.11.: **Left:** Transformation of a spike train to a binary RV by convolution with a box function. Here, the box function is defined as $\mathrm{box}(t|-\tau_\mathrm{on}/2, \tau_\mathrm{on}/2, 1)$, making the 1-states centered around the spikes. We shall see later that aligning the left flank of the box function with the spikes ($\mathrm{box}(t|0, \tau_\mathrm{on}, 1)$) as defined in Equation 4.150 is more meaningful, so this remains an illustrative example for the general case. **Right:** 1/0-states of a neuron pair with shared inputs. The red and blue lines represent $s_1(t)$ and $s_2(t)$, respectively, while the green lines represent the output spikes of the two neurons. Figures taken from Bytschok (2011).

with $\Theta(\cdot)$ representing the Heaviside step function. This transformation effectively represents a spike count during the time interval $(t - \tau_\mathrm{on}, t]$. We shall go one step further and not differentiate between states with different numbers of preceding spikes:

binary state of a spiking neuron

$$s(t) = \mathrm{sgn}\left[\tilde{s}(t)\right] = \begin{cases} 1 & \text{if } \exists\, t_s \in (t - \tau_\mathrm{on}, t] \ , \\ 0 & \text{otherwise} \ , \end{cases} \tag{4.150}$$

By virtue of the above definition, a neuron is said to be in the "1"-state for a duration τ_on following a spike and in the "0"-state otherwise (here, we use quotation marks to emphasize that the numerical values we associate with the two states are arbitrary; from here on, we drop them for readability). In other words, $s(t)$ encodes whether the neuron has spiked or not during the time interval $(t - \tau_\mathrm{on}, t]$. For now, τ_on is arbitrary, but we shall explore a deep connection to the representation of discrete probability spaces in Chapter 6. Figure 4.11 shows an exemplary transformation of a spike train with the method described above.

With the above transformation, we have effectively represented a spike train $\rho(t)$ as a binary RV $S = s(t)$. A pair of neurons can therefore be in one of four possible joint states at any point in time:

state space of two spiking neurons

$$\Omega(S_1, S_2) = \{(0,0), (0,1), (1,0), (1,1)\} \quad , \tag{4.151}$$

where we use the notation $(\cdot)$ to represent a tuple (not an interval). In order to improve readability, we further define the following shorthands:

$$^{n}p_x := p(S_n = x) \quad \text{and} \tag{4.152}$$

$$p_{xy} := p(S_1 = x, S_2 = y) \quad , \tag{4.153}$$

with which we can denote the probabilities of either neuron to be in the "1"-state by ${}^1p_1$ and ${}^2p_1$, and those of the four possible joint states by p_{00}, p_{01}, p_{10} and p_{11}, respectively. In this notation, the expression for the spike-train CC turns out to be quite simple:

state CC for a pair of spiking neurons

$$\begin{aligned}
\rho_{s_1,s_2} &= \frac{\mathrm{Cov}[S_1,S_2]}{\sqrt{\mathrm{Var}[S_1]\cdot\mathrm{Var}[S_2]}} \\
&= \frac{E[S_1,S_2]-E[S_1]E[S_2]}{\sqrt{\left(E[S_1^2]-E^2[S_1]\right)\left(E[S_2^2]-E^2[S_2]\right)}} \\
&= \frac{\sum\limits_{(s_1,s_2)\in\Omega} p(s_1,s_2)s_1s_2 - \sum\limits_{s_1\in\{0,1\}} p(s_1)s_1 \sum\limits_{s_2\in\{0,1\}} p(s_2)s_2}{\sqrt{\prod\limits_{n=1}^{2}\left[\sum\limits_{s_n\in\{0,1\}} p(s_n)s_n^2 - \left(\sum\limits_{s_n\in\{0,1\}} p(s_n)s_n\right)^2\right]}} \\
&= \frac{p_{11}-{}^1p_1{}^2p_1}{\sqrt{\left({}^1p_1-{}^1p_1^2\right)\left({}^2p_1-{}^2p_1^2\right)}} \quad . \qquad (4.154)
\end{aligned}$$

While the CC is an often-used spike train correlation measure[9], the way in which we are now effectively treating the spike output of a neuron as a binary RV might call for a more information-theoretical approach.

The amount of information in a message is routinely quantified by its entropy. The temporal evolution of a binary RV effectively spells out such a message, with an entropy defined by

entropy of a binary RV

$$H(S) = -\sum_{s\in\{0,1\}} p(s)\log_2 p(s) \quad . \qquad (4.155)$$

This is a particularly useful quantity, since it associates a neuron that "permanently does the same thing", i.e., $p(S=0)\to 1$ (never spiking) or $p(S=1)\to 1$ (always spiking), with a vanishing information content, as can be easily verified by l'Hôspital's rule:

$$H(S) \xrightarrow{p(S=x)\to 1} 0 \quad . \qquad (4.156)$$

For two correlated RVs, we can define a so-called conditional entropy, which quantifies the amount of information we can gain by measuring one of the RVs in addition to what we already know by having measured the other:

conditional entropy for two binary RVs

$$\begin{aligned}
H(S_1|S_2) &= \sum_{s_2\in\{0,1\}} p(s_2)H(S_1|s_2) \\
&= -\sum_{(s_1,s_2)\in\Omega} p(s_2)p(s_1|s_2)\log_2 p(s_1|s_2) \\
&= -\sum_{(s_1,s_2)\in\Omega} p(s_1,s_2)\log_2\frac{p(s_1,s_2)}{p(s_2)} \quad . \qquad (4.157)
\end{aligned}$$

[9] Albeit not necessarily in the same way as we do here. More often than using a box function, spike trains are convolved with exponential or Gaussian functions. Yet another popular method of processing a spike train is by binning, thereby effectively discretizing time and treating the output of a neuron as a firing rate.

The conditional entropy is sometimes also called noise entropy, especially in cases where S_1 represents a noisy version of S_2, so the remaining entropy of S_1 after having measured S_2 is the entropy of the noise. This already brings us closer to our goal of defining an entropy-based correlation measure, but the conditional entropy still has the drawback of being asymmetric (due to the denominator in the $\log_2$ depending only on s_2).

As we have already mentioned (Equation 4.8) the joint product of a set of RVs factorizes if and only if they are pairwise independent. In particular, for two RVs,

$$s_1 \perp s_2 \quad \Rightarrow \quad p(s_1, s_2) = p(s_1)p(s_2) \quad . \tag{4.158}$$

We can therefore characterize the mutual dependence of two RVs by comparing their joint distribution $p(S_1, S_2)$ with the product of their marginals $p(S_1)p(S_2)$. A standard measure for such a comparison is provided by the Kullback-Leibler (KL) divergence, which is defined as follows for an ordered[10] pair (p, q) of probability distributions over the same discrete[11] space Ω:

KL divergence

$$D_{\mathrm{KL}}\left(p \parallel q\right) = \sum_{x\in\Omega} p(x) \log \frac{p(x)}{q(x)} \quad . \tag{4.160}$$

The base of the logarithm is usually chosen depending on the nature of the RVs or on the preferred unit of information (bits, nats, etc.) – so in our case, $\log_2$ is a natural choice. We can now calculate the KL divergence of the joint probability distribution $p(S_1, S_2)$ from the product of marginals $p(S_1)p(S_2)$:

mutual information

$$\begin{aligned} D_{\mathrm{KL}}\left(p(S_1, S_2) \parallel p(S_1)p(S_2)\right) &= \sum_{(s_1,s_2)\in\Omega} p(s_1, s_2) \log_2 \left(\frac{p(s_1|s_2)}{p(s_1)}\right) \\ &= \sum_{(s_1,s_2)\in\Omega} p(s_1, s_2) \log_2 p(s_1|s_2) - \sum_{(s_1,s_2)\in\Omega} p(s_1, s_2) \log_2 p(s_1) \\ &= \underbrace{\sum_{(s_1,s_2)\in\Omega} p(s_1, s_2) \log_2 \frac{p(s_1, s_2)}{p(s_2)}}_{\text{Eq. 4.157}} - \underbrace{\sum_{s_1\in\{0,1\}} p(s_1) \cdot \log_2 p(s_1)}_{\text{Eq. 4.156}} \\ &= H(S_1) - H(S_1|S_2) =: I(S_1, S_2) \quad , \end{aligned} \tag{4.161}$$

which is the definition of the frequently used so-called mutual information (MI) $I(S_1, S_2)$. The MI has the very useful property of being a metric, therefore being, in particular, symmetric with respect to S_1 and S_2.

The remaining concern with using the MI as a correlation measure is the fact that it is unnormalized, therefore depending on, e.g., the input rates and weights of a pair of neurons with a fixed configuration of shared and private input channels. This problem can be easily solved by normalizing the MI to the entropies of both RVs, in order to keep it symmetric. Here, we use the so-called symmetric uncertainty (SU) as defined in Witten and Frank (2005):

symmetric uncertainty

$$\tilde{I}(S_1, S_2) := \frac{2I(S_1, S_2)}{H(S_1) + H(S_2)} \quad . \tag{4.162}$$

[10] The Kullback-Leibler divergence is not symmetric, i.e, in general,

$$D_{\mathrm{KL}}\left(p \parallel q\right) \neq D_{\mathrm{KL}}\left(q \parallel p\right) \quad . \tag{4.159}$$

[11] For a continuous Ω, the sum is replaced by an integral.

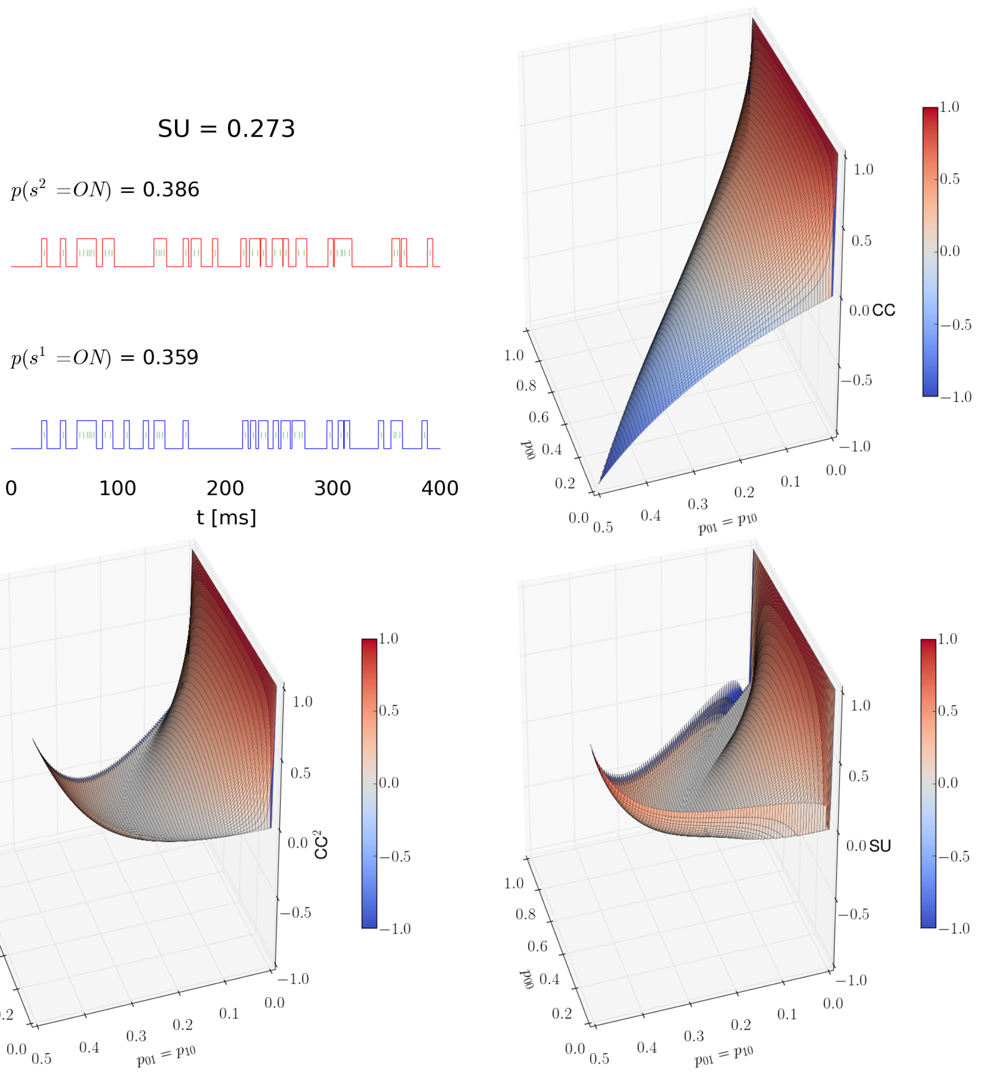


Figure 4.12.: CC vs. SU. **Top left:** Example spike trains (green) from two neurons with 5 shared and 2 private inputs and their state variables (red and blue). The positive CC and SU reflect the ratio of shared to total inputs, but the SU is lower since it is a convex function of $s/(s+p)$. The other three subplots depict various correlation measures over a large subspace of Ω: $p_{00} \in [0, 1]$ and $p_{01} = p_{10} \in [0, 0.5]]$. Figure taken from Bytschok (2011). **Top right:** CC. **Bottom left:** Squared CC. **Bottom right:** SU. Note the qualitative similarity between the squared CC and the SU.

In addition to being symmetric, it can be shown that the SU is normalized to the unit interval:

$$\tilde{I}(S_1, S_2) \in [0, 1] \quad . \tag{4.163}$$

Figure 4.12 shows an example of binarized neuron spike trains and their calculated SU. In contrast to the CC, which increases linearly with the ratio of private to total inputs (Equation 4.143), the SU is clearly a concave function of this ratio, with a value of 0.273 for $s/(s + p) = 5/7$ – as compared to the independent case $\tilde{I}(s/(s + p) = 0) = 0$ and the fully correlated case $\tilde{I}(s/(s+p) = 1) = 1$. Since calculating the SU requires the knowledge of the same probabilities ${}^1p_1$, ${}^2p_1$, p_{00}, p_{01}, p_{10} and p_{11} as the CC, we will restrict ourselves to only predicting the former, knowing that we could predict the latter as well.

In particular, we observe an interesting connection between the SU and the CC. Figure 4.12 depicts a sweep over a subspace of the entire spectrum of probability distributions over Ω (p_{10} and p_{01} are kept equal in order to allow a plot over only two degrees of freedom). Indeed, it turns out that the SU looks very similar to the square of the CC over the entire spectrum of probability distributions over Ω. However, we need to stress that this is not due to a deep mathematical relationship between the SU and the CC, but rather a direct consequence of the conditions imposed on our correlation measure: symmetry with respect to S_1 and S_2, normalization to $[0, 1]$ and extreme values for zero ($\tilde{I} = \rho^2 = 0$) and total ($\tilde{I} = \rho^2 = 1$) (anti)correlation.[12]

Before we move on, we should point out that the denomination of "symmetric uncertainty" is a quite unfortunate one. Intuitively, a high uncertainty should point to independence, since for two independent RVs, measuring one leaves us uncertain about the state of the other. For the SU, the exact opposite is the case. Nevertheless, we shall continue using this (declaredly confusing) denomination for historical reasons (Bytschok, 2011; Witten and Frank, 2005).

4.4.5. Spike-Based Correlations from Free Membrane Potential Statistics

We have previously defined the free membrane potential of a neuron under certain stimulus conditions to represent the would-be value of the membrane potential if the threshold was set to infinity. When comparing the time course of the membrane potential with the one of the free membrane potential (Figure 4.13), we can immediately identify a connection of the latter to the state variable of the neuron: the states $S = 1$ appears to coincide with the periods when the effective membrane potential $\tilde{u}$ is suprathreshold:

state variable inferred from suprathreshold free membrane potential

$$S(t) = 1 \overset{?}{\Longleftrightarrow} \tilde{u} \geq \vartheta \quad . \tag{4.164}$$

If this is, indeed, the case, then any calculations involving state variables could be performed using free membrane potentials, the (joint) distributions of which we are able to fully specify in closed form (Section 4.4.2). More precisely, the joint and marginal state

[12] Herein lies the main difference between the CC and the SU. The former is a measure of correlation, which discerns between positive ($\rho > 0$) and negative ($\rho < 0$) correlation, whereas the latter is a measure of dependence (mutual information). As usual, correlation implies dependence ($\rho \in \{-1, 1\} \Rightarrow \tilde{I} = 1$). Conversely, independence implies zero correlation ($\tilde{I} = 0 \Rightarrow \rho = 0$).

distributions are then given by

marginal and joint state distributions inferred from the free membrane potential distribution

$$ {}^{i}p_1 := p(S_i = 1) \overset{!}{=} p(\tilde{u}_i \geq \vartheta) = \int_{\vartheta}^{\infty} f(\tilde{u}, \mu_i, \sigma_i^2) d\tilde{u} \quad , \tag{4.165} $$

$$ p_{00} := p(S_1 = 0, S_2 = 0) \overset{!}{=} p(\tilde{u}_1 \leq \vartheta, \tilde{u}_2 \leq \vartheta) = \int_{-\infty}^{\vartheta} \int_{-\infty}^{\vartheta} f(\tilde{\boldsymbol{u}}, \boldsymbol{\mu}, \boldsymbol{\Sigma}) d\tilde{u}_1 d\tilde{u}_2 \quad , \tag{4.166} $$

$$ p_{01} := p(S_1 = 0, S_2 = 1) \overset{!}{=} p(\tilde{u}_1 \leq \vartheta, \tilde{u}_2 \geq \vartheta) = \int_{-\infty}^{\vartheta} \int_{\vartheta}^{\infty} f(\tilde{\boldsymbol{u}}, \boldsymbol{\mu}, \boldsymbol{\Sigma}) d\tilde{u}_1 d\tilde{u}_2 \quad , \tag{4.167} $$

$$ p_{10} := p(S_1 = 1, S_2 = 0) \overset{!}{=} p(\tilde{u}_1 \geq \vartheta, \tilde{u}_2 \leq \vartheta) = \int_{\vartheta}^{\infty} \int_{-\infty}^{\vartheta} f(\tilde{\boldsymbol{u}}, \boldsymbol{\mu}, \boldsymbol{\Sigma}) d\tilde{u}_1 d\tilde{u}_2 \quad , \tag{4.168} $$

$$ p_{11} := p(S_1 = 1, S_2 = 1) \overset{!}{=} p(\tilde{u}_1 \geq \vartheta, \tilde{u}_2 \geq \vartheta) = \int_{\vartheta}^{\infty} \int_{\vartheta}^{\infty} f(\tilde{\boldsymbol{u}}, \boldsymbol{\mu}, \boldsymbol{\Sigma}) d\tilde{u}_1 d\tilde{u}_2 \quad , \tag{4.169} $$

where f represents a Gaussian PDF (the arguments show whether it is univariate or bivariate). We have previously derived closed-form solutions for the mean μ (mean vector $\boldsymbol{\mu}$) and variance σ^2 (covariance matrix $\boldsymbol{\Sigma}$) for different neuron models in Sections 4.3.4 and 4.4.2. It can be easily verified that $\sum_{(xy)\in\Omega} p_{xy} = 1$. In order for the above equations to be useful, we need to ensure that the $\tilde{u}$-based prediction of the above probabilities is in accordance to the probabilities calculated from the spike trains via Equation 4.150. In particular, we need to determine numerical values for two free variables: the onset of the 1-state associated with a spike and its duration τ_{on}.

In our definition of the 1-state (Equation 4.150), we have assumed the switch from $S = 0$ to $S = 1$ to occur synchronously to the moment of spiking. In light of the connection between $S(t)$ and $\tilde{u}(T)$, we can now motivate this properly, since we want the 1-state to coincide with a suprathreshold free membrane potential – and it is the threshold crossing of the latter that triggers a spike (Figure 4.14).

The length τ_{on} of the box function which determines the 1-state duration for a single spike is a little more difficult to find. Intuitively, it should be in the order of the longest time constant that governs the evolution of the membrane potential (i.e., the falling flank of a PSP), since this time constant then also determines the time that the free membrane potential spends above the threshold following a spike. However, the free membrane potential can be suprathreshold for extended periods, during which the neuron would spike with a high rate (burst). A fixed τ_{on} should therefore remain valid for both single spike events as well as bursts of arbitrary length. For now, we shall determine τ_{on} experimentally, but we will provide a more thorough (analytical) approach to the calculation of burst lengths in Section 6.5.3.

The most elementary requirement is that, for a single neuron, the occurrence probability p_1 of the 1-state should be the same when calculated from both the free membrane potential (with the condition $\tilde{u} \geq \vartheta$) and the spike train (by convolution with $\mathrm{box}(t|0, \tau_{\mathrm{on}}, 1)$). While the former is fixed, we can study the latter for various input scenarios by sweeping

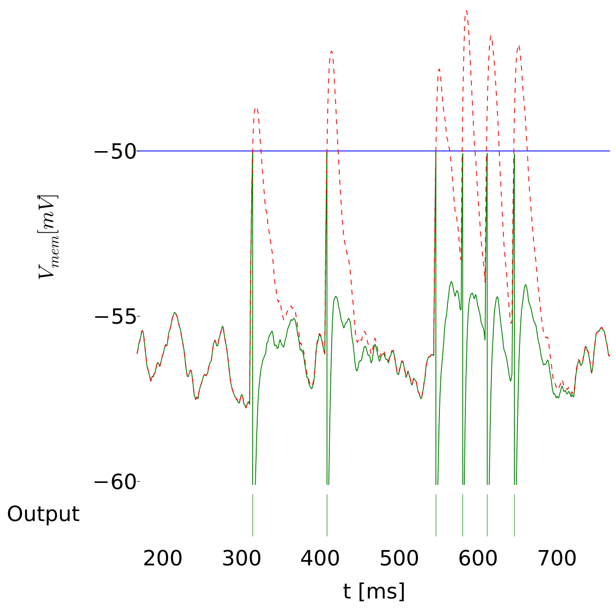


Figure 4.13.: Free membrane potential $\tilde{u}$ (dashed curve) vs. “true” membrane potential u (solid curve). Spiking (which corresponds to $S = 1$ in the state picture) coincides with a suprathreshold free membrane potential $\tilde{u} \geq \vartheta$. Figure taken from Bytschok (2011).

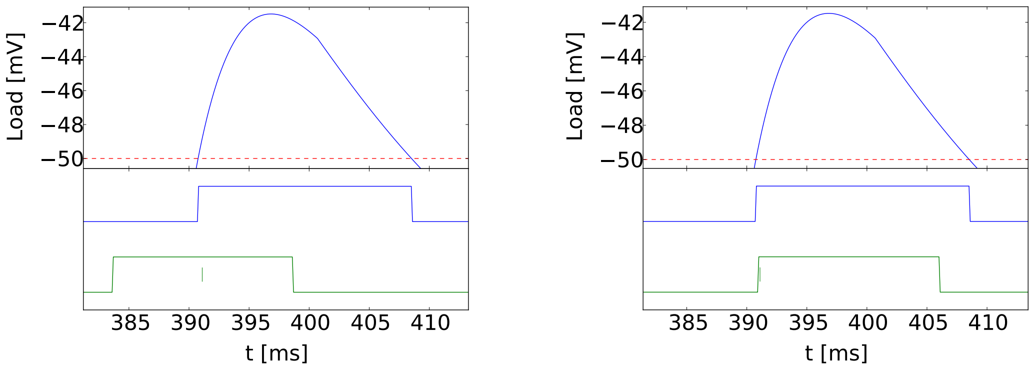


Figure 4.14.: A threshold crossing of the free membrane potential $\tilde{u}$ triggering a single output spike. The 1-state defined by the suprathreshold period of $\tilde{u}$ is depicted in blue. The same state resulting from the convolution of the output spike (train) with a box function shown in green. **Left:** Box function is centered around the spike, i.e., box $= \text{box}(t| - \tau_{\text{on}}/2, \tau_{\text{on}}/2, 1)$. **Right:** The left flank of the box function coincides with the timing of the spike, i.e., box $= \text{box}(t|0, \tau_{\text{on}}, 1)$. Naturally, the best overlap is achieved for latter case. Figure taken from Bytschok (2011).

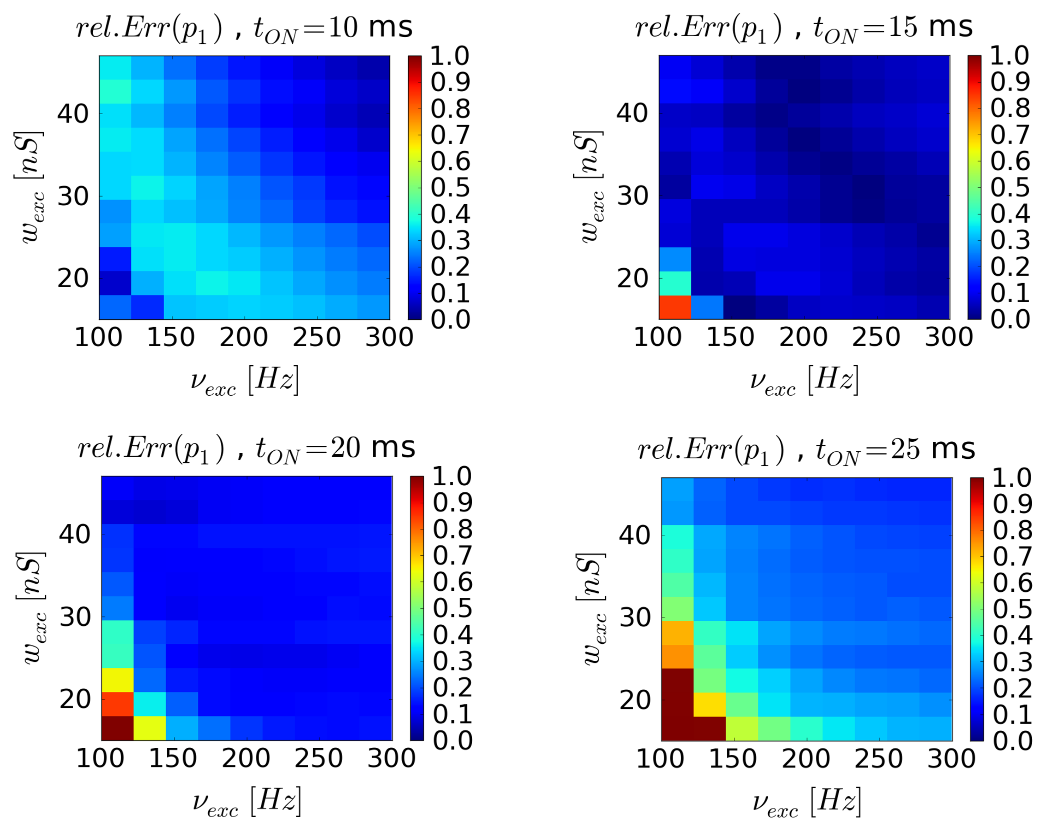


Figure 4.15.: Relative error $\Delta p_1 = \frac{p_1^{(\text{Eq. 4.165})} - p_1^{(\text{Eq. 4.150})}}{p_1^{(\text{Eq. 4.165})}}$ calculated from the free membrane potential distribution ($p(\tilde{u} \geq \vartheta)$, Equation 4.165) as compared to the p_1 obtained from convolving the output spike train with a box function of duration τ_{on} (Equation 4.150). The relative error is calculated for various input weights w and rates nu, as well as for different τ_{on}. The lowest relative error is obtained for $\tau_{\text{on}} = 15\,\text{ms}$ (top right). The relatively large errors that can be observed for weak input (bottom left corners of the plots) are due to non-Gaussian $\tilde{u}$-distributions. Figure taken from Bytschok (2011).

over the firing rates and synaptic weights of the inputs. Figure 4.15 shows the result of such a sweep, where only the excitatory weights and rates were changed, in order to ensure a fast coverage of the entire $p_1 \in [0, 1]$ interval. As it turns out, the difference (relative error) between the $\tilde{u}$-based and the spike-based calculation of p_1 has a minimum around $\tau_{\text{on}} = 15\,\text{ms}$, which corresponds roughly to the membrane time constant. The relative errors only become large for low rates and weights, which is expected: firstly, the Gaussian approximation of the free membrane potential distribution does not hold for low-rate stimuli; secondly, when stimulation is weak, spike events are rare and the finite simulation times come into play.

determination of τ_{on} from p_1

We can now test our chosen value of $\tau_{\text{on}} = 15\,\text{ms}$ by predicting the joint states p_{xy} with Equations 4.166-4.169 and comparing the results to the ones obtained from simulations (spike trains convolved with $\mathrm{box}(t|0,\tau_{\mathrm{on}},1)$). We repeat our sweeps over the input parameters w and ν from before. The 00- and 11-state distributions are compared in Figure 4.16. The distributions of the "mixed" 01- and 10-states are compared in Figure 4.17 (since the total input rates and weights received by the two neurons are identical, $p_{01} = p_{10}$, so only one of them is shown).

The p_{00}- and p_{11}-surfaces in the w-ν parameter space behave, qualitatively, as expected. Since spiking intensifies for increasing input rates and weights, p_{11} increases monotonically with w and ν. Conversely, p_{00} is monotonically decreasing with w and ν. The box function duration τ_{on} does not affect this qualitative relationship, but rather acts as a multiplier for p_{11} and $1 - p_{00}$. The same monotonicity in w and ν is expected from the suprathreshold (subthreshold) probability mass of the free membrane potential distribution (see Section 4.3.4).

monotonicity of p_{11} and p_{00}

The iso-probability loci for the p_{00}- and p_{11}-surfaces are represented by "pixels" with identical color. If our assumption about the equivalence between suprathreshold-$\tilde{u}$-states and spike-train-derived 1-states is correct, then an unchanged subthreshold (suprathreshold) probability mass of the free membrane potential distribution should leave all p_{xy} unchanged as well. The subthreshold probability mass is given by the CDF of the Gaussian $\tilde{u}$-distribution at the threshold:

$$F(\vartheta) = \frac{1}{2}\left[1 + \mathrm{erf}\left(\frac{\vartheta - \mu}{\sqrt{2}\sigma}\right)\right] \quad . \tag{4.170}$$

In Section 4.3.4 we have shown that, for both CUBA and COBA neurons in a low-conductance state, the mean μ of the free membrane potential scales with $w\nu$ and its variance σ^2 with $w^2\nu$. This implies that $F(\vartheta)$ does not change for

$$\frac{\vartheta - \mu}{\sigma} = \mathrm{const} \quad , \tag{4.171}$$

iso-probability loci of p_{11} and p_{00}

which gives us the iso-probability loci up to some multiplicative constants α and β:

$$w = \frac{\vartheta}{\alpha\nu + \beta\sqrt{\nu}} \quad . \tag{4.172}$$

Indeed, we can observe that the iso-probability loci for both p_{00} and p_{11} are approximately hyperbolic. For the particular value of $\tau_{\mathrm{on}} = 15\,\mathrm{ms}$, we can see that for the entire investigated w-ν parameter space, the values predicted from the $\tilde{u}$-distribution are in excellent agreement with the one obtained from the simulated spike trains (see Figure 4.16).

w-ν-dependence of p_{01} and p_{10}

The p_{01}- and p_{10}-surfaces paint a similar picture. Their qualitative dependence on the input weights and rates again corresponds to our intuitive expectations. For very weak and very strong background stimulation, the 00- and 11-states are, respectively, predominant, leaving the 01- and 10-states with only small probabilities. For intermediate w and ν values, p_{01} and p_{10} reach a maximum.

τ_{on}-dependence of p_{01} and p_{10}

Their non-monotonic dependence on τ_{on} is slightly less straightforward, but still intuitive. Consider the case of high w and ν, where the neurons both spike almost continuously: by decreasing τ_{on}, the relative occurrence of 1-states is reduced, thereby implicitly reducing the occurrence of 11-states. The missing probability mass is then distributed among the other three joint states, thereby increasing p_{01} and p_{10} as well. The converse argument applies for the weak-input region of the w-ν space.

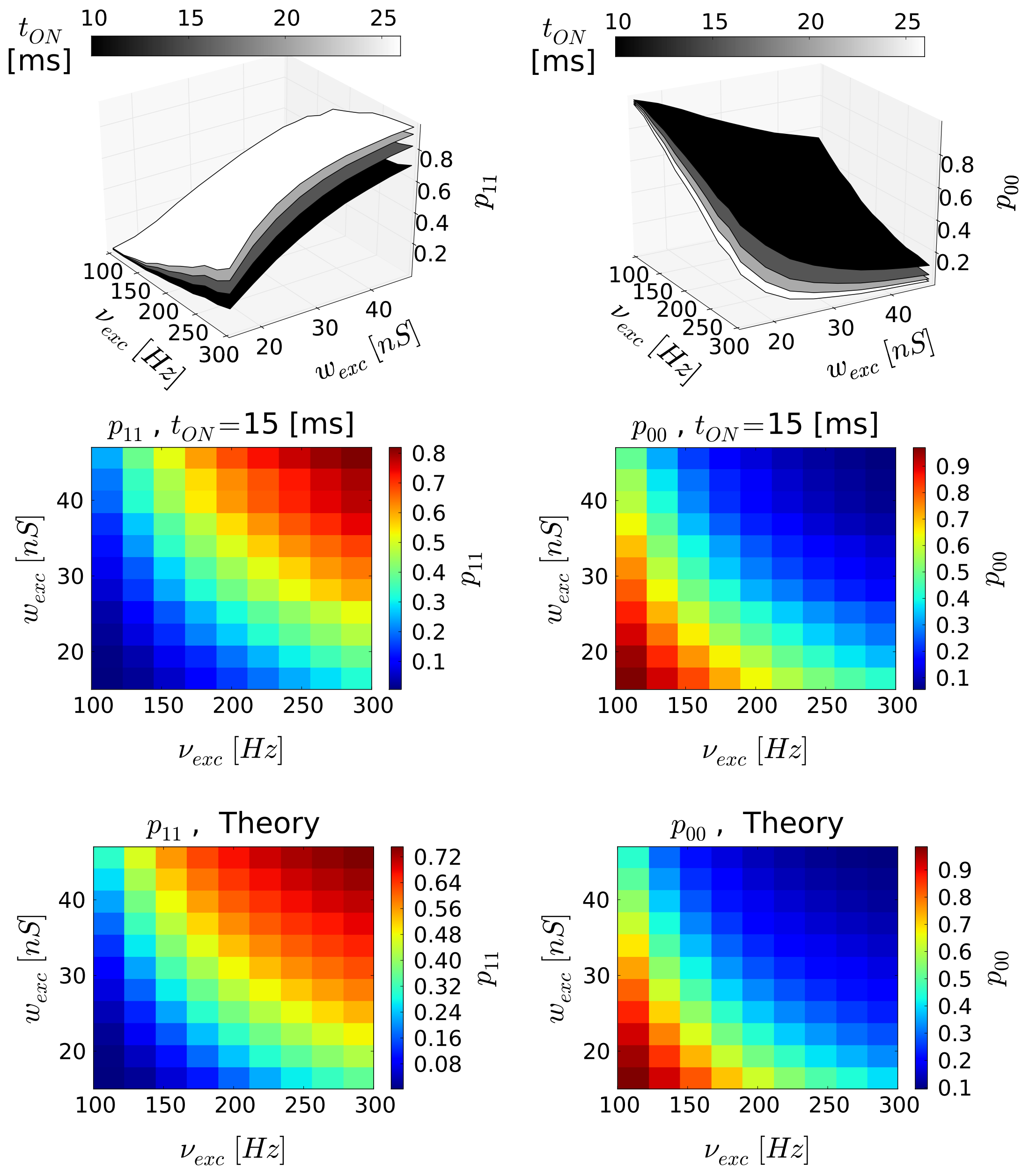


Figure 4.16.: 00- and 11-state distributions across a range of different input rates and weights. **Top:** p_{11} and p_{00} surfaces obtained from simulated spike trains for $\tau_{\text{on}} \in \{10, 15, 20, 25\}$ ms. The exact value of τ_{on} does not change the qualitative aspect of the $p_{(xy)}$-surfaces. As expected, higher τ_{on} values lead to a higher p_{11}, while lower τ_{on} values increase p_{00}. **Middle:** p_{11} and p_{00} color plots obtained from simulated spike trains for $\tau_{\text{on}} = 15\,\text{ms}$. The iso-probability loci (areas of same color) are of approximately hyperbolic shape, as predicted by the theoretical calculation based on the free membrane potential. **Bottom:** Theoretical prediction of p_{11} and p_{00} from Equations 4.166 and 4.169. With the previously found value of $\tau_{\text{on}} = 15\,\text{ms}$, we can observe a very good agreement between the $p_{(xy)}$ calculated from the spike trains obtained from simulations and the theoretical prediction from the $\tilde{u} \geq \vartheta$-condition. Figure taken from Bytschok (2011).

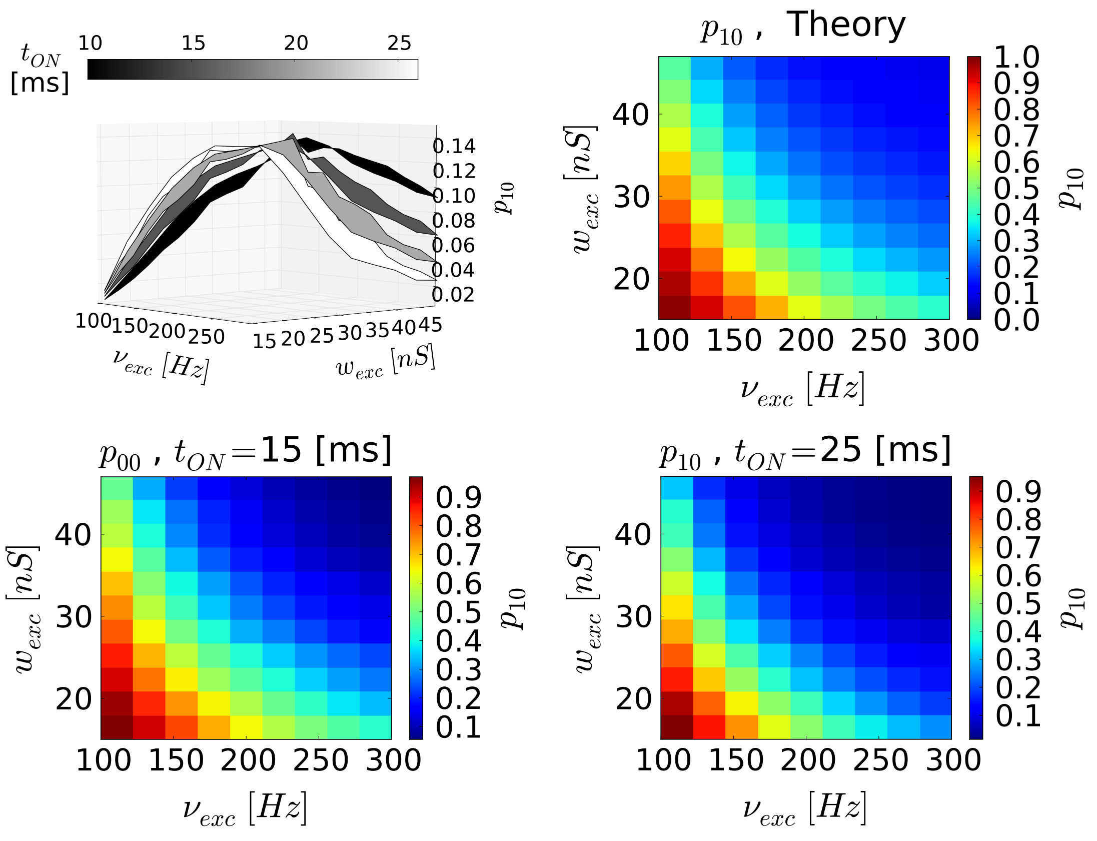


Figure 4.17.: 10-state distribution across a range of different input rates and weights. The 01-state behaves identically due to the symmetry of the experimental setup (the two neurons have identical stimulation parameters). **Top left:** p_{10} surfaces obtained from simulated spike trains for $\tau_{\text{on}} \in \{10, 15, 20, 25\}$ ms. As before, the exact value of τ_{on} does not change the qualitative aspect of the $p_{(xy)}$-surfaces. However, in contrast to p_{00} and p_{11}, τ_{on} scales the "mixed" state probabilities p_{01} and p_{10} non-monotonically. **Top right:** Theoretical prediction of p_{10} from Equation 4.168. Due to the same invariant suprathreshold probability mass considerations as for p_{00} and p_{11}, the iso-probability loci (areas of same color) are of approximately hyperbolic shape. **Bottom:** p_{10} color plots obtained from simulated spike trains for $\tau_{\text{on}} = 15$ ms and $\tau_{\text{on}} = 25$ ms. For the previously found value of $\tau_{\text{on}} = 15$ ms, the agreement between the $p_{(xy)}$ calculated from the spike trains obtained from simulations and the theoretical prediction from the $\tilde{u} \geq \vartheta$-condition is very good – and visibly better than for $\tau_{\text{on}} = 25$ ms.

iso-probability loci of p_{01} and p_{10}

The iso-probability-locus argument derived from the invariant free membrane potential distribution applies for the "mixed" states as well. As expected, we find the same hyperbolic iso-probability loci for the p_{01}- and p_{10}-surfaces as we did before for the p_{00}- and p_{11}-surfaces. The agreement between theory and simulations is, again, very good (see Figure 4.17)

4.4.6. Spike-Based Correlations: Theory vs. Experiment

The good predictions we were able to obtain for the marginal and joint probabilities ${}^{i}p_x$ and p_{xy} now enable us to predict the SU as well. In particular, we expect similarly good results as we have presented in the previous section, since the calculation of the SU only requires the abovementioned probabilities. Here, we only show results for COBA neurons, but the conclusions hold for CUBA neurons as well.

w-ν-s-dependence of the SU

We start by predicting the behavior of the SU for various values of the input rates and weights, as well as for several shared-to-total input channel ratios (Figure 4.18). As expected, for any configuration of input rates and weights, the SU increases with the proportion of shared channels. The increase is also not linear, as the p_{xy}-dependence of the SU (Figure 4.12) already suggests. The dependence on the input rates and weights, however, is more complex and not as intuitive. In particular, we note that the input firing rates have a significantly stronger impact than the input synaptic weights. These theoretically predicted dependencies represent interesting findings, if validated by experiments/simulations.

This has been done, and the comparison is depicted in Figure 4.19. We note that the overlap of the state variable $S(t)$ calculated from the free membrane potential time course compared to the one obtained from the output spike train is, in general, quite good, with several small exceptions due to the relatively large effective time constant, which causes slight disparities between the free and the "true" membrane potential. The theoretical prediction of the SU (denoted $SU_{\mathcal{N}}$)[13] fits almost perfectly with the SU calculated from the state variables derived from the simulated suprathreshold free membrane potential (denoted SU_{load})[14]. The SU values derived from the spike train (denoted SU_{sim}) deviate only slightly and within the error margins imposed by the finite simulation time. The good agreement between the theoretical prediction and the simulation data appears to hold for a wide range of input weights/rates configurations.

We need to mention that we suspect the deviations between theory and experiment to be, to some extent, systematic. The reason lies mainly in the abovementioned difference between the free and "true" membrane potential, which causes additional short 1-states that are not reflected in the spike trains. These become less pronounced for faster membranes and can also be tackled by refined theoretical methods, which are the subject of ongoing investigations.

Regimes also exist where the disparity between theory and simulations is larger, as can

[13] The notation represents a reference to the Gaussian approximation for the free membrane potential PDF.

[14] We have kept this notation for historical reasons. Earlier versions of our theory used a modified synaptic conductance trace instead of the free membrane potential, for which we used the term "load function".

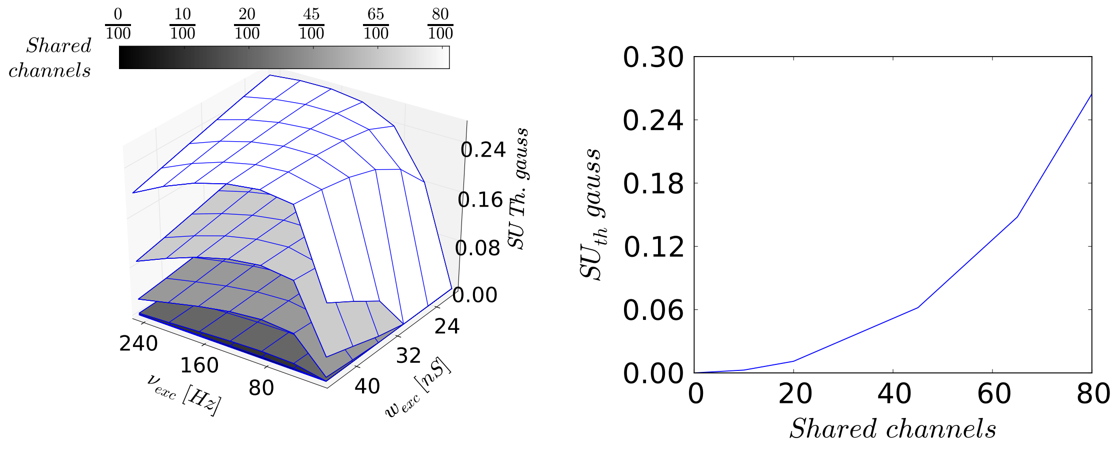


Figure 4.18.: Theoretical prediction of the SU for various input parameter configurations. **Left:** Sweep over the excitatory input rates and weights. The inhibitory ones were kept constant at $\nu_{\mathrm{inh}} = 50\,\mathrm{Hz}$ and $w_{inh} = 35\,\mathrm{nS}$. As expected, the SU increases with the ratio of shared to total inputs. The dependence of the SU on the input firing rates and weights is complex and not acessible by straightforward intuition. We note, however, that shape of the different SU surfaces for various shared-to-total ratios remains qualitatively similar and differs only in amplitude. **Right:** $SU(s)$ for a total number of $s + p = 100$ input channels with parameters $\nu_{\mathrm{inh}} = 50\,\mathrm{Hz}$, $w_{inh} = 35\,\mathrm{nS}$, $\nu_{\mathrm{exc}} = 90\,\mathrm{Hz}$ and $w_{exc} = 30\,\mathrm{nS}$. Figure taken from Bytschok (2011).

violation of Gaussian approximation at low ν

be seen in Figure 4.20. This happens, however, only for low input rates, which is easily understood when considering that in this regime, the CLT which we have invoked for motivating the Gaussian distribution of the membrane potential (see Section 4.3.2) no longer holds. However, the agreement between the SU calculated from the free membrane potential and the one obtained from the output spike trains (both simulated) validate our assumption that the two are practically equivalent generators of the state variables $S_i(t)$.

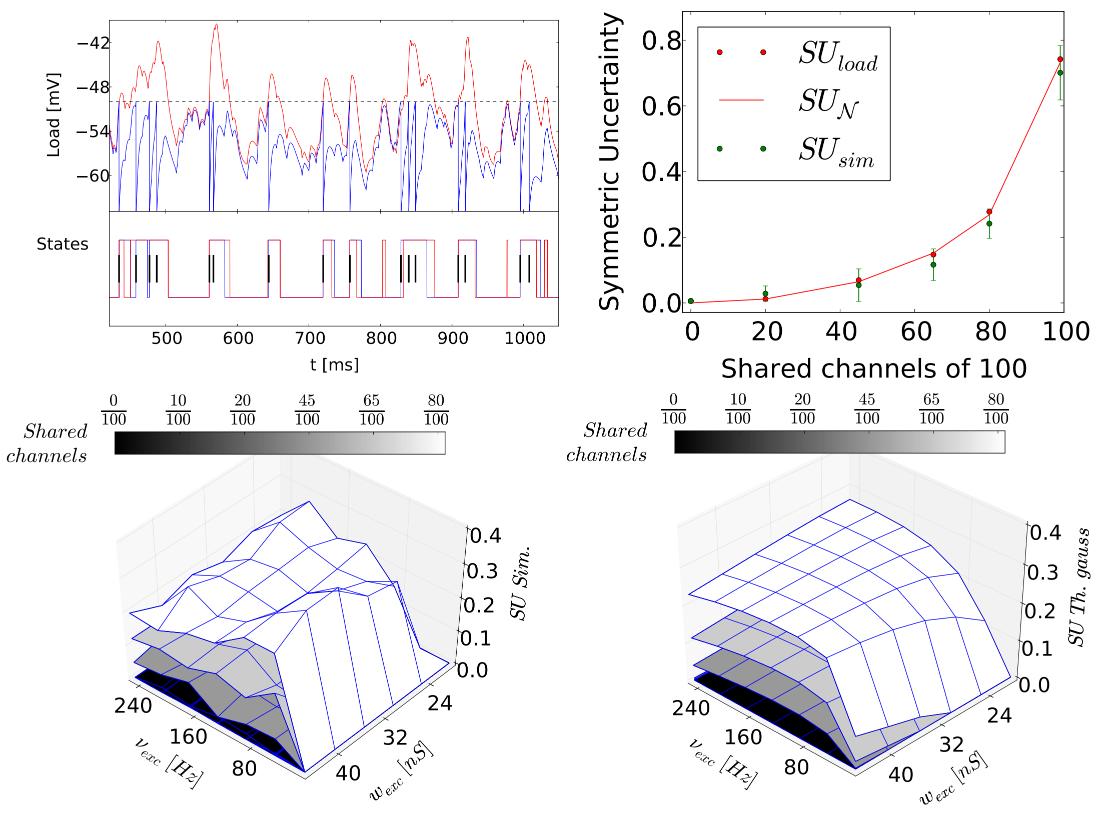
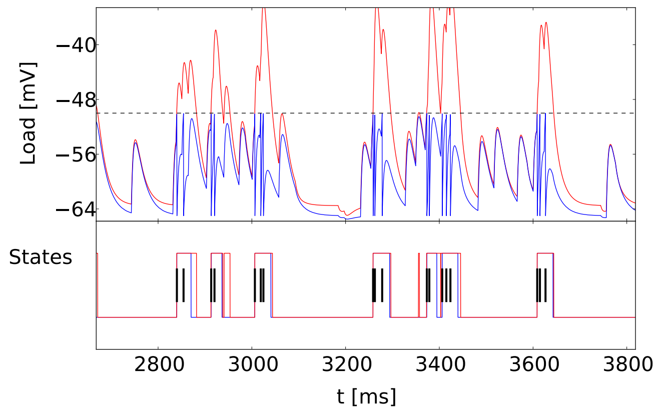
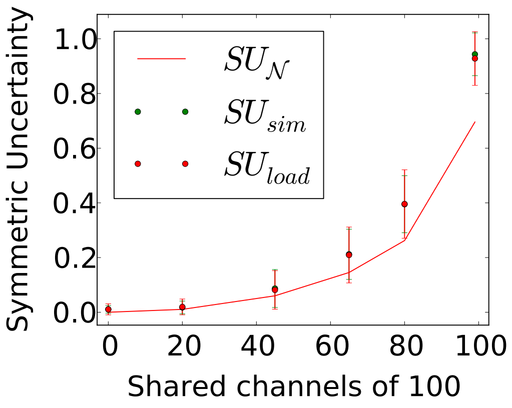


Figure 4.19.: State variables and their SU: theory vs. simulations. **Top left:** Membrane potential of a neuron under Poisson bombardment. We use two ways to calculate the temporal evolution of the state variable $S(t)$. Firstly, we can translate the time spent by free membrane potential in the suprathreshold regime directly to $S(t) = 1$ (red traces), which is also used in the theoretical prediction of the SU. Secondly, we can use the "true" membrane potential, i.e., the spike trains it generates, to calculate the state variables (blue traces) by convolution with a box function (Equations 4.148 and 4.148). We note that the two methods yield very similar results, with only few exceptions that can be traced back to the relatively large effective time constant of the membrane. **Top right:** $SU(s)$ for a total number of $s + p = 100$ input channels with parameters $\nu_{\text{inh}} = 170\,\text{Hz}$, $w_{inh} = 20\,\text{nS}$, $\nu_{\text{exc}} = 90\,\text{Hz}$ and $w_{exc} = 30\,\text{nS}$. The theoretical prediction of the SU (denoted $SU_{\mathcal{N}}$) is represented by the red curve. The SU calculated from the suprathreshold free membrane potential trace (denoted SU_{load}) is represented by red dots. As expected from the successful prediction of free membrane potential distributions (Section 4.4.3), the agreement between these two datasets is very good as well. The SU calculated from the output spike trains (denoted SU_{sim}) is represented by green dots. Error bars represent the standard deviation over multiple trials (the red error bars are smaller than the symbols). The analytical prediction fits the simulation data well within the measured errors. The remaining differences are, indeed, systematic, and can be traced back to the relatively large effective time constant. **Bottom left:** Measured values of the SU from simulated spike trains for various values of the shared-to-total input ratio, input firing rate and input synaptic weight. The jitter in the measured values is due to finite simulation times. **Bottom right:** Predicted values of the SU for the same parameter sets as in the simulations. Figure taken from Bytschok (2011).





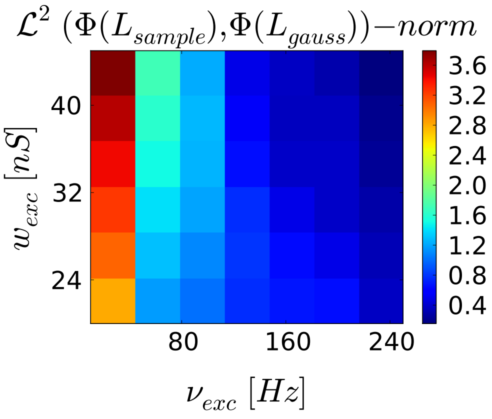


Figure 4.20.: The SU at low firing rates: theory vs. simulation. **Top:** Membrane potential of a neuron under low-rate Poisson bombardment. We use the same color coding as in Figure 4.19. We note again the good agreement between the state variable calculated from the free membrane potential (red) and the one calculated from the spike trains (blue). **Bottom left:** $SU(s)$ for a total number of $s + p = 100$ input channels with parameters $\nu_{\text{inh}} = 10\,\text{Hz}$, $w_{inh} = 20\,\text{nS}$, $\nu_{\text{exc}} = 20\,\text{Hz}$ and $w_{exc} = 50\,\text{nS}$. As expected from the good agreement in the state variable calculation (see above), SU_{load} and SU_{sim} have almost exactly the same values. The predicted $SU_{\mathcal{N}}$, however, lies systematically below the simulated values. The reason for this discrepancy lies in the low input rates, for which the Gaussian approximation (and therefore also $SU_{\mathcal{N}}$) no longer holds. **Bottom right:** Validity of the Gaussian approximation for various sets of input rates and weights. Here, we have plotted the Euclidean (L^2) norm of the difference between the simulated free membrane potential distribution and the Gaussian distribution assumed by the theory. As expected, the dependence on the input weights is only weak, since these can be reduced to a linear rescaling of the membrane potential. The input rates, however, are critical for the CLT argument we have provided in Section 4.3.2. Figure taken from Bytschok (2011).

4.5. Conclusions and Outlook

In this chapter, we have discussed the dynamics of both conductance- (COBA) and current-based (CUBA) LIF neurons with exponential synaptic kernels. We have derived closed-form expressions for the PSP shapes, which are exact for the CUBA case and good approximations for the COBA case in the high-conductance state. Building on these, we were able to derive closed-form expressions for the current, conductance and membrane potential distributions of LIF neurons and have shown them to accurately match simulation results. We then extended this formalism to study neural response correlations due to shared Poisson sources. We were able to find good predictions of the free membrane potential correlation coefficient, as well as for the SU of spike trains produced by pairs of correlated neurons.

Our results were obtained for one particular synaptic kernel shape which is often used in computational and theoretical neuroscience. However, the exact same formalism can be applied to derive expressions for arguably more biological PSC shapes such as α-functions or differences of exponentials. This becomes particularly interesting for understanding the behavior of neuromorphic circuits, where the precise PSC shape can be either measured or inferred from the design specifications and is usually not an exact exponential function of time. Having a closed-form expression for the PSP shape that explicitly depends on controllable hardware parameters is of obvious advantage for the calibration procedure.

higher-order correlations

Here, we have only studied pairwise correlations, but our framework can be extended to higher-order correlations as well. In particular, the analytical predictions we were able to derive here will be useful for understanding and quantifying the effects of shared-input correlations on the sampling networks we discuss in Chapter 6 and possibly also for finding appropriate compensation mechanisms. Such mechanisms, derived from the single-neuron statistics, have already been applied successfully to the functional networks discussed in Chapter 5.

built-in homeostasis

One particularly interesting outcome was the explanation of the counterintuitive features of membrane potential distributions in the high-conductance state, in particular the inverse proportionality of the distribution width to the input firing rate and its near-independence of the synaptic input weights. As already mentioned before, this can act as a homeostatic mechanism in plastic networks and, if shown to be functionally sufficient for models of unsupervised learning such as the ones proposed in, e.g., Habenschuss et al. (2012); Nessler et al. (2009, 2013), may save significant amounts of hardware and software resources that are used for additional homeostatic mechanisms, as proposed in, e.g., Breitwieser (2015).

ACP formalism

Our prediction of spike train correlations rely on the assumption that a suprathreshold free membrane potential is, on average, approximately equivalent to an $S(t) = 1$ state (Equation 4.164) for a particular choice of the 1-state kernel duration τ_{on}. We have determined this duration experimentally for a particular choice of parameters and have shown our initial assumption to represent a good approximation. However, as we exemplify in Figure 4.20, the initial assumption is not true for arbitrary parameter sets. In Chapter 6, we analyze the statistics of single neuron membranes from a different perspective (propagation of membrane autocorrelations, short: ACP), in particular with a well-defined interpretation of τ_{on}, and obtain very accurate results for broad parameter ranges. It appears likely that the ACP formalism can be applied to ensembles of neurons as well and we are actively investigating this idea at this time.

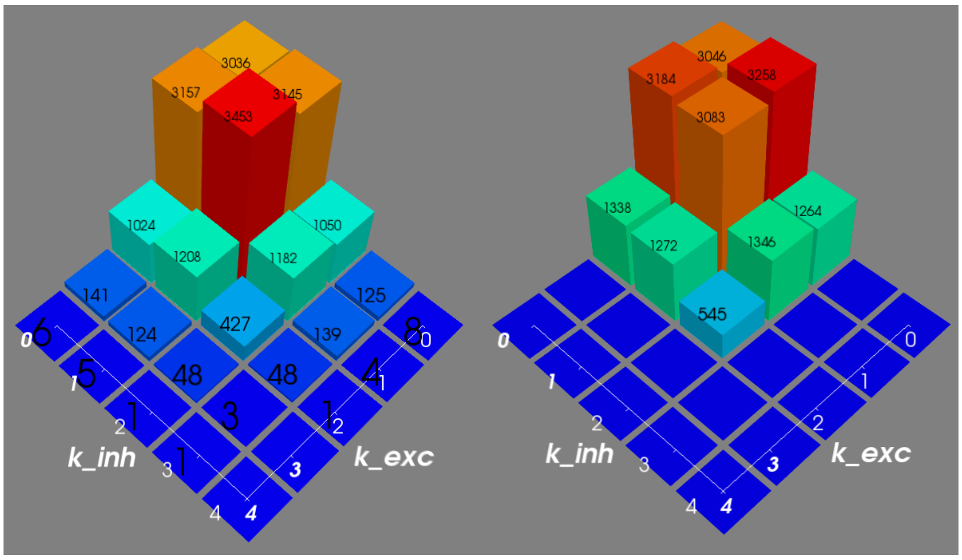


Figure 4.21.: Exemplary results of the graph-based algorithm for local optimization of the source-to-neuron mapping. The cost function to be minimized is multidimensional and is given by the number of shared sources per neuron pair. In this example, the total number of network neurons is 192 and the total number of available sources is 64, out of which 32 are excitatory and 32 are inhibitory. Each neuron must have a presynaptic pool of 4 excitatory and 4 inhibitory sources. The histograms (colored bars and black numbers) show the number of neuron pairs that share a particular number of excitatory and inhibitory sources (white numbers). **Left:** Random choice of input pools. **Right:** Input pools chosen by the graph-based heuristic algorithm. The algorithm succeeds in providing all neurons with the required number of sources while eliminating all pool configurations that have a pairwise overlap larger than 2. Figure taken from Petrovici and Bill (2009).

When neural receptive fields become sufficiently large, shared-input correlations become inevitable in finite-size neural substrates. This is particularly relevant for neuromorphic devices, where internal Poisson generators are expensive in terms of chip area and external noise sources must occupy the already limited bandwidth between the neuromorphic device and the host computer. Under these constraints, it often becomes an important problem to find a mapping between the pool of noise sources and the neurons in the functional network that minimizes these correlations. This can be expressed as an optimization problem where one searches for a minimally overlapping set of neuron input pools. In turn, this problem can be shown to be mathematically equivalent to the well-known maximum independent vertex set problem from graph theory (Petrovici and Bill, 2009). This problem is known to be NP-hard, but we have designed an heuristic algorithm (see also Halldórsson and Radhakrishnan, 1997, for a related discussion) that is able to efficiently find good solutions for small-scale networks (see Figure 4.21 for an example with 192 neurons and 64 sources). The subsets found by running the algorithm with varying starting parameters define multiple sets of shared and private input configurations, which can then be fed into our SU prediction to find a locally optimal noise input configuration.

graphical optimization problem

5. Cortical Models on Neuromorphic Hardware

As I see it, the only way of overcoming this magical view of what "I" and consciousness are is to keep on reminding oneself, unpleasant though it may seem, that the "teethering bulb of dread and dream" that nestles safely inside one's own cranium is a purely physical object made up of completely sterile and inanimate components, all of which obey exactly the same laws that govern all the rest of the universe [...]. Only if one keeps bashing up against this disturbing fact can one slowly begin to develop a feel for the way out of the mystery of consciousness: that the key is not the stuff out of which brains are made, but the patterns that can come to exist inside the stuff of a brain.

Douglas Hofstadter, Gödel, Escher, Bach, 1999

Along with the many advantages it offers, the neuromorphic approach also comes with limitations of its own. These have various causes that lie both in the hardware itself and the control software. We will later identify these causes, which we henceforth refer to as distortion mechanisms. The neural network emulated by the hardware device can therefore differ significantly from the original model, be it in terms of pulse transmission, connectivity between populations or individual neuron or synapse parameters. We refer to all the changes in network dynamics (i.e., deviations from the original behavior defined by software simulations) caused by hardware-specific effects as hardware-induced distortions.

hardware-induced distortions

Due to the complexity of state-of-the-art neuromorphic platforms and their control software, as well as the vast landscape of emulable neural network models, a thorough and systematic approach is essential for providing reliable information about causal mechanisms and functional effects of hardware-induced distortions in model dynamics and for ultimately designing effective compensation methods. In this chapter, we discuss this systematic analysis and compensation techniques for several hardware-specific distortion mechanisms.

compensation techniques

First and foremost, we identify and quantify the most important sources of model distortions. We then proceed to investigate their effect on network functionality. In order to cover a wide range of possible network dynamics, we have chosen three very different cortical network models to serve as benchmarks. In particular, these models implement several prototypical cortical paradigms of computation, relying on winner-take-all structures (attractor networks, Section 5.3), precise spike timing correlations (synfire chains, Section 5.4) or balanced activity (self-sustained asynchronous irregular states, Section 5.5).

benchmark

functionality criteria

For every emulated model, we define a set of functionality criteria, based on specific aspects of the network dynamics. This set should be complex enough to capture the characteristic network behavior, from a microscopic (e.g., membrane potentials) to a mesoscopic level (e.g., firing rates) and, where suitable, computational performance at a specific task. Most importantly, these criteria need to be precisely quantified, in order to facilitate an accurate comparison between software simulations and hardware emulations or between different simulation/emulation back-ends in general. The chosen functionality criteria should also be measured, if applicable, for various relevant realizations (i.e., for different network sizes, numbers of functional units etc.) of the considered network.

Because multiple distortion mechanisms occur simultaneously in hardware emulations, it is often difficult, if not impossible, to understand the relationship between the observed effects (i.e., modifications in the network dynamics) and their potential underlying causes. Therefore, we investigate the effects of individual distortion mechanisms by implementing them, separately, in software simulations. As before, we perform these analyses over a wide range of network realizations, since - as we will show later - these may strongly influence the effects of the examined mechanisms.

After having established the relationship between structural distortions caused by hardware-specific factors and their consequences for network dynamics, we demonstrate various compensation techniques in order to restore the original network behavior.

In the final stage, for each of the studied models, we simulate an implementation on the hardware back-end by running an appropriately configured executable system specification (see Section 3.3.4), which includes the full panoply of hardware-specific distortion mechanisms. Using the proposed compensation techniques, we then attempt to deal with all these effects simultaneously. The results from these experiments are then compared to results from software simulations, thus allowing a comprehensive assertion of the effectivity of our proposed compensation techniques, as well as of the capabilities and limitations of the neuromorphic emulation device.

Owing to the detailed understanding of hardware-induced distortive effects on the model functionality, we were able to scale down two of the studied models (the L2/3 and synfire chain models) to a size that is amenable to emulation on the Spikey chip. While some particular functional properties can only be properly observed in the large-scale versions of these networks, it was possible to reproduce many fundamental characteristics in the small-scale models as well. The Spikey emulations of the L2/3 and synfire chain models are described in Sections 5.3.9 and 5.4.10, respectively.

This work is the result of a close collaboration with Paul Müller, Bernhard Vogginger, Oliver Breitwieser, Mikael Lundqvist and Lyle Muller. The material for all waferscale-related Sections in Chapter 5 is taken from Petrovici et al. (2014), whereas the material for the Spikey emulations is taken from Pfeil et al. (2013). Both of these publications have been coauthored by the author of this thesis. Some related results have also been described in Breitwieser (2011); Brüderle et al. (2011); Müller (2011); Vogginger (2010).

5.1. Investigated Distortion Mechanisms

Reviewing the hardware and software components of the BrainScaleS wafer-scale system (Section 3.3) leaves us with a number of mechanisms that can negatively affect the emulation of neural network models:

distortion mechanisms

- neuron and synapse models are cast into silicon and can not be altered after chip production
- limited ranges for neuron and synapse parameters
- discretized and shared parameters
- limited number of neurons and synapses
- restricted connectivity
- synapse loss due to non-optimal algorithms for NP-hard mapping
- parameter variations due to transistor level mismatch and limited re-write precision
- non-configurable pulse delays and jitter
- limited bandwidth for stimulation and recording of spikes

It is clear that, for all of the above distortion mechanisms, it is possible to find a corner case where network dynamics are influenced strongly. However, a few of these effects stand out: on one hand, they are of such fundamental nature to mixed-signal VLSI that they are likely to play some role in any similar neuromorphic device; on the other hand, they are expected to influence any kind of emulated network to some extent. We have therefore directed our focus towards these particular effects, which we summarize in the following. In order to allow general assessments, we investigate various magnitudes of these effects, also beyond the values we expect for our particular hardware implementation.

Neuron and Synapse Models

While some network architectures employ relatively simple neuron and synapse models for analytical and/or numerical tractability, others rely on more complex components in order to remain more faithful to their biological archetypes. Such models may not allow a straightforward translation to those available on the hardware, requiring a certain amount of fitting. In our particular case, we search for parameters to Equations 3.3 – 3.10 that best reproduce low-level dynamics (e.g. membrane potential traces for simple stimulus patterns) and then tweak these as to optimally reproduce high-level network behaviors. Additionally, further constraints are imposed by the parameter ranges permitted by the hardware (Table 3.3).

Synapse Loss

Above a certain network size or density, the mapping process may not be able to find enough hardware resources to realize every single synapse. We use the term “synapse loss” to describe this process, which causes a certain portion of synaptic connections to be lost after mapping. In a first stage, we model synapse loss as homogeneous, i.e., each synapse is deleted with a fixed probability between 0 and 50 %. To ease the analysis of distortions, we make an exception for synapses that mediate external input, since, in principle, they can be prioritized in the mapping process such that the probability of losing them practically vanishes. Ultimately however, the compensation methods designed for homogeneous synapse loss are validated against a concrete mapping scenario.

Non-Configurable Axonal Delays

Axonal delays on the wafer are not configurable and depend predominantly on the processing speed of digital spikes within one HICANN, but also on the physical distance of the neurons on the wafer. In all simulations, we assume a constant delay of 1.5 ms for all synaptic connections in the network, which represents an average of the expected delays when running the hardware with a speedup of 10^4 with respect to real time.

Synaptic Weight Noise

As described in Section 3.3.1, the variation of synaptic weights is assumed to be the most significant source of parameter variation within the network. This is due to the coarser discretization (4-bit weight vs. 10 bit used for writing the analog neuron parameters) as well as the large number of available synapses, which prohibits the storage of calibration data for each individual synapse. The quality of the calibration only depends on the available time and number of parameter settings, while the trial-to-trial variability and the limited setting resolution remains. To restrict the parameter space of the following investigations (Section 5.2), only the synaptic weights are assumed to be affected by noise. In both software and ESS simulations, we model this effect by drawing synaptic weights from a Gaussian centered on the target synaptic weight with a standard-deviation-to-mean-ratio between 0 and 50 %. Occasionally, this leads to excitatory synapses becoming inhibitory and vice versa, which can not happen on the hardware. Such weights are clipped to zero. Note that this effectively leads to an increase of the mean of the distribution, which however can be neglected, e.g., for 50 % noise the mean is increased by 0.425 %. For ESS simulations we assume a synaptic weight noise of 20 %, as test measurements on the hardware indicate that the noise level can not be reduced to below this number.

It has to be noted that the mechanism of distortion plays a role in the applicability of the compensation mechanisms. The iterative compensation in Section 5.5.7.2 is only applicable when the dominant distortion mechanism is fixed-pattern noise. The other compensation methods, which do not rely on any kind of knowledge of the fixed-pattern distribution, function independently of the distortion mode.

5.2. Characterization and Compensation of Network-Level Distortions: a Systematic Workflow

In the following, we analyze the effects of hardware-specific distortion mechanisms on a set of neuronal network models and propose adequate compensation mechanisms for restoring the original network dynamics. The aim of these studies is twofold. On one hand, we propose a generic workflow which can be applied for different neural network models regardless of the neuromorphic substrate, assuming it possesses a certain degree of configurability (Figure 5.1). On the other hand, we seek to characterize the universality of the BrainScaleS neuromorphic device by assessing its capability of emulating very different large-scale network models with minimal, if any, impairment to their functionality. *workflow*

In order to allow a comprehensive overview, the set of benchmark experiments is required to cover a broad range of possible network architectures, parameters and function modi. To this end, we have chosen three very different network models, each of which highlights crucial aspects of the biology-to-hardware mapping procedure and poses unique challenges for the hardware implementation. In order to facilitate the comparison between simulations of the original model and their hardware implementation, all experimental setups were implemented in PyNN, running the same set of instructions on either simulation back-end. *benchmarks*

For each of our benchmark models we define a set of specific well-quantifiable functionality criteria. These criteria are measured in software simulations of the ideal, i.e., undistorted network, which is then further referenced as the “original”. *functionality criteria*

Assuming that the broad range of hardware-specific distortion mechanisms affects various network parameters, their impact on these measures are investigated in software simulations and various changes to the model structure are proposed in order to recover the original functionality. The feasibility of these compensation methods is then studied for the BrainScaleS neuromorphic platform with the help of the ESS described in Section 3.3.4.

All software simulations were performed with NEST (Diesmann and Gewaltig, 2002) or Neuron (Hines and Carnevale, 2003).

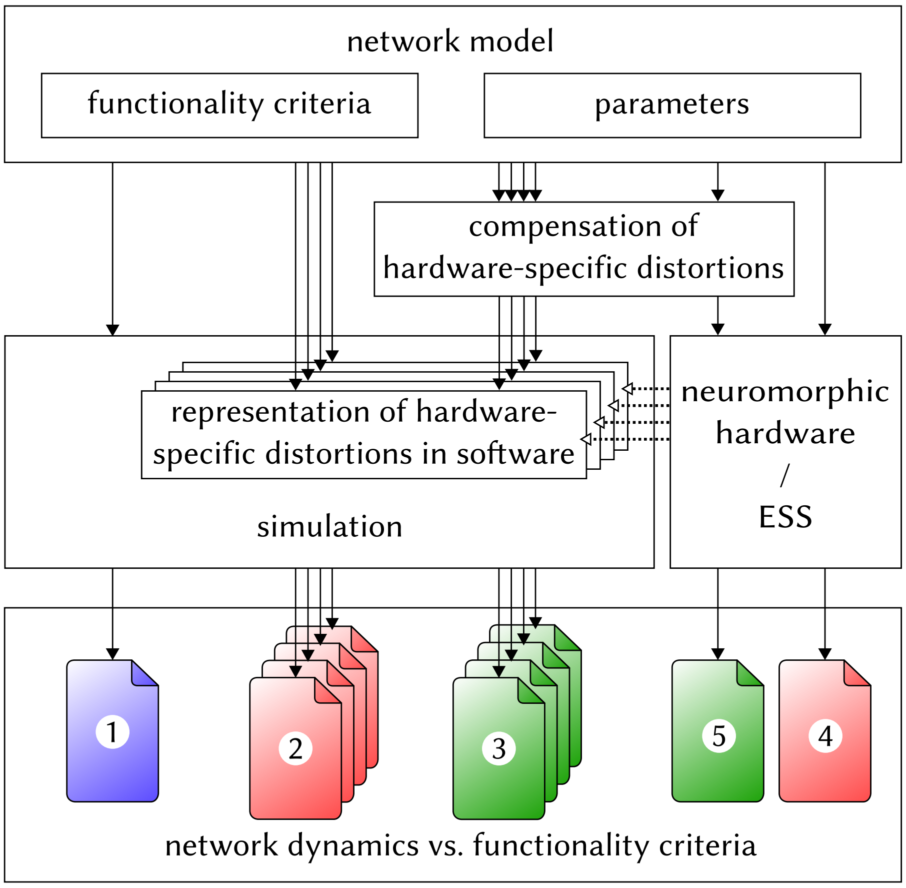

network model
functionality criteria
parameters
compensation of hardware-specific distortions
representation of hardware-specific distortions in software
simulation
neuromorphic hardware / ESS
1
2
3
5
4
network dynamics vs. functionality criteria

Figure 5.1.: Schematic of the workflow we have used for studying and compensating hardware-induced distortions of network dynamics. (1) A given network model is defined by providing suitable parameters (for its connectivity and components) and well-defined functionality criteria. (2) The distortions that are expected to occur natively on the hardware back-end are implemented and studied individually in software simulations. (3) Compensation methods are designed and tested, with the aim of recovering the original network dynamics as determined by the functionality criteria. (4) The network model is run on the hardware (here: the ESS) without any compensation to evaluate the full effect of the combined distortion mechanisms. (5) The compensation methods are combined and applied to the hardware (here: the ESS) simulation in order to restore the original network dynamics.

5.3. Cortical Layer 2/3 Attractor Memory

attractor network

As our first benchmark, we have chosen an attractor network model of the cerebral cortex which exhibits characteristic and well-quantifiable dynamics, both at the single-cell level (membrane voltage UP and DOWN states) and for entire populations (gamma band oscillations, pattern completion, attentional blink). For this model, the mapping to the hardware was particularly challenging, due to the complex neuron and synapse models required by the original architecture on the one hand, as well as its dense connectivity on the other. In particular, we observed that the shape of synaptic conductances strongly affects the duration of the attractor states. As expected for a model with relatively large populations as functional units, it exhibits a pronounced robustness to synaptic weight noise. Homogeneous synapse loss, on the other hand, has a direct impact on single-cell dynamics, resulting in significant deviations from the expected high-level functionality, such as the attenuation of attentional blink. As a compensation for synapse loss, we suggest two methods: increasing the weights of the remaining synapses in order to maintain the total average synaptic conductance and reducing the size of certain populations and thereby decreasing the total number of required synapses. After mapping to the hardware substrate, synapse loss is not homogeneous, due to the different connectivity patterns of the three neuron types required by the model. However, we were able to apply a population-wise version of the suggested compensation methods and demonstrate their effectiveness in recovering the previously defined target functionality measures.

5.3.1. Architecture

L2/3 model

cortical columns

As described in Lundqvist et al. (2006) and Lundqvist et al. (2010), this model (henceforth called L2/3 model) implements a columnar architecture (Buxhoeveden and Casanova, 2002; Mountcastle, 1997). The connectivity is compliant with data from cat cerebral cortex (Thomson et al., 2002). The key aspect of the model is its modularity, which manifests itself on two levels. On a large scale, the simulated cortical patch is represented by a number N_{HC} of hypercolumns (HCs) arranged on a hexagonal grid. On a smaller scale, each HC is further subdivided into a number N_{MC} of minicolumns (MCs) (Buxhoeveden and Casanova, 2002; Mountcastle, 1997). Such MCs should first and foremost be seen as functional units, and could, in biology, also be a group of distributed, but highly interconnected cells (Kampa et al., 2006; Perin et al., 2011; Song et al., 2005). In the model, each MC consists, in turn, of 30 pyramidal (PYR), 2 regular spiking non-pyramidal (RSNP) and 1 basket (BAS) cells (Markram et al., 2004a; Peters and Sethares, 1997). Within each MC, PYR neurons are mutually interconnected, with 25% connectivity, such that they will tend to be co-active and code for similar input.

hyper-columns, mini-columns

PYR, RSNP, BAS cells

attractor/-pattern orthogonal-ity

The functional units of the network, the MCs, are connected in global, distributed patterns containing a set of MCs in the network (Figure 5.2). Here the attractors, or patterns, contain exactly one MC from each HC. We have only considered the case of orthogonal patterns, which implies that no two attractors share any number of MCs.[1] Due to the mutual excitation within an attractor, the network is able to perform pattern

[1] Orthogonal patterns are much more comfortable to study than non-orthogonal ones, since the response of the network to experimental scenarios such as pattern completion is easily classified as "correct" or "wrong". However, allowing patterns to share MCs can greatly increase the memory capacity of the network, i.e., the number of patterns it can "correctly" recall under certain well-defined conditions, where the "correctness" is, itself, a parameter to be defined in a sensible way. Although not a part of

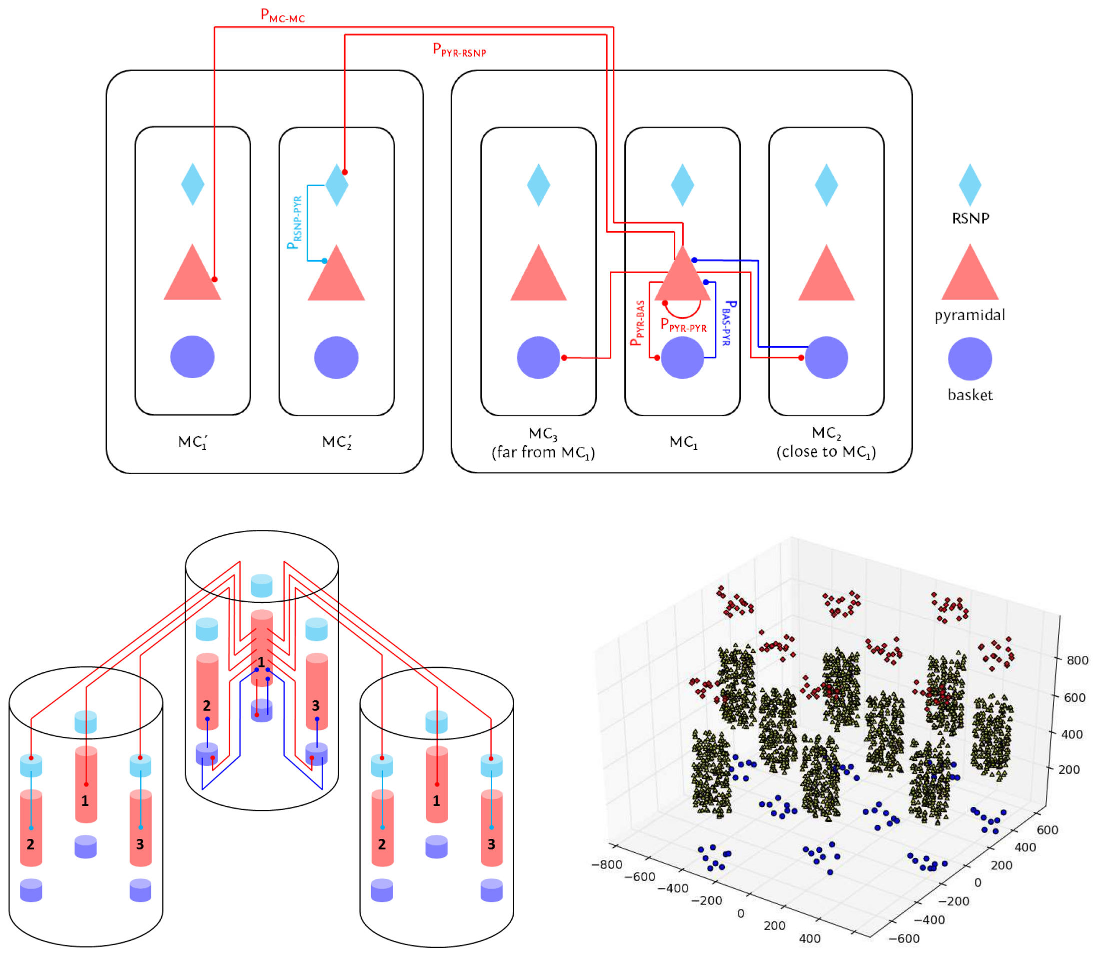


Figure 5.2.: Layer 2/3 model architecture. **Top:** Connection probabilities among the individual populations. Excitatory cells are depicted in red, inhibitory in blue. Only a subset of MCs within an HC are depicted. MC_3 is not among the 7 closest neighbors to MC_1 and therefore its basket cells do not project onto the pyramidal cells of MC_1. **Bottom left:** Pseudo-3D schematic of the geometric arrangement of the HCs and MCs therein. For clarity, only a subset of the active connections during an UP-state of attractor 1 are shown. **Bottom right:** Geometrical arrangement of the cells in the simulation (red: RSNP, green: PYR, blue: BAS). Figure taken from Petrovici et al. (2014).

completion, which means that whenever a subset of MCs in an attractor is activated, the activity tends to spread throughout the entire attractor. *pattern completion*

Pattern rivalry results from competition between attractors mediated by short and long-range connections via inhibitory interneurons. *pattern rivalry* Each HC can be viewed as a soft winner-take-all (WTA) module which normalizes activity among its constituent MCs (Lundqvist et al., 2010). *soft WTA* This is achieved by the inhibitory BAS cells, which receive input from the PYR cells from the 8 closest MCs and project back onto the PYR cells in all the MCs within the home HC. Apart from providing long-range connections to PYR cells within the same pattern, the PYR cells within an MC project onto RSNP cells in all the MCs which do not belong to the same pattern and do not lie within the same HC. The inhibitory RSNP cells, in turn, project onto the PYR cells in their respective MC. The effect of this connectivity is a disynaptic inhibition between competing patterns. Figure 5.2 shows a schematic of the default architecture, emphasizing the connectivity pattern described above. It consists of $N_{\mathrm{HC}} = 9$ HCs, each containing $N_{\mathrm{MC}} = 9$ MCs, yielding a total of 2673 neurons. Due to its modular structure, this default model can easily be scaled up or down in size with preserved dynamics (Section 5.3.2).

When a pattern receives enough excitation, its PYR cells enter a state reminiscent of a so-called local UP-state (Cossart et al., 2003), which is characterized by a high average membrane potential, several mV above its rest value, and elevated firing rates. *local UP-state* Pattern rivalry leads to states where only one attractor may be active (with all its PYR cells in an UP-state) at any given time. Inter-PYR synapses feature an STD mechanism which weakens the mutual activation of PYR cells over time and prevents a single attractor from becoming persistently active. Additionally, PYR neurons exhibit spike-frequency adaptation, which also suppresses prolonged firing. These mechanisms impose a finite life-time on the attractors such that after their termination more weakly stimulated or less excitable attractors can become active, in contrast to what happens in classical WTA networks.

The inputs to the layer 2/3 PYR cells arrive from the cortical layer 4 (L4), which is represented by 5 cells per MC. *L4* The L4 cells project onto the L2/3 PYR cells and can be selectively activated by external Poisson spike trains. Additionally, the network receives unspecific input representing activity in various connected cortical areas outside the modeled patch. This input is modeled as diffuse noise and generates a background activity of several Hz.

More details on the model architecture, as well as neuron and synapse parameters, can be found in Table A.2.3.1.

5.3.2. Network Scaling

Due to the modularity of this network model, several straightforward possibilities exist for increasing or decreasing its size without affecting its basic functionality. One can vary the total number of neurons simply by modifying the number of cells per MC. One can also vary the number of MCs per attractor by varying the total number of HCs. And finally, one can change the number of attractors by changing the number of MCs per HC accordingly.

this discussion, we point to two related studies on non-orthogonal patterns in the L2/3 model, namely Breitwieser (2011) and Rivkin (2014).

Connection	Scaled connection probability $\tilde{p}$
PYR → PYR (same MC)	$29/(N_{\text{PYR}} - 1) \cdot p$
PYR → PYR (different MC)	$30/N_{\text{PYR}} \cdot 8/(N_{\text{HC}} - 1) \cdot p$
PYR → RSNP	$30/N_{\text{PYR}} \cdot 8/(N_{\text{HC}} - 1) \cdot p$
PYR → BAS	$30/N_{\text{PYR}} \cdot p$
RSNP → PYR	$2/N_{\text{RSNP}} \cdot p$
BAS → PYR (enlarging)	$1/N_{\text{BAS}} \cdot p$
BAS → PYR (shrinking)	$1/N_{\text{BAS}} \cdot 8/N_{\text{MC}} \cdot p$

Table 5.1.: Scaling rules for the connection densities of the L2/3 model. N_x represents the number of units of type x (the original values are found in Table A.5). p represents the original connection probability as found in Table A.6. Whenever a scaled probability $\tilde{p}$ exceeded 1, it was clipped to 1, but the weights of the corresponding synapses were also increased by $\tilde{w}^{\text{syn}} = w^{\text{syn}} \cdot \tilde{p}$.

fan-in conservation

All such changes need to be accompanied by corresponding modifications in connectivity in order to preserve the network dynamics. This has been done by keeping the average input current per neuron within an active attractor constant, which is equivalent to conserving the fan-in for every neuron from every one of its afferent populations and leads to the scaling rules shown in Table 5.1. In order to facilitate a comparison with the original results from Lundqvist et al. (2006) and Lundqvist et al. (2010), we have only considered homogeneous changes, meaning that all modules (MCs, HCs) were equal in size and symmetrically connected.

The connections to the BAS cells required special treatment for two reasons. Firstly, during an active state, they receive input from a single MC, but are excited by all MCs in an HC during the competition period between active attractors. Only one aspect can be preserved when scaling and we have considered the dynamics during UP states as most important, leading to a PYR → BAS scaling rule independent of N_{MC}. Secondly, because PYR cells in MCs only project to the nearest 8 BAS cells, there are always precisely 8 active BAS cells per HC within an active attractor, which yields a simple BAS → PYR scaling rule. When decreasing the number of attractors however, the number of existing BAS cells per HC also decreases, making an appropriate connection density scaling necessary. This is the reason for the two different BAS → PYR scaling rules found in Table 5.1.

Table A.11 shows the combinations of N_{HC} and N_{MC} used for the quantification of synapse loss after mapping the L2/3 model to the hardware in Figure 5.11. In these mapping sweeps, the diffusive background noise was modeled, as for the large-scale network ported to the ESS (Section 5.3.8), with a background pool of 5000 Poisson sources and every PYR cell receiving input from 100 of the sources.

5.3.3. Functionality Criteria

Figure 5.3 shows some characteristic dynamics of the L2/3 model, which have also been chosen as functionality criteria and are described below.

spontaneous attractors

The core functionality of the original model is easily identifiable by its distinctive display of spontaneously activating attractors in, e.g., raster plots (A) or voltage star plots

(D) (for an explanation of star plots see Section A.2.3.5). However, in particular for large network sizes, spontaneous attractors become increasingly sparse. Additionally, many further indicators of functionality can be found, such as the average membrane potential or the gamma oscillations observed in UP states. Finally, when receiving L4 stimulation in addition to the background noise, the original model displays important features such as pattern completion and attentional blink, which need to be reproducible on the hardware as well. Consequently, we consider several measures of functionality throughout our analyses.

dwell time

When an attractor becomes active, it remains that way for a characteristic dwell time τ_{ON}. The dwell time depends strongly on the neuron and synapse parameters (as will be discussed in the following sections) and only weakly on the network size (C, F), since the scaling rules ensure a constant average fan-in for each neuron type. Conversely, this makes τ_{ON} sensitive to hardware-induced variations in the average synaptic input. The detection of active attractors is performed automatically using the spike data (for a description of the algorithm, see Section A.2.3.2).

competition

We describe the periods between active attractors as competition phases and the time spent therein as the total competition time. The competition time varies strongly depending on the network size (H). One can observe that the competition time is a monotonically increasing function of both N_{HC} and N_{MC}. For an increasing number of HCs, i.e., a larger number of neurons in every pattern, the probability of a spontaneous activation of a sufficiently large number of PYR cells decreases. For an increasing number of MCs per HC, there is a larger number of competing patterns, leading to a reduced probability of any single pattern becoming dominant.

average PYR spike rate

When an attractor becomes active, the average spike rate of its constituent PYR cells rises sharply and then decays slowly until the attractor becomes inactive again (J). Two independent mechanisms are the cause of this decay: neuron adaptation and synaptic depression. The characteristic time course of the spike rate depends only weakly on the size of the network.

average PYR membrane potential

As described in Section 5.3.1, PYR cells within active attractors enter a so-called local UP state, with an increased average membrane potential and an elevated firing rate (K). While inactive or inhibited by other active attractors, PYR cells are in a DOWN state, with low average membrane potential and almost no spiking at all (L). In addition to these characteristic states, the average PYR membrane potential exhibits oscillations with a period close to τ_{ON}. These occur because the activation probability of individual attractors is an oscillatory function of time as well. In the immediate temporal vicinity of an active period (i.e., assuming an activation at $t = 0$, during $[-\tau_{\mathrm{ON}}, 0) \cup [\tau_{\mathrm{ON}}, 2\tau_{\mathrm{ON}})$) the same attractor must have been inactive, since PYR populations belonging to an activated attractor need time to recuperate from synaptic depression and spike-triggered adaptation before being able to activate again.

gamma oscillations

An essential emerging feature of this model are oscillations of the instantaneous PYR spike rate in the gamma band within active attractors (M). The frequency of these oscillations are independent of size and rather depend on excitation levels in the network (Lundqvist et al., 2010). Although the gamma oscillations might suggest periodic spiking, it is important to note that individual PYR cells spike irregularly The irregularity of the inter-spike intervals (ISIs) is quantified by their CV, which is $\langle \mathrm{CV}_{\mathrm{ISI}} \rangle = 1.36 \pm 0.36$ within active attractors.

ISI, $\mathrm{CV}_{\mathrm{ISI}}$

Apart from these statistical measures, two behavioral properties are essential for defin-

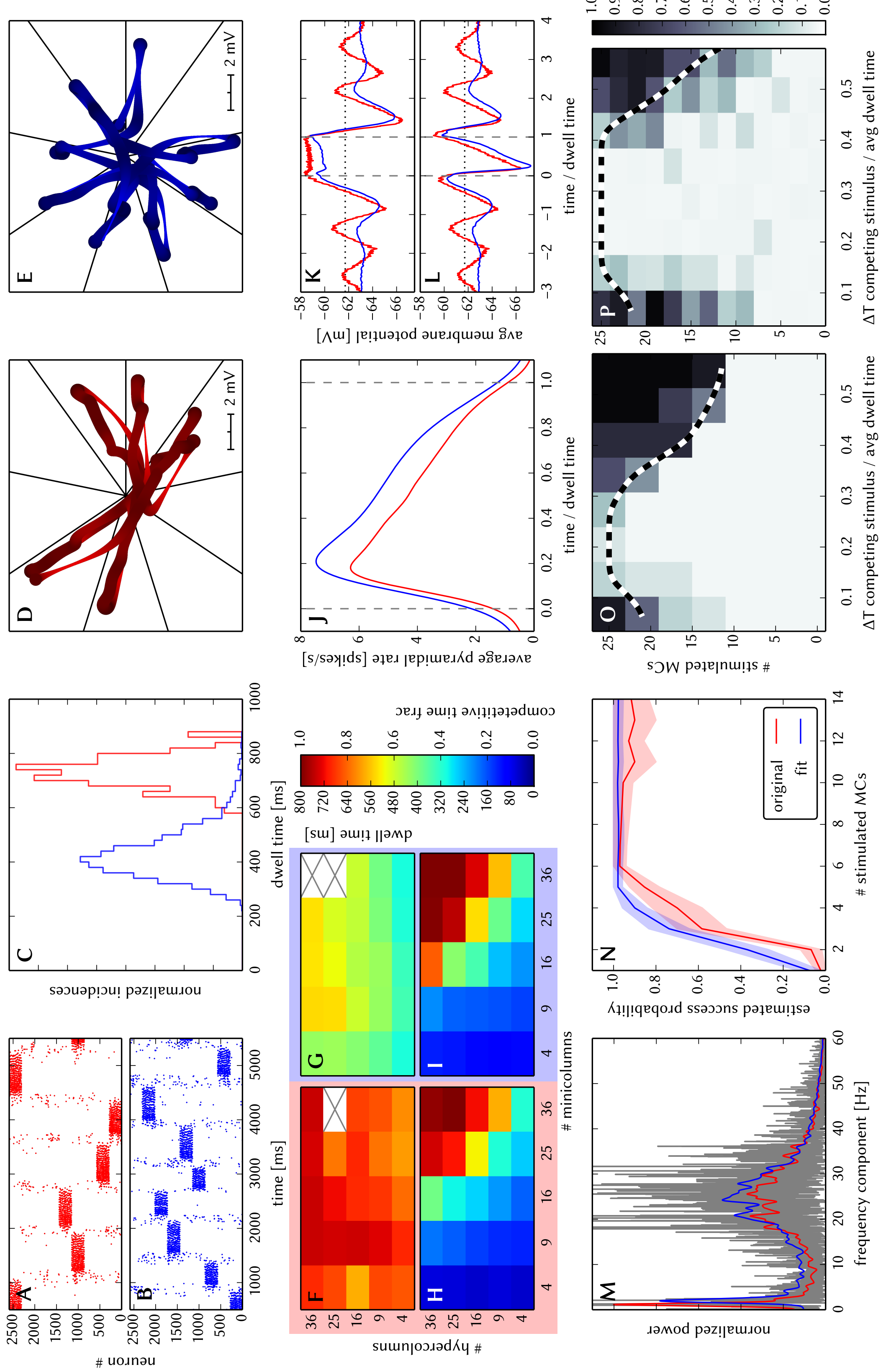

A
B
neuron #
time [ms]
C
normalized incidences
dwell time [ms]
D
2 mV
E
2 mV
F
G
H
I
# hypercolumns
# minicolumns
dwell time [ms]
competetitive time frac
J
average pyramidal rate [spikes/s]
time / dwell time
K
L
avg membrane potential [mV]
time / dwell time
M
normalized power
frequency component [Hz]
N
estimated success probability
# stimulated MCs
original
fit
O
# stimulated MCs
ΔT competing stimulus / avg dwell time
P
ΔT competing stimulus / avg dwell time
success ratio

Figure 5.3.: Comparison between original and adapted L2/3 network models. Unless explicitly stated otherwise, the default network model (9HC×9MC) was used. Measurements from the original model are depicted (or highlighted) in red, while those from the adapted model are depicted (or highlighted) in blue. **(A, B)** Raster plots of spiking activity. Attractors activate spontaneously only due to diffuse background noise. Only PYR cells are shown. The MCs are ordered such that those belonging to the same attractor (and *not* those within the same HC) are grouped together. **(C)** Attractor dwell time distributions. The shorter average dwell times in the adapted model are caused by sharper PSPs which miss the long NMDA time constants. **(D, E)** Star plots of average PYR cell voltages from a sample of 5 PYR cells per MC. Details on this representation of multidimensional data can be found in A.2.3.5. **(F, G)** Average dwell time for various network sizes. **(H, I)** Fraction of time spent in competitive states (i.e. no active attractors) for various network sizes. While dwell times remain relatively constant, competition times increase with network size, suppressing spontaneous attractors in large networks. **(J)** Average firing rate of PYR within an active period of their parent attractor. **(K)** Average voltage of PYR cells before, during and after their parent attractor is active (UP state). **(L)** Average voltage of PYR cells before, during and after an attractor they do not belong to is active (DOWN state). For subplots **J**, **K** and **L**, the abscissa has been subdivided into multiples of the average attractor dwell time in the respective simulations. The oscillations of the average voltages occur due to spike-triggered adaptation: after an active period, PYR cells need to recover before being able to support an active period of their home attractor, during which time they are inhibited by other active attractors. The more pronounced attenuation of the oscillations in the adapted model happens due to a higher relative variability of dwell times (compare subplot **C**). In subplots **K** and **L** the dotted line indicates the leak potential E_l of the PYR cells. **(M)** Smoothed power spectrum of PYR firing rate averaged over all MCs. The gray curve in the background represents the unsmoothed spectral density for the original model. Attractor switches (≈ 2 Hz) and gamma oscillations (≈ 25 Hz) can be clearly observed. **(N)** Pattern completion in a 25HC×25MC network. Estimated probability of an attractor to fully activate (success ratio) as a function of the number of stimulated constituent MCs, measured over 25 trials per abscissa value. **(O,P)** Attentional blink in a 25HC×25MC network. Two attractors are stimulated (the second one only partially, i.e. a certain number of constituent MCs) with a temporal lag of ΔT in between. Activation probability of the second attractor and $p = 0.5$ iso-probability contours, measured over 14 trials per (ΔT, #MCs) pair. Figure taken from Petrovici et al. (2014).

ing the functionality of the network: the pattern completion and attentional blink mentioned above. The pattern completion ability of the network can be described as the successful activation probability of individual patterns as a function of the number of stimulated MCs (N). Similarly, the attentional blink phenomenon can also be quantified by the successful activation rate of an attractor as a function of the number of stimulated MCs if it is preceded by the activation of some other attractor with a time lag of Δ_T (O). For small Δ_T, the second attractor is completely "blinked out", i.e., it can not be activated regardless of the number of stimulated MCs. To facilitate the comparison between different realizations of the network with respect to attentional blink, we consider the 50% iso-line, which represents the locus of the input variable pair which leads to an attractor activation ratio of 50%. These functional properties are easiest to observe in large networks, where spontaneous attractors are rare and do not interfere with stimulated ones.

pattern completion attentional blink

5.3.4. Neuron and Synapse Model Translation

A particular feature of this benchmark model is the complexity of both neuron and synapse models used in its original version. Therefore, the first required type of compensation concerns the parameter fitting for the models implemented on the hardware. Some exemplary results of this parameter fit can be seen in Figure 5.4. More details can be found in Section A.2.3.6.

Neurons

In general, the typical membrane potential time course during a spike of a Hodgkin-Huxley neuron can be well approximated by the exponential term in the AdEx equation (Brette and Gerstner, 2005). However, when fitting for spike timing, we found that spike times were best reproduced when eliminating the exponential term, i.e. setting $\Delta_\mathrm{T} = 0$.

Adaptation is an essential feature of both the PYR and the RSNP cells in the original model, where it is generated by voltage-dependent ${\mathrm{K}^+}_{\mathrm{Ca}^{++}}$ channels. We were able to reproduce the correct equilibrium spike frequency by setting the AdEx adaptation parameters a and b to nonzero values. One further difference resides in the original neurons being modeled as having several compartments, whereas the hardware only implements point neurons. The passive neuron properties (membrane capacitances and leak conductances) were therefore determined by fitting the membrane potential time course under stimulation by a step current which was not strong enough to elicit spikes.

multicompartment to point neuron mapping

Synapses

We have performed an initial estimation of synaptic weights and time constants by fitting the membrane potential time course of the corresponding neurons in a subthreshold regime. However, two important differences remain between the synapses in the original model and the ones available on our hardware.

In the original model, PYR-PYR and PYR-RSNP synapses contain two types of neurotransmitters: Kainate/AMPA and NMDA (see Table A.4). Due to the vastly different time constants for neurotransmitter removal at the postsynaptic site (6 ms and 150 ms, respectively), the PSPs have a characteristic shape, with a pronounced peak and a long

multiple synaptic time constants

tail (red curve in Figure 5.4B). While, in priniciple, the HICANN supports several excitatory time constants per neuron (see Section 3.3.1), the used version of the PyNN API as well as the mapping process support only one excitatory time constant per neuron. With this limitation the PSP shape can not be precisely reproduced.

saturating synapses

One further difference lies in the saturating nature of the postsynaptic receptor pools after a single presynaptic spike. In principle, this behavior could be emulated by the TSO plasticity mechanism by setting $U = 1$ and $\tau_{\text{rec}} = \tau^{\text{syn}}$. However, this would conflict with the TSO parameters required for modeling short-term depression of PYR synapses and would also require parameters outside the available hardware ranges.

For these reasons, we have further modified synaptic weights and time constants by performing a behavioral fit, i.e., by optimizing these parameters towards reproducing the correct firing rates of the three neuron types in two scenarios - first without and then subsequently with inhibitory synapses. Because the original model was characterized by relatively long and stable attractors, we further optimized the excitatory synapse time constants towards this behavior.

Post-Fit Model Behavior

Figure 5.3 shows the results of the translation of the original model to hardware-compatible dynamics and parameter ranges. Overall, one can observe a very good qualitative agreement of characteristic dynamics with the original model. In the following, we discuss this in more detail and explain the sources of quantitative deviations.

shortened dwell times

When subject to diffuse background noise only, the default size network clearly exhibits its characteristic spontaneous attractors (B). Star plots exhibit the same overall traits, with well-defined attractors, characterized by state space trajectories situated close to the axes and low trajectory velocities within attractors (E). Attractor dwell times remain relatively stable for different network sizes, while the competition times increase along with the network size (G and I). The average value of dwell times, however, lies significantly lower than in the original (C). The reason for this lies mainly in the shape of EPSPs: the long EPSP tails enabled by the large NMDA time constants in the original model caused a higher average membrane potential, thereby prolonging the activity of PYR cells.

higher dwell time variation

Within attractors, active and inactive PYR cells enter well-defined local UP and DOWN states, respectively (K and L). Before and after active attractors, the dampened oscillations described in Section 5.3.3 can be observed. In the adapted model, attenuation is stronger due to a higher coefficient of variation of the dwell times ($\frac{\sigma}{\mu} = 0.20$ as compared to 0.08 in the original model).

Average PYR firing rates within active attractors have very similar time courses (J), with a small difference in amplitude, which can be attributed to the difference in EPSP shapes discussed earlier. Both low-frequency switches between attractors ($<$ 3 Hz, equivalent to the incidence rate) and high-frequency gamma oscillations arising from synchronous PYR firing (with a peak around 25 Hz) can be clearly seen in a power spectrum of the PYR firing rate (M).

Pattern completion occurs similarly early, with a steep rise and nearly 100% success rate starting at 25% of stimulated MCs per attractor (N). Attentional blink follows the same qualitative pattern (P, Q), although with a slightly more pronounced dominance of the first activated attractor in the case of the adapted network, which happens due to the slightly higher firing rates discussed above.

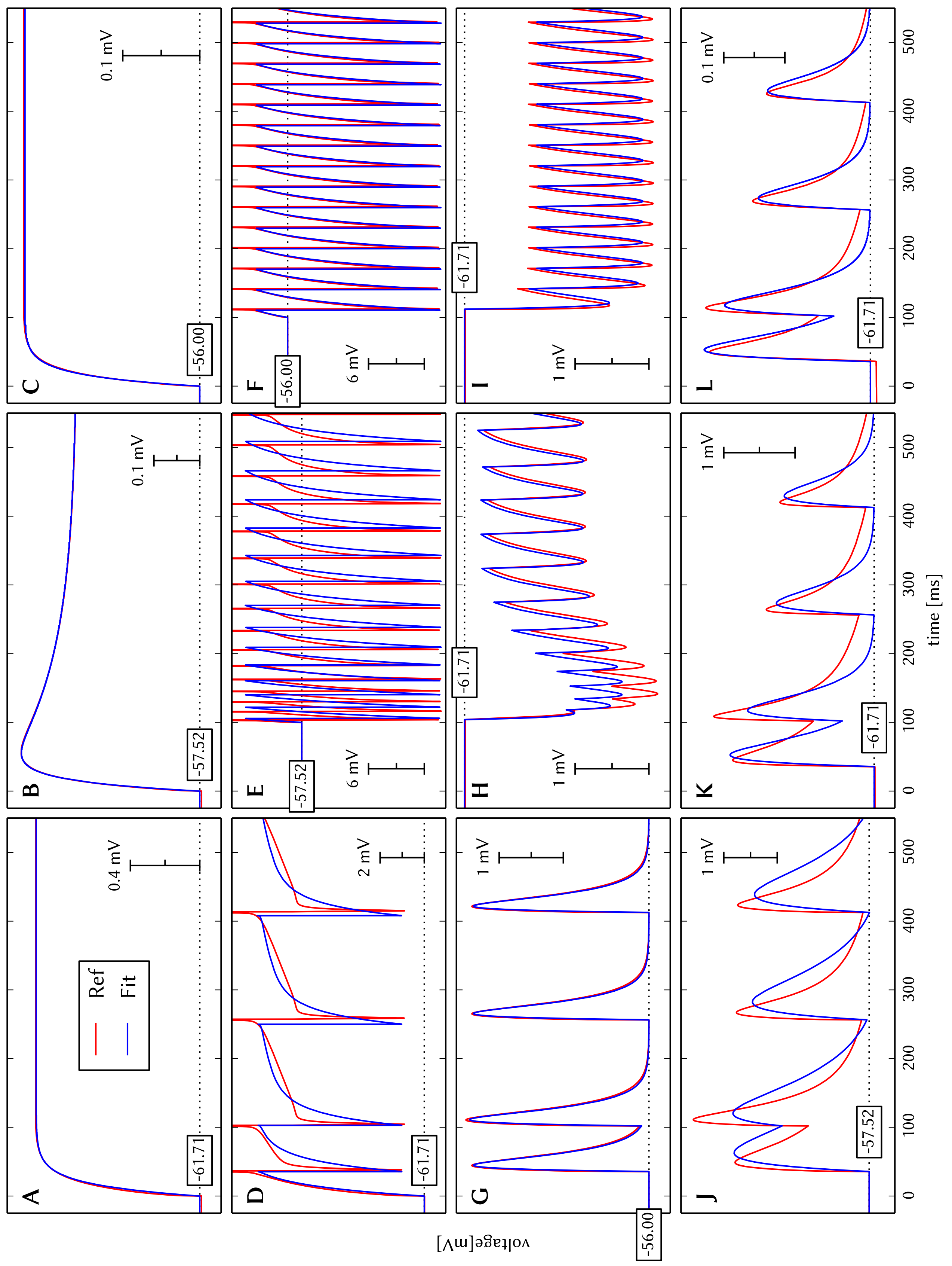

A
B
C
D
E
F
G
H
I
J
K
L
Ref
Fit
-61.71
-57.52
-56.00
0.4 mV
0.1 mV
2 mV
6 mV
1 mV
voltage[mV]
time [ms]
0
100
200
300
400
500

Figure 5.4.: Comparison of original neuron and synapse dynamics to the fitted dynamics of hardware-compatible models. **(A - C)** Membrane potential of the three different cell types (PYR, RSNP, and BAS, respectively) under subthreshold current stimulation. These were used to determine the rest voltage E_l, total equivalent membrane capacitance C_m, membrane time constant τ_m and the adaptation coupling parameter a. **(D - F)** Membrane potential of the three different cell types (PYR, RSNP, and BAS, respectively) under spike-inducing current stimulation. While the precise membrane potential time course of the original neuron model can not be reproduced by a single-compartment AdEx neuron, it was possible to reproduce the spike timing and especially average firing rates quite accurately. A small deviation of spiking frequency can be observed for RSNP cells during the first 50 ms – in the original model, they adapt slower than their AdEx counterparts. From these fits, the values for the absolute refractory period τ_ref, reset voltage E_r, threshold voltage E_T, slope factor Δ_T, spike-triggered adaptation b and adaptation time constant τ_w were extracted. **(G - L)** PSP fit results for all synapse types of the L2/3 model (PYR$\rightarrow$BAS, RSNP$\rightarrow$PYR, BAS$\rightarrow$PYR, PYR$\rightarrow$RSNP, PYR$\rightarrow$PYR within an MC, and PYR$\rightarrow$PYR between MCs, respectively). The output spikes from **D - F** have been used as input. These fits were used to determine synaptic weights w^syn, time constants τ^syn and the TSO parameters U and τ_ref. Because the hardware synapses only support a single conductance decay time constant, as opposed to the two different time constants in the original model for AMPA/kainate and NMDA, we have chosen an intermediate value for τ^syn, which constitutes the main reason for the difference in PSP shapes. A second reason lies in the saturating nature of synaptic conductances in the original model, which can not be emulated on the hardware without affecting the required TSO parameters. Figure taken from Petrovici et al. (2014).

Having established the quality of the model fit and in order to facilitate a meaningful comparison, all following studies concerning hardware-induced distortions and compensation thereof use data from the adapted model as reference.

5.3.5. Synapse Loss

Effects

suppressed attractors

With increasing synapse loss, the functionality of the network gradually deteriorates (Figure 5.5). Attractors become shorter or disappear entirely, with longer periods of competition in between (D, K, O).

reduced net inhibition

While average excitatory conductances are only affected linearly by synaptic loss, inhibitory conductances feel a compound effect of synapse loss, as it affects both afferent and efferent connections of inhibitory interneurons. Therefore, synapse loss has a stronger effect on inhibition, leading to a net increase in the average PYR membrane potential (R, S). Additionally, since all connections become weaker, the variance of the membrane potential becomes smaller, as observed in the corresponding star plots as well (E). The weaker connections also decrease the self-excitation of active attractors while decreasing the inhibition of inactive ones, thereby leading to shorter attractor dwell times (P). Somewhat surprisingly, the maximum average PYR firing rate in active attractors remains almost unchanged when subjected to synapse loss. However, the temporal evolution of the PYR firing rate changes significantly (Q).

suppressed pattern completion and blink

The pattern completion ability of the network suffers particularly in the region of weak stimuli, due to weaker internal excitation of individual attractors. The probability of triggering a partially stimulated pattern can drop by more than 50% (T). Due to the decreased stability of individual attractors discussed above, rival attractors are easier to excite, thereby significantly suppressing the attentional blink phenomenon (U).

Compensation

As a first-order approximation, we can consider the population average of the neuron conductance as the determining factor in the model dynamics. For synapses with exponential conductance courses, the average conductance generated by the ith synapse is proportional to both synaptic weight w_{ij} and afferent firing rate ν_j. Because conductances sum up linearly, the total conductance that a neuron from population i receives from some other population j is, on average (see Section 4.3.1 and in particular Equation 4.94),

$$\langle g^{\mathrm{syn}} \rangle = N_j p_{ij} \langle w_{ij} \rangle \langle \nu_j \rangle \tau^{\mathrm{syn}} \quad , \tag{5.1}$$

conservation of $\langle g^{\mathrm{syn}} \rangle$

where N_j represents the size of the presynaptic population and p_{ij} represents the probability of a neuron from the presynaptic population to project onto a neuron from the postsynaptic population. Since homogeneous synapse loss is equivalent to a decrease in p_{ij}, we can compensate for synapse loss that occurs with probability p_{loss} by increasing the weights of the remaining synapses by a factor $1/(1 - p_{\mathrm{loss}})$.

increased $\mathrm{Var}[g^{\mathrm{syn}}]$

Figure 5.7 shows the results of this compensation strategy for $p_{\mathrm{loss}} = 0.5$. In all aspects, a clear improvement can be observed. The remaining deviations can be mainly attributed to two effects. First of all, preserving the average conductance by compensating homogeneous synapse loss with increasing synaptic weights leads to an increase in the variance of the total input conductance (see Equations 4.94 and 4.95). Secondly, finite population sizes coupled with random elimination of synapses lead to locally inhomogeneous synapse loss and further increase the variability of neuronal activity.

PYR population reduction

Instead of compensating for synapse loss after its occurrence, it is also possible to circumvent it altogether after having estimated the expected synapse loss in a preliminary mapping run. For the L2/3 model, this can be done without altering the number of functional units (i.e., the number of HCs and MCs) by changing the size of the PYR cell populations. For this approach, however, the standard scaling rules (see Section 5.3.2 and in particular Table 5.1) need to be modified. These rules are designed to keep the average number of inputs per neuron constant and would increase the total number of PYR-incident synapses by the same factor by which the PYR population is scaled. This would inevitably lead to an increased number of shared inputs per PYR cell, with the immediate consequence of increased firing synchrony. Instead, when reducing the PYR population size, we compensate for the reduced number of presynaptic partners by increasing relevant synaptic weights instead of connection probabilities. This modified downscaling leads to a net reduction of the total number of synapses in the network, thereby potentially reducing synaptic loss between all populations.

Figure 5.7 shows the effects of scaling down the PYR population size until the total remaining number of synapses is equal to the realized number of synapses in the distorted case (50% of the total number of synapses in the undistorted network). More detailed plots of the effects of PYR population downscaling can be found in Figure 5.6. The two presented compensation methods can also be combined to further improve the final result, as we show in Section 5.3.8.

5.3.6. Synaptic Weight Noise

One would not expect the synaptic weight noise to affect the L2/3 model strongly, as it should average out over a large number of connections between the constituent populations. It turns out that the surprisingly strong impact of synaptic weight noise is purely due to the implementation of background stimulus in this model and can therefore be easily countered.

Effects

The relative variation of the total synaptic conductance scales with (see Equations 4.94 and 4.95)

$$E\left[g\right]/\sqrt{\mathrm{Var}[g]} \propto 1/\sqrt{\nu_{\mathrm{input}}} \propto 1/\sqrt{N} \quad , \tag{5.2}$$

where ν_{input} is the total input frequency and N the number of presynaptic neurons. Therefore, interactions between large populations are not expected to be strongly affected by synaptic weight noise.

The only connections where an effect is expected are the RSNP→PYR connections, because the presynaptic RSNP population consists of only 2 neurons per MC. However, long-range inhibition also acts by means of a second-order mechanism, in which an active MC activates its counterpart in some other HC, which then in turn inhibits all other MCs in its home HC via BAS cells. This mechanism masks much of how synaptic weight noise affects RSNP→PYR connections.

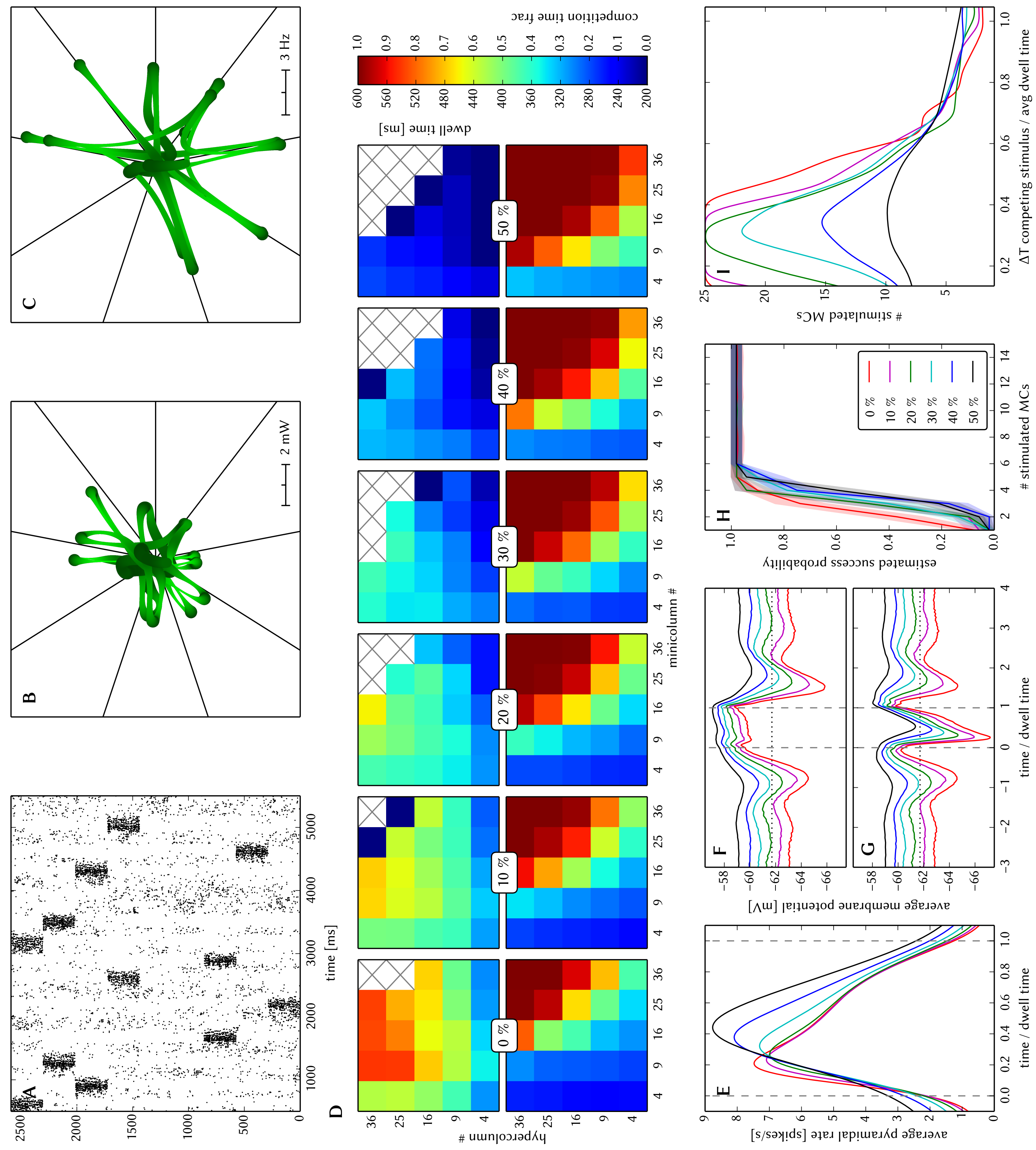

A
neuron #
time [ms]
B
2 mW
C
3 Hz
D
hypercolumn #
minicolumn #
0 %
10 %
20 %
30 %
40 %
50 %
dwell time [ms]
competition time frac
E
average pyramidal rate [spikes/s]
time / dwell time
F
G
average membrane potential [mV]
H
estimated success probability
# stimulated MCs
I
# stimulated MCs
ΔT competing stimulus / avg dwell time

Figure 5.5.: Effects of homogeneous synapse loss on the L2/3 model. Unless explicitly stated otherwise, the default network model (9HC×9MC) was used. The topmost 3 figures exemplify the dynamics of the network at 50% synapse loss, all other figures show the effects of various degrees of synapse loss (0-50%). **(A)** Raster plot of spiking activity. Only PYR cells are shown. The MCs are ordered such that those belonging to the same attractor (and *not* those within the same HC) are grouped together. **(B)** Star plot of average PYR cell voltages from a sample of 5 PYR cells per MC. **(C)** Star plot of average PYR cell firing rates. **(D)** Average dwell times and relative competition times for various network sizes. **(E)** Average firing rate of PYR cells during an UP state. **(F)** Average voltage of PYR cells before, during and after their parent attractor is active (UP state). **(G)** Average voltage of PYR cells before, during and after an attractor they do not belong to is active. For the previous three plots, the abscissa has been subdivided into multiples of the attractor dwell time. In subplots **F** and **G** the dotted line indicates the leak potential E_l of the PYR cells. **(H)** Pattern completion in a 25HC×25MC network. **(I)** Attentional blink in a 25HC×25MC network: $p = 0.5$ iso-probability contours. Figure taken from Petrovici et al. (2014).

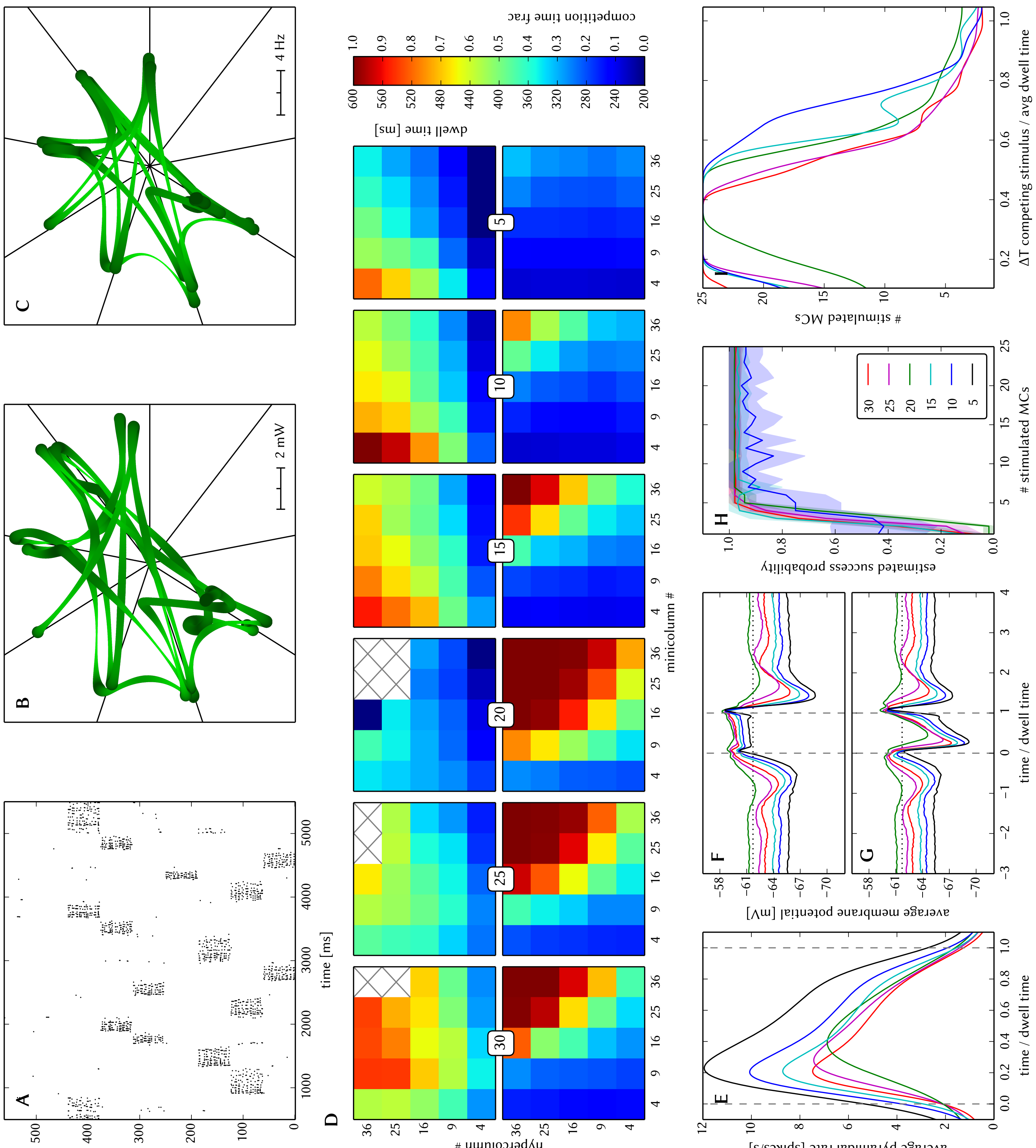

A
neuron #
time [ms]
B
2 mW
C
4 Hz
D
hypercolumn #
minicolumn #
dwell time [ms]
competition time frac
30
25
20
15
10
5
E
average pyramidal rate [spikes/s]
time / dwell time
F
G
average membrane potential [mV]
time / dwell time
H
estimated success probability
# stimulated MCs
I
# stimulated MCs
ΔT competing stimulus / avg dwell time

Figure 5.6.: Effects of PYR population size scaling on the L2/3 model. Unless explicitly stated otherwise, the default network model (9HC×9MC) was used. The topmost 3 figures exemplify the dynamics of the network at 50% of its original PYR population size, all other figures show the effects of various degrees of PYR population reduction (0-50%). **(A)** Raster plot of spiking activity. Only PYR cells are shown. The MCs are ordered such that those belonging to the same attractor (and *not* those within the same HC) are grouped together. **(B)** Star plot of average PYR cell voltages from a sample of 5 PYR cells per MC. **(C)** Star plot of average PYR cell firing rates. **(D)** Average dwell times and relative competition times for various network sizes. **(E)** Average firing rate of PYR cells during an UP state. **(F)** Average voltage of PYR cells before, during and after their parent attractor is active (UP state). **(G)** Average voltage of PYR cells before, during and after an attractor they do not belong to is active. For the previous three plots, the abscissa has been subdivided into multiples of the attractor dwell time. In subplots **F** and **G** the dotted line indicates the leak potential E_l of the PYR cells. **(H)** Pattern completion in a 25HC×25MC network. **(I)** Attentional blink in a 25HC×25MC network: $p = 0.5$ iso-probability contours. In **H** and **I**, the dataset for 5 PYR cells per MC was omitted because of its extremely low validity rate. Figure taken from Petrovici et al. (2014).

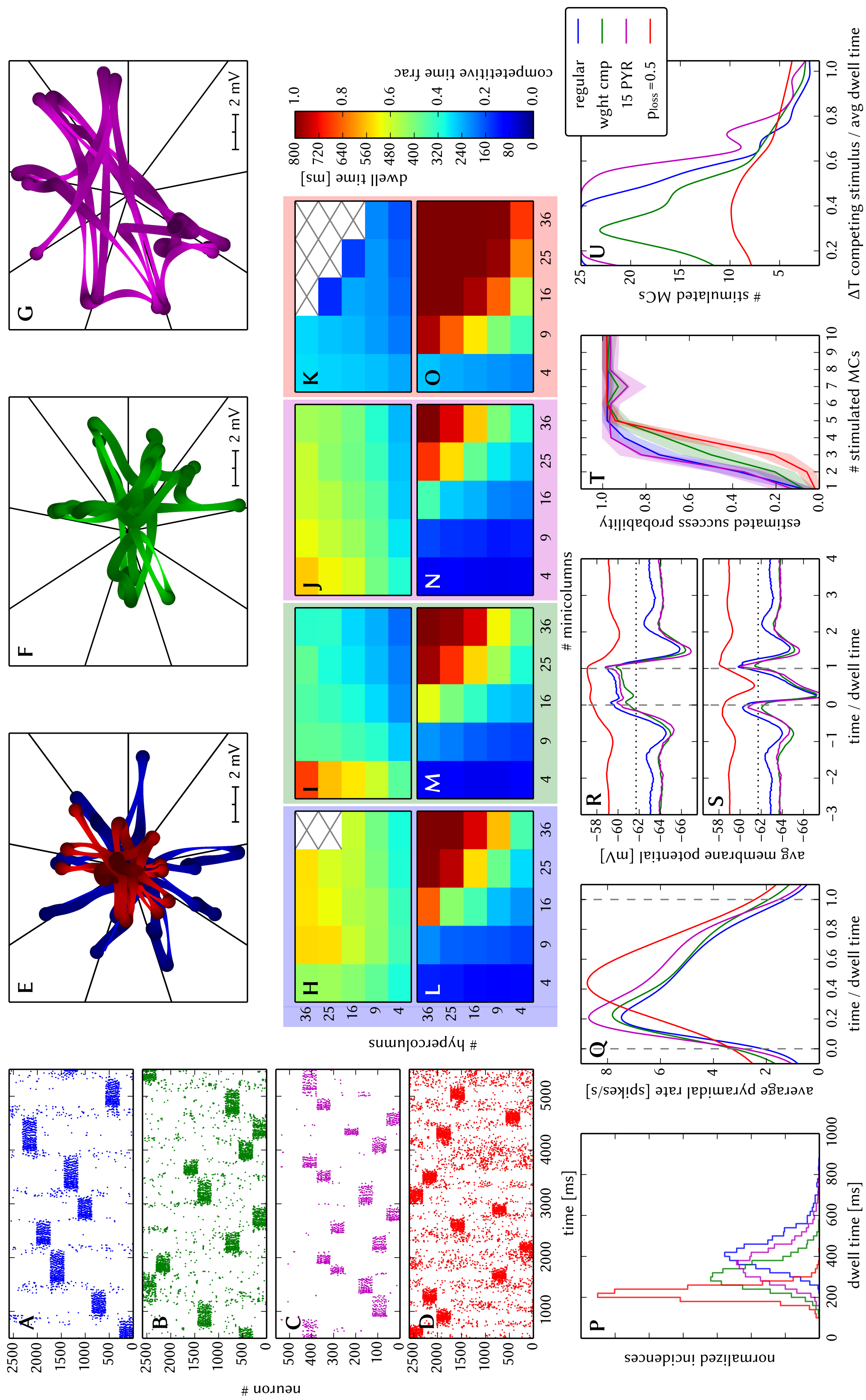

A
B
C
D
neuron #
time [ms]
E
F
G
2 mV
H
I
J
K
L
M
N
O
# hypercolumns
# minicolumns
dwell time [ms]
competetitive time frac
P
normalized incidences
dwell time [ms]
Q
average pyramidal rate [spikes/s]
time / dwell time
R
S
avg membrane potential [mV]
time / dwell time
T
estimated success probability
# stimulated MCs
U
# stimulated MCs
ΔT competing stimulus / avg dwell time
regular
wght cmp
15 PYR
$p_{loss} = 0.5$

Figure 5.7.: Compensation of homogeneous synaptic loss in the L2/3 model. Unless explicitly stated otherwise, the default network model (9HC×9MC) was used. Here, we use the following color code: blue for the original model, red for the distorted case (50% synapse loss), green for the compensation via increased synaptic weights and purple for the compensation by scaling down the size of the PYR populations. **(A) - (D)** Raster plots of spiking activity. The MCs are ordered such that those belonging to the same attractor (and *not* those within the same HC) are grouped together. Synapse loss weakens the interactions within and among MCs, causing shorter dwell times and longer competition times. Both compensation methods successfully counter these effects. These phenomena can also be observed in subplots **H-P**. **(E) - (G)** Star plots of average PYR voltages from a sample of 5 PYR cells per MC. Synapse loss leads to a less pronounced difference between the average PYR membrane potential within and outside of active attractors. After compensation, the differences between UP and DOWN states become more pronounced again. These phenomena can also be observed in subplots **R** and **S**. **(H) - (K)** Average dwell time for various network sizes. **(L) - (O)** Fraction of time spent in competitive states (i.e. no active attractors) for various network sizes. **(P)** Distributions of dwell times. **(Q)** Average firing rate of PYR cells within an active period of their parent attractor. **(R)** Average voltage of PYR cells before, during and after their parent attractor is active (UP state). **(S)** Average voltage of PYR cells before, during and after an attractor they do not belong to is active (DOWN state). For subplots **Q**, **R** and **S**, the abscissa has been subdivided into multiples of the average attractor dwell time in the respective simulations. In subplots **R** and **S** the dotted line indicates the leak potential E_l of the PYR cells. **(T)** Pattern completion in a 25HC×25MC network. Estimated activation probability from 25 trials per abscissa value. Synapse loss shifts the curve to the right, i.e., more MCs need to be stimulated to achieve the same probability of activating their parent attractor. Both compensation methods restore the original behavior to a large extent. **(U)** Attentional blink in a 25HC×25MC network: $p = 0.5$ iso-probability contours, measured over 14 trials per $(\Delta_\mathrm{T}, \#\mathrm{MCs})$ pair. Synapse loss suppresses attentional blink, as inhibition from active attractors becomes to weak to prevent the activation of other stimulated attractors. Compensation by increasing the weight of the remaining synapses alleviates this effect, but scaling down the PYR population sizes directly reduces the percentage of lost synapses and is therefore more effective in restoring attentional blink. Figure taken from Petrovici et al. (2014).

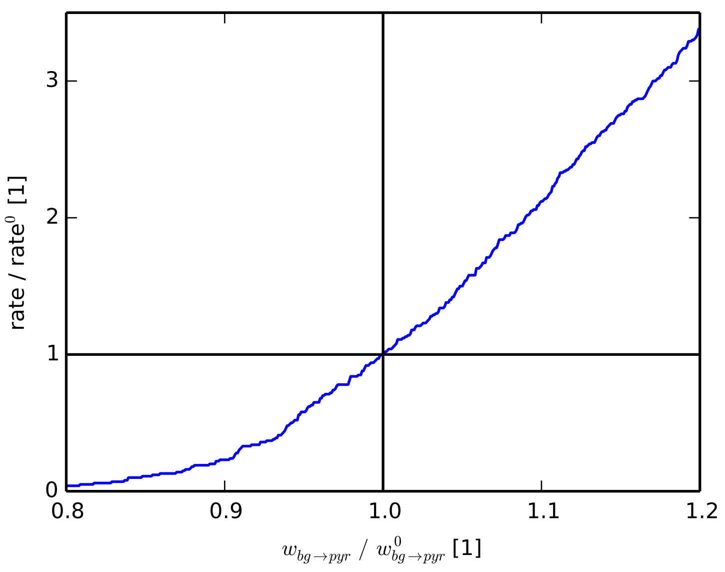


Figure 5.8.: Single PYR cell firing rate for different synaptic input weights. Each weight configuration was simulated for 100 s.

Poisson sources

Nevertheless, synaptic weight noise appears to have a strong effect on network dynamics (Figure 5.9, red curves). The reason for this lies in the way the network is stimulated. In the original model, each PYR cell receives input from a single Poisson source. This is of course a computational simplification and represents diffuse noise arriving from many neurons from other cortical areas. However, having only a single noise source connected by a single synapse to the target neuron makes the network highly sensitive to synaptic weight noise.

As can be seen in Figure 5.8, the firing rate of single PYR cells is highly dependent on the synaptic input weight that connects them to their respective Poisson source. For example, a variation of 20% in the input weight can cause the firing rate to either effectively vanish or more than triple. This heavily distorts network dynamics as PYR cells within MCs will exhibit highly disparate firing rates, thereby disrupting the network's ability to maintain stable UP states (in which all participating PYR cells should fire roughly with the same rate).

Compensation

The compensation for this effect was done by increasing the number of independent noise sources per neuron, thereby reducing the statistically expected relative noise conductance variations per PYR cell. The only limitation lies in the total number of available external spike sources and the bandwidth supplied by the off-wafer communication network (see Section 3.3.2). Once this limit is reached, the number of noise inputs per PYR cell can still be increased even further if PYR cells are allowed to share noise sources. The question therefore becomes the following: given a total number of available Poisson sources N and a noise population size of n sources per PYR cell, what is the average pairwise overlap between two such populations?

We can start by calculating the probability of choosing two subsets S_1 and S_2 of size $|S_1| = |S_2| = n$ with overlap k from a set S of cardinality $|S| = N$. Without loss of generality, we can assume S_1 to be fixed and simply count all possible realizations of S_2 such that k elements of S_2 are in S_1 and the other $n-k$ are in $S\backslash S_1$:

$$p(|S_1 \cap S_2| = k) = \frac{\binom{n}{k}\binom{N-n}{n-k}}{\binom{N}{n}} \quad , \tag{5.3}$$

where the normalization factor $\binom{N}{n}$ represents the total number of possible choices for S_2. The average overlap is then given by the expectation value of k:

$$\begin{aligned} E\left[k\right] = \sum_{k=0}^{n} k \cdot p(|S_1 \cap S_2| = k) &= \sum_{k=0}^{n} k \frac{\binom{n}{k}\binom{N-n}{n-k}}{\binom{N}{n}} \\ &= \frac{n}{\binom{N}{n}} \sum_{k=1}^{n} \binom{n-1}{k-1}\binom{N-1-(n-1)}{n-1-(k-1)} \quad . \end{aligned} \tag{5.4}$$

The product of binomial coefficients on the RHS can then be transformed with the relation

$$\sum_{k=0}^{m} \binom{m}{k}\binom{N-m}{n-k} = \binom{N}{n} \tag{5.5}$$

to yield

average pairwise presynaptic overlap

$$E\left[k\right] = \frac{n}{\binom{N}{n}} \binom{N-1}{n-1} = \frac{n^2}{N} \quad , \tag{5.6}$$

which gives us the average pairwise overlap between two background noise populations.

As long as the average overlap remains small enough, the overlap-induced spike correlations will not affect the network dynamics. In our example (Figure 5.9, green curves), we have chosen $n = 100$, while the total number of Poisson sources is set at $N = 5000$. We can now estimate the expected free membrane potential correlation of two PYR cells with Equation 4.143:

shared-input correlation

$$E\left[\rho_{u_1,u_2}\right] = \frac{E\left[k\right]}{n} = \frac{n}{N} = 0.02 \quad . \tag{5.7}$$

We consider this small enough to be negligible. Note how this relatively simple compensation method efficiently restores most functionality criteria (Figure 5.9, green curves). The most significant remaining differences can be seen in pattern completion and attentional blink (T, U) and appear mainly due to the affected RSNP$\rightarrow$PYR connections.

In addition to the investigation of synaptic weight noise on the default model, we repeated the same experiments for the model with reduced PYR population sizes (Figure 5.9, purple curves), which we have previously suggested as a compensation method for synaptic weight noise (in Section 5.3.5). The fact that PYR population reduction does not affect the network functionality in the case of (compensated) synaptic weight noise is an early indicator for the compatibility of the suggested compensation methods when all distortion mechanisms are present (Section 5.3.8).

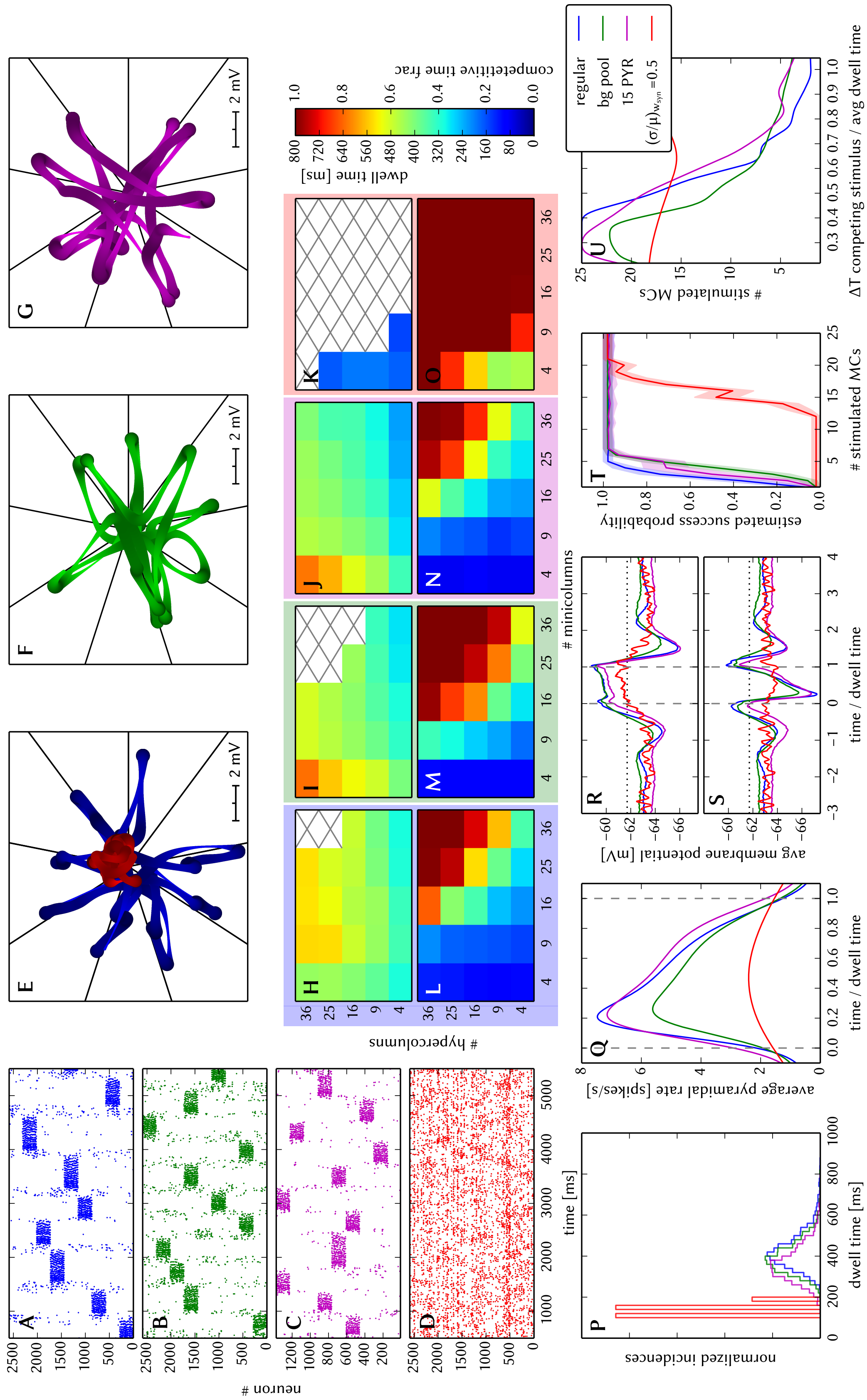

A
B
C
D
neuron #
time [ms]
E
F
G
2 mV
H
I
J
K
L
M
N
O
# hypercolumns
# minicolumns
dwell time [ms]
competetitive time frac
P
normalized incidences
dwell time [ms]
Q
average pyramidal rate [spikes/s]
time / dwell time
R
S
avg membrane potential [mV]
time / dwell time
T
estimated success probability
# stimulated MCs
U
# stimulated MCs
ΔT competing stimulus / avg dwell time
regular
bg pool
15 PYR
$(\sigma/\mu)_{w_{syn}}=0.5$

Figure 5.9.: Compensation of synaptic weight noise in the L2/3 model. Unless explicitly stated otherwise, the default network model (9HC×9MC) was used. Here, we use the following color code: blue for the original model, red for the distorted case (50% synaptic weight noise), green for the compensation via multiple background sources per PYR cell and purple for the same compensation method, but with scaled down PYR populations. Altogether, we note that the observed effects happen almost exclusively due to each PYR cell receiving background input via a single synapse. When compensated via the inclusion of multiple background sources, the network exhibits remarkable robustness towards synaptic weight noise. **(A)** - **(D)** Raster plots of spiking activity. The MCs are ordered such that those belonging to the same attractor (and *not* those within the same HC) are grouped together. When each PYR cell has a single background source, high levels of synaptic weight noise cause some PYR cells to become completely silent, while others spike disproportionately often. This can completely disrupt the stability of attractors, resulting in largely random spiking, with long competition times between the occasional appearance of weak, unstable attractors. The inclusion of multiple background sources per PYR cell efficiently counters these effects. This compensation strategy works just as well for downscaled PYR populations. The phenomena described above can also be observed in subplots **H-P**. **(E)** - **(G)** Star plots of average PYR voltages from a sample of 5 PYR cells per MC. The disrupted attractor behavior and erratic PYR spiking result in weak fluctuations of average PYR voltages with essentially no clear UP or DOWN states. After compensation, the differences between UP and DOWN states become more pronounced again. These phenomena can also be observed in subplots **R** and **S**. **(H)** - **(K)** Average dwell time for various network sizes. **(L)** - **(O)** Fraction of time spent in competitive states (i.e. no active attractors) for various network sizes. **(P)** Distributions of dwell times. **(Q)** Average firing rate of PYR cells within an active period of their parent attractor. **(R)** Average voltage of PYR cells before, during and after their parent attractor is active (UP state). **(S)** Average voltage of PYR cells before, during and after an attractor they do not belong to is active (DOWN state). For subplots **Q**, **R** and **S**, the abscissa has been subdivided into multiples of the average attractor dwell time in the respective simulations. In subplots **R** and **S** the dotted line indicates the leak potential E_l of the PYR cells. **(T)** Pattern completion in a 25HC×25MC network. Estimated activation probability from 25 trials per abscissa value. Due to erratically firing PYR cells in the distorted network, much stronger stimulation is needed to guarantee the appearance of an attractor. Compensation restores the original behavior to a large extent. **(U)** attentional blink in a 25HC×25MC network: $p = 0.5$ iso-probability contours, measured over 14 trials per (Δ_T, #MCs) pair. Due to the highly unstable attractors in the distorted network, attentional blink is completely suppressed. Compensation restores blink, but not to its original strength, due to the synaptic weight noise within the network itself. Figure taken from Petrovici et al. (2014).

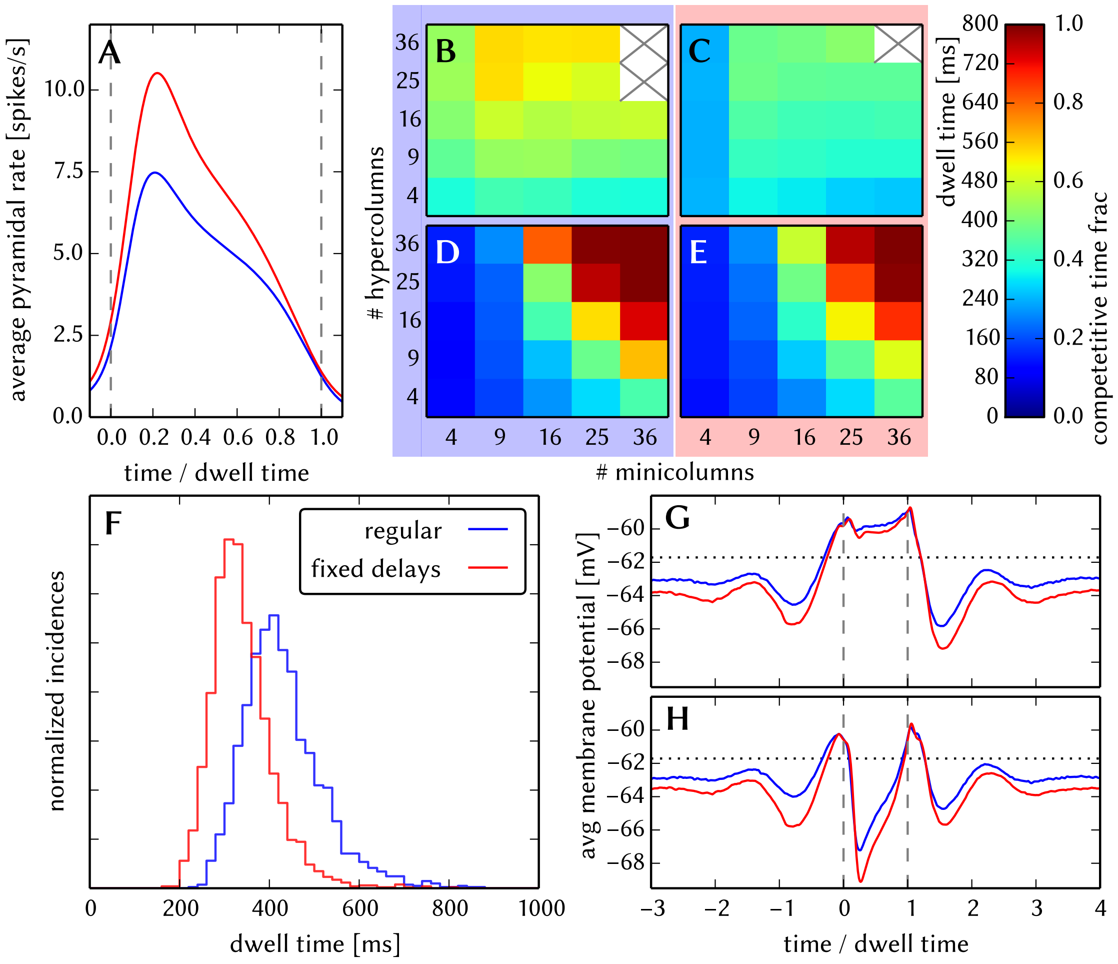


Figure 5.10.: Effects of fixed axonal delays on the L2/3 model. Unless explicitly stated otherwise, the default network model (9HC×9MC) was used. Data from the regular and distorted models is depicted (or highlighted) in blue, and red, respectively. **(A)** Average firing rate of PYR cells within an active period of their parent attractor. **(B)**, **(C)** Average dwell time for various network sizes. **(D)**, **(E)** Fraction of time spent in competitive states (i.e. no active attractors) for various network sizes. **(F)** Distributions of dwell times. **(G)** Average voltage of PYR cells before, during and after their parent attractor is active (UP state). **(H)** Average voltage of PYR cells before, during and after an attractor they do not belong to is active (DOWN state). For subplots **A**, **G** and **H**, the abscissa has been subdivided into multiples of the average attractor dwell time in the respective simulations. In subplots **G** and **H** the dotted line indicates the leak potential E_l of the PYR cells. Figure taken from Petrovici et al. (2014).

5.3.7. Non-Configurable Axonal Delays

In the original model, axonal delays between neurons are proportional to the distance between their home HCs. At an axonal spike propagation velocity of 0.2 m/ms, the default (9HC×9MC) network implements axonal delays distributed between 0.5 and 8 ms. While PYR cells within an MC tend to spike synchronously in gamma waves, the distribution of axonal delays reduces synchronicity between spike volleys of different MCs.

increased synchronicity

Fixed delays, on the other hand, promote synchronicity, thereby inducing subtle changes to the network dynamics (Figure 5.10). The synchronous arrival of excitatory spike volleys causes PYR cells in active attractors to spike more often (A). Their higher firing rate in turn causes shorter attractor dwell times, due to their spike frequency adaptation mechanism (B, C, F). During an active attractor, the elevated firing rate of its constituent PYR cells causes a higher firing rate of the inhibitory interneurons belonging to all other attractors. This, in turn, leads to a lower membrane potential for PYR cells during inactive periods of their parent attractor (G, H). As these effects are not fundamentally disruptive and also difficult to counter without significantly changing other functional characteristics of the network, we chose not to design a compensation strategy for this distortion mechanism in the L2/3 network.

5.3.8. Full Simulation of Combined Distortion Mechanisms

In a final step, we emulate the L2/3 model on the ESS (Section 3.3.4), and compensate simultaneously for all of the effects discussed above. We first investigate how much synapse loss to expect for different network sizes, and then realize the network at two different scales in order to investigate all of the chosen functionality criteria. The default network (9HC×9MC) is used to analyze spontaneous attractors, while a large-scale model (25HC×25MC) serves as the test substrate for pattern completion and pattern rivalry.

Synapse Loss

The synapse loss after mapping the L2/3 model onto the BrainScaleS hardware is shown in Figure 5.11 for different sizes, using the scaling rules defined in Section 5.3.2. Synapse loss starts to occur already at small sizes and increases rapidly above network sizes of 20 000 neurons. The jumps can be attributed to the different ratios between number of HCs and number of MCs per HC (Table A.11).

Small-Scale Model

The default model (9HC×9MC) can, in principle, be mapped onto the hardware without any synapse loss if the full wafer is available for use. Nevertheless, in some scenarios, a full wafer might not be available, due to faulty components or part of its area being used for emulating other parts of a larger parent network. We simulate this scenario by limiting the usable wafer area to 4 reticles (out of a total of 48 on the full wafer). With the reduced available hardware size, the available pulse bandwidth of the off-wafer communication network decreases as well, such that diffusive background noise can not be modeled with one individual Poisson source per neuron. Hence, each pyramidal neuron receives input from 9 out of 2430 background sources.

reduced available wafer area

The total synapse loss for the given network setup amounts to 22.2 % and affects different projection types with varying strength (Table 5.2). Also external synapses are

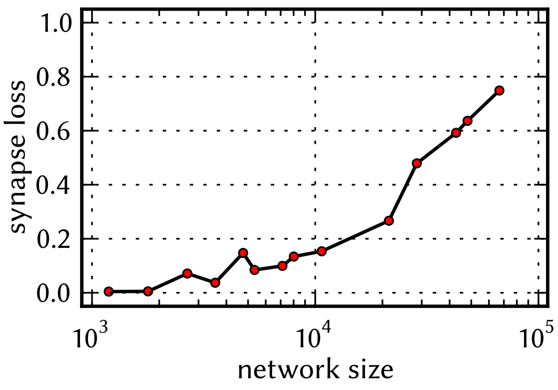


Figure 5.11.: Total synapse loss in the L2/3 model after mapping to the wafer. A more detailed listing can be found in Table 5.2. Figure taken from Petrovici et al. (2014).

lost, since, in contrast to the synapse loss study (Section 5.3.5), they have not been prioritized in the mapping process in this case. Additionally, we applied 20 % synaptic weight noise and simulated the network with a speedup factor of 12 000.

The behavior of the L2/3 network on the ESS is shown in Figure 5.12. The distorted network shows no spontaneous attractors (C), which can be mainly attributed to the loss of over 32 % of the background synapses. To recover the original network behavior, we first increased the number of background neurons per cell from 9 to 50 to compensate for synaptic weight noise, and also scaled the weights by $1/(1 - p_{\text{loss}})$ for each projection type with extracted synapse loss values p_{loss} (Table 5.2), following the synapse loss compensation method described in Section 5.3.5. Note that here the complete PyNN experiment is re-run: synaptic weights are scaled in the network definition leading to a new configuration of g_{max} and the digital weights on the HICANNs (Section 3.3.1) after the mapping process. These measures effectively restored the attractor characteristics of the network (Figure 5.12). The attractor dwell times remained a bit smaller than for the regular network (G), which can be ascribed to the non-configurable delays (Section 5.3.7).

Large-Scale Model

The ability of the network to perform pattern completion and exhibit pattern rivalry was tested on the ESS for the large-scale model with 25 HCs and 25 MCs per HC. From the start, we use a background pool with 5000 Poisson sources and 100 sources per neuron to model the diffusive background noise, as used for the compensation of synaptic weight noise (Section 5.3.6).

As with the small-scale network, the total synapse loss of 17.9 % shows significant heterogeneity (see Table 5.2), and affects mainly projections from PYR to inhibitory cells, but also connections from the background and L4 stimulus. In contrast to the idealized case in Section 5.3.5, where each synapse is deleted with a given probability, the synapse loss here happens for entire projections at the same time, i.e. all synapses between two populations are either realized completely or not at all. We note that the realization of all PYR-RSNP synapses is a priori impossible, as each RSNP cell has $24 \times 24 \times 30 = 17280$

projection-wise synapse loss

	9HC×9MC		25HC×25MC	
projection	dist	comp	dist	comp
PYR → PYR (local)	21.1	21.0	0.9	0.3
PYR → PYR (global)	20.8	21.2	8.0	0.4
PYR → RSNP	22.6	21.9	37.0	28.8
PYR → BAS	8.2	7.6	15.0	0.2
BAS → PYR	23.3	39.4	0.5	0.2
RSNP → PYR	22.7	39.9	0.0	3.9
L4 → PYR	44.1	45.4	15.5	2.3
background → PYR	32.3	31.3	17.3	1.3
total	22.2	25.2	17.9	9.8

Table 5.2.: Projection-wise synapse loss of the L2/3 model in % after the mapping process for the default (9HC×9MC) and large-scale (25HC×25MC) network.

potential pre-synaptic neurons (given the scaling rules in Section 5.3.2), which is more than the maximum possible number of pre-synaptic neurons per HICANN (14336, see Section 3.3.1).

The simulation results with 20 % synaptic weight noise for pattern completion and pattern rivalry are shown in Figure 5.12K, L (red curves). In both cases the network functionality is clearly impaired. In particular, the ability of an active pattern to suppress other patterns is noticeably detoriated, which can be traced back to the loss of 37 % of PYR-RNSP connections.

In order to restore the functionality of the network we used a two-fold approach: First, we attempted to reduce the binary loss of PYR-RSNP projections by reducing the number of PYR cells per MC from 30 to 20, which decreases the total number of neurons in the network, as well as the number of potential pre-synaptic neurons per RNSP cell. The synapse loss was thereby reduced to 28.8 % for PYR-RSNP projections and was eliminated almost completely for all other projections (see Table 5.2). Secondly, we compensated for the remaining synapse loss by scaling the synaptic weights as described in Section 5.3.5.

After application of these compensation mechanisms, we were able to effectively restore the original functionality of the network. Both pattern completion and attentional blink can be clearly observed. The small remaining deviations from the default model can be attributed to the inhomogeneity of the synapse loss and the fixed delays on the wafer.

5.3.9. Emulation on the Spikey Chip

When scaling down the original model (2673 neurons) to the maximum size available on the Spikey chip (192 neurons, see Figure 5.13B for software simulation results), we made use of the essential observation that the number of PYR cells can simply be reduced without compensating for it by increasing the corresponding projection probabilities (see Section 5.3.5). Also, for less than 8 MCs per HC, all BAS cells within an HC have identical afferent and efferent connectivity patterns, therefore allowing to treat them as a single population. Their total number was decreased, while increasing their efferent projection

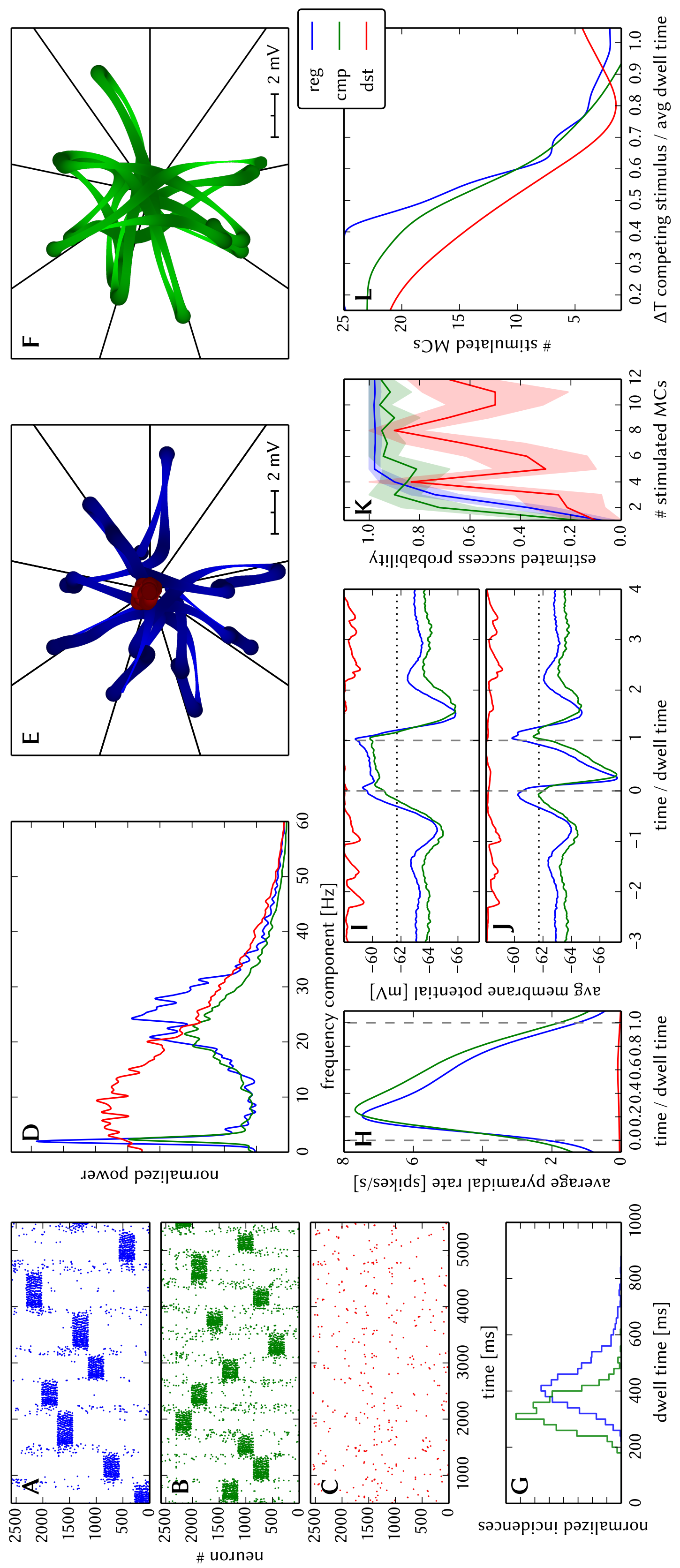

A
B
C
neuron #
time [ms]
G
normalized incidences
dwell time [ms]
D
normalized power
frequency component [Hz]
H
average pyramidal rate [spikes/s]
time / dwell time
I
J
avg membrane potential [mV]
time / dwell time
E
F
2 mV
K
estimated success probability
# stimulated MCs
L
# stimulated MCs
ΔT competing stimulus / avg dwell time
reg
cmp
dst

Figure 5.12.: ESS emulation of the L2/3 model. Unless explicitly stated otherwise, the default network model (9HC×9MC) was used. Here, we use the following color code: blue for the original model, red for the distorted case on the ESS (with 20 % synaptic weight noise and approx. 20 % synapse loss), and green for the compensated case on the ESS. **(A) - (C)** Raster plots of spiking activity. The MCs are ordered such that those belonging to the same attractor (and *not* those within the same HC) are grouped together. A synapse loss of 32 % on the background synpases (see Table 5.2) is the main reason for which no spontaneous attractors are evoked. For this reason, there are no red curves in **G**, **H**, **I** and **J**. Applying a weight compensation and increasing the number of background sources from 9 to 50 effectively restores the original behavior. **(D)** Power spectrum of global activity. Since no spontaneous attractors are evokes, neither attractor switching ($\sim 3\,\mathrm{Hz}$) nor gamma oscillations ($\sim 25\,\mathrm{Hz}$) can be observed. The spectrum of the distorted network complies with the asynchronous irregular firing observed in **C**. Compensation restores both of the characteristic peaks in the spectrum. **(E** and **F)** Star plots of average PYR voltages from a sample of 5 PYR cells per MC. The disrupted attractor behavior results in a weak fluctuations of average PYR voltages with essentially no clear UP or DOWN states. After compensation, the differences between UP and DOWN states become more pronounced again. **(G)** Distributions of dwell times. The disrupted network effectively shows no spontaneous attractors. As expected from the software simulations, the dwell times remain, on average, slightly shorter after compensation. **(H)** Average firing rate of PYR cells within an active period of their parent attractor. The higher firing rates after compensation are caused by the fixed, short delays, which promote synchronous firing and therefore stronger mutual excitation among PYR cells. **(I)** Average voltage of PYR cells before, during and after their parent attractor is active (UP state). **(J)** Average voltage of PYR cells before, during and after an attractor they do not belong to is active (DOWN state). For subplots **H**, **I** and **J**, the abscissa has been subdivided into multiples of the average attractor dwell time in the respective simulations. In subplots **I** and **J** the dotted line indicates the leak potential E_l of the PYR cells. **(K)** Pattern completion in a 25HC×25MC network. Estimated activation probability from 25 trials per abscissa value. (**L**) Attentional blink in a 25HC×25MC network: $p = 0.5$ iso-probability contours, measured over 14 trials per (Δ_T, #MCs) pair. Since the distorted network showed no spontaneous attractors (**C**), we used the average dwell time from the pattern completion experiment (**K**) for normalization. Figure taken from Petrovici et al. (2014).

probabilities accordingly. In general (i.e., except for PYR cells), when number and/or size of populations were changed, projection probabilities were scaled in such a way that the total fan-in for each neuron was kept at a constant average (see Section 5.3.2). When the maximum fan-in was reached (one afferent synapse for every neuron in the receptive field), the corresponding synaptic weights were scaled up by the remaining factor.

Because neuron and synapse models on the Spikey chip are different to the ones used in the original model, we have performed a heuristic fit in order to approximately reproduce the target firing patterns. Neuron and synapse parameters were first fitted in such a way as to generate clearly discernable attractors with relatively high average firing rates (see Figure 5.13D). Additional tuning was needed to compensate for missing neuronal adaptation, limitations in hardware configurability, parameter ranges and fixed-pattern noise affecting hardware parameters.

During hardware emulations, apart from the appearance of spontaneous attractors given only diffuse Poisson stimulation of the network (Figure 5.13C), we were able to observe two further interesting phenomena which are characteristic for the original attractor model: UP states of PYR cells during active attractors and pattern completion. These in-silico observations nicely match the characteristics of the model that we have previously discussed based on software simulations (both with Neuron and the ESS).

Figure 5.13E shows the emergence of such UP-states on hardware. The onset of an attractor is characterized by a steep rise in PYR cell average membrane voltage, which then decays towards the end of the attractor due to synaptic short-term depression and/or competition from other attractors temporarily receiving stronger stimulation. On both flanks of an UP state, the average membrane voltage shows a slight undershoot, due to the inhibition by other active attractors.

A second important characteristic of cortical attractor models is their capability of performing pattern completion (see Section 5.3.3). This means that a full pattern can be activated by stimulating only a subset of its constituent PYR cells (in the original model, by cells from cortical Layer 4, modeled by us as additional Poisson sources). To demonstrate pattern completion, we have used the same setup as in the previous experiments, except for one pattern receiving additional stimulation. From an initial equilibrium between the three attractors (approximately equal active time), we can observe how the stimulated attractor increasingly dominates the other two when an increasing subset of the PYR cells in its four minicolumns receive L4 stimulus (Figure 5.13F).

The implementation of the attractor memory model is a particularly comprehensive showcase of the configurability and functionality of the Spikey chip due to the complexity of both model specifications and emergent dynamics. As discussed in previous sections, the next-generation wafer-scale hardware will be able to much more accurately model biological behavior, thanks to a more flexible, adapting neuron model and especially a significantly increased number of available resources (i.e., allowing a larger network size).

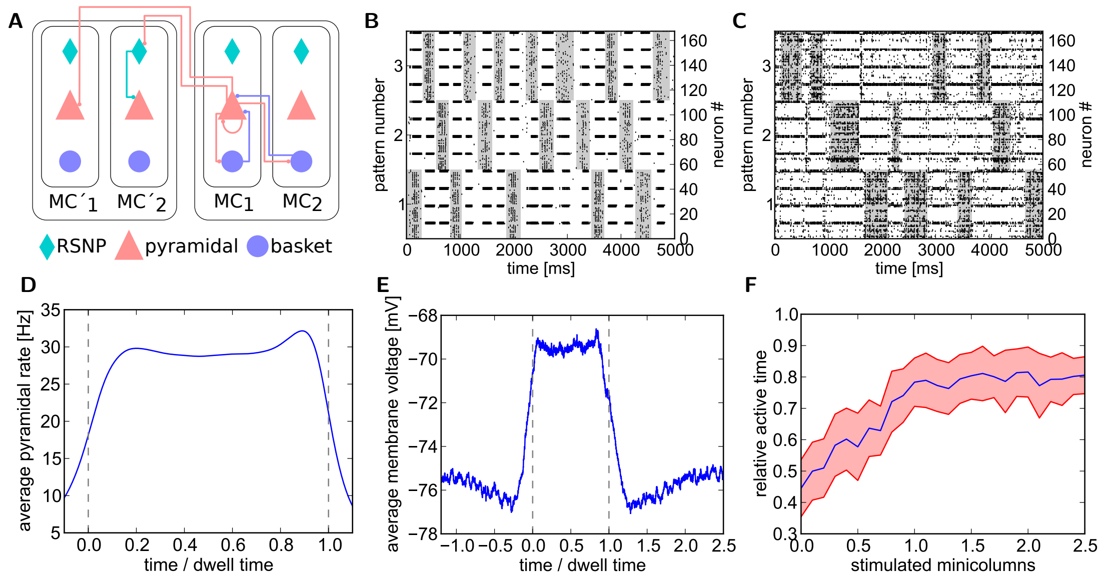


Figure 5.13.: L2/3 model on the Spikey chip. **(A)** Schematic of the cortical layer 2/3 attractor memory network. Two HCs, each containing two MCs, are shown. For better readability, only connections that are active within an active pattern are depicted. **(B)** Software simulation of spiking activity in the cortical attractor network model scaled down to 192 neurons (only PYR and RSNP cells shown, BAS cells spike almost continuously). MCs belonging to the same pattern are grouped together. The broad stripes of activity are generated by PYR cells in active attractors. The interlaced narrow stripes of activity represent pairs of RSNP cells, which spike when their home MC is inhibited by other active patterns. **(C)** Same as **B**, but on Spikey. The raster plot is noisier and the duration of attractors (dwell time) are less stable than in software due to fixed-pattern noise on neuron and synapse circuits. For better readability, active states are underlined in gray in **B** and **C**. **(D)** Average firing rate of PYR cells on Spikey inside active patterns. To allow averaging over multiple active periods of varying lengths, all attractor dwell times have been normalized to 1. **(E)** Average membrane potential of PYR cells on Spikey inside and outside active patterns. **(F)** Pattern completion on Spikey. Average values (from multiple runs) depicted in blue, with the standard deviation shown in red. From a relatively equilibrated state where all patterns take turns in being active, additional stimulation (see text) of only a subset of neurons from a given attractor activates the full pattern and enables it to dominate over the other two. The pattern does not remain active indefinitely due to short-term depression in excitatory synapses, thereby still allowing short occasional activations of the other two patterns. Figure taken from Pfeil et al. (2013).

5.4. Synfire Chain with Feedforward Inhibition

synfire chain

Our second benchmark network is a model of a series of consecutive neuron groups with feed-forward inhibition, called a synfire chain (Kremkow et al., 2010b). This network acts as a selective filter to a synchronous spike packet that is applied to the first neuron group of the chain. The behavior of the network is quantified by the dependence of the filter properties on the strength and temporal width of the initial pulse. Our simulations show that synapse loss can be compensated in a straightforward manner. Further, the major impact of weight noise on the network functionality stems from weight variations in background synapses, which can be countered by modification of synaptic and neuronal parameters. The effect of fixed axonal delays on the filtering properties of the network can be countered only to a limited extent by modifying synaptic time constants and the strength of local inhibition. Simulations using the ESS show that the developed compensation methods are applicable simultaneously. Furthermore, they highlight some further sources of potential failure of pulse propagation that originate from bandwidth limitations in the off-wafer communication infrastructure.

5.4.1. Architecture

Feed-forward networks with a convergent-divergent connection scheme provide an ideal substrate for the investigation of activity transport. Insights have been gained regarding the influence of network characteristics on its response to different types of stimulus (Aertsen et al., 1996; Diesmann et al., 1999; Vogels and Abbott, 2005). Similar networks were also considered as computational entities rather than purely as a medium for information transport (Abeles et al., 2004; Kremkow et al., 2010a; Schrader et al., 2010). The behavior of this particular network has been shown to depend on the connection density between consecutive groups, on the balance of excitation and inhibition as well as on the presence and magnitude of axonal delays in Kremkow et al. (2010b). This makes it sensitive to hardware-specific effects such as an incomplete mapping of synaptic connectivity, the variation of synaptic weights, bandwidth limitations which cause loss of individual spike events and limited availability of adjustable axonal delays and jitter in the spike timing that may be introduced by different hardware components.

RS, FS cells

The feed-forward network comprises a series of successive neuron groups, each group containing one excitatory and one inhibitory population. The excitatory population consists of 100 regular-spiking (RS), the inhibitory one of 25 fast-spiking (FS) cells. Both cell types are modeled as LIF neurons with exponentially shaped synaptic conductance without adaptation, as described in Section 3.3.1. Both RS and FS neurons are parameterized using identical values (Table A.12).

feedforward inhibition

Each excitatory population projects to both populations of the consecutive group while the inhibitory population projects to the excitatory population in its local group (Figure 5.14A). There are no recurrent connections within the RS or FS populations.

In the original publication (Kremkow et al., 2010b), each neuron was stimulated independently by a Gaussian noise current. Because the hardware system does not offer current stimulus for all neurons, all neurons in the network received stimulus from independent Poisson spike sources. For Gaussian current stimulus, as well as in the diffusion

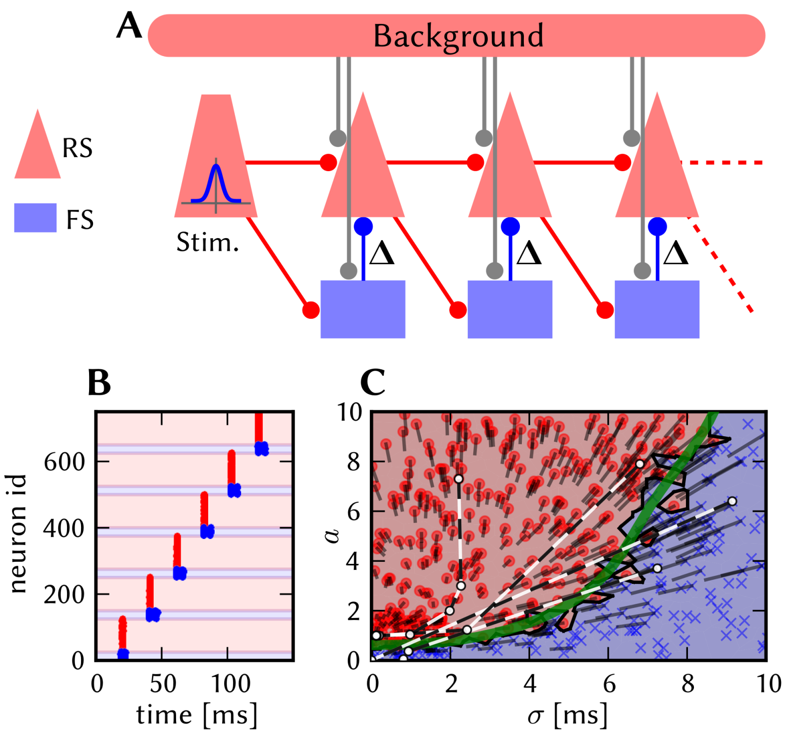


Figure 5.14.: Synfire chain network. **(A)** Architecture of the synfire chain with feedforward inhibition. Excitatory projections are shown in red, inhibitory ones in blue. In the default realization the network consists of six consecutive groups. The local FS $\to$ RS projection has an adjustable delay Δ, which affects the network dynamics. The intergroup delay is set to 20 ms for visualization purposes following Kremkow et al. (2010b); this has no influence on the filter properties because the delay of both intergroup projections is equal. The background stimulus is realized using random Gaussian current (original) and Poisson background spikes (adapted version for the hardware). The parameters for neurons and connections are given in Tables A.12 and A.13. **(B)** Exemplary raster plot of the network behavior. The first group receives a pulse packet with $a = 1$ and $\sigma_0 = 1$ ms, which propagates as a synchronous spike volley along the chain. **(C)** Characterization of the network behavior in the (σ, a) state space. Each marker represents the initial stimulus parameters (σ_0, a_0). The stimulus parameters were selected randomly from the region ($a_0 < 10$, $\sigma_0 \leq 10$ ms). The region with ($a_0 < 2$, $\sigma_0 \leq 2$ ms) was simulated more frequently to increase the resolution near the convergence points of the propagation. The marker color is linearly scaled with the activity in the last group, a_6, being blue for $a_6 = 0$ and red for $a_6 = 1$ and is set to red for $a_6 > 1$. To improve visibility, the background is colored according to the color of the nearest marker, red for $a_6 \geq 0.5$ and blue otherwise. Experiments where the RS group did not fire are marked as $\times$. The gray lines originating from each marker denote the direction towards the pulse volley parameters (σ_1, a_1). The green line shows a fit to the separatrix between zero and nonzero activity at the last group of the synfire chain (see Section 5.4.3 for details). This approximation is used to compare the behavior of different modifications of the original network. The dashed black and white lines show four exemplary trajectories through the (σ, a) state space. Figure taken from Petrovici et al. (2014).

limit of Poisson stimulus (high input rates, low synaptic weights), the membrane potential is stationary Gaussian, with an autocorrelation dominated by the largest dynamic time constant, which for this model is the membrane time constant (see Sections 4.3.4 and 6.5.2). The only remaining differences are due to the finite, but small, synaptic time constants. The rate and synaptic weight of the background stimulus were adjusted to obtain similar values for the mean and variance of the membrane potential, resulting in a firing rate of 2 kHz with a synaptic weight of 1 nS.

The initial synchronous stimulus pulse is emitted by a population of spike sources, which has the same size and connection properties as a single RS population within the network. *pulse packet* A temporally localized pulse packet was used as a stimulus, whereby each of the 100 spike sources emitted a_0 spikes that were sampled from a Gaussian distribution with a common mean time and a given standard deviation σ_0. The variables (σ_i, a_i) are later used to describe the characteristics of the activity in the ith group of the chain, referring to the temporal pulse width and number of spike pulses per neuron, respectively.

5.4.2. Network Scaling

In the default setup studied here, the synfire chain consists of 6 groups of 125 neurons (100 excitatory and 25 inhibitory). In order to quantify the amount of synapse loss after mapping the network to the BrainScaleS wafer-scale hardware for different network sizes, we define the following network scaling rules. *constant fan-in* When increasing the network size, we vary both the number of synfire groups and the number of neurons per group while keeping the number of incoming synapses per neuron constant (see Table A.13). The fraction of inhibitory neurons always amounts to 20 %. Neuron and synapse parameters are not altered. Table 5.3 lists the combinations of group size and group count used for the synapse loss estimation in Figure 5.19A.

Poisson background The background Poisson stimulus is scaled as follows. For the hardware implementation of the synfire chain we can not use one individual Poisson source for each neuron due to input bandwidth limitations. Instead, we assume one pool of 32 Poisson sources for each synfire group, and each neuron receives input from 8 random sources from that pool. The size of the background pool is then scaled with the number of neurons per synfire group, while always drawing 8 sources from the pool per neuron. This scaling of the background pool was chosen to make the total number of background sources proportional to the total number of neurons and independent of the group count.

5.4.3. Functionality Criteria

The functionality of the feed-forward network is assessed by examining the propagation of a synchronous pulse after the stimulus is applied to the first group in the chain (Figure 5.14B). The propagation is quantified by applying initial stimuli of varying strength $a_0 \in [0, 10]$ and temporal spread $\sigma_0 \in [0\,\text{ms}, 10\,\text{ms}]$. For each synfire group $i \in \{1, ..., 6\}$, the activation is determined by setting a_i to the number of emitted spikes divided by the number of neurons in the RS population. The standard deviation of the spike pulse times is denoted by σ_i. Typically, the resulting "trajectory" in the (σ, a) space (Figure 5.14) is *fixed points* attracted to one of two fixed points: either near $(\sigma = 0\,\text{ms}, a = 1)$, i.e., the pulse packet propagates as a synchronous spike volley, and $(0\,\text{ms}, 0)$, i.e., the propagation dies out (e.g., Figure 5.15A).

groups	group size	total neurons	groups	group size	total neurons
8	125	1000	20	500	10 000
16	125	2000	40	500	20 000
24	125	3000	60	500	30 000
20	200	4000	40	1000	40 000
25	200	5000	50	1000	50 000
15	400	6000	30	2000	60 000
20	350	7000	20	3500	70 000
20	400	8000	20	4000	80 000
30	300	9000	30	3000	90 000
25	400	10 000	25	4000	100 000

Table 5.3.: Scaling table for the synfire chain used for the synapse loss estimation in Figure 5.19A.

The network behavior is characterized by the separating line between successful and extinguished propagation in the state space (σ, a) of the initial stimulus; this line will be called separatrix from here on. The differentiation between extinguished and successful propagation is defined by $a_6 \geq 0.5$ resp. $a_6 < 0.5$ in the last (6th) group. This is justified because in the undistorted case, a is clustered around the values 0 and 1. Due to the statistic nature of the connectivity, background stimulus and pulse packet, the macroscopic parameters σ and a do not fully determine the behavior of the system. This means that in the reference simulation, there is a small region around the separatrix where the probability of a stable pulse propagation is neither close to zero nor to one. Thus, in addition to the location of the separatrix (Section A.2.4.2), the width of this region is taken as a functionality criterion.

separatrix

The background stimulus is adjusted such that the spontaneous firing rate in the network is below 0.1 Hz, in accordance with Kremkow et al. (2010b). In cases in which distortion mechanisms induce a much stronger background firing, the spike trains are filtered before the analysis by removing spikes which appear not to be within a spike volley (Section A.2.4.3).

5.4.4. Synapse Loss

On average, homogeneous synapse loss affects the strength of excitatory and inhibitory projections equally. Additionally, the number of incoming spikes seen by a single neuron varies as synapses are removed probabilistically, in contrast to the undistorted model with a fixed number of incoming connections for each neuron type (Table A.13). Synapse loss was applied to all internal connections as well as to the connection from the synchronized stimulus population to the first group in the network; background stimulus was not affected (see Section 5.1).

Figure 5.15A shows a single experiment with synapse loss of 37.5 %, contrasting with the undistorted case (Figure 5.14A). Above a certain value of synapse loss, the signal fails to propagate to the last group. As shown in Figure 5.15C and E for one stimulus parameter set, successful propagation stops at a synapse loss value between 30% and 40%. The pulse width increases with rising synapse loss due to the increasing variation of

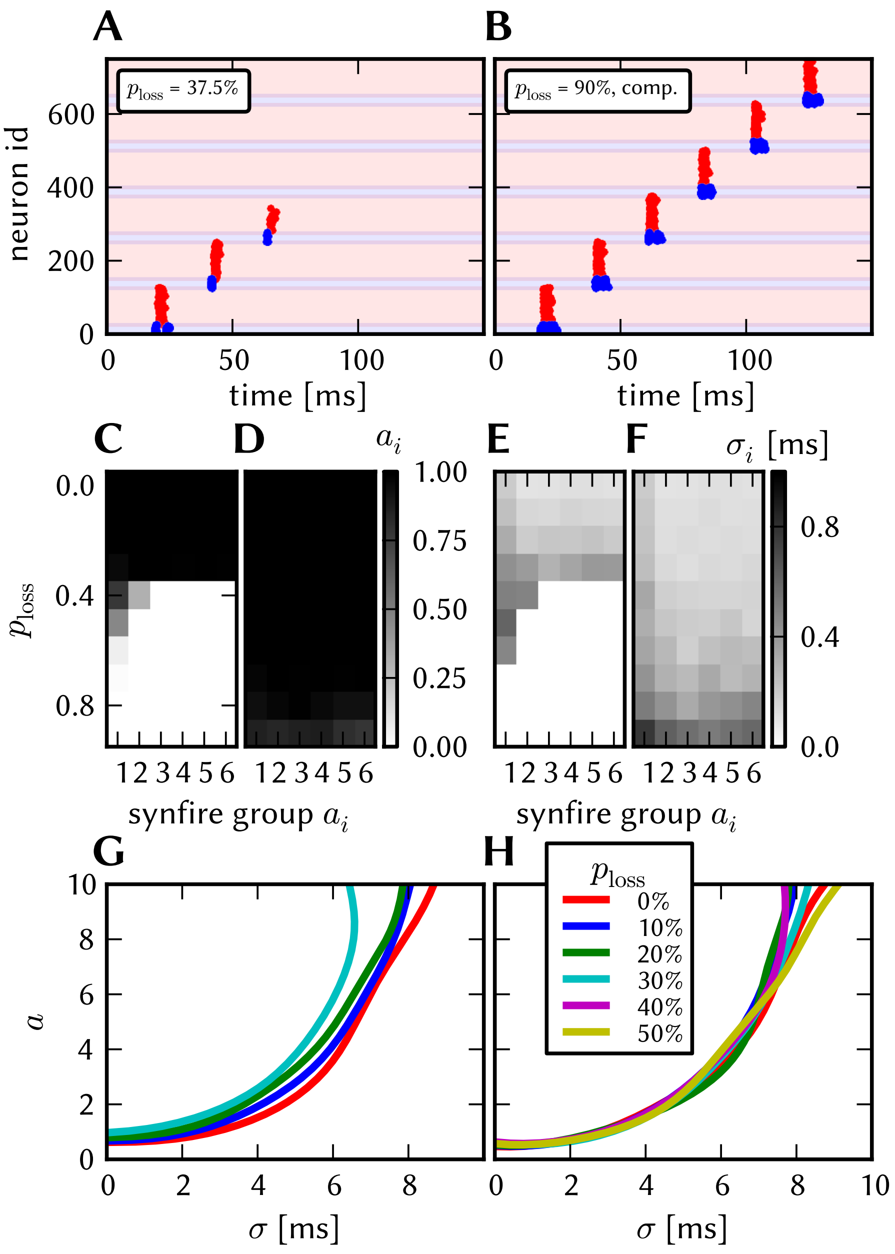


Figure 5.15.: Effect and compensation of synapse loss for the synfire chain network. **(A)** Synfire chain with 37.5% synapse loss applied to all connections within the network. External connections (synchronous stimulus and background) are not affected. **(B)** Raster plot with active compensation at 90% synapse loss. **(C)** Activation a_i in each group i with varying values of synapse loss. **(D)** a_i as in **C** but with active compensation. **(E)** Pulse width σ_i in each group i with varying values of synapse loss. **(F)** Pulse width as in **E** but with active compensation. **(G)** Comparison of approximated separatrix locations for synapse loss values from 0% to 50%. The lines for 40% and 50% are missing because no stable region exists. **(H)** Approximated separatrix locations with active compensation. Figure taken from Petrovici et al. (2014).

synaptic conductance for individual neurons (E). The effect is reversed by increasing all synaptic weights in the network by a factor of $1/(1-p_{\text{loss}})$, with p_{loss} being the probability of synapse loss. This compensation strategy can effectively counter synapse loss of up to 90 % (B, D) and the pulse width increase is shifted to larger values of synapse loss (F).

weight compensation

The distortion mechanism has only a minor effect on the a-value of the separatrix in the depicted region (G). However, the location of the separatrix at $\sigma_0 = 0$ rises with synapse loss until it reaches the fixed point at approximately $(0.1\,\text{ms}, 1)$, at which point a bifurcation occurs and the the attractor region for $(0.1\,\text{ms}, 1)$ disappears (as described in Diesmann et al. (2001) for the case of varying weights). In the compensated case, the separatrix locations are identical with the undistorted case within the measurement precision.

bifurcation

5.4.5. Synaptic Weight Noise

The effect of synaptic weight noise is shown in Figure 5.16. Similarly to the effect of synapse loss, the region of stable propagation shrinks (B); additionally, the border between the regions of stable and extinguished propagations becomes less sharp (A). This is caused by two effects: varying strength of the background stimulus, and varying strength of the synaptic connections within the network.

The first effect is significant because the background stimulus to each neuron is provided through a single synapse. Thus, the effective resting potential of each neuron is shifted, significantly changing its excitability and, in some cases, inducing spontaneous activity. One possibility of countering this effect is to utilize several synapses for background stimulus, thereby averaging out the effect of individual strong or weak synapses, as has been done in the case of the L2/3 model in Section 5.3.6. Here, we propose a different method. The resting potential E_{l} can be raised while simultaneously lowering the synaptic weight from the background stimulus. We can choose the parameters in such a way that the mean and variance of the distribution of free membrane voltages (see Equations 4.69 and 4.77) in each neuron population are kept at the default value (without synaptic weight noise):

$E[u]$ and $\text{Var}[u]$ conservation

$$E[u] \approx w_0 E[K] + E_{\text{l}} \quad \text{and} \tag{5.8}$$

$$\text{Var}[u] \approx w_0^2 \text{Var}[K] + \text{Var}[w^{\text{syn}}] \Big(E[K]^2 + \text{Var}[K] \Big) \quad , \tag{5.9}$$

where

$$K(t) = \sum_{\text{spk } j} \kappa(t - t_j) \tag{5.10}$$

represents the effect of the background stimulus, κ being the PSP kernel, and $\text{Var}[w^{\text{syn}}] = w_0^2 \sigma^2$ appears due to synaptic weight noise.

In the distorted case, the width of this distribution is a combined effect of the random background stimulus and the weight variation, while in the original case it originates from the stochasticity of the stimulus only. In the undistorted case, $\text{Var}[w^{\text{syn}}] = 0$, and only the first term contributes to $\text{Var}[u]$. With increasing σ^2, the contribution of the second term to $\text{Var}[u]$ increases, which is compensated by changing w_0 accordingly, keeping $\text{Var}[u]$ at the original level. This, in turn, changes $E[u]$, which is compensated by a change of E_{l}.

The effect of synaptic weight noise within the network itself is less significant compared to its impact on the background stimulus. Figure 5.16C shows that removing the effect of background stimulus noise alone is sufficient to counteract synaptic noise values of up to at least 50%.

5.4.6. Non-Configurable Axonal Delays

Figure 5.17A shows the effect of varying axonal delays between the inhibitory and excitatory population of a single synfire group. As was shown in Kremkow et al. (2010b), the delay can be employed to control the position of the separatrix between stable and unstable propagation. Because the axonal delay is not configurable for on-wafer connections, a different method is required to regain the ability to control the separatrix. While Sections 5.4.4 and 5.4.5 show that synaptic weight noise and synapse loss can influence the location of the separatrix, a method is required that is independent of those distortion mechanisms.

Diesmann (2002) shows that several parameters, including group size and noise level, can modify the separatrix location, albeit for a model without feed-forward inhibition. Here, we investigate to which extent parameter modification can reproduce the effect of variable delays.

$\tau_{\mathrm{i}}^{\mathrm{syn}}$ *delay compensation*

For very short delays (in this case, 0.1 ms, not shown), stable propagation does not occur, because the onset of local inhibition is nearly synchronous with the onset of external excitation. This effect was countered by increasing the synaptic time constant and simultaneously decreasing the synaptic weight for local inhibition, thus extending the duration of inhibition that acts on the RS population. The inhibitory synaptic time constant was increased by a factor of 3 while simultaneously reducing the synaptic weight of the inhibitory projection.

Figure 5.17B shows the result of the compensation for 1.5 ms local inhibition delay. For both values of axonal delay, the location of the separatrix can be controlled by changing the weight of inhibition. However, its shape differs from the delay-induced case because of the modified delay mechanism of inhibition. Reduction of the weight beyond a certain point is not possible, as balanced inhibition is required for network functionality (Kremkow et al., 2010b). It is important to note that this kind of compensation is specific to the state space region which is examined, and that it can not be extended to arbitrarily large delays.

5.4.7. Additional Simulations

To check that the proposed compensation methods do not interfere with each other, all distortion mechanisms were applied simultaneously with weight noise values of 20 % and 50 % and synapse loss values of 30 % and 50 %, with an axonal delay of 1.0 ms. Without compensation, no stable region exists in all four cases. Figure 5.18 shows the result with all compensation methods applied.

Note that when several methods required modification of a network parameter, all modifications were applied. For instance, in the case of the synaptic weight which needed to be scaled by both synapse loss and delay compensation methods, both scaling factors were multiplied. In all four cases, the applied compensation methods are successful in restoring the original input selectivity of the network.

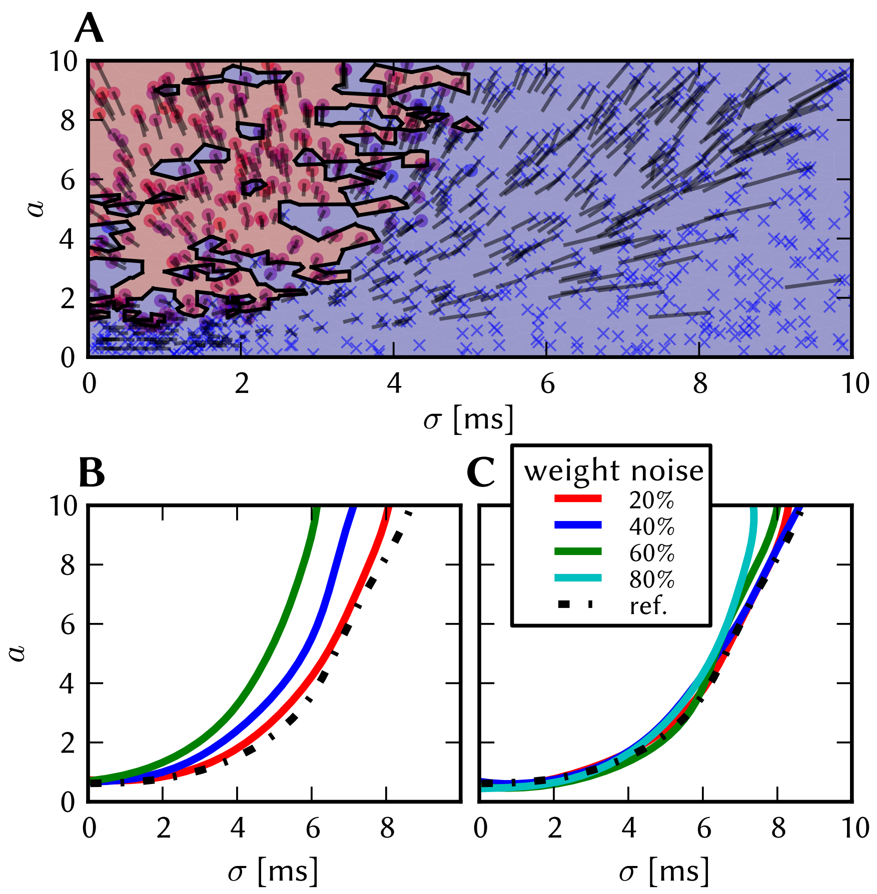


Figure 5.16.: Effect of synaptic weight noise on the synfire chain model. The spike data for all three plots was filtered to remove spontaneous spikes in individual neurons, which stem from weight increase in some background synapses due to weight noise. (The filter parameters were $T = 10\,\mathrm{ms}, N = 25$, see Section A.2.4.3) **(A)** State space at 80% weight noise. The set of inputs that evokes activity in the last group is patchy as a consequence of the distortion mechanism. In the compensated case the separation is sharp again, as shown in Figure A.3. **(B)** Approximate separatrix locations for smaller values of weight noise. **(C)** Approximate separatrix locations for the compensated case. Figure taken from Petrovici et al. (2014).

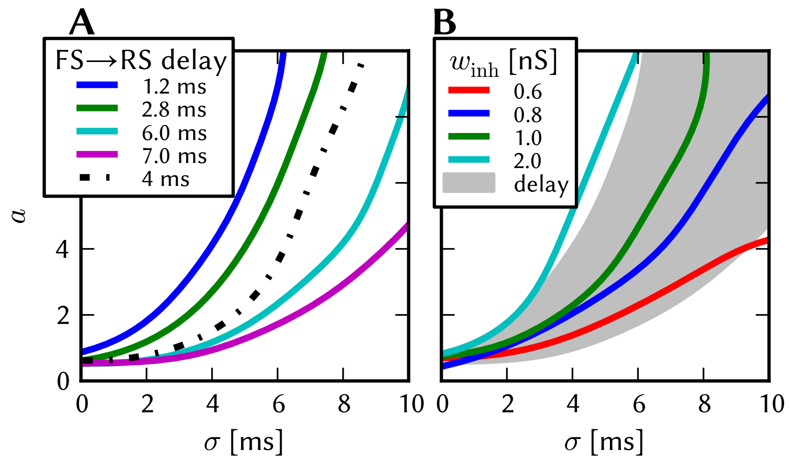


Figure 5.17.: Delays in the synfire chain model. **(A)** Reproduction of Kremkow et al. (2010b), fig. 4c. The location of the separatrix is modified by changing the axonal delay of local inhibition. For a value of 0.4 ms, no stable region is present. **(B)** The location of the separatrix is modified by varying weights for synapses taking part in local inhibition. The axonal delay of local inhibition was fixed at 1.5 ms and the inhibitory time constant was increased by a factor of 3. The gray region shows the range of the separatrix location for delay values from 1.2 ms to 7 ms (the range in plot **A**) as reference. Figure taken from Petrovici et al. (2014).

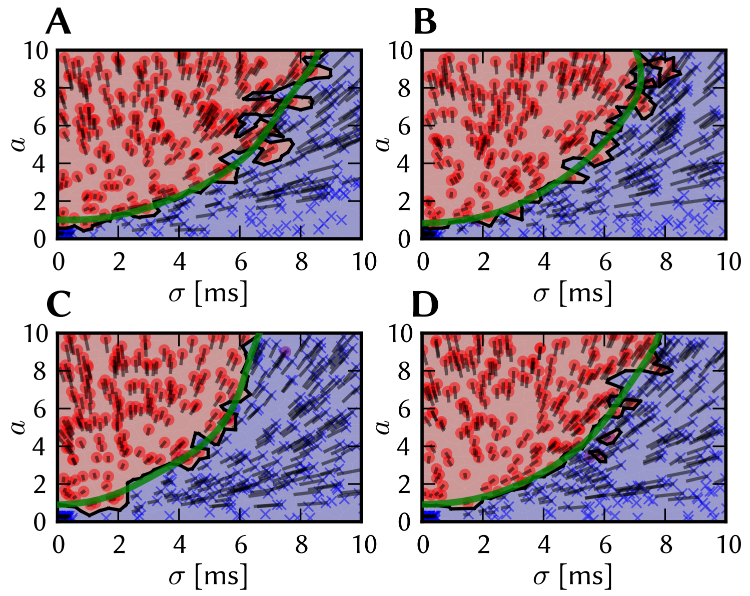


Figure 5.18.: (σ, a) state space of the synfire chain model with all compensation methods applied for four different levels of distortion. **(A)** 30 % synapse loss, 20 % weight noise . **(B)** 30 % synapse loss, 50 % weight noise . **(C)** 50 % synapse loss, 20 % weight noise . **(D)** 50 % synapse loss, 50 % weight noise. Figure taken from Petrovici et al. (2014).

5.4.8. Full Simulation of Combined Distortion Mechanisms

In a final step, we simulate the synfire chain with the ESS and compensate simultaneously for all the causes of distortions addressed above. We proceed with a quantification of synapse loss after mapping the synfire chain for different network sizes to the hardware. For the ESS simulations, we limit the model to very few hardware resources to artificially generate synapse loss, such that all of the above distortion mechanisms are present (similarly to the L2/3 model, see Section 5.3.8). Additional hardware simulations investigating the influence of spike loss and jitter on the network functionality are described in Section 5.4.9.

reduced hardware size

Synapse Loss

We mapped the synfire chain at different network sizes onto the BrainScaleS wafer-scale hardware in order to quantify the amount of synapse loss (Figure 5.19A). For this purpose we developed network scaling rules that depend on the number and the size of the synfire groups (Section 5.4.2). Due to its modular structure and feed-forward connectivity scheme, there is no synapse loss for networks with up to 10 000 neurons. However, for network sizes above 30 000 neurons, the ratio of lost synapses increases abruptly. With increasing network size more neurons have to be mapped onto one HICANN, thereby reducing the number of hardware synapses per neuron. Moreover, as the group size grows with the network size (see Table 5.3), also the number of pre-synaptic neurons for all neurons mapped onto one HICANN increases, so that the maximum number of inputs to a HICANN, i.e., the synapse drivers, becomes a limiting constraint. The combination of both factors unavoidably leads to synapse loss.

Distorted and Compensated Simulation

For the ESS simulation, we applied several modifications to the benchmark network.

Originally, each cell in the network receives Poisson background stimulus from an individual source with 2000 Hz. Because the off-wafer pulse routing network does not support such high bandwidths (see Section 3.3.2), we reduce the total number of background sources from 750 to 192 and let each neuron receive input from 8 sources, while decreasing the Poisson rate by a factor of 8, using the same mechanism as for the compensation of synaptic weight noise in the L2/3 model (see Section 5.3.6). For the same reason, the network was emulated with a speedup factor of 5000 compared to biological real-time, whereby the effective bandwidth for stimulation and recording is doubled with respect to the normal operation with a speedup of 10 000.

As seen before, no synapse loss occurs for small networks. However, as discussed for the L2/3 model in Section 5.3.8, one can consider situations where only a small part of the wafer is available for experiments, or where some neurons or synaptic elements are defective or missing a calibration. Therefore, in order to generate synapse loss in the feed-forward network, we limited the network to only 8 out of 48 reticles of the wafer and furthermore declare half of the synapse drivers as not available. This resulted in a total synapse loss of 27.4 %. As with the L2/3 model, the synapse loss was not homogeneous but depended strongly on the projection type (Table 5.4).

We simulated the synfire chain with default neuron and synapse parameters on the ESS with 20 % synaptic weight noise and the synapse loss described above. The (σ, a) state

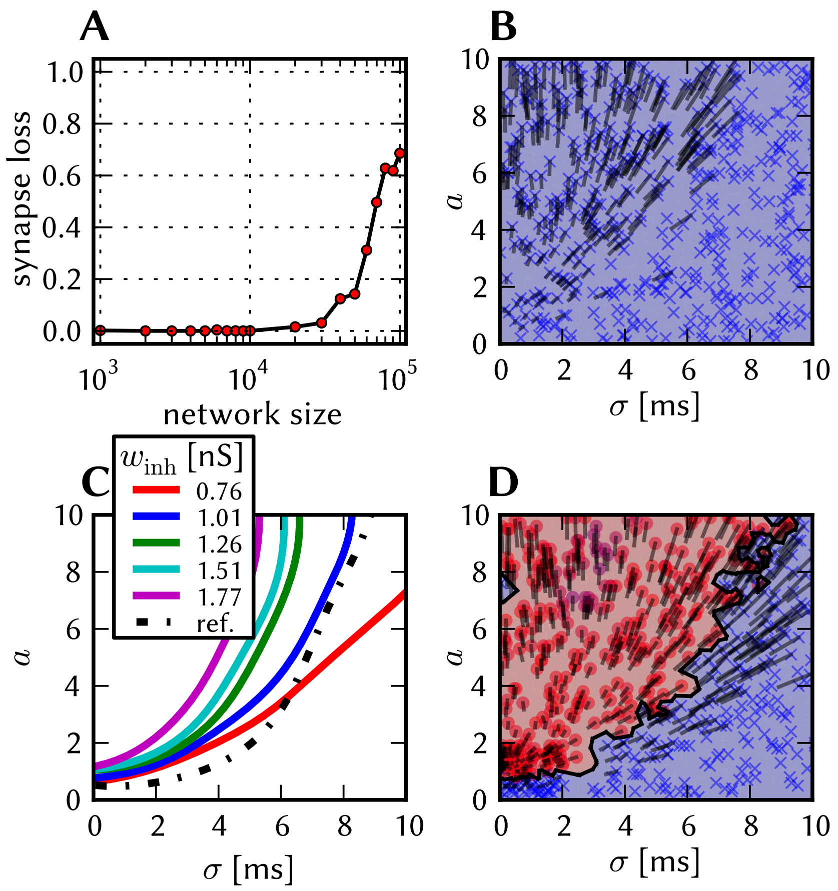


Figure 5.19.: Distorted and compensated simulations of the feedforward synfire chain on the ESS. **(A)** Synapse loss after mapping the model with different numbers of neurons onto the BrainScaleS System. **(B)** (σ,a) state space on the ESS with default parameters, 20 % weight noise, and 27.4 % synapse loss. **(C)** After compensation for all distortion mechanisms, different separatrices are possible by setting different values of the inhibitory weight. **(D)** Compensated state space belonging to the blue separatrix in **C**. Figure taken from Petrovici et al. (2014).

projection	synapse loss [%]
Pulse Packet $\rightarrow$ RS_0	21.3
Pulse Packet $\rightarrow$ FS_0	12.7
RS_n $\rightarrow$ RS_{n+1}	32.4
RS_n $\rightarrow$ FS_{n+1}	32.0
FS_n $\rightarrow$ RS_n	20.8
Poisson background $\rightarrow$ ALL	0
total	27.4

Table 5.4.: Projection-wise synapse loss of the synfire chain model after the mapping process.

space (Figure 5.19B) shows no stable point of propagation. This can be mainly attributed to the small and non-configurable axonal delays which are in the range of 0.6 ms to 1.1 ms for the chosen speedup factor of 5000.

In order to recover the original behavior, we applied the previously developed compensation methods (Sections 5.4.4 to 5.4.6). Synapse loss was compensated separately for each projection type using Table 5.4. For synaptic weight noise effectively two compensation methods were applied, as, by using 8 Poisson sources per neuron instead of one, the effect of weight variations is already reduced. This was straightforwardly accounted for by replacing $\mathrm{Var}\left[w^{\mathrm{syn}}\right]$ with $\frac{1}{8}\mathrm{Var}\left[w^{\mathrm{syn}}\right]$ in Equation 5.9. With our proposed compensation strategies, we were able to compensate for all distortion mechanisms while still maintaining control over the position of the separatrix (Figure 5.19C).

However, we also encountered some additional hardware-specific effect, which can be seen in Figure 5.19D.

When looking at the (σ, a) state space for one of the separatrices, for $\sigma \approx 3$ ms and $a > 7$ we can recognize a purple region indicating that not all RS cells of the last group spiked. In fact, spikes did occur for all RS cells in the simulated hardware network, but not all spikes were recorded because they were lost in the off-wafer communication network (Section 3.3.2).

For very small σ_0 an additional effect can appear. Input bandwidth limitations can result in very dense pulse volleys not being propagated through the synfire chain, as can be seen for the blue point with $\sigma_0 = 0.02$ ms and $a_0 = 3.3$ on the very left of the (σ, a) state space. In that particular case, the large majority of input spikes were lost in the off-wafer communication network so that they did not even reach the first synfire group. We remark that this effect only appeared for σ_0 smaller than 0.1 ms.

5.4.9. Further ESS Simulations

Distortion and Compensation Without Synapse Loss

For the ESS simulation in Section 5.4.8, we enforced a certain amount of synapse loss by restricting the synfire chain network to very limited hardware resources. However, due to its feed-forward structure, the network can be easily mapped onto the BrainScaleS hardware without any synapse loss (Figure 5.19A). Thus, we also investigated the network without synapse loss, such that the active distortion mechanisms in the ESS simulations were synaptic weight noise, non-configurable axonal delays as well as spike loss and jitter. The state space of the distorted network (Figure 5.20A) contains only a small and loosely connected region of sustained activity which indicates unreliable separation. Applying the compensation mechanism for synaptic weight noise and axonal delays fully restores the filter property of the synfire chain, as can be seen in Figure 5.20B, where different separatrices mimic different delay-dependent realizations. Compared to the compensation for all distortion mechanisms, the compensated state space without synapse loss does not show any flaws (C).

Effect of Spike Loss and Jitter

We investigated the effect of spike loss and jitter in the HICANN, where the spikes of the neurons connected to the same on-wafer routing bus are processed subsequently

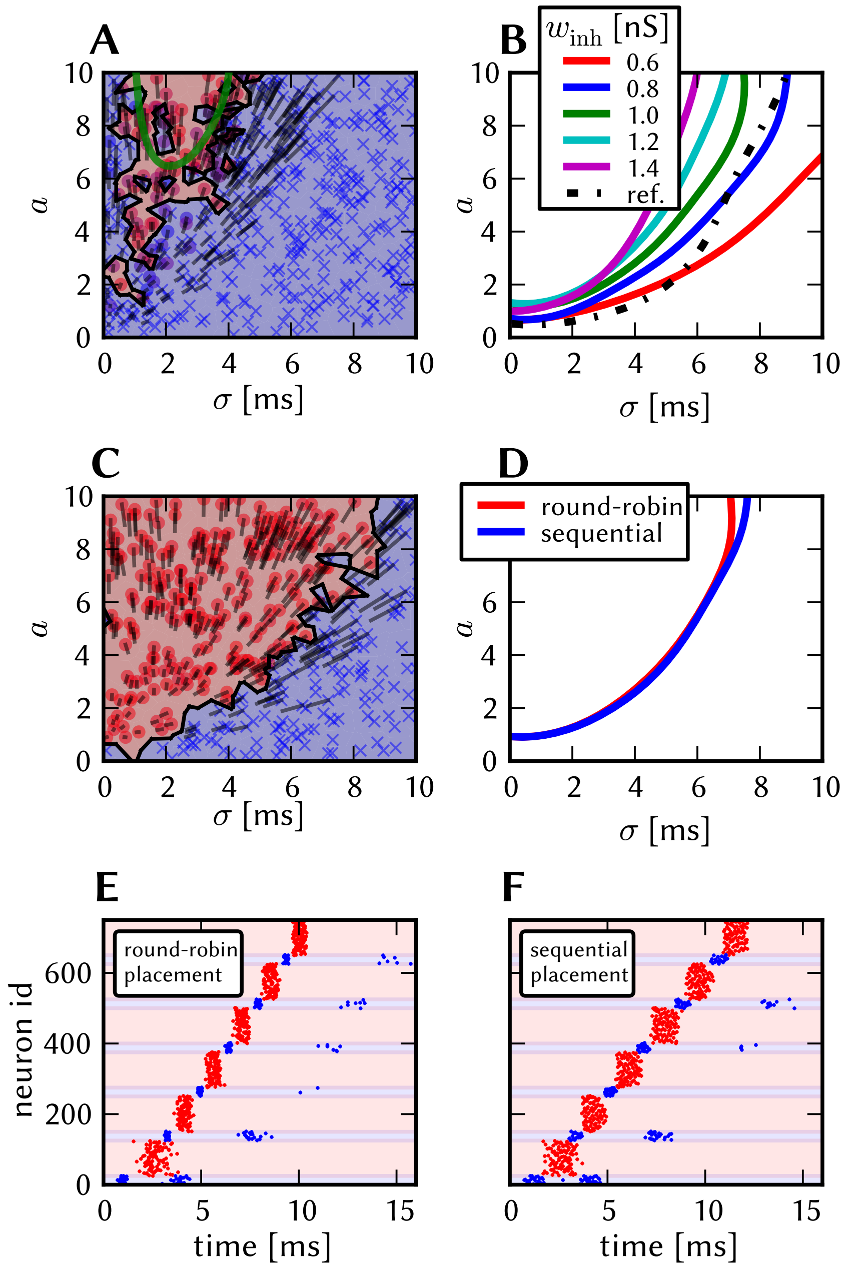


Figure 5.20.: Additional simulations of the feed-forward synfire chain on the ESS without synapse loss. **(A)** (σ,a) state space on the ESS with default parameters and 20% weight noise. **(B)** After compensation of for all distortion mechanisms, different separatrices are possible by setting different values of the inhibitory weight. **(C)** Compensated state space belonging to the blue separatrix in **B**. w refers to the synaptic weight of local inhibition. **(D)**-**(F)** Investigation of effects of spike loss and jitter by using two different approaches for neuron placement. **(D)** Separatrices for round-robin and sequential neuron placement with parameters as for the green curve in **B**. Raster plots for round-robin (**E**) and sequential (**F**) neuron placement. Stimulus parameters: $a_0 = 1$ and $\sigma_0 = 1\,\text{ms}$. Figure taken from Petrovici et al. (2014).

(Section 3.3.2), which can lead to spike time jitter and in rare cases to spike loss when firing is highly synchronized.

Which 64 neurons inject their spikes into a routing bus is determined by the placement of the neurons on the HICANN. Hence, in order to study the effect of spike loss and jitter, we simulated the synfire chain network in two different placement setups. First, neurons of the same synfire group were placed sequentially onto the same routing bus, and second, neurons were distributed in a round-robin manner over different routing buses, such that neurons of different groups injected their spikes into one routing bus. Hence, we expect the spiking activity on each routing bus to be more synchronous in the first case than in the second. In both setups, the utilized hardware and the number of neurons per routing bus was equal, allowing a fair competition between both. The separatrices for the two different placement strategies with otherwise identical parameters are almost indistinguishable (Figure 5.20D).

Nevertheless, the raster plots (Figure 5.20E, F) reveal the effect of the introduced jitter. For sequential placement, the spread of spike times within a group is roughly double than for round-robin placement and also the onset of the volley in the last group comes 1.5 ms later. In contrast to the reference simulation, the fixed point of succesful propagation is not (0.12 ms, 1) but (0.21 ms, 1) for round-robin and (0.36 ms, 1) for sequential placement.

We conclude that, especially for dense pulses, the subsequent processing of spikes in the hardware leads to a temporal spread of the pulse volley, which however has only very little influence on the filter properties of the synfire chain.

5.4.10. Emulation on the Spikey Chip

The original network model could not be mapped directly to the Spikey chip because it requires 125 neurons per group, while on the chip only 192 neuron circuits are available. Further constraints were caused by the fixed synaptic delays, which are determined by the speed of signal propagation on the chip. The magnitude of the delay is approximately 1 ms in biological time. By simple modifications of the network, we were able to qualitatively reproduce two target behavioral properties:

- stable signal propagation up to (and including) the last population of an initial $(\sigma, a) = (0\,\text{ms}, 1)$ pulse and
- the existence of a separatrix in the (σ, a) state space surrounding a similar region of successful propagation as the original one (see Figure 5.14).

Two different network configurations were used, each adjusted to the requirements of one of the two benchmarks. In the following, we describe these differences, as well as the results for each benchmark.

synfire loop

To demonstrate a stable propagation of pulses, a large number of consecutive group activations was needed. The chain was configured as a loop by connecting the last group to the first, allowing the observation of more pulse packet propagations than there are groups in the network.

The time between two passes of the pulse packet at the same synfire group needed to be maximized to allow the neurons to recover (see voltage trace in Figure 5.21B). This was accomplished by increasing the group count and consequently reducing the group size. As

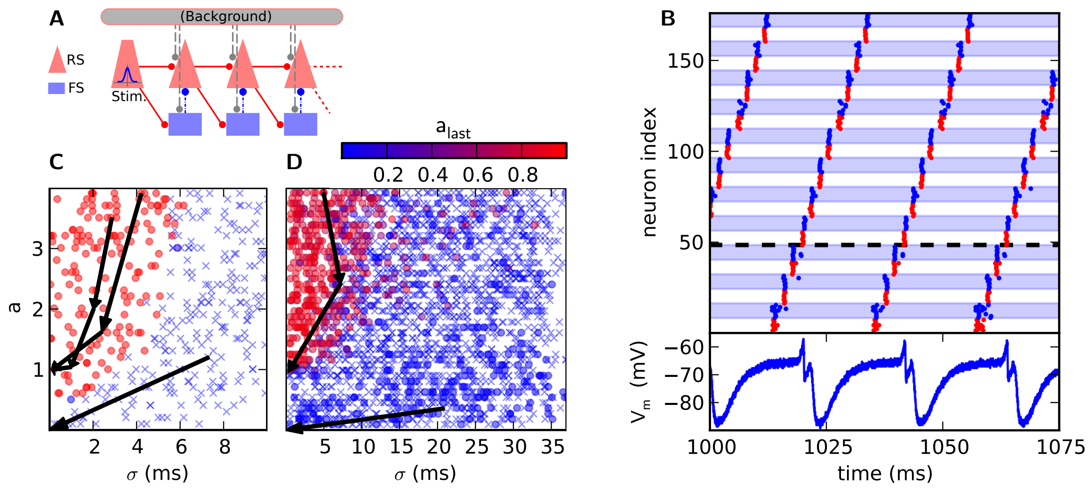


Figure 5.21.: **(A)** Synfire chain with feedforward inhibition. The background is only utilized in the original model, where it is implemented as random Gaussian current. For the presented hardware implementation it has been omitted due to network size constraints. As compensation for missing background stimuli, the resting potential was increased to ensure a comparable excitability of the neurons. **(B)** Hardware emulation. Top: Raster plot of pulse packet propagation 1000 ms after initial stimulus. Spikes from RS groups are shown in red and spikes from FS groups are denoted by blue color and background. Bottom: Membrane potential of the first neuron in the fourth RS group, which is denoted by a dashed horizontal line. The cycle duration is approximately 20 ms. **(C)** State space generated with software simulations of the original model. The position of each marker indicates the (σ, a) parameters of the stimulus while the color encodes the activity in the RS population of the third synfire group. Lack of activity is indicated with a cross. The evolution of the pulse packet parameters is shown for three selected cases by a series of arrows. Activity either stably propagates with fixed point $(\sigma, a) = (0.1\,\text{ms}, 1)$ or extinguishes with fixed point $(\sigma, a) = (0\,\text{ms}, 0)$. **(D)** Same as (C), but emulated on the FACETS chip-based system. The activity in the last group is located either near $(\sigma, a) = (0\,\text{ms}, 0)$ or $(0.3\,\text{ms}, 1)$. Figure taken from Pfeil et al. (2013).

too small populations cause an unreliable signal propagation, mainly due to variability in the neuron behavior, $n_{\mathrm{RS}} = n_{\mathrm{FS}} = 8$ was chosen as a satisfactory trade-off between propagation stability and group size. Likewise, the proportion of FS neurons in a group was increased to maintain a reliable inhibition.

To further improve propagation properties, the membrane time constant was lowered for all neurons by raising g_{l} to its maximum value. The strength of inhibition was increased by setting the inhibitory synaptic weight to its maximum value and lowering the inhibitory reversal potential to its minimum value. Finally, the synaptic weights $\mathrm{RS}_i \to \mathrm{RS}_{i+1}$ and $\mathrm{RS}_i \to \mathrm{FS}_{i+1}$ were adjusted.

With these improvements, we were able to observe persisting synfire propagation on the oscilloscope even after 2 h wall-clock time after a single initial stimulation. This corresponds to more than 2 years in biological real-time, which is a nice demonstration of the possibility to run extremely long-lasting experiments in very short time due to the high acceleration factor of the hardware system.

long experiments

The second network configuration demonstrates the filtering properties of a hardware-emulated synfire chain with feedforward inhibition. This use case required larger synfire groups than in the first case, since the total excitatory conductance caused by a pulse packet with large σ would otherwise not be smooth enough due to the low number of spikes. Thus, three groups were placed on a single chip with $n_{\mathrm{RS}} = 45$ and $n_{\mathrm{FS}} = 18$. The resulting evolution of pulse packets is shown in Figure 5.21D. After passing through three groups, pulse packet emitted by the last group was either very weak or it was located near the point $(0.3\,\mathrm{ms}, 1)$, as illustrated in Figure 5.21D.

These emulations on hardware differ from our earlier software simulations in two important points. Firstly, the separation in the parameter space of the initial stimulus is not as sharply bounded, which is demonstrated by the fact that occasionally, significant activity in the last group can be evoked by stimuli with large σ and large a, as seen in Figure 5.21D. This is a combined effect due to the reduced population sizes and the fixed pattern noise in the neuronal and synaptic circuits. Secondly, a stimulus with a small a can evoke weak activity in the last group, which is attributed to a differing balance between excitation and inhibition. In hardware, a weak stimulus causes both the RS and FS populations to respond weakly, which leads to a weak inhibition of the RS population, allowing the pulse to reach the last synfire group. Hence, the pulse fades slowly instead of being extinguished completely. In the original model, the FS population is more responsive and prevents the propagation more efficiently.

Nevertheless, the filtering properties of the in-silico network are apparent. As discussed above and shown in Section 5.4.8, the quality of the filter is clearly improved when using larger group sizes, which is only possible on an appropriately large neuromorphic device. Because the synfire chain model itself does not require sustained external stimulus, it could then be employed as an autonomous source of periodic input to other experiments.

5.5. Self-Sustained Asynchronous Irregular Activity

AI firing

Our third benchmark is a cortically inspired network with random, distance-dependent connectivity which displays self-sustained asynchronous and irregular firing (short: “AI network”). We define functionality measures on several levels of abstraction, starting from single network observables such as the network firing rate, the correlation coefficient and the coefficient of variation, the properties of the power spectrum of the network activity, up to global behavior such as the dependence of network dynamics on the internal synaptic weights $g_\mathrm{i}^\mathrm{syn}$ and $g_\mathrm{e}^\mathrm{syn}$. We then test two compensation strategies based on a mean field approach and on iterative modification of individual neuron parameters. While the first method offers a way to control the mean firing rate in the presence of synapse loss, the second is applicable to synapse loss and fixed-pattern weight noise simultaneously, in contrast to the other presented compensation methods. Non-configurable axonal delays do not significantly affect the network functionality because the intrinsic hardware delay is approximately equal to the delay utilized in the model. A scaling method for the network size is introduced and the effectivity of the second compensation method is demonstrated using the ESS on a large network with mapping-induced synapse loss and imposed fixed-pattern synaptic weight noise.

5.5.1. Architecture

dynamical attractor self-sustained firing

Under appropriate parametrization, the self-sustained regime can constitute an attractor of a dynamical system (Amit and Brunel, 1997). In spiking neural networks, self-sustained states are known to be exquisitely sensitive to the correlation dynamics generated by recurrent activity (El Boustani and Destexhe, 2009; Kumar et al., 2008). Because of this sensitivity, a model of self-sustained activity within the asynchronous-irregular regime can provide a strong comparison between hardware and software platforms, by requiring the hardware network to reproduce the low firing, weakly correlated, and highly irregular dynamics of this state. Notably, it is often observed that this activity regime provides a good match to the dynamics observed experimentally in the awake, activated cortex (Brunel, 2000; Destexhe and Pare, 1999; Destexhe et al., 2003). Additionally, the self-sustained activity regime provides an interesting use-case for the BrainScaleS hardware system, as in this state, the model network is not driven by external Poisson input, but has dynamics dominated by internally generated noise (Destexhe and Contreras, 2006), beyond the initial brief Poisson stimulation to a small percentage of the network. Networks based on this principle have also already been implemented in neuromorphic VLSI hardware (Giulioni et al., 2012).

PYR and INH cells

Here, we use a reduced model based on Destexhe (2009). Neurons in the network follow the AdEx equations 3.3 to 3.5. Their parameters are chosen as in Muller and Destexhe (2012), in order to model regular spiking pyramidal cells (PYR) with spike frequency adaptation (Connors and Gutnick, 1990) and fast spiking inhibitory cells (INH) with relatively little spike frequency adaptation. Instead of explicitly modeling the thalamocortical or corticocortical networks, as in the previous work, we have chosen to modify the model, simplifying it to a single two-dimensional toroidal sheet and adding local connections and conduction delays. The addition of local connectivity follows the experimental observation that horizontal connections in neocortex project, for the most part, to their immediate surroundings (Hellwig, 2000), while the choice of linear conduction delays reflects elec-

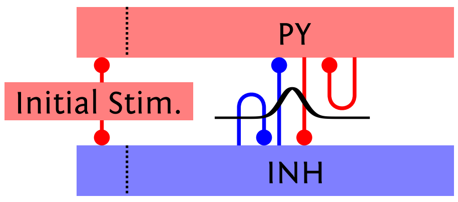


Figure 5.22.: Schematic of the connectivity of the random cortical network. Excitatory PY and inhibitory INH neurons are connected randomly with a spatial, Gaussian connection probability profile. The connection properties are given in Section A.2.5. A small part of the network is stimulated in the beginning of the experiment. Figure taken from Petrovici et al. (2014).

trophysiological estimates of conduction velocity in these unmyelinated horizontal fibers, in the range of 0.1 to $0.5\,\mathrm{m\,s^{-1}}$ (Bringuier et al., 1999; González-Burgos et al., 2000; Hirsch and Gilbert, 1991; Murakoshi et al., 1993; Telfeian and Connors, 2003). Propagation delays are known to add richness to the spatiotemporal dynamics of neural network models (Roxin et al., 2005) and in this case are observed to expand the region in the 2D space spanned by the excitatory and inhibitory conductances that supports self-sustained activity, albeit only slightly.

The default model consists of 3920 neurons (80 % pyramidal and 20 % inhibitory) equally distributed on a two-dimensional lattice of $1 \times 1\ \mathrm{mm}^2$ folded to a torus. The connection probability is distance-dependent and is normalized such that each neuron receives synaptic input from 200 excitatory and 50 inhibitory neurons. All simulations run for 10 s. 2 % of all neurons in the network are initially stimulated by one individual Poisson source for 100 ms in order to induce initial network activity. The default size was chosen such that the model can be fully realized on the wafer-scale hardware without losing any synaptic connections in the mapping step (Section 3.3.3), thereby allowing us to compare topologically equivalent software simulations, with the only remaining difference lying in the non-configurable delays and dynamic constraints on the ESS.

Figure 5.22 shows a schematic of the AI network with its distance-dependent connectivity. A small part of the neurons is stimulated at the beginning of the experiment. Depending on its parameters, the network is then able to sustain asynchronous irregular firing activity. The details about the architecture and the parameters used are given in Section A.2.5.

5.5.2. Network Scaling

For this network, scaling is quite straightforward due to its overall homogeneity. When the network is scaled up in size, we only increase the number of neurons while keeping the number of afferent synapses per neuron constant. All other parameters concerning the connectivity do not change, including the size of the cortical sheet, the distance-dependent delays and connection probability, as well as the ratio of excitatory to inhibitory cells. Neuron and synapse parameters remain unaltered.

5.5.3. Functionality Criteria

The global functionality criterion for this network consists of the ability to sustain activity in an asynchronous and irregular activity regime. The activity is considered self-sustained upon persistence to the end of the chosen simulation period. The survival time is defined as the last spike time in the network.

The mean firing rate of all PYR neurons is used to classify the overall activity of the network. The variance of the firing rates across the PYR neurons measures the homogeneity of their response. For a better comparison, we look at the relative variance, i.e., the coefficient of variation of the firing rates $\mathrm{CV}_{\mathrm{rate}} = \frac{\sigma(\nu)}{\bar{\nu}}$, where $\bar{\nu}$ and $\sigma(\nu)$ are the mean and standard deviation of the average firing rates ν of the inidividual neurons. $\mathrm{CV}_{\mathrm{rate}}$

The coefficient of variation of interspike intervals ($\mathrm{CV}_{\mathrm{ISI}}$) serves as an indicator of spiking regularity. It is calculated as

$\mathrm{CV}_{\mathrm{ISI}}$

$$\mathrm{CV}_{\mathrm{ISI}} = \frac{1}{N}\sum_{i=1}^{N} \frac{\sigma_i(\mathrm{ISI})}{\overline{\mathrm{ISI}}_i} \quad , \tag{5.11}$$

where $\sigma_i(\mathrm{ISI})$ is the standard deviation of interspike intervals in the i-th spike train, while $\overline{\mathrm{ISI}}_i$ is the mean interspike interval in the same spike train. N is the number PYR cells in the simulation. $\mathrm{CV}_{\mathrm{ISI}} = 0$ for a regular spike train and approaches 1 for a sufficiently long Poisson spike train.

The correlation coefficient (CC) is defined as

CC

$$\mathrm{CC} = \frac{1}{P}\sum_{j,k}^{P} \frac{\mathrm{Cov}(S_j, S_k)}{\sigma(S_j)\sigma(S_k)} \quad . \tag{5.12}$$

The sum runs over $P = 5000$ randomly chosen pairs of spike trains (j, k) from the excitatory population. S_i is the time-binned spike count in the i-th spike train with a bin width of $\Delta = 5\,\mathrm{ms}$. $\sigma(S_i)$ denotes the standard deviation of S_i, and $\mathrm{Cov}(S_j, S_k)$ the covariance of S_j and S_k. CC approaches 0 for sufficiently long independent spike trains and is 1 for linearly dependent (S_j, S_k). The simulation results were cross-checked with a bin width of $\Delta = 2\,\mathrm{ms}$.

The power spectrum $S(\omega)$ of a spike train is calculated as

$S(\omega_k)$

$$S(\omega_k) = |A_k|^2 N\Delta \quad , \tag{5.13}$$

where

$$\omega_k = \frac{2\pi k}{N\Delta} \quad \text{and} \tag{5.14}$$

$$A_k = \sum_{m=0}^{N-1} r_m \exp\left(-2\pi i \frac{mk}{N}\right) \qquad k = 0, \dots, N-1 \quad . \tag{5.15}$$

Here, we use the time-binned population firing rate r_i with $i \in \{0, ..., N-1\}$ with a bin width of Δ for a spike train of length $N\Delta$ (see, e.g., Rieke et al., 1997). For the AI network, we used a bin width of $\Delta = 1\,\mathrm{ms}$ for calculating the raw power spectra, and a $\sigma = 5\,\mathrm{Hz}$ for the Gauss-filtered versions which where then used to determine the peak frequency (i.e., the first non-zero peak in the power spectrum). In case of the L2/3

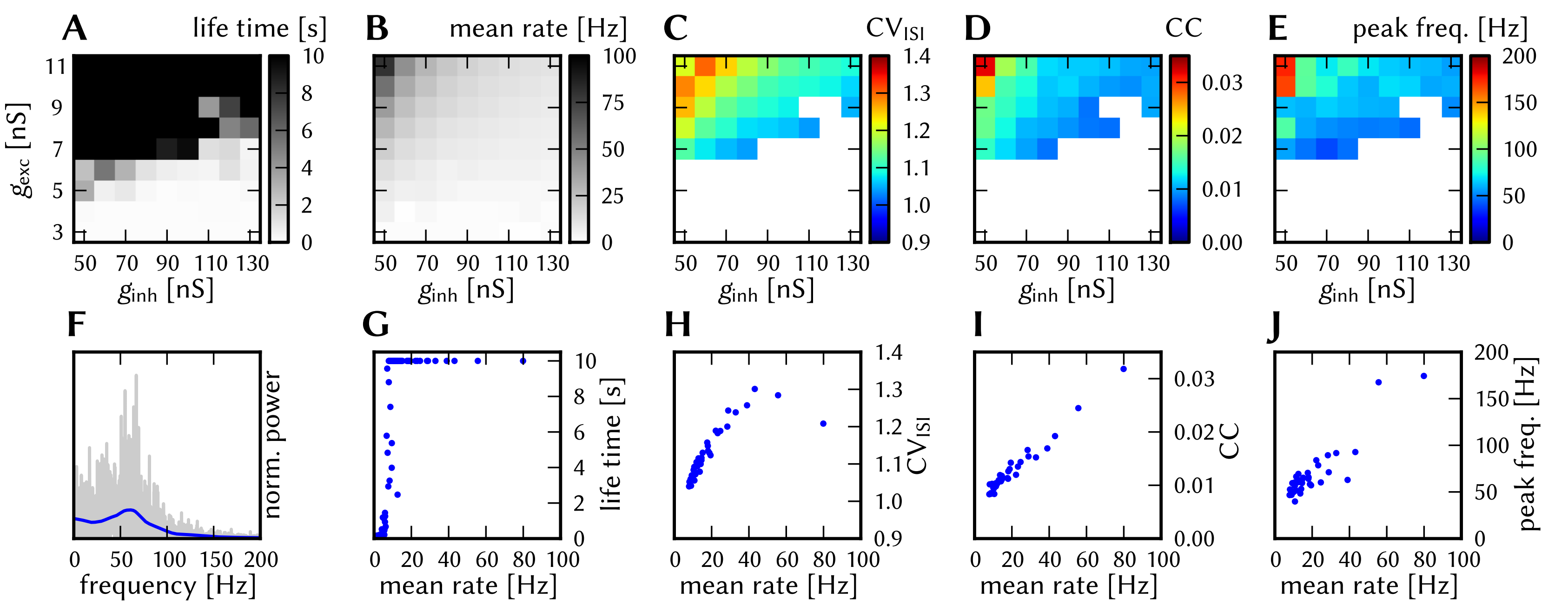


Figure 5.23.: Behavior of the undistorted AI network. Survival time **(A)**, mean firing rate **(B)**, coefficient of variance CV_{ISI} **(C)**, coefficient of correlation CC **(D)** and position of peak in power spectrum of global activity **(E)** in the parameter space for g_e^{syn} and g_i^{syn} for the default network with 3920 neurons without any distortions. **(F)** Power spectrum of the global pyramidal activity for the state ($g_e^{\text{syn}} = 9\text{nS}$, $g_i^{\text{syn}} = 90\text{nS}$). The population activity was binned with a time of 1ms, the raw spectrum is shown in gray, the blue curve shows a Gauss-filtered ($\sigma = 5\,\text{Hz}$) version for better visualization. The position of the peak in the filtered version was used for **E**. In **(G)** - **(J)**, the dependence of single criteria on the mean firing rate is shown: survival time (**G**), CV_{ISI} (**H**), CC (**I**), position of peak in power spectrum (**J**). In the last three plots, only surviving states of the (g_e^{syn}, g_i^{syn}) space were considered. Figure taken from Petrovici et al. (2014).

model, the power spectra were calculated from Gauss-filtered ($\sigma = 5\,\mathrm{ms}$) spike data with a bin width of $\Delta = 0.1\,\mathrm{ms}$ and (unless otherwise stated) smoothed with a $\sigma = 0.3\,\mathrm{ms}$ Gauss-filter.

These criteria were evaluated for a range of excitatory and inhibitory synaptic weights $g_\mathrm{e}^\mathrm{syn}$ and $g_\mathrm{i}^\mathrm{syn}$ for the default network consisting of 3920 neurons. Figure 5.23A shows the region in the ($g_\mathrm{e}^\mathrm{syn}$, $g_\mathrm{i}^\mathrm{syn}$) parameter space that allows self-sustained activity, which is achieved at PYR firing rates above 8 Hz (G).

$\mathrm{CV}_\mathrm{rate} < 0.2$

The coefficient of variation of the firing rates across neurons ($\mathrm{CV}_\mathrm{rate}$) is small ($< 0.2$, see the 0 % weight noise data in Figure 5.26B), as all neurons have identical numbers of afferent synapses with identical weights in each network realization. In addition to the parameter space plots in the top row of Figure 5.23, we plot the other criteria against the mean firing rate in the bottom row and recognize the latter as the principal property of each state that mostly determines all other criteria.

$\mathrm{CV}_\mathrm{ISI} > 1$

The activity is irregular ($\mathrm{CV}_\mathrm{ISI} > 1$) across all states (C) and is mainly determined by the network firing rate: the CV_ISI first increases with the firing rate, then saturates and decreases for rates higher than 50 Hz (H).

$\mathrm{CC} < 0.03$

Over the entire parameter space, the spike trains of the pyramidal cells are only weakly correlated, with a CC between 0.01 and 0.03. The average CC increases with the firing rate, which can be attributed to local areas in which neurons synchronize over short time periods.

$\arg\max_{\omega\neq 0}(S) \propto \bar{\nu}$

At last, we look at the power spectrum of the global pyramidal activity, exemplarily for the (9 nS, 90 nS) state in (F). As a comparison for further studies we follow Brunel (2000) and use the position of the non-zero peak in the power-spectrum $\arg\max_{\omega\neq 0}(S)$, which is shown for each ($g_\mathrm{e}^\mathrm{syn}$, $g_\mathrm{i}^\mathrm{syn}$) point (E) and as a function of the firing rate (J). The position of the power spectrum peak frequency increases linearly with the mean firing rate.

5.5.4. Non-Configurable Axonal Delays

For the analysis of the effects of non-configurable delays, we repeated the ($g_\mathrm{e}^\mathrm{syn}$, $g_\mathrm{i}^\mathrm{syn}$) sweep with all axonal delays set to 1.5 ms (see Section 5.1). The lack of distance-dependent delays did not affect any of the functionality criteria, as each neuron still received synaptic input comparable to the reference case.

mean delay

One might expect an influence on the power spectrum of global activity as fixed delays change the temporal correlation of the effect of a neuron on all of its postsynaptic partners. However, the power spectra did not change significantly (Figure 5.25A, B, C), which can be partly explained when considering the average delays in the network. In the reference case, the average of all distance-dependent delays amounts to 1.55 ms (see Figure 5.24), which is close to the constant delay value of 1.5 ms we use to model the non-configurable delays on the hardware. In this particular case, the hardware delay matches the average delay in the network such that no distortion is introduced. Accordingly, parameter space sweeps on the ESS yielded the same results.

To further investigate the influence of synaptic delays, we ran additional simulations where all delays in the network were set to 0.1 ms (D) and 3 ms (E). Lowering delays increases the speed of activity propagation such that the position of the peak in the power spectrum is shifted towards higher frequencies. For higher delays, the peak frequency decreases as expected, but also the region of sustained activity diminishes significantly.

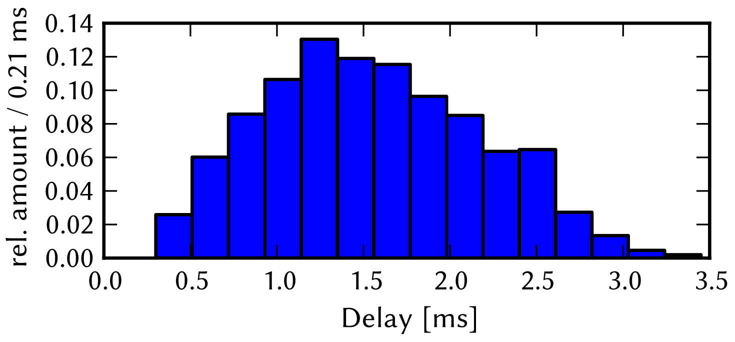


Figure 5.24.: Histogram of delays in the AI network. The mean delay is 1.55 ms. Figure taken from Petrovici et al. (2014).

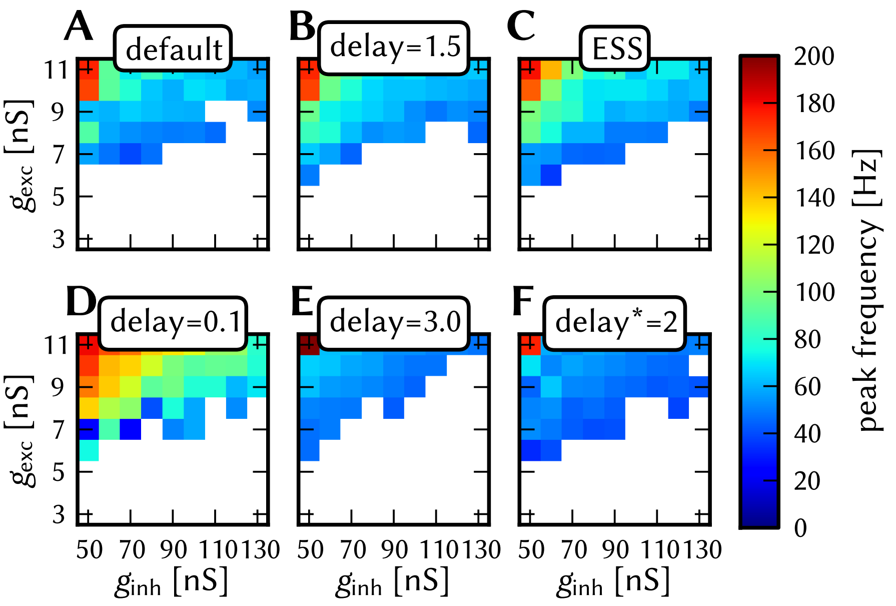


Figure 5.25.: Effects of axonal delays on the AI network. (g_e^{syn}, g_i^{syn}) spaces with the peak frequency of the global PYR activity for different axonal delay setups: **(A)** default with distance-dependent delays; **(B)** delay of 1.5 ms; **(C)** on on the ESS, but without synapse loss or synaptic weight noise; **(D)** constant delay of 0.1 ms; **(E)** constant delay of 3.0 ms; **(F)** distance-dependent delays scaled by factor of 2 with respect to default setup. Figure taken from Petrovici et al. (2014).

For comparison, we also performed simulations with distance-dependent delays scaled by a factor of 2 with respect to the baseline model, thus having an average delay of 3.1 ms (F). While the peak frequency is in good agreement with the 3 ms simulations, the region of sustained states is extended and even larger than in the baseline setup. Our simulations thereby confirm that distance-dependent delays expand the region of self-sustained states in the ($g_\mathrm{e}^\mathrm{syn}$, $g_\mathrm{i}^\mathrm{syn}$) space, as we have already mentioned in Section 5.5.1.

We note that for variants of this benchmark where the average network delay is higher or lower than 1.5 ms, there exists a straightforward but effective compensation strategy: the emulation speedup of the emulation on the hardware can simply be changed such that the average network delay is directly mapped onto the hardware delay (see also Section 5.5.7.1). Assume, for example, a modified experiment where the average delay amounts to 3 ms. By choosing a speedup of 20 000, this delay can be directly mapped to the 150 ns average delay on the hardware.

However, it should be made clear that such a change of emulation speed is not completely trivial, as one has to make sure that the neural dynamics can still be emulated at the chosen speed (i.e, the parameters still lie within the supported ranges, see Table 3.3). Furthermore, the reduced bandwidth for the pulse communication, especially for external stimulation, must be considered. While this may not raise immediate problems for this particular self-sustained network model, these conditions must be also fulfilled for potential other networks that are connected to the AI network.

5.5.5. Synaptic Weight Noise

The effects of synaptic weight noise between 10 % and 50 % on the AI network are shown in Figure 5.26. The region of self-sustained states in the ($g_\mathrm{e}^\mathrm{syn}$, $g_\mathrm{i}^\mathrm{syn}$) space is increased by this distortion mechanism, as marked by the circles in (C) representing states that survived with 50 % synaptic weight noise but not in the undistorted case. The firing rate increases with the amplitude of the weight noise (A). In general, the change becomes larger for lower $g_\mathrm{e}^\mathrm{syn}$, but it is less pronounced for states with an already high firing rate in the undistorted case (C). Synaptic weight noise leads to an increase of the variation of firing rates ($\mathrm{CV}_\mathrm{rate}$), with the change being stronger for high population firing rates (B). The CV_ISI as a function of firing rates remains unchanged for low rates, but decreases for higher firing rates in proportion to the noise level (E). Furthermore, weight noise introduces randomness into the network, thereby reducing synchrony: the pairwise correlation between neurons decreases linearly with the amount of weight noise (F). The power spectrum of the global activity is not affected by this distortion mechanism.

5.5.6. Synapse Loss

Synapse loss has a similar influence on the network behavior as synaptic weight noise. Figure 5.27 shows the results of the $g_\mathrm{e}^\mathrm{syn}$-$g_\mathrm{i}^\mathrm{syn}$ sweeps for synapse loss values between 10 % and 50%. The region of sustained states increases with synapse loss but not as strongly as for weight noise (C). The firing rate increases with synapse loss (A). Compared to the change caused by synaptic weight noise, however, the effect is much stronger here. The same holds for the variance of the firing rates across the PYR neurons, which again increases with synapse loss, as can be seen in (B). Note that the $\mathrm{CV}_\mathrm{rate}$ does not follow a

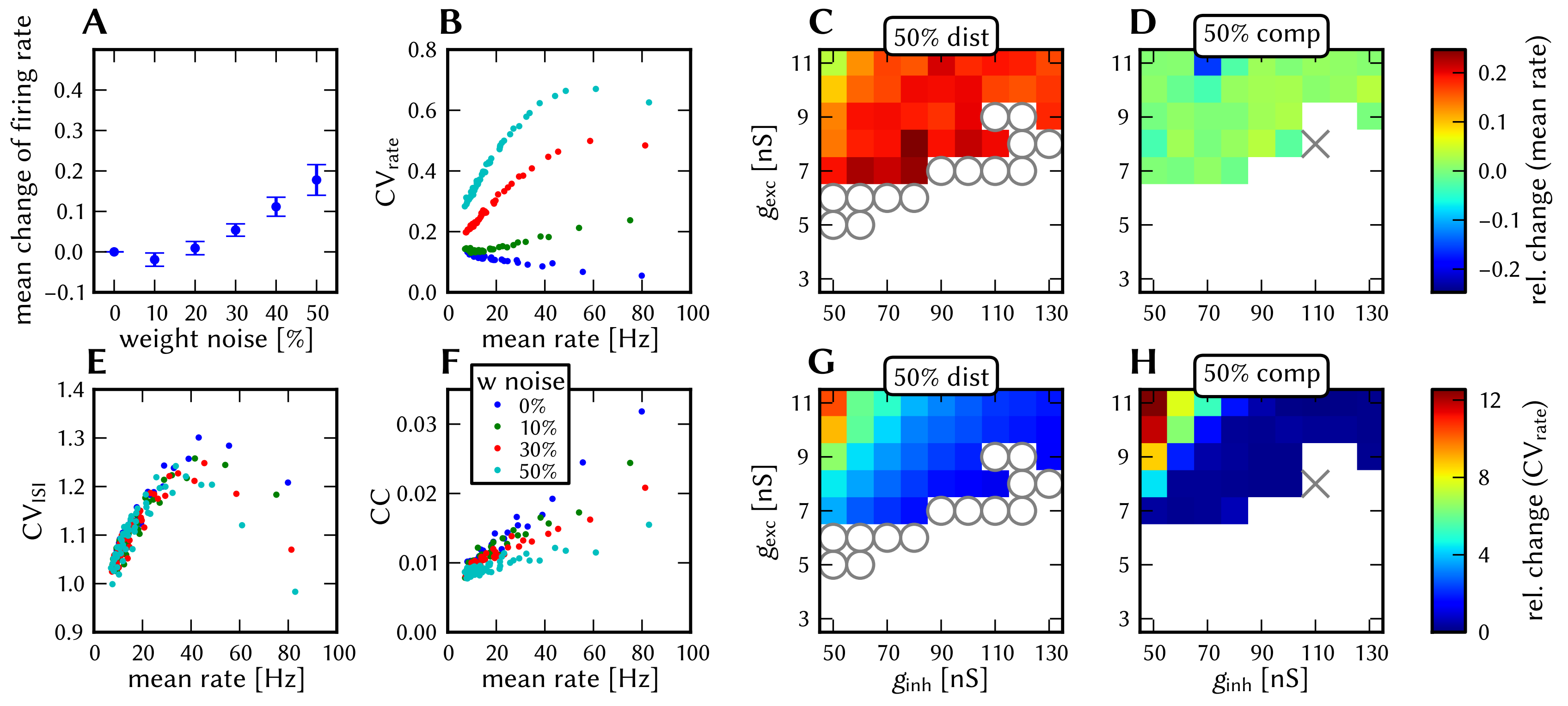


Figure 5.26.: Effect and compensation of synaptic weight noise in the AI network. **(A)** Relative change of the firing rate with respect to the undistorted network averaged over all sustained states for varying synapse weight noise. **(B)** CV_{rate} as a function of mean rate for every survived state for varying synapse weight noise. **(C)** Relative change of the firing rate with respect to the undistorted for each state for 50 % synapse weight noise. **(D)** Same as **C**, but after compensation. **(E)** CV_{ISI} as a function of mean rate for varying synaptic weight noise. **(F)** CC as a function of mean rate for varying synaptic weight noise. **(G)** Relative change of CV_{rate} with respect to the undistorted network for each state for 50 % synaptic weight noise. **(H)** Same as **G**, but after compensation. In **C**, **D**, **G** and **H**: A cross marks a state that was sustained in the undistorted but not sustained in the compared case. A circle marks a state that was not sustained in the original but sustained in the compared case. Figure taken from Petrovici et al. (2014).

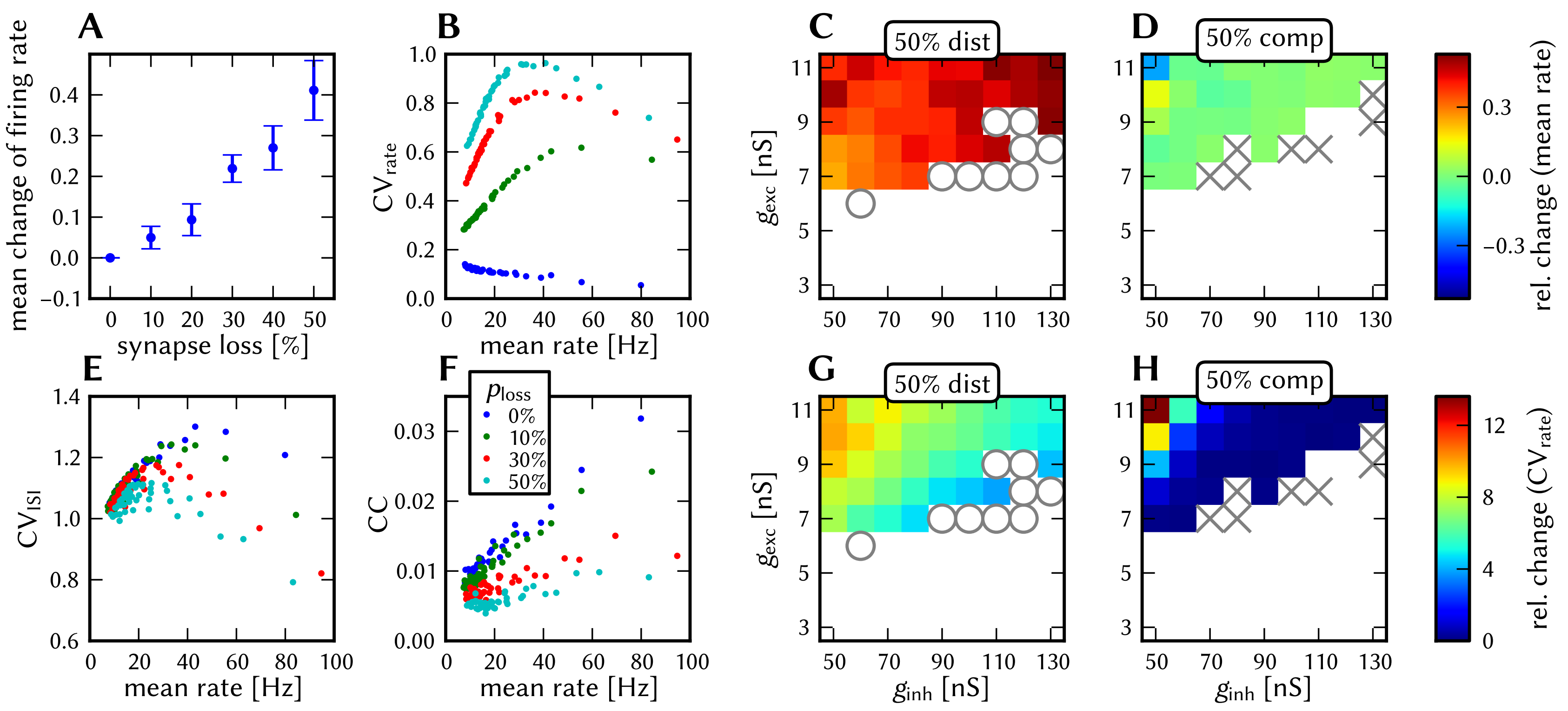


Figure 5.27.: Effect and compensation of synapse loss in the AI network. **(A)** Relative change of the firing rate with respect to the undistorted network averaged over all sustained states for varying synapse loss. **(B)** CV_{rate} as a function of mean rate for every survived state for varying synapse loss. **(C)** Relative change of the firing rate with respect to the undistorted case for each state for 50 % synapse loss. **(D)** Same as **C**, but after compensation. **(E)** CV_{ISI} as a function of mean rate for varying synapse loss. **(F)** CC as a function of mean rate for varying synapse loss. **(G)**, **(H)** Relative change of CV_{rate} with respect to the undistorted case for each state for 50 % synapse loss. **(H)** Same as **G**, but after compensation. In **C**, **D**, **G** and **H**: A cross marks a state that was sustained in the undistorted but not sustained in the compared case. A circle marks a state, that was not sustained in the original but sustained in the compared case. Figure taken from Petrovici et al. (2014).

monotonic function. We should also remark that for high synapse loss, some neurons did not fire at all. Both the irregularity and the correlation of firing decrease with increasing synapse loss, leaving the network still in an asynchronous irregular state (E, F). Synapse loss shows no effect on the power spectrum of global PYR activity.

5.5.7. Compensation Strategies

The hardware-induced distortions of the AI network behavior analyzed in the previous sections leave two major criteria that need to be recovered: the population firing rate and the variation of firing rates across the population. We consider the other effects (change of CC, $\mathrm{CV}_{\mathrm{ISI}}$, peak frequency in power spectrum) as minor because they are mainly determined by the mean rate and neglect them in the following.

One quite simple approach for recovering the original firing rate would be to change the strengths of the synaptic weights $g_{\mathrm{e}}^{\mathrm{syn}}$ and $g_{\mathrm{i}}^{\mathrm{syn}}$. Considering the conducted $(g_{\mathrm{e}}^{\mathrm{syn}}, g_{\mathrm{i}}^{\mathrm{syn}})$ parameter space sweeps, we could simply select the distorted state that best matches the criteria of the undistorted reference. However, this method requires to scan $g_{\mathrm{e}}^{\mathrm{syn}}$ and $g_{\mathrm{i}}^{\mathrm{syn}}$ over a wide range to finally get to the desired result. A compensation method that can be applied to a single experiment and works without time-consuming parameter sweeps would be preferable.

5.5.7.1. Mean Field Compensation for Rate Changes

The mean firing rate in the network rises with increasing synapse loss. This effect can be understood using a mean-field approach (see, e.g., Kumar et al., 2008) in which the response rate ν_i of a single neuron is assumed to be a function of the mean excitatory and inhibitory network firing rates:

response function

$$\nu_i = f\left(\nu_{\mathrm{in,exc}}, \nu_{\mathrm{in,inh}}\right) \quad . \tag{5.16}$$

This approach is similar to the one in Brunel (2000), where the afferent neurons are replaced by independent Poisson processes with equal instantaneous rate in a sparse random network. Under the assumption that both PYR and INH neurons fire with the same average rate, we can now use Equation 5.16 to calculate the mean firing rate in a self-sustained state as a stable, self-consistent solution of

$$\nu_i = f(\nu_i, \nu_i) \quad , \tag{5.17}$$

in which we can also include synapse loss to yield

$$\hat{\nu}(p_{\mathrm{loss}}) = f\left(N_{\mathrm{exc}}(1 - p_{\mathrm{loss}})\hat{\nu}, N_{\mathrm{inh}}(1 - p_{\mathrm{loss}})\hat{\nu}\right) \quad . \tag{5.18}$$

Here, N_{exc} and N_{inh} are the number of pre-synaptic connections of a given neuron and p_{loss} the synapse loss value.

Figure 5.28A shows the response function f of PYR and INH neurons for $p_{\mathrm{loss}} = 0$, which yields the stable solution $\tilde{\nu}(0) \approx 14\,\mathrm{Hz}$ as the intersection of the first diagonal and the gain function. Analogously, the solution for $p_{\mathrm{loss}} = 0.5$ can be determined as the intersection with the $y = 2x$ line (considering $\nu_{\mathrm{in}}(p_{\mathrm{loss}}) = p_{\mathrm{loss}} \cdot \nu_{\mathrm{in}}$). The result also justifies the assumption of the mean firing rate of inhibitory and excitatory neurons being

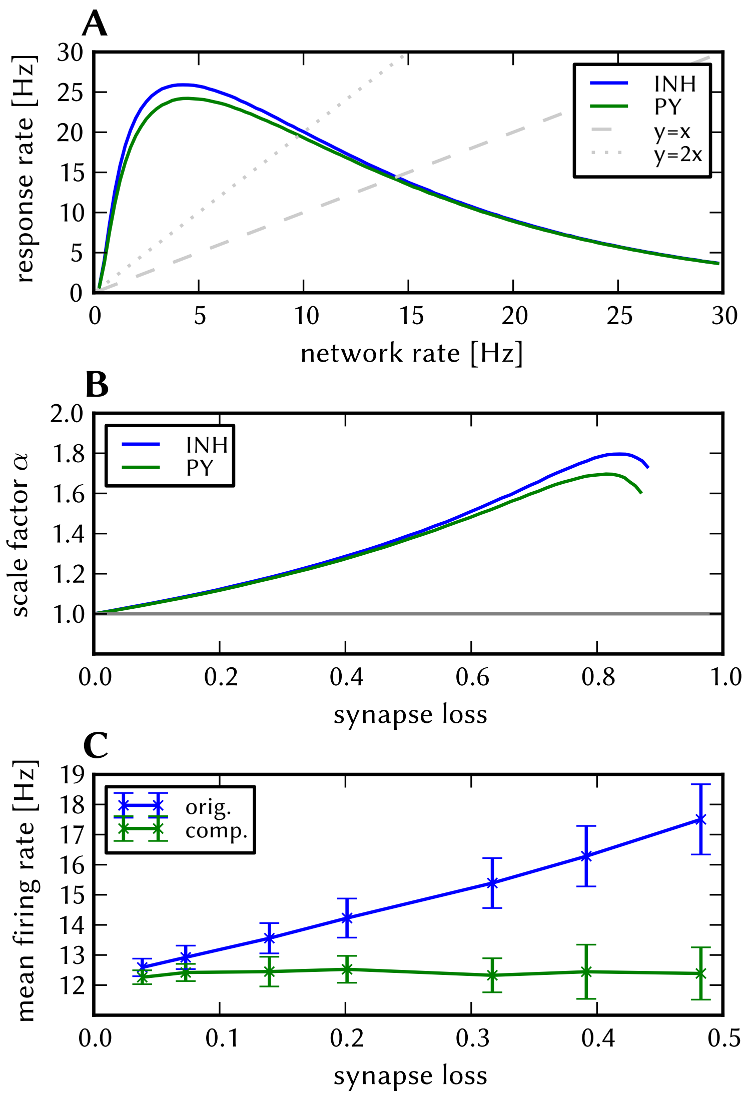


Figure 5.28.: Mean field compensation method for the AI network. **(A)** Mean firing rate of a single PYR and INH neuron given a Poisson stimulus by the external network with a given rate. **(B)** Compensation factor α calculated from the data in **A**. **(C)** Compensation applied to the self-sustained network (with parameters $g_i^{syn} = 90\,nS$, $g_e^{syn} = 9\,nS$). The error bars denote the standard deviation of mean firing rates across all neurons. The scaling of internal delays has negligible effects on the firing rate (not shown). Figure taken from Petrovici et al. (2014).

equal for $p_{\text{loss}} < 0.5$.

We now need to calculate the parameter change that is necessary to restore the original mean firing rate. The idea is to speed up or slow down the dynamics of the entire network, thereby modulating the firing rate as well. Let us denote by $\boldsymbol{x}$ the state of all dynamic variables within a network:

$$\dot{\boldsymbol{x}}(t) = \boldsymbol{F}(\boldsymbol{x}, t) \quad . \tag{5.19}$$

We can now define $\boldsymbol{y}$ to describe a network which functionally follows the same time dependence, but where time itself is scaled by a factor α:

$$\boldsymbol{y}(t) := \boldsymbol{x}(\alpha t) \quad \Longrightarrow \quad \dot{\boldsymbol{y}}(t) = \alpha \dot{\boldsymbol{x}}(\alpha t) = \alpha \boldsymbol{F}(\boldsymbol{y}(t), \alpha t) \quad . \tag{5.20}$$

Therefore, by defining

$$\tilde{\boldsymbol{F}}(\boldsymbol{x}, t) := \alpha \boldsymbol{F}(\boldsymbol{x}, \alpha t) \tag{5.21}$$

we obtain accelerated dynamics for $\boldsymbol{y}$ in real time that are identical to the original dynamics of $\boldsymbol{x}$ in scaled time. *accelerated dynamics*

As the given random cortical network shows self-sustained behavior, the transition from $\boldsymbol{F}$ to $\tilde{\boldsymbol{F}}$ requires only the modification of internal network parameters (external input would have to also be modified otherwise). More specifically, the transition requires the scaling of all time constants: τ_{m}, τ_x^{syn}, τ_{ref}, τ_{w}, as well as the synaptic delays by α. The scaling factor α is calculated from the measured gain function as

$$\alpha = \frac{\tilde{\nu}(p_{\text{loss}})}{\tilde{\nu}(0)} \quad . \tag{5.22}$$

The effects of these parameter modifications on the output firing rates are shown in Figure 5.28C. Along with the good results of our compensation method, we should also note that the variance of the firing rates across neurons grows as the synapse loss rises. This happens due to the probabilistic synapse loss, which causes a spread in the neuronal fan-in. In principle, this effect could also be countered by applying the mean-field-based compensation method on a neuron-to-neuron basis, but this would require knowledge of the actual network realization (which is available only after the mapping step), as well as the measurement of the response function f for all occurring configurations of presynaptic inhibitory and excitatory neuron numbers. We shall tackle these issues as well when we discuss a different method below.

In conclusion, the mean-field method can be applied when the synapse loss of individual neurons does not have too much spread and requires the knowledge of neural response functions. More importantly, this method does not require the detailed understanding of the ensemble dynamics of the complete network. We should point out that, in general, this method requires the ability to modify synaptic delays according to the scaling rule. However, for this network, the influence of synaptic delays on the mean firing rate is negligible (see Section 5.5.4).

5.5.7.2. Iterative Compensation

The iterative compensation method aims at reducing two distortion effects: the change of the mean firing rate of the PYR neurons and its variance across neurons, which are both

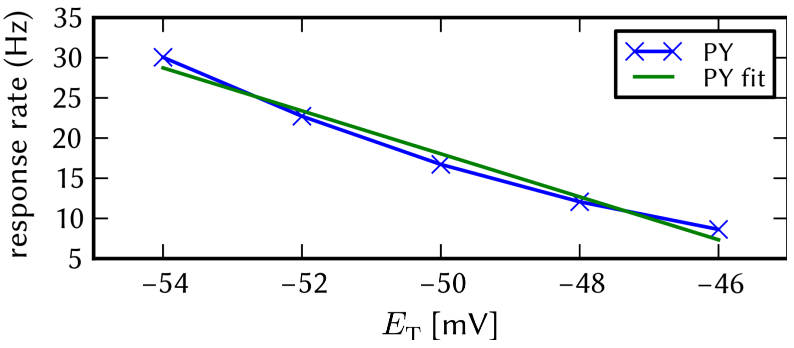


Figure 5.29.: Calculation of the compensation factor c_{comp} for the state ($g_{\mathrm{e}}^{\mathrm{syn}} = 9\,\mathrm{nS}$, $g_{\mathrm{i}}^{\mathrm{syn}} = 90\,\mathrm{nS}$) of the AI network. A PYR neuron is stimulated by 200 excitatory and 50 inhibitory Poisson sources with rate 12.38 Hz and its spike initiation threshold E_{T} is varied. The slope $m = -2.6745\frac{\mathrm{Hz}}{\mathrm{mV}}$ of a linear fit is then used to calculate the compensation factor $c_{\mathrm{comp}} = \frac{0.5}{m} = -0.18695\frac{\mathrm{mV}}{\mathrm{Hz}}$. Figure taken from Petrovici et al. (2014).

apparent for synapse loss and synaptic weight noise. It relies on the controllability of the hardware neuron parameters, since it requires the tuning of AdEx parameters for every individual neuron (see Section 3.3.1).

The iterative compensation is quite straightforward. We start with the simulation results of the reference network, as well as the distorted one. From the reference simulation, we can extract the target mean rate ν^{tgt} of the neurons in a population. For each neuron in the distorted network, we compare its actual firing rate against ν^{tgt}, and modify the excitability of the neuron in proportion to the difference between target and measured firing rate. The distorted network with modified neuron parameters is then simulated and the output is compared again to the reference network. This iterative compensation step is repeated until the characteristics of the last step approximately match those of the reference simulation.

In our simulations, we modified the spike initiation threshold E_{T}, with its change being proportional to the difference between the current and the target firing rate. In particular, we used

E_{T} *update*

$$E_{\mathrm{T}}{}^{\mathrm{n+1,i}} = E_{\mathrm{T}}{}^{\mathrm{n,i}} + (\nu^{\mathrm{tgt}} - \nu^{\mathrm{n,i}})c_{\mathrm{comp}} \quad , \tag{5.23}$$

where $E_{\mathrm{T}}{}^{\mathrm{n,i}}$ and $\nu^{\mathrm{n,i}}$ are the threshold voltage and firing rate of the ith neuron in the nth compensation step and c_{comp} is the compensation factor that links the firing response and the threshold voltage.

The target rate ν^{tgt} is computed separately for the excitatory and inhibitory population from the reference simulations. We can then choose the compensation factor for each ($g_{\mathrm{e}}^{\mathrm{syn}}, g_{\mathrm{i}}^{\mathrm{syn}}$) state as follows. Similar to the mean-field approach in Section 5.5.7.1, we consider the response rate of an excitatory neuron given a network firing rate of ν^{tgt}. The stimulus it experiences within the network is equivalent to being stimulated by 200 excitatory and 50 inhibitory Poisson sources with rate ν^{tgt}. We can then vary the threshold voltage of the neuron between −54 mV and −46 mV and thereby determine the relation between the response rate and the threshold voltage. From a linear fit of this dependence, we can extract the slope m, and set the compensation factor to $c_{\mathrm{comp}} = \frac{0.5}{m}$ (Figure 5.29). The factor of 0.5 was chosen to limit the change of the mean rate in each step in order to avoid oscillations in the compensation procedure. Additionally, whenever we changed the spike initiation voltage E_T, we also shifted the spike detection voltage V_spike equally.

With this appropriate choice of c_comp, we found that 10 iterations are sufficient to restore the mean and variance of the firing rates in the undistorted network. While the compensated mean rate exactly corresponds to ν^tgt, the compensated $\mathrm{CV}_\mathrm{rate}$ is higher than in the reference network, but reliably below the 1.2-fold of the reference value.

deterministic mapping

We remark that the proposed iterative compensation requires a controllable, deterministic mapping, which guarantees that in each iteration the neurons and synapses are always mapped onto the same hardware elements. Hence, whenever we change the random seed that is used to generate the probabilistic connectivity between the neurons, the iterative compensation needs to be run anew. However, the majority of the time required by this method is spent with the network simulation itself, which is not really an issue with a hardware speed-up factor of 10^4. In between simulations, parameter updates must be performed for each neuron individually, but this can be easily parallelized once the simulation data is available.

Synaptic Weight Noise

In order to verify the iterative compensation strategy, we applied it to the 50 % synaptic weight noise case. Note that, here and in Section 5.5.8, weight noise was implemented as being always the same in all iterations, representing the case where fixed-pattern noise, and not trial-to-trial variability, determines the synaptic weight noise.

The results of the iterative compensation are shown in Figure 5.26, which displays the relative difference of the mean and variance of the firing rates with respect to the reference simulation in (D) and (H). The region of sustained activity in the ($g_\mathrm{e}^\mathrm{syn}$, $g_\mathrm{i}^\mathrm{syn}$) parameter space of the compensated network matches the one of the reference simulation very well. The mean and variance of firing rates could be successfully recovered for most of the states. The only exception are excitation-dominated states with a mean firing rate above 25 Hz, where both criteria still differ notably from the reference after 10 iterations (upper left regions in the ($g_\mathrm{e}^\mathrm{syn}$, $g_\mathrm{i}^\mathrm{syn}$) spaces). For such extreme cases, the performance of the iterative compensation might be further improved by a specific tuning of the compensation factor c_comp for high firing rates. The other criteria such as CV_ISI and peak frequency could be fully recovered, following the assumption made earlier, that those criteria mainly depend on the firing rate. However, the coefficient of pairwise cross-correlation (CC) of the compensated networks is lower than in the reference simulation, showing that some of the randomness introduced by the synaptic weight remains even after compensation.

Synapse Loss

The effects of the iterative compensation strategy for 50 % synapse loss are shown in Figure 5.27, which displays the relative difference of the mean and variance of the firing rates with respect to the reference simulation in (D) and (H). While the restoration of the mean and variance of firing rates was successful, the compensation also led to a reduction in the self-sustained area of the ($g_\mathrm{e}^\mathrm{syn}$, $g_\mathrm{i}^\mathrm{syn}$) space. Again, for high firing rates, the iterative compensation performed only moderately well (upper left regions in the ($g_\mathrm{e}^\mathrm{syn}$, $g_\mathrm{i}^\mathrm{syn}$) spaces). The other functionality criteria show the same behavior as for the weight noise compensation, i.e., the peak frequency and $\mathrm{CV_{ISI}}$ are in good match with the reference values, while the pairwise correlation (CC) decreased due to the randomness introduced by the synapse loss. We repeated the iterative compensation for the ($g_\mathrm{e}^\mathrm{syn}$, $g_\mathrm{i}^\mathrm{syn}$) space with 30 % synapse loss. The results (not shown) were comparable to the 50 % case, but exhibited fewer unstable states, i.e., there were more combinations of $g_\mathrm{e}^\mathrm{syn}$ and $g_\mathrm{i}^\mathrm{syn}$ where the network activity was self-sustained.

5.5.8. Full simulation of Combined Distortion Mechanisms

In a final step, the iterative compensation method designed for the AI network was tested in ESS simulations. In order to observe synapse loss, we scaled the network up to a relatively large size (compared to the default model). This large-scale network was then emulated on the ESS and compared to the undistorted reference simulation with NEST. Afterwards, we applied the compensation strategy developed in the previous section to restore the original behavior of the AI network.

Mapping such homogeneous networks that lack any modularity represents the worst-case scenario for the mapping process, as they have little room for optimization. In Figure 5.30A, the relative synapse loss is plotted for various network sizes using the scaling method described in Section 5.5.2.

synapse loss

Somewhat unexpectedly, synapse loss can be seen to already occur for low numbers of neurons, although there are sufficient hardware synapses and synapse drivers. This is a consequence of the sparseness of the on-wafer routing switches, which causes some routing buses to not be able to find a free switch to connect to their respective target HICANNs.

A kink in the graph of the synapse loss can be seen at around 20 000 neurons, where at least 64 neurons are mapped onto one HICANN (see Table 3.2). As 14 336 is the maximum number of presynaptic neurons that can send their pulses to one HICANN and the connectivity in the AI network is probabilistic, the chance to find groups of 64 neurons whose pool of pre-synaptic neurons is smaller than 14 336 is close to zero.

large-scale model

In order to produce a demanding scenario, we scaled the model to a size of 22 445 neurons. The size was chosen such that the network almost occupies an entire wafer, while mapping up to 64 neurons onto one HICANN. This large-scale network has a total of approximately 5.6 million synapses. The statistics of the reference simulation can be found in Table 5.6.

projection type	synapse loss [%]
PY $\rightarrow$ PY	26.9
PY $\rightarrow$ INH	28.1
INH $\rightarrow$ PY	31.1
INH $\rightarrow$ INH	33.4
STIM $\rightarrow$ PY	77.5
STIM $\rightarrow$ INH	89.4
total	28.1

Table 5.5.: Projection-wise synapse loss of the large-scale AI network after mapping.

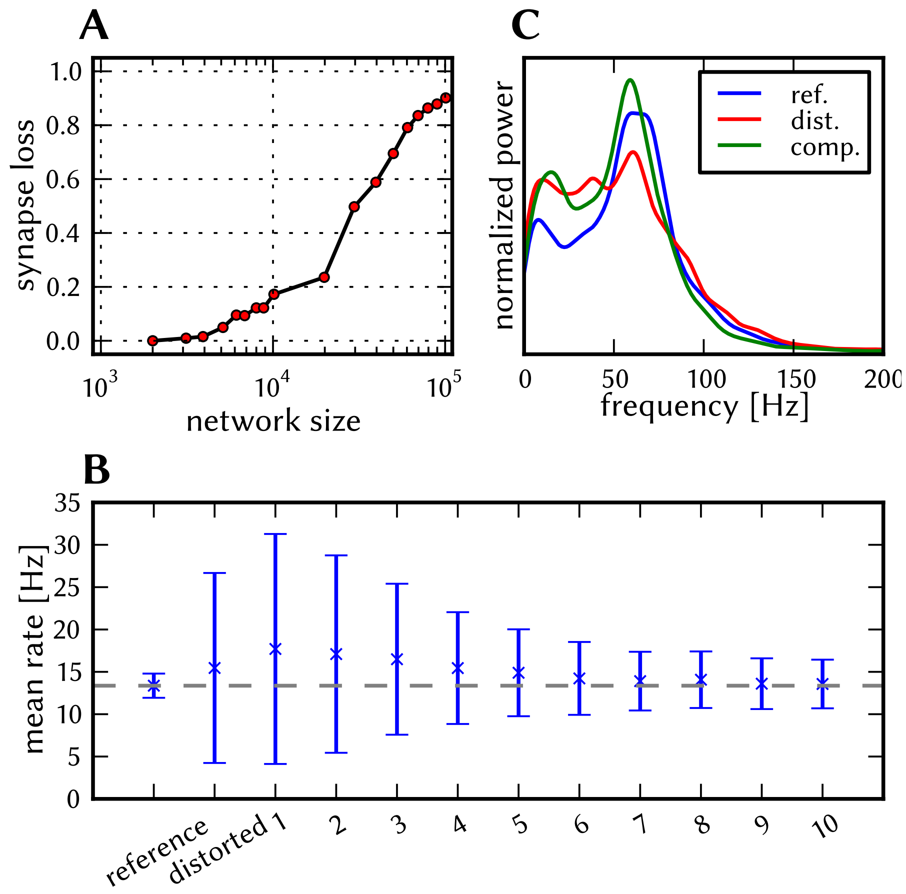


Figure 5.30.: AI network on the ESS. **(A)** Synapse loss after mapping the network with different sizes onto the BrainScaleS system. **(B)** Iterative compensation of the large-scale network with 22 445 neurons on the ESS: evolution of mean and standard deviation of firing rates for 10 iterations. **(C)** Gauss-filtered power spectrum of global activity of the pyramidal neurons in the large-scale network. Reference spectrum shown in blue (simulated with NEST), distorted and compensated spectra in red and green, both simulated with the ESS. Figure taken from Petrovici et al. (2014).

In the above scenario, 28.1 % of synapses were lost during the mapping process (for projection-wise numbers see Table 5.5). We remark that the synapse loss at this size is higher than during the synapse loss sweep in Figure 5.30A, as we used a sequence of mapping algorithms that aims for balance between synapse loss of excitatory and inhibitory connections. Additionally, we applied a fixed-pattern noise of 20 % to the synaptic weights in the ESS simulation. The result of this simulation can be found in Table 5.6: the network still survived until the end of the simulation, but the firing rate and its variance increased compared to the reference simulation, in accordance with our previous predictions from the distortion analysis.

We then used the iterative compensation method (Section 5.5.7) to compensate the abovementioned distortions and repeated the ESS simulation with the modified network. The evolution of the firing rates over 10 iterations is shown in Figure 5.30B. One can clearly see how, step by step, the firing rate approaches the target rate and that at the same time the variance of firing rates decreases. The statistics of the final iteration are listed in Table 5.6. The mean firing rate was fully recovered, while the variation of firing across neurons (CV_{rate}) was significantly reduced from 0.726 to 0.212 (but was still twice as large as in the reference network). The other functionality criteria match the reference simulation very well, as does the power spectrum of global activity in Figure 5.30C.

functionality criterion	reference (NEST)	distorted (ESS)	compensated (ESS)
Rate [Hz]	13.4	15.5	13.6
CV_{rate}	0.107	0.726	0.212
CV_{ISI}	1.12	1.11	1.09
CC	0.00103	0.0011	0.00166
Peak Frequency[Hz]	60.3	60.7	59.0

Table 5.6.: Functionality criteria of the large-scale AI network before and after compensation.

5.6. Conclusions and Outlook

In this study, we have presented a systematic comparison between neural network simulations carried out with ideal software models and a specific implementation of a neuromorphic computing system. The results for the neuromorphic system were obtained with a detailed simulation of the hardware architecture. The core concept is, essentially, a functionalist one: neural networks are defined in terms of functional measures on multiple scales, from individual neuron behavior up to network dynamics. The various neuron and synapse parameters are then tuned to achieve the target performance in terms of these measures.

The comparison was based on three cortically inspired benchmark networks: a layer 2/3 columnar architecture, a model of a synfire chain with feed-forward inhibition and a random network with self-sustained, irregular firing activity. We have chosen these specific network architectures for two reasons. First of all, they implement very different, but widely acknowledged computational paradigms and activity regimes found in neocortex: winner-take-all modules, spike-correlation-based computation, self-sustained activity and asynchronous irregular firing. Secondly, due to their diverse properties and structure, they pose an array of challenges for their hardware emulation, being affected differently by the studied hardware-specific distortion mechanisms.

All three networks were exposed to the same set of hardware constraints and a detailed comparison with the ideal software model was carried out. The agreement was quantified by looking at several chosen microscopic and and macroscopic observables on both the cell and network level, which we dubbed "functionality criteria". These criteria were chosen individually for each network and were aimed at covering all of the relevant aspects discussed in the original studies of the chosen models.

Several hardware constraint categories have been studied: the dynamics of the embedded neuron and synapse models, limited parameter ranges, synapse loss due to limited hardware resources, synaptic weight noise due to fixed-pattern and trial-to-trial variations, and the lack of configurable axonal delays. The final three effects were studied in most detail, as they are expected to affect essentially every hardware-emulated model. The investigated distortion mechanisms were studied both individually, as well as combined, similarly to the way they would occur on a real hardware substrate. As expected, above certain magnitudes of the hardware-specific distortion mechanisms, substantial deviations of the functionality criteria were observed.

For each of the three network models and for each type of distortion mechanism, several compensation strategies were discussed, with the goal of tuning the hardware implementation towards maximum agreement with the ideal software model. With the proposed compensation strategies, we have shown that it is possible to considerably reduce, and in some cases even eliminate the effects of the hardware-induced distortions. We therefore regard this study as an exemplary workflow and a toolbox for neuromorphic modelers, from which they can pick the most suitable strategy and eventually tune it towards their particular needs.

other distortion mechanisms

In addition to the investigated mechanisms, several other sources of distortions are routinely observed on neuromorphic hardware. A (certainly not exhaustive) list might include mismatch of neuron and synapse parameters, shared parameter values (i.e., not individually configurable for each neuron or synapse) or limited parameter programming resolution. These mechanisms are highly back-end-specific and therefore difficult to generalize. However, although they are likely to pose individual challenges by themselves, some of their ultimate effects on the target network functionality can be alleviated with the compensation strategies proposed here.

Our proposed strategies aim at neuromorphic implementations that compete in terms of network functionality with conventional computers but offer major potential advantages in terms of power consumption, simulation speed and fault tolerance of the used hardware components. If implemented successfully, such neuromorphic systems would serve as fast and efficient simulation engines for computational neuroscience. Their potential advantages would then more than make up for the overhead imposed by the requirement of compensation.

From this point of view, hardware-induced distortions are considered a nuisance, as they hinder precise and reproducible computation. In an alternative approach, one might consider the performance of the system itself at some computational task as the "fitness function" to be maximized. In this context, some particular architecture of an embedded model, together with an associated target behavior, would then become less relevant. Instead, one would design the network structure specifically for the neuromorphic substrate or include training algorithms that are suitable for such an inherently imperfect back-end. The use of particular, "ideal" software models as benchmarks might then given up altogether in favor of a more hardware-oriented, stand-alone approach. Here, too, the proposed compensation strategies can be actively embedded in the design of the models or their training algorithms.

hardware-oriented modeling

The hardware architecture used for our studies is, indeed, suited for both approaches. It will be an important aspect of future research with neuromorphic systems to develop procedures that tolerate or even actively embrace the temporal and spatial imperfections inherent to all electronic circuits. These questions need to be addressed by both model and hardware developers, in a common effort to determine which architectural aspects are important for the studied computational problems, both from a biological and a machine learning perspective.

biological parameter variability

On a final note, we consider it important to point out that the parameter variability observed in neuromorphic components only superficially resembles the variability observed in biological neural networks. While it is true that even for some of the simplest biological neural networks the network parameters exhibit very large animal-to-animal variability, the location of these parameter sets in the entire potential parameter space are far from being random (Marder and Taylor, 2011). The variability observed in nature is, indeed, much rather the consequence of fine-tuning through various structural plasticity and homeostatic mechanisms. The fact that many biochemical processes are partly redundant in terms of their effect on the network functionality (such as different ion channel types, see, e.g., Marder, 2011; Marder and Goaillard, 2006) offers additional evidence for the hypothesis that evolution has favored the existence of redundant "control knobs" precisely in order to allow a robust fine-tuning of the network functionality.

structural plasticity, homeostasis, redundance

It is therefore essential for future hardware generations to also offer comprehensive means of structural plasticity in order to allow a departure from the sometimes difficult tuning strategies that are currently required. Concerning the BrainScaleS waferscale devices, two important steps in this direction have already been taken: the availability of long-term plasticity (STDP, see Schemmel et al., 2007, 2008) and the implementation of a general-purpose plasticity processor (Friedmann et al., 2013). What the learning algorithms should look like that shall be implemented with the help of these additional components remains an open question and an extremely interesting venue of active research. We point to Breitwieser (2015) and Weilbach (2015) for some recently developed ideas on this topic.

plasticity processor

6. Probabilistic Inference in Neural Networks

Did the sun just explode? (It's night, so we're not sure.) A neutrino detector measures whether the sun has gone nova. Then, it rolls two dice. If they both come up six, it lies to us. Otherwise, it tells the truth.
Frequentist statistician: "Let's try. Detector! Has the sun gone nova?"
Detector: "Roll... Yes".
Frequentist statistician: "The probability of this result happening by chance is $1/36 = 0.027$*. Since* $p < 0.05$*, I conclude that the sun has exploded."*
Bayesian statistician: "Bet you $50 it hasn't."

Adapted from Randall Munroe, XKCD 1132

bottom-up modeling

Throughout the previous chapter, we have discussed several models that have been specifically inspired by the architecture of the cortex. These models have been constructed in a bottom-up fashion: given a raw connectivity map, as suggested by electrophysiological and neuroanatomic experimental results, one superimposes a finer architectural granularity in order to enable a desired spectrum of dynamics and/or functionality.

deterministic models

Let us, for now, leave aside the AI-states model, which is designed as a "cortical template" of sorts and therefore focuses less on specific functionality. If one looks at how the other two models react to specific stimuli, one notices a predominantly deterministic behavior. Attractors in the L2/3 model, for instance, have a very sharp transition from silent to active as a function of stimulus strength (see, e.g., Figure 5.3). A stimulus presented to the synfire chain is either propagated or not, depending on its location in state space (see, e.g., Figure 5.14). Obviously, for the tasks these networks are thought to perform in the cortex (working memory retrieval, signal propagation), the requirement of reliability strongly implies a deterministic response. However, one does not have to look far in order to find situations, be they in everyday life or in a laboratory environment, where brains (or subunits thereof) react in a highly stochastic manner. Whether this type of apparent randomness has a functional correlate is still a subject of major debate among neuroscientists (see, e.g., Pouget et al., 2013), but there exists increasing evidence that information representation and processing in the cortex is performed, at least to some degree, in a stochastic manner.

neuronal stochasticity

Much in contrast to "conventional" neural network simulations, stochasticity in biological systems seems deeply embedded at all levels of neural information processing (Rolls and Deco, 2010). Ion channels in neurons and the synaptic release of neurotransmitters *intrinsic noise*

are, for instance, inherently probabilistic (Chow and White, 1996; Schneidman et al., 1998). In large-scale simulations, this intrinsic noise is often neglected, as computational resources rarely suffice to allow the implementation of such detailed mechanisms.[1] The response of individual neurons to sensory stimuli shows a large degree of trial-to-trial variability, especially in vivo (Azouz and Gray, 1999). This occurs partly due to the phenomena mentioned above, on the one hand, but also due to stochastic input from their afferents, on the other. The latter kind of stochasticity is a result of the interplay between the dynamics of single neurons and their mutual interaction (as prominently exemplified by the AI-state-model described in Section 5.5). Many models regard this this extrinsic noise as a nuisance more than a computational resource. In simulations, this input noise is implemented by adding random, weak input spikes (usually generated by a Poisson process, as done in, e.g., the KTH L2/3 model, see Section 5.3) or by embedding the model network in a "sea of noise" (as done in, e.g., the synfire chain model, see Section 5.4). Ideally, these models then exhibit robustness towards this kind of randomness.

extrinsic noise

What if, however, some of this temporal variability actually carried information? Indeed, it is not only the response of individual neurons, but also the behavioral response of organisms that also exhibits large trial-to-trial variability (Brascamp et al., 2006). Consider, for example, well-known ambiguous images such as the duck-rabbit or Necker cube illusion[2] (Figure 6.1). Our perception jumps randomly[3] between the two different interpretations of the images. This can be viewed as a good probabilistic representation of the (perceived) underlying reality: either of the two perceptions can be the correct one – with approximately equal probability. In this case, the variability of the perception clearly encodes information, namely the posterior probability associated with the underlying object, conditioned on the ambiguous visual stimulus. Even more, the "state of the brain" at any point in time seems to not encode this probability itself, but rather one of the two possible states – we might even go so far as to say that it seems to sample from the abovementioned posterior. We shall return to this idea later in more detail.

ambiguous optical illusions

behavioral variability encoding information

Beyond the plausibility arguments offered by these perceptual ambiguity scenarios, the evidence of stochastic computation in the brain is steadily increasing. As a reference, we point to two more recent experimental studies involving behavior (Körding and Wolpert, 2004) and spike recordings (Berkes et al., 2011) which strongly support the "Bayesian brain" hypothesis. Such ideas are further strengthened by theoretical work on potential spiking network implementations (Buesing et al., 2011; Deneve, 2008; Rao, 2005; Steimer et al., 2009, only to name a few). We shall explore some of these approaches in more detail later on.

Bayesian inference

Throughout this chapter, we will discuss how ensembles of spiking neurons can perform statistical – and in particular, Bayesian – inference in well-defined probability spaces. This can either be done analytically, implying an explicit computation of the sought probability distribution, or stochastically, by sampling from the underlying distribution over the random variables of interest.

[1]A notable exception can be found, e.g., in the highly detailed simulations of cortical columns in the Blue Brain Project (Markram, 2006).

[2] Technically, this is not really an illusion, since the perception of either the duck or the rabbit does not contradict any established objective reality. However, sensory stimuli that evoke interpretational ambiguity are routinely grouped into the category of cognitive illusions.

[3] Albeit with a significant degree of conscious control.

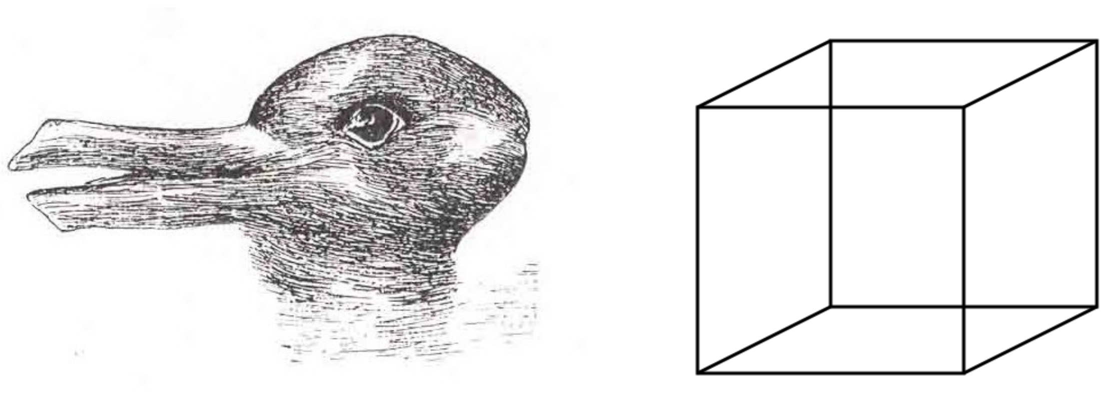


Figure 6.1.: Two optical illusions based on ambiguity. **Left:** The duck-rabbit illusion. As its name already implies, an observer's perception switches between seeing a duck (with its bill pointing towards the left) and a rabbit (with its snout pointing towards the right). **Right:** Necker cube. The 2D wireframe representation allows two possible interpretations of the orientation of the perceived 3D cube, where either of the two squares may be viewed as the front face of the cube.

We start by providing a brief overview of graphical models and belief propagation in Section 6.1, which we also use to introduce important concepts and notations.

In Section 6.3, we describe analytical inference in neural networks that implement previously discussed probabilistic graphical models. In this approach, local spiking activity in the network (more precisely: firing rates) explicitly encodes probability values and communication between neuron populations can be interpreted as Bayesian belief propagation. We shall see that, while providing good estimates for simple distributions, these networks inherit (by design) the drawbacks of belief propagation and also scale rather badly when dependencies between random variables become more complex.

As an alternative to rate-based analytical inference, we turn to spike-based stochastic inference in Section 6.4. After reviewing some basic probabilistic network models (Sections 6.4.1-6.4.2), we describe a theoretical framework for interpreting the activity of abstract neural network models as Markov Chain Monte-Carlo sampling (Section 6.4.3). On top of this model of "neural sampling", we develop a new theory that establishes an analogy between the aforementioned mathematical model and more biological neural dynamics (Section 6.5). With the mathematics in place, we were able to create a software framework that automatically translates any given Boltzmann Machine into a network of spiking LIF neurons (Section 6.5.4).

An essential insight from the theory, which also enabled an important functional aspect of the software, is the possibility for any neuron to correctly calculate conditional distributions regardless of most of its parameters. This concept of "self-calibration" is essential for the realization of neural sampling in analog neuromorphic hardware (Section 6.8). Such a highly accelerated and massively parallel implementation fosters a wealth of applications both within and outside the realm of brain science.

Finally, we turn to more sophisticated network architectures based on neural sampling. In particular, the proposed LIF-based implementation of Boltzmann machines can be extended to multilayered ("deep") networks, which have been extensively studied in the field of machine learning. State-of-the-art training algorithms can be translated to the neural domain (Section 6.6.2), enabling the neural implementation of powerful generative models for complex datasets.

Further extending the scope of LIF-based sampling, we also show how the previously discussed neural sampling framework can be extended from Boltzmann distributions to arbitrary distributions over discrete spaces and demonstrate the implementation of Bayesian networks with LIF neurons (Section 6.7).

6.1. Graphical Models

Graphical[4] models are, first and foremost, useful tools for representing probability distributions. *graphical model, vertex, edge*

In "basic" graphical models – Bayesian networks (Section 6.1.1) and Markov random fields (Section 6.1.1) – each RV is assigned a vertex (or node) in a graph, and the edges (or links) are taken to represent probabilistic relationships between the RVs. The connectivity of the graph then captures the way in which the joint distribution can be decomposed into factors (factorization). *factorization*

In a next step, these factors themselves become vertices in more complex structures called factor graphs (Section 6.1.3). This class of graphical model encompasses the other two in the sense that it can be used to implement the factorization properties of both Bayesian networks and Markov random fields.

Finally, we will then discuss how belief propagation can be performed with the help of these structures, and how efficient message passing algorithms allow the computation of posterior distributions (short: posteriors), i.e., inference (Section 6.1.4). *posterior*

In our discussion of graphical models, we will always assume RVs to be discrete. The generalization to continuous RVs simply requires replacing sums by integrals.

The following section only offers a brief overview of the most important concepts which we will later require. Fore a more in-depth discussion of graphical models, we recommend Chapter 8 of the textbook by Bishop (2009).

6.1.1. Directed Graphs: Bayesian Networks

The most intuitive class of graphical models are the so-called Bayesian networks (BNs). They can be constructed for any probability distribution by repeated application of Bayes' rule (Equation 4.7). Consider an arbitrary joint distribution $p(z_1, \ldots, z_N)$ over n RVs $\{Z_1, \ldots, Z_N\}$. We can always write this as a product over n conditional distributions, one for each RVs: *Bayesian networks*

$$\begin{aligned} p(Z_1 = z_1, \ldots, Z_N = z_N) &= p(z_1|z_2, \ldots, z_N)p(z_2, \ldots, z_N) \\ &= p(z_1|z_2, \ldots, z_N)p(z_2|z_3, \ldots, z_N)p(z_3, \ldots, z_N) \\ &= \ldots \\ &= \prod_{k=1}^{N} p(z_k|z_{k+1}, \ldots, z_N) \quad . \end{aligned} \tag{6.1}$$

We can now represent each of these conditionals $p(z_k|z_{k+1}, \ldots, z_N)$ by a set of directed edges, one for each $Z_i \in \{Z_{k+1}, \ldots, Z_N\}$, pointing from Z_i to Z_k. The left panel in Figure 6.2 shows an example for three RVs. If there exists an edge pointing from vertex z_i to vertex z_j, then z_i is called a parent of z_j and z_j a child of z_j. If there exists a directed path (each step of the path follows the direction of the edges) from vertex z_i to vertex z_j, then z_i is called an ancestor of z_j and z_j a descendant of z_j. An important restriction of *directed edge* *parent, child* *directed path* *ancestor, descendant*

[4] In this context, the word "graphical" stands for graph-related, since these models use graphs to represent probability distributions. By etymological serendipity, it can also be interpreted as "visual", since these graphs are, indeed, a form of visualization of an abstract concept.

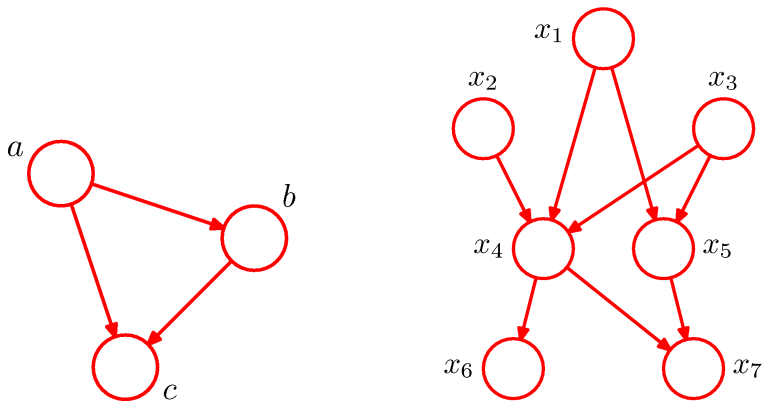


Figure 6.2.: Examples of Bayesian networks. The represented joint probability distributions can be read directly from the graph's connectivity structure (Equation 6.2). **Left:** Generic BN for an arbitrary probability distribution over 3 RVs: $p(a, b, c) = p(c|a, b)p(b|a)p(a)$. **Right:** BN representing a particular joint distribution over 7 RVs: $p(x_1, \ldots, x_7) = p(x_1)p(x_2)p(x_3)p(x_4|x_1, x_2, x_3)p(x_5|x_1, x_3)p(x_6|x_4)p(x_7|x_4, x_5)$.
Figures taken from Bishop (2009).

acyclic graph

these graphical models is that they must be acyclic, i.e., that no directed path may exist from a vertex back to itself.[5]

Some distributions may factorize into "smaller" factors, in which case particular edges might be missing. In general, we consider the joint distribution defined by a Bayesian graph to be the product of conditional distributions, one for each RV, with its value conditioned on the values of its parent variables:

factorization of the joint distribution in BNs

$$p(\boldsymbol{z}) = \prod_{k=1}^{K} \frac{1}{Z} \Phi_k(\boldsymbol{z}_k) := \prod_{k=1}^{N} p(z_k|\mathbf{pa}_k) \quad , \tag{6.2}$$

factor order

where each factor $\Phi_k(\boldsymbol{z}_k)$ is given by the conditional distribution $p(z_k|\mathbf{pa}_k)$ of a node given the state of its parents and $\mathbf{pa}_k$ represents the state vector of the parents of Z_k. The factor $\Phi_k(\boldsymbol{z}_k)$ is called an n^{th}-order factor if it depends on n RVs or rather $|\mathbf{pa}_k| = n - 1$. In BNs, the order of a factor associated with a vertex is therefore equal to the number of parent vertices plus one. An example of the factorization defined by such a BN can be seen in the right panel of Figure 6.2.

While being easily "readable", BNs are not as immediately intuitive in terms of independence relationships between their constituent vertices. In the following, we shall describe

[5] BNs must be acyclic in order to guarantee that their underlying probability distribution is normalized to 1. It is quite easy to prove that this is the case, by starting at a vertex with no parents (which must exist, otherwise the graph would contain a cycle) and marginalizing it out, then repeating the procedure until all vertices have been accounted for.
This is no longer guaranteed to be the case if the graph has a cycle and a counterexample is readily found. Consider the cyclic graph $A \rightarrow B \rightarrow C$ where the value of each parent fully determines the value of its child, e.g., $p(B = x|A = x) = 1$. If we now sum the joint distribution over all possible states (A, B, C), then all states of type (x, x, x) have joint probability 1, and all other states have probability 0. Clearly, the sum over multiple "ones" is larger than one.

the notion of conditional independence (CI) by using the example of gene inheritance.

conditional independence

Consider the case of two (non-twin) siblings whose mother we know does not carry a particular rare gene G. Whether they are carriers themselves (a, b) then only depends on whether the father is a carrier of G or not (c). The associated graph is shown in the left panel of Figure 6.3. The joint distribution reads:

$$p(a,b,c) = p(c)p(a|c)p(b|c) \quad . \tag{6.3}$$

We can verify the independence of a and b by checking whether their joint distribution – which we obtain by marginalizing $p(a,b,c)$ over c – factorizes:

$$p(a,b) = \sum_c p(c)p(a|c)p(b|c) \overset{?}{=} p(a)p(b) \quad . \tag{6.4}$$

In general, this is not the case: if one sibling is found to carry G, this directly implies that the father is a carrier too, increasing the probability of the other sibling to be a carrier as well (from “rare” to ca. 50%[6]). In this graph, a and b are therefore not independent, given no further information:

$$a \not\perp\!\!\!\perp b \mid \emptyset \quad . \tag{6.5}$$

If we now observe c, the joint distribution of a and b (conditioned on our observation) becomes:

$$p(a,b|c) = \frac{p(a,b,c)}{p(c)} = p(a|c)p(b|c) \quad . \tag{6.6}$$

Conditioned on c, a and b have apparently become (conditionally) independent:

$$a \perp\!\!\!\perp b \mid c \quad . \tag{6.7}$$

If we know, for example, that the father carries only one copy of G, then the observation of either sibling's genotype tells us nothing about the other's, since the inheritance of G is independent for the two (with 50% probability).

Now, consider the inheritance of G in the case of two parents (a, b) of a single child (c). The associated graph is shown in the right panel of Figure 6.3. The joint distribution reads:

$$p(a,b,c) = p(a)p(b)p(c|a,b) \quad . \tag{6.8}$$

When verifying the independence of a and b, we find

$$p(a,b) = \sum_c p(a)p(b)p(c|a,b) = p(a)p(b)\sum_c p(c|a,b) = p(a)p(b) \quad . \tag{6.9}$$

Evidently, a and b are independent:

$$a \perp\!\!\!\perp b \mid \emptyset \quad , \tag{6.10}$$

which makes perfect sense under the assumption that the parents are not related.

[6] Actually, the probability is always higher than 50%, since the father could be homozygous at the locus of G.

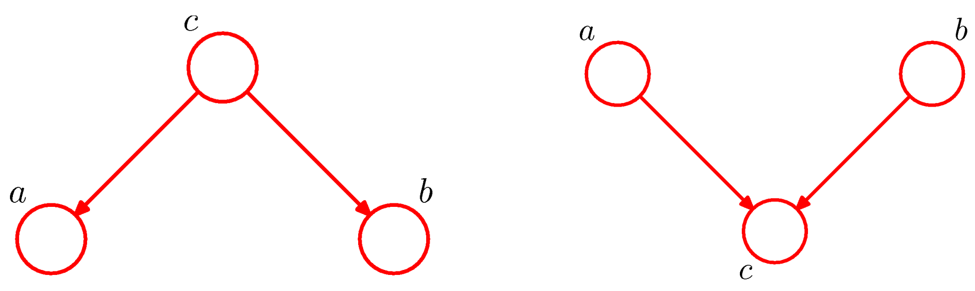


Figure 6.3.: Two examples of conditional (in)dependence in simple BNs. **Left:** Tail-to-tail scenario for two directed edges. Children are, in general, not independent. Conditioned on their parents, however, they become independent. All parents of a vertex therefore belong to its Markov blanket. **Right:** Head-to-head scenario for two directed edges. Unconnected parents are, in general, independent. Conditioned on their children, however, they become dependent, as illustrated by the explaining away phenomenon. Therefore, both children and co-parents of a vertex belong to its Markov blanket. Figures taken from Bishop (2009).

However, conditioned on c, we obtain

$$p(a,b|c) = \frac{p(a,b,c)}{p(c)} = \frac{p(a)p(b)p(c|a,b)}{p(c)} \quad , \tag{6.11}$$

which in general does not factorize into $p(a|c)p(b|c)$. The observation of c has made a and b conditionally dependent:

$$a \not\perp\!\!\!\perp b \mid c \quad , \tag{6.12}$$

which is also readily exemplified. Let us, for example, assume that the child is a carrier of G and we additionally observe the genotype of the father. If we find that the father does not carry G, then it immediately follows that the mother must be a carrier herself (otherwise the child could not carry G). Conversely, should the father be observed to carry G as well, then the chance that the mother is a carrier is reduced to "rare" again. This phenomenon is commonly dubbed "explaining away", since the observation of b explains away the observation of c and thereby changes our prediction for a. It is an immediate consequence of the factor $p(c|a,b)$ in the joint distribution (Equation 6.8) being a function over all three RVs.

explaining away

Our discussion of CI raises an immediate question: given a particular graph $\mathcal{G}$ containing a vertex z_k, which set of RVs $\mathcal{M}_k$ needs to be observed in order to render z_k independent of all other RVs? The set $\mathcal{M}_k$ that satisfies this condition, which can formally be defined as

Markov blanket

$$\mathcal{M}_k : p(z_k|\mathcal{M}_k) = p(z_k|\boldsymbol{z}_{\setminus k}) \quad , \tag{6.13}$$

is called the Markov blanket of z_k. Here, we have used $\boldsymbol{z}_{\setminus k}$ to denote the set of all vertices in $\mathcal{G}$ except for z_k. In the case of BNs, the Markov blanket of a vertex evidently includes all its parents and children, but also all of its co-parents (all other parents of its children), due to the explaining away effect. We shall return to the concept of Markov blankets in BNs in Section 6.7. For now, we only note that the structure of the Markov blanket is rather complicated if the graphical model has directed edges.

Markov blankets in BNs

6.1.2. Undirected Graphs: Markov Random Fields

Since directed graphs lead to somewhat complicated Markov blankets, we can discard the directionality of the edges to produce so-called Markov random fields (MRFs). In such graphs, the asymmetry between parents and children is removed, rendering the very notion of parents and children unapplicable. For undirected edges, two connected vertices are called neighbors. The Markov blanket of a vertex should now simply consist of all of its neighbors (Figure 6.4). Of course, this requires different factorization properties of the underlying joint probability distribution (i.e., MRFs are "read" differently from BNs), which we shall discuss in the following.

Markov random field

neighbor

Markov blankets in MRFs

From our above definition of the Markov blanket in MRFs, it follows that any two unconnected vertices are independent given all other vertices in the graph, since the set of all other vertices includes all neighbors of either node, i.e., their respective Markov blankets:

$$p(z_i, z_j|\boldsymbol{z}_{\setminus\{i,j\}}) \overset{!}{=} p(z_i|\boldsymbol{z}_{\setminus\{i,j\}})p(z_j|\boldsymbol{z}_{\setminus\{i,j\}}) \quad . \tag{6.14}$$

In order for this to hold for any joint distribution associated with the graph, the two nodes may not be variables of the same factor. We can therefore define the factors $\Phi_k(\boldsymbol{z}_k)$ over cliques $\boldsymbol{z}_k$.[7] A clique is a fully connected set of vertices. For the most general case which fulfills our Markov blanket definition, we shall use maximal cliques, since they contain all other cliques as subsets, i.e., the factorization into potential functions over maximal cliques allows a finer factorization over subsets of these. We can now write our joint distribution as

clique

maximal clique

$$p(\boldsymbol{z}) = \frac{1}{Z}\prod_k \Phi_k(\boldsymbol{z}_k) \quad , \tag{6.15}$$

where the product runs over all maximal cliques $\boldsymbol{z}_k$ in the graph. Division by a so-called partition function Z

partition function

$$Z = \sum_{\boldsymbol{z}}\prod_k \Phi_k(\boldsymbol{z}_k) \tag{6.16}$$

ensures correct normalization.

The individual potential functions must be non-negative and can, for all practical purposes, be required to be positive. This makes it convenient to express them as exponentials over energy functions $E(\boldsymbol{z}_k)$:

energy function

$$\Phi_k(\boldsymbol{z}_k) = \exp[-E(\boldsymbol{z}_k)] \quad . \tag{6.17}$$

The exponential representation is often useful, since it effectively transforms the product over potential functions into a sum over energy functions. This, in turn, is intuitively pleasing from a physicist's perspective: configurations with a small total energy are more probable then high-energy states. Due to its formal equivalence to the probability function of microstates in a canonical ensemble, the exponential representation of a joint distribution is also called a Boltzmann distribution.

Boltzmann distribution

As mentioned above, the choice of cliques depends on the modeled problem, with smaller cliques implying factors with fewer arguments. A particularly useful choice is to

[7] More formally, this argument is underpinned by the Hammersley–Clifford theorem.

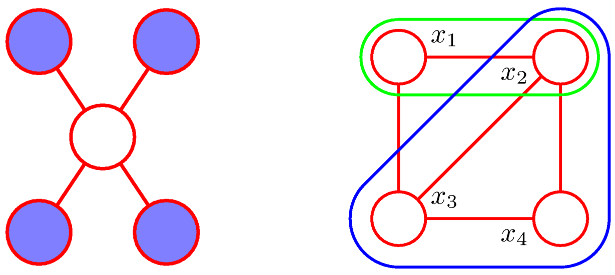


Figure 6.4.: **Left:** The Markov blanket of a vertex in an MRF is given by the set of its neighbors. **Right:** Cliques in an MRF are defined as fully connected subgraphs. The green clique is not maximal, since it can be extended by including x_3. The blue clique is maximal, since the inclusion of x_1 would violate the requirement of full connectivity. Figures taken from Bishop (2009).

restrict the cliques to a maximum size of two; in this setting, the energy of the system can be interpreted as a result of pairwise interactions between its constituent units, with obvious parallels to physical systems and useful mathematical properties. Section 6.4 is dedicated entirely to these so-called Boltzmann machines.

While the CI properties of undirected graphs are very useful, their underlying joint distribution is, in general, also their most major limitation. Calculating the partition function of an MRF requires computing a sum over all possible states, which, in the worst case, is exponential in the total number of RVs. The computation of local conditionals, however, can be done efficiently, since it involves the ratio of two marginals, for which the partition functions in the numerator and denominator cancel each other out. This plays an important role in the sample-based evaluation of joint distributions (Sections 6.4.2 and 6.4.3).

6.1.3. Factor Graphs

As we have previously illustrated, the CI properties of a graphical model are intimately connected with the factorization of its underlying joint distribution. Indeed, the fact that all CI properties are directly readable from a graph represents an elegant and useful property of graphical models. As it turns out, the sets of CI properties that can be expressed by BNs and MRFs are not identical.

Consider, for example, the BN depicted in the left panel of Figure 6.5, which satisfies the CI properties $A \perp\!\!\!\perp B \mid \emptyset$ and $A \not\perp\!\!\!\perp B \mid C$ (the latter statement represents the explaining away effect). No MRF over the same three variables can satisfy both these properties, since the first one implies that A and B are not connected, which then directly violates the second CI property.

On the other hand, the MRF depicted in the right panel of Figure 6.5 satisfies the properties $A \not\perp\!\!\!\perp B \mid \emptyset$, $C \not\perp\!\!\!\perp D \mid \emptyset$, $A \perp\!\!\!\perp B \mid C \cup D$ and $C \perp\!\!\!\perp D \mid A \cup B$. A BN could only satisfy these conditions if each vertex had exactly one parent and one child, which would then imply the existence of a cycle, which in turn is forbidden by the requirement that BNs must be acyclic.

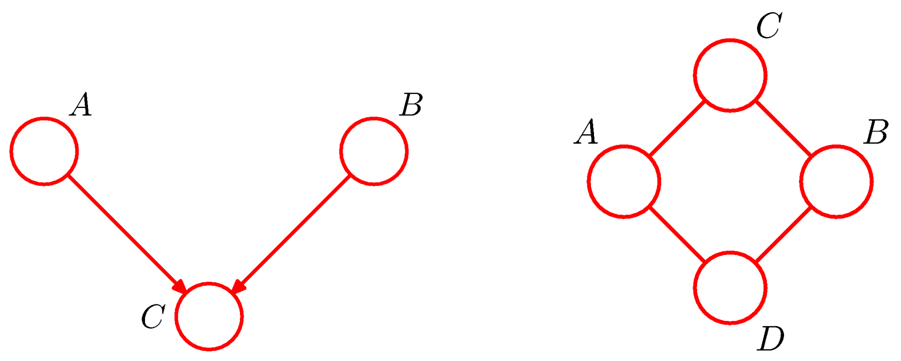


Figure 6.5.: Examples of graphs that can not be translated from directed to undirected and vice-versa while preserving all CI properties. **Left:** This BN can not be translated to an MRF, since MRFs can not exhibit explaining away effects (Markov blankets in MRFs do not extend beyond nearest neighbors). **Right:** This MRF can not be translated to a BN, since it would require the existence of a cycle (a more detailed explanation is given in the main text). Figures taken from Bishop (2009).

Essentially, what we require is a graphical representation of either type of factors, be they conditional distributions or potential functions. This can be achieved by introducing an additional type of node, so-called factor nodes, which are used to generalize the representation of factors in a joint distribution. The resulting graphs are called factor graphs and can be used to represent the CI properties of both BNs and MRFs.

factor node

factor graph

More formally, a factor graph (FG) is a bipartite graph, i.e., it can be partitioned into two sets (or types) of nodes, where no two nodes of the same type may have a connection. The first type are the variable nodes which we already know from BNs and MRFs, and the second type are the newly introduced factor nodes (see Figure 6.6). The factor nodes represent the factors f_s in the joint distribution and the neighbors $\boldsymbol{x}_s$ of f_s are its arguments. The joint distribution therefore reads

bipartite graph

factorization of the joint distribution in FGs

$$p(\boldsymbol{x}) = \prod_s f_s(\boldsymbol{x}_s) \quad . \tag{6.18}$$

The introduction of factor nodes allows additional freedom in our graphical representations, with several examples given in Figures 6.7 and 6.8. Furthermore, it supports an efficient marginalization algorithm, which we describe in the following.

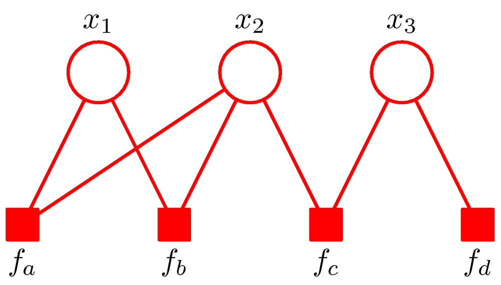


Figure 6.6.: Example FG, with an explicitly depicted partition into variable (circles) and factor (squares) nodes. The joint distribution is a product of the factors over their neighbor variables, i.e., $p(x_1, x_2, x_3) = f_a(x_1, x_2)f_b(x_1, x_2)f_c(x_2, x_3)f_d(x_3)$. Figure taken from Bishop (2009).

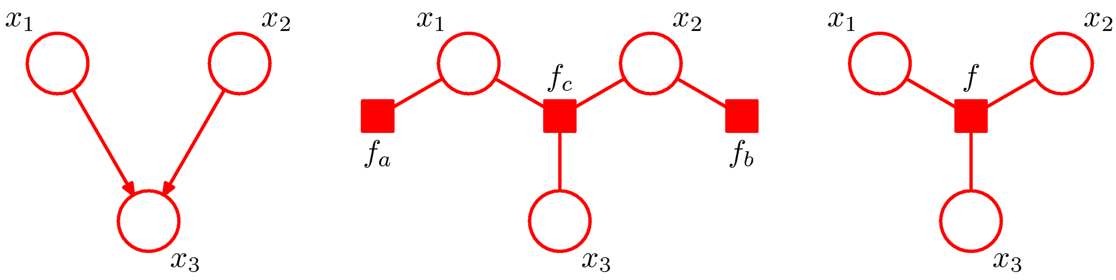


Figure 6.7.: Transforming a BN into an FG. **Left:** BN with the factorization $p(x_1, x_2, x_3) = p(x_1)p(x_2)p(x_3|x_1, x_2)$. **Middle:** FG with the same factorization, if we define $f_a := p(x_1)$, $f_b := p(x_2)$ and $f_c := p(x_3|x_1, x_2)$. **Right:** Alternative translation of the same BN to an FG, where the factor simply represents the entire joint distribution $p(x_1, x_2, x_3)$. Figures taken from Bishop (2009).

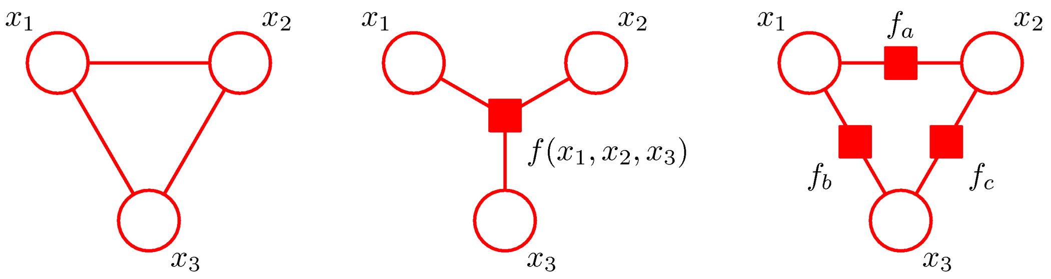


Figure 6.8.: Transforming an MRF into an FG. **Left:** MRF over 3 RVs which does not factorize in general, since it is fully connected, so the only maximal clique is the graph itself. **Middle:** FG with the same joint distribution, where $f := p(x_1, x_2, x_3)$. **Right:** If the joint distribution does factorize into, e.g., potential functions over pairs of RVs, this can be readily accomodated by an FG representation. This is, in particular, the case for the Boltzmann machines in Section 6.4.2. Figures taken from Bishop (2009).

6.1.4. Inference in Factor Graphs

A rather generic formulation of what could be called "the inference problem" would be the following: given a joint distribution over a set of RVs and a number of observations on a subset of these, what can we infer about the possible states of one or more RVs of interest? In other words, an inference problem is formally a problem of marginalization:

1. We start with the joint distribution $p(\boldsymbol{x})$.
2. Observing a subset of RVs turns the joint distribution into a conditional, which, if we do not care about normalization[8], has zero computational cost since it simply requires replacing these variables in the joint distribution by their particular observes values $\boldsymbol{x}_{\mathrm{o}}$:
$$p(\boldsymbol{x}) \longrightarrow p(\boldsymbol{x}_{\backslash \mathrm{o}}|\boldsymbol{x}_{\mathrm{o}}) = p(\boldsymbol{x}_{\backslash \mathrm{o}}, \boldsymbol{x}_{\mathrm{o}})/Z \quad , \tag{6.19}$$
where, from Bayes' rule, $Z = p(\boldsymbol{x}_{\mathrm{o}}) = \sum_{\boldsymbol{x}_{\backslash \mathrm{o}}} p(\boldsymbol{x}_{\backslash \mathrm{o}}, \boldsymbol{x}_{\mathrm{o}})$.
3. Inferring the distribution over some RVs of interest $\boldsymbol{x}_{\mathrm{i}}$ now requires us to sum over all possible values of all unobserved RVs that are not of interest (i.e., marginalize them out):
$$p(\boldsymbol{x}_{\mathrm{i}}) = \sum_{\boldsymbol{x}_{\backslash \mathrm{i}, \backslash \mathrm{o}}} p(\boldsymbol{x}_{\backslash \mathrm{o}}|\boldsymbol{x}_{\mathrm{o}}) \quad . \tag{6.20}$$

As already discussed (Section 4.1), it is the computational cost of the marginalization that scales exponentially with the number of RVs to be marginalized out. The most computationally costly case is therefore the one where no RVs are observed and we are interested in the marginal distribution over a single RV of interest. We shall further assume that our FG is a tree graph, i.e., that there exist no (undirected) cycles. It is quite easy to see that, in principle, any FG can be transformed into a tree FG by absorbing factor nodes that lie within a cycle into a single factor node.[9] *tree graph*

In a factor graph, the marginal distribution over a single RV x is given by
$$p(x) = \sum_{\boldsymbol{x} \backslash x} p(\boldsymbol{x}) = \sum_{\boldsymbol{x} \backslash x} \prod_{s} f_s(\boldsymbol{x}_s) \quad . \tag{6.21}$$

If we consider the above factorization and use the left panel of Figure 6.9 as guidance, we can see how the variable node x partitions the joint distribution into larger factors $F_s(x, \boldsymbol{X}_s)$, which contain all factors that lie "behind" the neighbor factor nodes f_s of x (subgraph factorization):
$$p(x) = \sum_{\boldsymbol{x} \backslash x} \prod_{s} f_s(\boldsymbol{x}_s) =: \sum_{\boldsymbol{x} \backslash x} \prod_{s \in \mathrm{ne}(x)} F_s(x, \boldsymbol{X}_s) \quad . \tag{6.22}$$

[8] We may only not care about normalization if it can be performed at the very end of the inference process, when the number of remaining variables and therefore the computational cost is small (since normalization itself involves marginalization). This is, indeed, the case for the sum-product algorithm which we present in this section.

[9] It is important to note that this is not a "universal solution", since the absorption of factors into larger factors leads to an increase in the number of arguments of the larger factor, thereby rendering the local computations of the sum-product algorithm, which involve local marginalization (Equation 6.25), more costly.

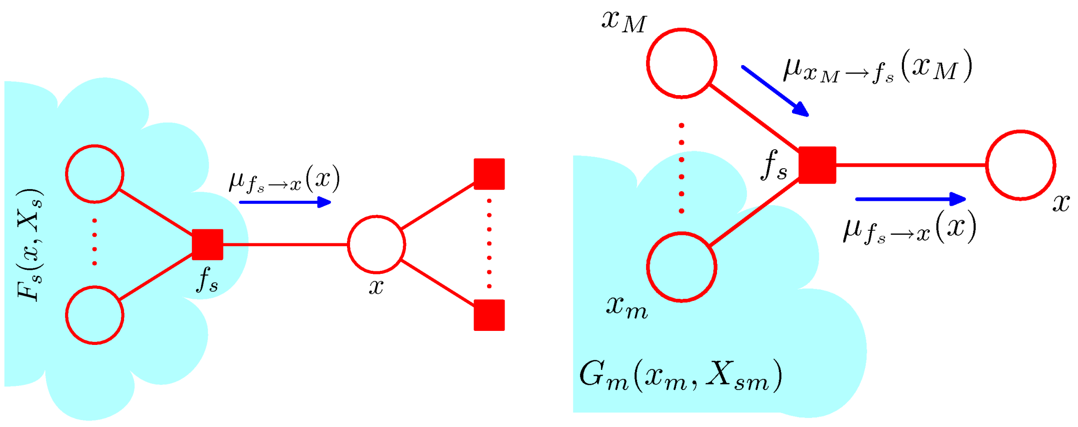


Figure 6.9.: Visualization of the different stages of message passing in the sum-product algorithm. The computability of an outgoing message as (a sum over) a product over all incoming messages is a consequence of the assumed tree structure of the FG, which allows a partitioning into subtrees at every stage of the message passing protocol. **Left:** The marginal distribution of any RV as a product of incoming messages from neighbor factor nodes (Equation 6.23). The same formal rule applies for calculating the outgoing message from a variable node (Equation 6.26). **Right:** The outgoing message from a factor node as a function of the incoming messages from neighbor variable nodes (Equation 6.25). Figures taken from Bishop (2009).

As the sets of variables encompassed by the different $\boldsymbol{X}_s$ are pairwise disjoint (since the graph is a tree, see again the left panel of Figure 6.9), we can interchange the sum and the product:

marginal as product over incoming messages

$$\begin{aligned} p(x) &= \sum_{\boldsymbol{x} \setminus x} \prod_{s \in \mathrm{ne}(x)} F_s(x, \boldsymbol{X}_s) \\ &= \sum_{\boldsymbol{X}_1} \cdots \sum_{\boldsymbol{X}_N} \prod_{s \in \mathrm{ne}(x)} F_s(x, \boldsymbol{X}_s) \\ &= \sum_{\boldsymbol{X}_1} F_1(x, \boldsymbol{X}_1) \ldots \sum_{\boldsymbol{X}_N} F_N(x, \boldsymbol{X}_N) \\ &= \prod_{s \in \mathrm{ne}(x)} \underbrace{\sum_{\boldsymbol{X}_s} F_s(x, \boldsymbol{X}_s)}_{\mu_{f_s \to x}(x)} \end{aligned} \tag{6.23}$$

In this formulation, we can interpret the marginal distribution of x as a product over "messages" $\mu_{f_s \to x}(x)$ arriving from its neighbor factor nodes.

In the next step, we now evaluate the messages $\mu_{f_s \to x}(x)$. Again, we partition the tree that lies "behind" f_s into disjoint subtrees $G_m(x_m, \boldsymbol{X}_{sm})$ (see Figure 6.9, right panel):

$$F_s(x, \boldsymbol{X}_s) = f_s(x, x_1, \ldots, x_M) G_1(x_1, \boldsymbol{X}_{s1}) \ldots G_M(x_M, \boldsymbol{X}_{sM}) \quad . \tag{6.24}$$

We can now plug this equation into Equation 6.23 and use the same switching of sums and products as we did before to obtain

recursive computation of messages from factor to variable nodes

$$\begin{aligned}\mu_{f_s \to x}(x) &= \sum_{\boldsymbol{X}_s} F_s(x, \boldsymbol{X}_s) \\ &= \sum_{\boldsymbol{X}_s} f_s(x, x_1, \dots, x_M) \prod_{x_m \in \text{ne}(f_s) \setminus x} G_m(x_m, \boldsymbol{X}_{sm}) \\ &= \sum_{\boldsymbol{x}_s = (\text{ne}(f_s) \setminus x)} f_s(x, \boldsymbol{x}_s) \prod_{x_m \in \text{ne}(f_s) \setminus x} \underbrace{\sum_{\boldsymbol{X}_{sm}} G_m(x_m, \boldsymbol{X}_{sm})}_{\mu_{x_m \to f_s}(x_m)} \end{aligned} \tag{6.25}$$

Messages from factor nodes to variable nodes can therefore be calculated over incoming messages from neighboring variable nodes. Note that this step can become computationally very costly if factors are large (i.e., if they have many variable node neighbors).

The same logic that we applied to calculating the marginal over x in Equation 6.23 can now be applied to express the messages from variable to factor nodes as products over messages from factor to variable nodes:

recursive computation of messages from variable to factor nodes

$$\mu_{x_m \to f_s}(x_m) = \prod_{l \in \text{ne}(x_m) \setminus f_s} \underbrace{\sum_{\boldsymbol{X}_{ml}} F_l(x_m, \boldsymbol{X}_{ml})}_{\mu_{f_l \to x_m}(x_m)} \tag{6.26}$$

sum-product algorithm

Equations 6.25 and 6.26 define the sum-product algorithm (SPA) as a message passing algorithm along the edges of an FG. Every outgoing message from a node can be calculated recursively from incoming messages to that node. The recursion ends when leaf nodes are reached, for which the output messages are also easily found within the above framework:

leaf and node output messages

$$\mu_{x_{\text{leaf}} \to f}(x) = 1 \tag{6.27}$$

$$\mu_{f_{\text{leaf}} \to x}(f) = f_{\text{leaf}}(x) \tag{6.28}$$

In order to calculate the marginal distribution of a different RV, the SPA must not be repeated all over again. Imagine that we have randomly chosen some variable node x as a root node. Starting from the leafs of the tree, we can now propagate messages all the way down to the root. This allows calculating the marginal $p(x)$. However, now that x has received all incoming messages, it can also send out messages along all its incident edges. We can therefore continue the propagation of messages all the way back to the leaves. For each edge, we now have two messages going into the two possible directions, with only twice the computational cost of calculating $p(x)$. However, since we now know all incident messages for all of the variable nodes, we can calculate the marginals of all the variables in the FG. We shall make use of this in our neural network implementation of the SPA (Section 6.3).

So far, we have not discussed normalization. Even when the joint distribution has been previously normalized, it can become an issue when the observation of a subset of RVs would require the computation of conditionals and therefore marginalization. However, in this case we would treat the resulting conditional simply as an unnormalized version of the original joint distribution. Since the SPA performs local computations correctly and in the end produces distributions over single variables x, a final normalization step of $p(x)$ is both trivial and computationally cheap.

On a final note, we should mention that the restriction to tree graphs is a necessary formal requirement (since it allows the partitioning of the graph into disjoint subtrees at each stage of the message passing protocol), it is not that strict from a practical point of view. Even if a graph has cycles, we can still use the SPA, since the computation of messages is local. However, the messages themselves travel along the cycles and will start updating themselves once a cycle is complete. In contrast to the SPA, this so-called loopy belief propagation is not guaranteed to converge.

loopy belief propagation

6.1.5. The Sum-Product Algorithm in Forney Factor Graphs

In the particular case in which each variable node is connected to no more than two factor nodes, the SPA takes on a particularly simple form. The computation performed at individual variable nodes (Equation 6.26) reduces to a simple passing on of the incoming message. In this situation, the variable nodes can be dropped altogether and the edges themselves represent the RVs. Such a graph is called a Forney factor graph (FFG) and has the obvious advantage that the SPA reduces to a single equation for the factor nodes:

Forney factor graph

messages in an FFG

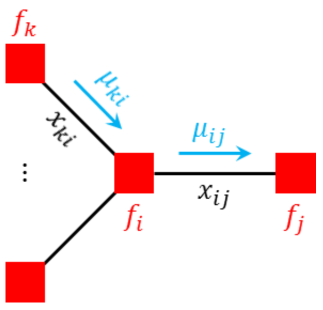
$$\mu_{ij} := \mu_{f_i \to f_j}(x_{ij}) = \sum_{\boldsymbol{x}_i := (x_{ki} | f_k \in \mathrm{ne}(f_i) \setminus f_j)} f_i(x_{ij}, \boldsymbol{x}_i) \prod_{f_k \in \mathrm{ne}(f_i) \setminus f_j} \mu_{ki}(x_k) \quad , \tag{6.29}$$

where $\boldsymbol{x}_i$ represents the state vector of the variables that lie "behind" a factor node and μ_{ki} the incoming messages across the respective edges. A graphical visualization of the above equation is shown in the left panel of Figure 6.10.

The condition imposed on the factor dependencies (only two factors may depend on the same variable) can be circumvented by "copying" variables with the help of "equality constraint factors". An equality constraint factor that creates n copies of some variable x represents the function $\prod_{i=1}^{n} \delta(x - x_i)$. Other hard constraints can also be represented by delta functions over appropriately chosen functions. An example FFG with both hard constraints and probabilistic nodes is shown in the right panel of Figure 6.10.

equality constraint factor

Due to the simplified form of the SPA, our first neural network implementation of probabilistic inference will be a representation of an FFG.



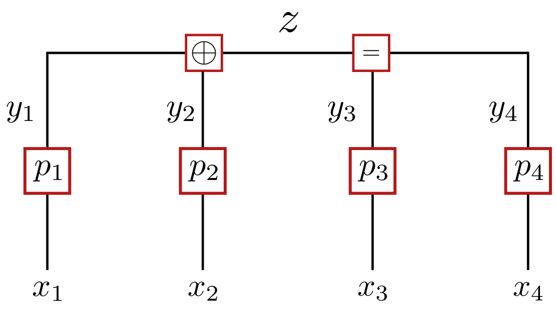


Figure 6.10.: **Left:** SPA in an FFG. The variables only have two neighboring factors and therefore simply pass on incoming messages. The SPA then reduces to a single equation for factor nodes. **Right:** Example of an FFG with probabilistic nodes and hard constraints. The graph represents a noisy binary channel with four valid messages $(y_1, \ldots, y_4) \in \{(0,0,0,0),(0,1,1,1),(1,0,1,1),(1,1,0,0)\}$. The set of valid messages is fully specified by the following constraint: the last two bits encode the parity of the sum of the first two bits. This can be imposed upon the noised bits with the help of an additional variable z and two hard constraint factors:

- z represents the parity of $y_1 + y_2$. This is encoded by the $f_\oplus$ factor: $f_\oplus(y_1, y_2, z) = \delta(y_1 \oplus y_2 \oplus z)$, where $\oplus$ denotes the XOR operator.
- The last two bits and z are equal. This is encoded by the $f_=$ factor: $f_=(z, y_3, y_4) = \delta(z - y_3)\delta(z - y_4)$.

The channel noise may flip each bit with a certain probability, which is directly encoded by the factors p_i. Applying the SPA to this graph effectively yields an estimate of the likelihood of every (unobserved) input bit y_i given the observed values x_i of the bits after they have been passed through the noisy channel

$$p(y_1, \ldots, y_4 | x_1, \ldots, x_4) = f_\oplus(y_1, y_2, z) f_=(z, y_3, y_4) \prod_{i=1}^{4} p(x_i | y_i) \quad . \tag{6.30}$$

Figure taken from Petkov (2012).

6.2. Liquid State Machines

Since factor nodes in FFGs can be chosen to represent arbitrary functions over their adjacent variables, we would ideally require a (sub)network model that is in principle able to perform this kind of universal computation, together with an appropriate training algorithm. *liquid state machine* The rather natural choice are liquid state machines (LSMs, see in particular Jaeger, 2001; Maass et al., 2002), which we shall briefly discuss in the following. Additionally, before moving to belief propagation with LSMs, we will present a successful implementation of an LSM on neuromorphic hardware and its application to a particular classification task. These experiments are the result of a collaboration with Sebastian Jeltsch and others and have already been reported at the 2010 Capo Caccia workshop (Brüderle et al., 2010), as well as published in Pfeil et al. (2013).

6.2.1. Network Model

The principle behind an LSM is simple, but may sound counterintuitive at first. Imagine that we have two pebbles and we wish to measure their momentum. In addition to several obvious ways of doing it, one alternative would include having them fly into a pond and inferring their momentum from the amplitude of the ripples they produce on the surface. One advantage of this seemingly convoluted method is that the observation of the splash would simultaneously allow us to retrieve information about their shape, size, density etc. Even more than that, we could also directly retrieve the results of simple operations, such as the sum of their momenta, from the interference patterns they produce – once again, by only observing the behavior of the water.

Let us put this in more formal terms, with Figure 6.11 serving as guidance. Consider a time-varying input $\boldsymbol{u}(t)$ that belongs to an n-dimensional space $\mathcal{U}^n$, of which we need to *liquid filter* compute some function $\Phi\boldsymbol{u}(t)$. The liquid can be represented as a filter function L that projects the n-dimensional input into the much higher-dimensional function space:

$$\boldsymbol{x}^k(t) = (L^k\boldsymbol{u})(t) \quad , \quad L^k : \mathcal{U}^n \mapsto (\mathbb{R}^\mathbb{R})^k \quad . \tag{6.31}$$

In order to allow the computation of arbitrary functions, the liquid must fulfill the so-called *separation property* separation property, which requires different inputs to be mapped to different outputs. In principle, this is easily achievable as long as the liquid is complex enough.

Assuming that the observation of a liquid that fulfills the separation property yields the result of the sought computation, we require a second component of the LSM which can be trained to perform this observation. Since we assume that the "hard" (i.e., nonlinear) part of the computation has already been performed by the liquid, we can limit *linear readout* the readout to simply implement a linear transformation of the liquid state:

$$y^m(t) = f^m(\boldsymbol{x}^k)(t) \quad , \quad f^m : (\mathbb{R}^\mathbb{R})^k \mapsto \mathbb{R}^m \quad . \tag{6.32}$$

The linear map f^m can be represented by a set of (unconnected) perceptrons, which can be efficiently trained with simple algorithms. This is where the high-dimensional projection performed by the liquid becomes important, since it is assumed to transform an initially

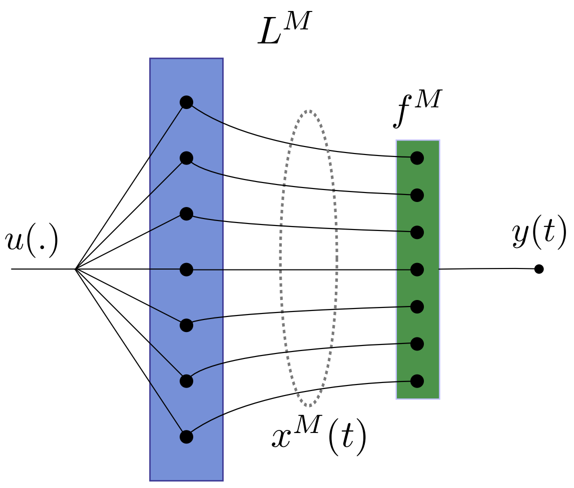


Figure 6.11.: Stages of computation within an LSM. The input $\boldsymbol{u}$ is projected into a high-dimensional space by the liquid filter L. A simple linear readout f can then be trained to approximate a given function $\boldsymbol{y} = \Phi \boldsymbol{u}$ of the input. Figure taken from Petkov (2012).

nonlinear separation problem into a linear one.[10] The only requirement for the readout itself is that it must fulfill the so-called approximation property, i.e., its output must be in principle able to approximate any result of the sought computation.

approximation property

The required computation is now represented by a two-step chain of liquid and readout:

$$\boldsymbol{u}(t) \xrightarrow{L} \boldsymbol{x}(t) \xrightarrow{f} \boldsymbol{y}(t) \quad . \tag{6.33}$$

The key advantage here is that the liquid must not be trained at all and the training of the readout is simple and efficient. Should we require to compute a different function Φ', we only need to train an additional readout f'. This compares favorably with other approaches such as multilayer perceptrons, deep belief networks etc.

In practice, some additional "tweaking" of the liquid may be needed. An important requirement, which we shall only briefly mention, is that the liquid may not be "too chaotic", since then any amount of noise on the initial input would render a practicable training of the readout impossible. The liquid therefore ideally treads on the "edge of chaos", where the separation property is fulfilled well, but without causing trouble for the subsequent training. The chaoticity of the liquid is usually controlled by changing the scale of interaction between its microscopic components – in a neural network, this would be given by the synaptic density and strength.

edge of chaos

Another advantage of having a liquid as a preprocessing stage is the possibility to take into account past information as well. Depending on the relaxation time constants within the liquid, the readout may gain access to previous states of the input, endowing the LSM with fading memory:

fading memory

$$\boldsymbol{y}(t) = fL(u(t'))_{t' \in [t-T,t]} \quad , \tag{6.34}$$

[10] At this point, one might recognize the similarity between this approach and the well-known "kernel trick" from machine learning. The difference is, however, that the state of the liquid constitutes an explicit representation of the projection of the input into a higher-dimensional space, whereas kernel functions are specifically designed to circumvent this explicit projection by directly computing only inner products (which, in turn, encode similarity) within the high-dimensional space.

kernel trick

where T represents the memory duration.

Up to this point we have deliberately remained unspecific about the nature of the liquid itself. This is because, in principle, any sufficiently complex device that features the separation property can be used as a liquid. In practice, when requiring a physical realization of an LSM, one would choose a particular implementation. In the particular context of neural networks, LSMs are, of course, represented by populations of neurons. The idea of having a generic microcircuit for arbitrary computations with individual readouts being trained for particular tasks, is, of course, quite appealing as a hypothesis for the architecture of the cortex itself. As we shall see in the following, even small neuronal LSMs can be quite efficient at performing simple operations on input spike trains. Unfortunately however, as we shall explore later on, the efficiency of this concept does not scale that well when applied to more complicated tasks.

6.2.2. A Multitasking Neuronal LSM

We shall highlight the most important computational properties of an LSM with an exemplary neuronal implementation from (Maass et al., 2002). The liquid itself consists of two columns, each containing 88 excitatory and 27 inhibitory CUBA LIF neurons with distance-dependent connectivity. The dominant relaxation time constants are those of synaptic depression, which are on the order of several hundred ms. The readouts are represented by populations of 51 unconnected, non-spiking CUBA LIF neurons (essentially, perceptrons with a low-pass filtered input) and have been trained with a modified perceptron learning rule.

The input consists of four 2 s spike trains which project to different neurons in the liquid. Each one of the 6 readouts was trained for a different task on a batch of 150 training inputs. In particular, some of the tasks explicitly required knowledge of past states of the input. The output of the readouts is a single number and is given by their normalized population firing rate. Figure 6.12 shows the output of the readouts for a particular novel (i.e., not part of the training batch) test input compared to the analytically calculated target output.

This experiment is particularly instructive, since it showcases the following properties of LSMs, which we shall try to exploit later:

- both the liquid and the readouts require only modest resources;
- multiple readouts can be trained for different tasks on the same liquid;
- readout training is simple and can be parallelized;
- the network is able to generalize its computational capabilities to inputs it has never "seen" before;
- the fading memory allows computations on past input states.

We stress again that these valuable properties are highly task-dependent and do not necessarily carry over to more complicated problems.

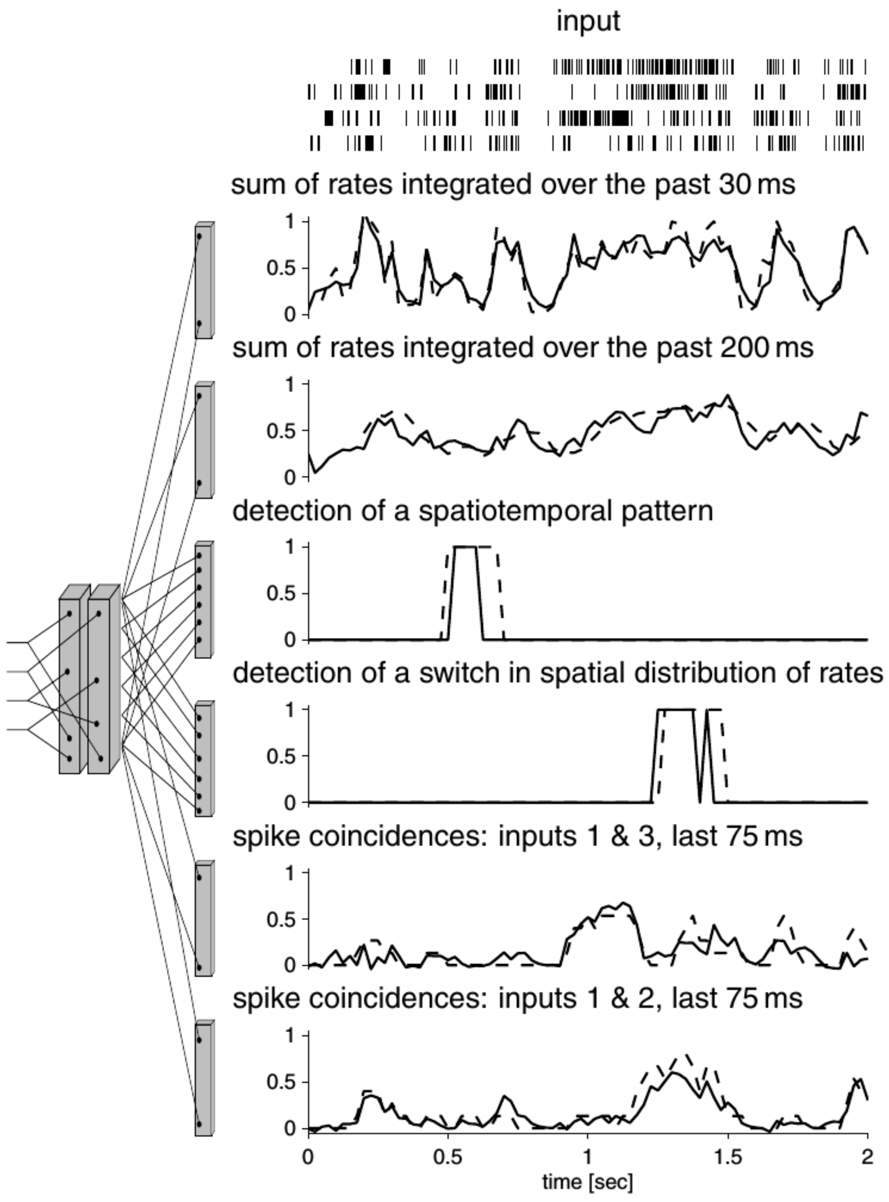


Figure 6.12.: LSM trained to simultaneously perform six computational tasks (with six different readouts) on an input consisting of four spike trains. Continuous lines represent the output of the readouts, while dashed lines denote the target output. The first two tasks involve the integration of the input population rate over 30 and 200 ms, respectively. The good performance of the LSM is also a showcase of its relatively long fading memory due to synaptic STP. The third task requires the detection of a particular spatiotemporal pattern. The fourth task requires the LSM to detect when the summed firing rates of inputs 1 and 2 increase and the summed firing rates of inputs 3 and 4 decrease simultaneously. The final two readouts are trained to represent firing correlations between different pairs of input channels. Figure taken from Maass et al. (2002).

6.2.3. Neuromorphic Implementation

We have argued above that the exact nature of a liquid is, in principle, not important, as long as the liquid is complex enough. In particular, its components must not be tuned towards some particular, precise dynamics. Furthermore, the computations performed by the liquid are robust to a certain level of noise (Maass et al., 2002). This makes liquid computing an ideal candidate for implementation in substrates that can not be perfectly controlled, as is the case for essentially all neuromorphic platforms which rely on analog components.

In the following, we describe some exemplary experiments with liquids on the Spikey chip (see Section 3.2). These results have already been reported in Pfeil et al. (2013).

balanced network

Our neuromorphic LSM consisted of two major components: the recurrent liquid network itself and a spike-based classifier operating as a readout (Figure 6.13A). As a general purpose liquid needs to meet the separation property (see Section 6.2.1), we aimed for a neuronal implementation which operates in a balanced state. In particular, short-term plasticity has been shown to endow network models with a stable firing regime (Sussillo et al., 2007), which we can explicitly exploit on Spikey. Such a network topology has been already successfully implemented on Spikey (Bill et al., 2010), allowing us to use the same network topology for our liquids. However, the global effects of short-term plasticity on Spikey would have interfered with the dynamics of the readout, which required us to remove this feature while otherwise maintaining the network structure. As it turned out, the resulting network was still balanced enough to support the LSM implementation.

Our balanced network consists of an excitatory and inhibitory population with a ratio of 80:20 excitatory to inhibitory neurons. As we use an entire block of 192 neurons, only 64 drivers remain for external input, which we assign to 32 excitatory and 32 inhibitory sources. From these, each neuron in the liquid receives 4 excitatory and 4 inhibitory inputs. All other connection probabilities are illustrated in Figure 6.13A.

tempotron

The readout is realized by means of a tempotron (Gütig and Sompolinsky, 2006), which is compatible with our hardware due to its spike-based nature. Furthermore, as it essentially consists of a single neuron, it leaves most hardware resources to the liquid.

A tempotron can be trained to discern between two classes of temporal patterns by spiking for all patterns in one class (which we denote as $\oplus$) and remaining silent for all patterns in the other (which we denote as $\ominus$). Gütig and Sompolinsky (2006) describes a gradient-descent-based training method for the tempotron's afferent weights:

tempotron learning rule

$$\Delta w_i^n = \begin{cases} 0 & \text{for a correct response} \\ \alpha(n)\sigma(n) \sum\limits_{t_i < t_\text{max}} \kappa(t_\text{max} - t_i) & \text{for an erroneous response} \end{cases} , \tag{6.35}$$

where Δw_i^n is the weight update corresponding to the ith afferent neuron after the nth learning iteration with learning rate $\alpha(n)$ and κ represents the PSP kernel (see Section 4.2). The nature of the training sample in a particular training run is given by $\sigma(n)$, which defines the sign of the weight update:

$$\sigma(n) = \begin{cases} -1 & \text{for a } \ominus \text{ training pattern} \\ 1 & \text{for a } \oplus \text{ training pattern} \end{cases} . \tag{6.36}$$

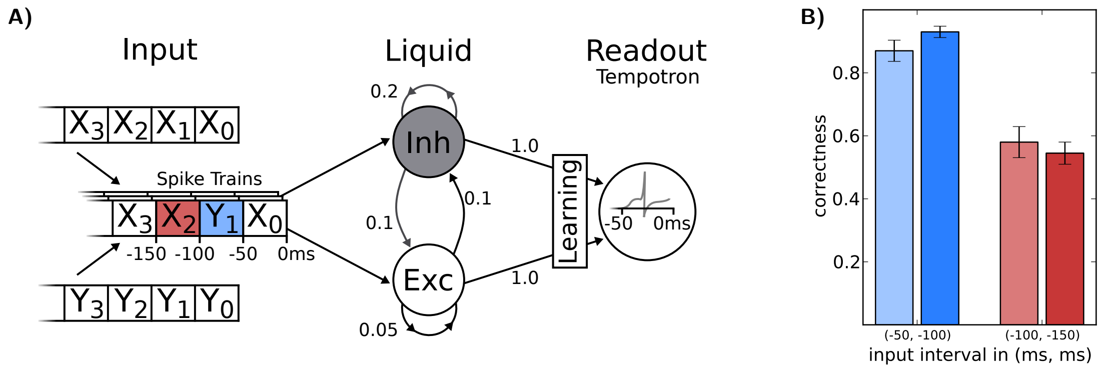


Figure 6.13.: LSMs on Spikey. **(A)** Schematic of the LSM and the given task. Spike sources are composed of 50 ms segments drawn from two template spike trains (X and Y). These patterns are streamed into the liquid (with descending index), which is a network consisting of 191 neurons, leaving one neuron for the tempotron. Connection probabilities are depicted next to each connection (arrows). In two experiments, the tempotron is trained to either classify the origin (X or Y) of the spike train segment with index 1 or 2. **(B)** The classification performance of the LSM measured over 200 samples after 1000 training iterations for both hardware (lighter) and software (darker) implementation. Figure taken from Pfeil et al. (2013).

The time at which the tempotron has reached its highest membrane potential in the respective training run is denoted by t_{max} (which is equivalent to t_{spike} in case the tempotron has spiked). The reasoning behind this learning rule is to “punish” erroneous firing appropriately. In case of a false positive, i.e., for $\ominus$ trials where the tempotron has spiked, the excitatory afferents with a causal contribution to this spike are weakened and inhibitory ones are strengthened, as $\Delta w_i^n \propto \sigma(n) < 0$. In case of a false negative, i.e., for $\oplus$ trials where the tempotron did not spike even though it should have, the direction of the weight modulation is inversed, i.e., excitatory weights are strengthened and inhibitory ones are weakened.

Upon training, the tempotron distinguishes between two input classes by emitting either one or no spike within a certain time window. The former is artificially enforced by blocking all further incoming spikes after the first spike occurrence.

binary decision problem

As the tempotron is a binary classifier, we needed to define an appropriate binary decision problem. Here, we have adapted a simple binary task from Maass et al. (2002) in order to evaluate the performance of the LSM. The challenge was to distinguish spike train segments in a continuous data stream composed of two templates with identical rates (denoted X and Y in Figure 6.13A). In order to generate the input, we cut the template spike trains into segments of 50 ms duration. We then composed the spike sequence to be presented to the network by randomly picking a spike segment from either X or Y in each time window (see Figure 6.13 for a schematic). Additionally, we added spike timing jitter from a normal distribution with a standard deviation of $\sigma = 1\,\text{ms}$ to each spike.

For each experiment run, both for training and evaluation, the composed spike sequence was then streamed into the liquid. Tempotrons were given the liquid activity as input and trained to identify whether the segment within the previous time window originated from sequence X or Y. In a second attempt, we trained the tempotron to identify the origin of the pattern presented in the window at -100 to -150 ms (that is, the second to the last window). Not only did this task allow to determine the classification capabilities of the LSM, but it also put the liquid's fading memory to the test, as classification of a segment further back in time becomes increasingly difficult.

fading memory

The hardware implementation of the tempotron was not straightforward, due to several differences between the theoretically optimal model and the physical implementation of neurons and synapses on the Spikey chip.

Firstly, the theory assumes the same PSP kernel κ for each incoming spike, but this is not the case for Spikey, which features only conductance-based synapses. However, with the reversal potentials set at a large distance from the dynamic range of the membrane potential, COBA dynamics can be well approximated by the CUBA equations. Nevertheless, the asymmetric position of excitatory and inhibitory reversal potentials with respect to the mean membrane potential $\bar{u}$ needed compensation. In the CUBA approximation, this can be done straightforwardly by scaling all excitatory weights with $(\bar{u} - E_\mathrm{i}^\mathrm{rev})/(\bar{u} - E_\mathrm{e}^\mathrm{rev})$, where $\bar{u}$ corresponds to the mean neuron membrane voltage and $E_\mathrm{e}^\mathrm{rev}$ and $E_\mathrm{i}^\mathrm{rev}$ are the excitatory and inhibitory reversal potentials, respectively.

Secondly, the tempotron learning rule may also cause synapses to switch their sign. For a physical implementation of COBA synapses (be it biological or in silico), switching to a different reversal potential is not straightforward. For our Spikey tempotron, we chose the easiest, albeit non-optimal, solution of simply prohibiting such transitions.

Finally, the shunting of the membrane potential after the first spike of the tempotron, which is required by the theoretically optimal implementation, is also difficult, both in biology and on Spikey. However, (Gütig and Sompolinsky, 2006) themselves have already proposed simply neglecting this mechanism, which we also adopted for our Spikey implementation.

Even though the tempotron was robust against fixed-pattern noise due to on-chip learning, the liquid itself required a certain degree of tuning. Ideally, firing thresholds should be adjusted independently to optimize the memory capacity and avoid violations of the separation property. This can be done in software simulations, but not on Spikey, since hardware neurons share firing thresholds (see Table 3.1). Additionally, the learning curve $\alpha(n)$ was chosen individually for software and hardware due to the limited resolution of synaptic weights on the latter.

The results for our software and hardware implementations of the LSM described above are illustrated in Figure 6.13B. Both LSMs performed at around 90% classification correctness for the spike train segment that lied 50 ms to 100 ms in the past with respect to the end of the stimulus. The LSM can therefore be characterized as successful in identifying spatiotemporal patterns even when a small time lag (on the order of 100 ms) is present, due to the transient activity of the liquid. This capability declines, of course, as the time lag increases. For inputs lying twice as far in the past, performances dropped to chance level (50% for a binary task), independently of the simulation back-end.

This experiment is, indeed, only a rather simple one. In principle, this implementation allows a larger variety of tasks to be performed. For example, some work has been done on hand-written digit recognition with the very same setup on the Spikey chip (Jeltsch, 2014). It should also be noted that even without a liquid, our implementation of the tempotron (or populations thereof) represents a promising choice for a neuromorphic classifier, given its bandwidth-friendly sparse response and robustness against fixed-pattern noise.

6.3. Rate-Based Belief Propagation with LIF Neurons

Building on the belief propagation algorithms discussed in the previous section, we can now formulate a framework in which spiking neural networks can perform these computations. In particular, messages required by the SPA (Section 6.1.4) shall be represented by firing rates and the local computations in each factor will be carried out by liquid state machines. These studies are based on work by (Steimer et al., 2009) and were done in collaboration with Venelin Petkov and Dominik Schmidt. The results have also been previously reported in their Diploma thesis (Petkov, 2012) and internship report (Schmidt, 2012), respectively. We shall end this section with an important discussion of the scalability of this approach to more complex problems.

6.3.1. A Neural Implementation of Binary Forney Factor Graphs

binary RVs

We shall start with a further simplification of the FFGs discussed in Section 6.1.5 by restricting the sample space of the constituent RVs to $\{0,1\}$, i.e., by making them binary. Since a message over an edge x_{ij} should encode a probability distribution over the RV x_{ij}, it now suffices to represent $p(x_{ij}=1)$, since $p(x_{ij}=0)$ follows directly from the unitarity condition for p.[11] Messages therefore assume a somewhat simpler form

discrete-time messages in binary FFGs

$$\mu_{ij} = \sum_{\boldsymbol{x}_i \in \{0,1\}^m} f_i(x_{ij}=1, \boldsymbol{x}_i) \prod_{f_k \in \mathrm{ne}(f_i) \setminus f_j} p_{ki}(x_{ki}) \quad , \tag{6.37}$$

where $m = |\mathrm{ne}(f_i) \setminus f_j| = |\mathrm{ne} f_i - 1|$ represents the number of RVs that lie "behind" f_i and p_{ki} can be calculated from μ_{ki} according to the abovementioned unitarity condition:

$$p_{ki}(x_{ki}) = \begin{cases} \mu_{ki}, & \text{if } x_{ki} = 1 \\ 1 - \mu_{ki}, & \text{otherwise} \end{cases} \tag{6.38}$$

flooding schedule

Since a neural implementation of the SPA in FFGs will represent a dynamical system, time will start playing an important role. In the classical SPA, the order in which messages are transmitted is usually chosen to follow the so-called flooding schedule in discrete time steps. At each point in time, all factors which have received enough input messages with respect to an output edge will send an output message along that edge. In the case of systems evolving in continuous time, such as the neural networks we are going to construct, factors will pass on messages in all directions at all times. The messages from leaf nodes towards other nodes in the graph will be correct, since they simply encode priors over certain RVs (or posteriors, if they have been observed). All other messages will be incorrect at the onset of the algorithm, but will become increasingly correct as the information from the leaf nodes diffuses throughout the network. Instead of enforcing a particular schedule, we shall therefore merely require that the output messages converge towards their target values:

continuous-time messages in binary FFGs

$$\tau \dot{\mu}_{ij}(t) = -\mu_{ij}(t) + \frac{1}{Z(t)} \sum_{\boldsymbol{x}_i \in \{0,1\}^m} f_i(x_{ij}=1, \boldsymbol{x}_i) \prod_{f_k \in \mathrm{ne}(f_i) \setminus f_j} p_{ki}(x_{ki}, t-D) \quad , \tag{6.39}$$

[11] Since in FFGs edges symbolize RVs, we use the same notation for both.

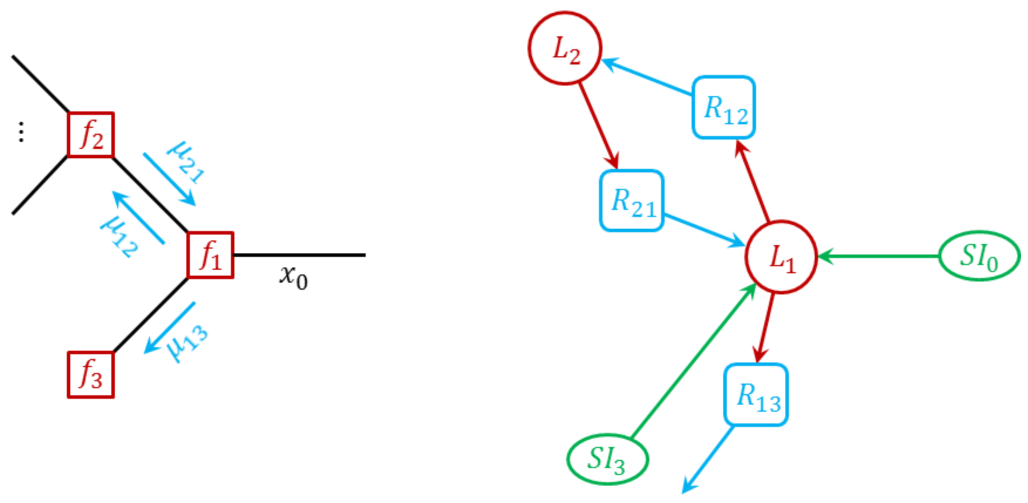


Figure 6.14.: Translation of an FFG to an LFG. **Left:** Example FFG with a main node f_1 (for which we describe the liquid/readout translation), a non-leaf node f_2, a leaf node f_3 and a half-edge representing an observed RV x_0. **Right:** Equivalent LFG. The main node is represented by a liquid L_1, with readouts that compute all necessary outgoing messages (here: R_{12} and R_{13} encode μ_{12} and μ_{13}, respectively). Non-leaf nodes such as f_2 themselves require to be implemented as liquids (here: L_2), with a readout R_{21} projecting to L_1, as well as receiving input from L_1 via R_{12}. Leaf nodes such as f_3 provide constant output messages and can therefore simply be replaced by Poisson (spike input) sources (here: SI_3). They do, however, connect to other nodes via RV edges, which also carry messages in the other direction, which we need to be able to read out in order to compute the probability distribution over the corresponding RVs. Therefore the readout R_{13} is still needed, although it does not project to anything else. Finally, half edges corresponding to observed RVs such as x_0 are also simply translated to Poisson sources (here:SI_0) that fire with either maximum or minimum rate, in order to encode $x_0 = 1$ or $x_0 = 0$, respectively. Evidently, a readout for observed RVs is superfluous.

where τ is a time constant that represents the intrinsic "reaction speed" of the system, $Z(t)$ is a normalization constant that ensures $0 \leq \mu_{ij}(t) \leq 1$ and D denotes the transmission delay within the system.

firing rates as messages

Now that messages are represented by single numbers (as opposed to probability vectors for RVs with more than 2 potential values), we can associate a population firing rate $r_{ij} \in [r_{\min}, r_{\max}]$ to each $\mu_{ij} \in [0, 1]$.[12] For simplicity, the mapping will be linear: $r_{ij} = \alpha\mu_{ij}$, with $\alpha = 90\,\text{Hz}$. The associated populations will be the readouts for liquids representing the individual factors f_i. The assumption is that a sufficiently complex liquid will be able to simultaneously calculate all the required messages $\mu_{ij} \forall f_j \in \text{ne}(f_i)$ and a readout R_{ij} can then be trained to extract the correct message μ_{ij} in the form of a population

[12] Note that multiple populations could, in principle, represent multivariate probability distributions as well.

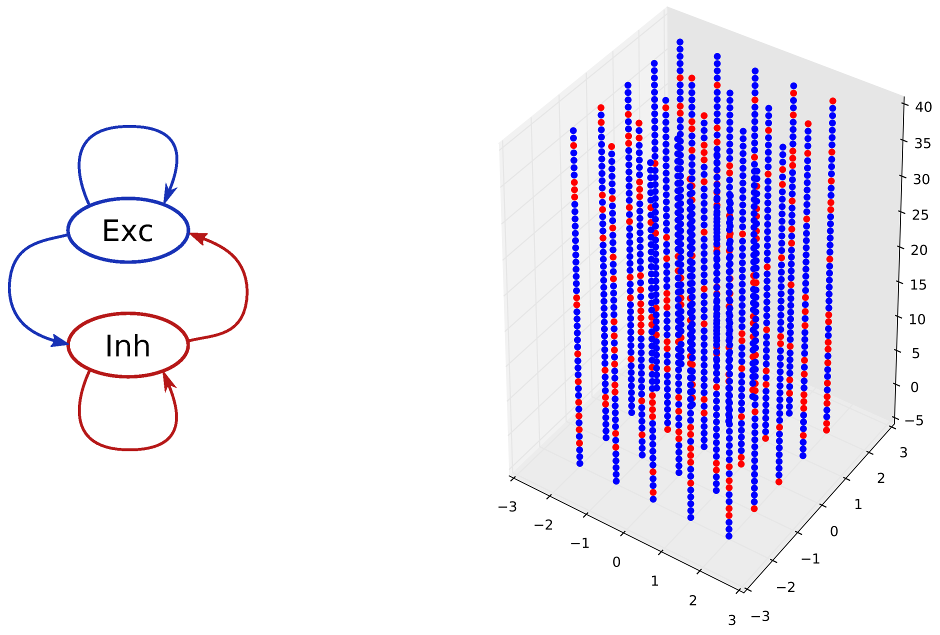


Figure 6.15.: Neuronal realization of liquids for the implementation of LFGs. **Left:** The liquid consists of two subpopulations of excitatory and inhibitory CUBA LIF neurons, respectively, which project onto themselves as well as each other. **Right:** Neurons are placed on a cuboid grid and 20% of them are randomly chosen as inhibitory. Figures taken from Petkov (2012).

liquid factor graph

firing rate r_{ij} and pass it on to the liquid representing the factor f_j. This defines the architecture of our liquid factor graph (LFG) implementation, an example of which can be found in Figure 6.14.

We have added the following simplifications in order to reduce the computational load on the LFG network:

- observed RVs are represented by homogeneous Poisson sources with firing rates that correspond to the observed value $r_{ij} \in \{r_{\min}, r_{\max}\}$;
- leaf nodes f_l are also represented by homogeneous Poisson sources with firing rates that encode the prior $r_l = \alpha p(x_{lj} = 1)$.

The only remaining nodes that need to be implemented as liquids are therefore those that represent factors of at least 2nd order, as well as those that must be introduced to translate arbitrary joint distributions to FFGs (i.e., equality nodes for copying RVs, see Section 6.1.5).

liquid architecture

The liquids themselves are implemented as seen in Figure 6.15, following the architecture suggested in Maass et al. (2002) and Steimer et al. (2009). Neurons are arranged on a cuboid lattice, with dimensions depending on the complexity of the factor. A random subpopulation of 20% of the liquid neurons are chosen as inhibitory and the remaining

80% as excitatory. The "small world" connectivity is implemented by a Gaussian stochastic rule, where each pair of neurons has a distance-dependent probability of being connected:

liquid connectivity

$$p(\boldsymbol{a}, \boldsymbol{b}) = C \exp\left(-\frac{\|\boldsymbol{a} - \boldsymbol{b}\|^2}{\lambda^2}\right) \quad , \tag{6.40}$$

where C represents a normalization factor and λ the density parameter.[13] The full set of generic parameters for the liquid pools can be found in Tables A.15 and A.16. We shall address application-specific parameters such as size and connectivity later on.

6.3.2. Training Readouts for Message Passing

The ultimate goal of the readout training is to define a set of weights $\boldsymbol{w}_{ij}$ between a liquid L_i and a readout R_{ij} such that the population average output rate $r_{ij}(t)$ of the readout correctly encodes the message $\mu_{ij}(t)$ at all times. For simplicity, we shall assume that each neuron in the readout population R_{ij} receives the same afferent weight vector $\boldsymbol{w}_{ij}$. Therefore, all readout neurons should fire with the same rate $r_{ij}(t)$ up to small deviations due to the variability in their parameters, which we shall discuss later.

If a readout neuron was a classical perceptron, its input weight vector $\boldsymbol{w}_{ij}$ would directly determine its output before binarization (which would be continuous and not spiking) via

classical perceptron I/O relation

$$r_{ij} = \boldsymbol{w}_{ij} \boldsymbol{r}_i + b_{ij} \quad , \tag{6.41}$$

where $\boldsymbol{r}_i$ represents the output vector of L_i and b_{ij} a bias that we shall henceforth neglect, since it can be provided by the liquid itself and included in $\boldsymbol{w}_{ij}$.[14] In our case, both the liquid and the readout have outputs that are non-continuous functions of time (spike trains are usually represented by sums of delta functions, see Equation 2.30). Furthermore, the output rate of a neuron is not equal to a weighted sum of its inputs.

We shall deal with these two issues by expressing the liquid output as a synaptic input vector $\boldsymbol{L_i}(t)$, which, at the same time, also represents the input vector for the readout. For our exponential synapse model, the liquid-to-readout synapses simply convolve the output spike train vector $\boldsymbol{\rho}(t)$ of the liquid with an exponential function $\exp\left(\frac{t-t_s}{\tau^{\text{syn}}}\right)$ (see Equations 2.47 and 2.50):

liquid output as synaptic input vector

$$\boldsymbol{L_i}(t) = \boldsymbol{\rho(t)} * \exp\left(-\frac{t-t_s}{\tau^{\text{syn}}}\right) = \begin{pmatrix} \sum\limits_{\text{spikes } s \text{ of liquid neuron } 1} \Theta(t-t_s) \exp\left(-\frac{t-t_s}{\tau^{\text{syn}}}\right) \\ \vdots \\ \sum\limits_{\text{spikes } s \text{ of liquid neuron } n} \Theta(t-t_s) \exp\left(-\frac{t-t_s}{\tau^{\text{syn}}}\right) \end{pmatrix} \tag{6.42}$$

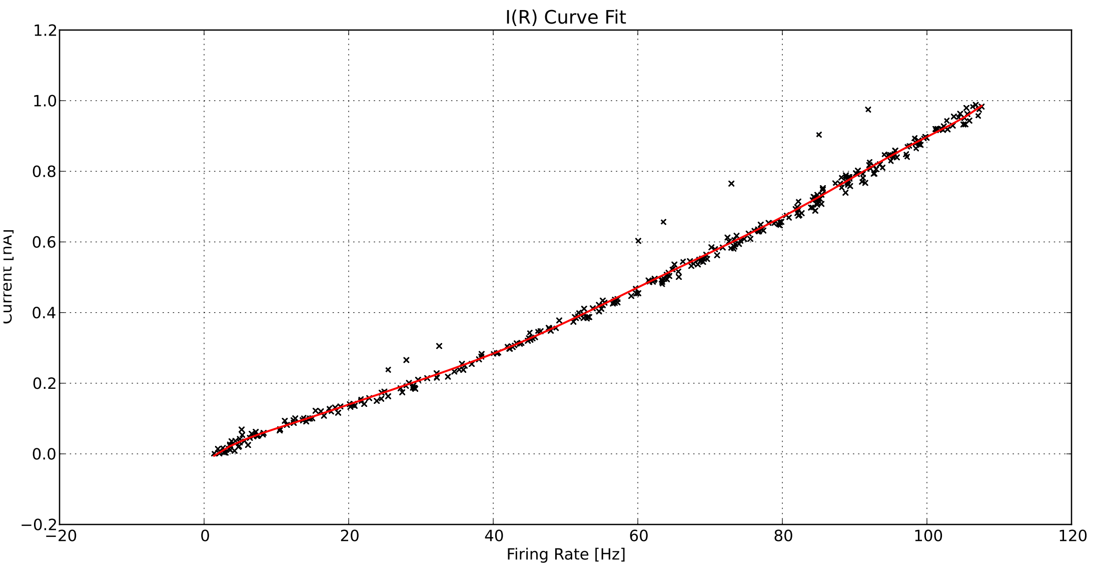
The synaptic input current of a readout can now be represented analogously to the continuous output case:

readout input current

$$I_{ij}^{\text{syn}} = \boldsymbol{w}_{ij} \boldsymbol{L}_i \quad . \tag{6.43}$$

[13] The density parameter λ gives the main contribution to the chaoticity of the liquid (see Section 6.2) and must be balanced carefully to keep the system on the edge of chaos.

[14] We use single indices (and vectors) to denote the output of liquids and double indices for the output of readouts.



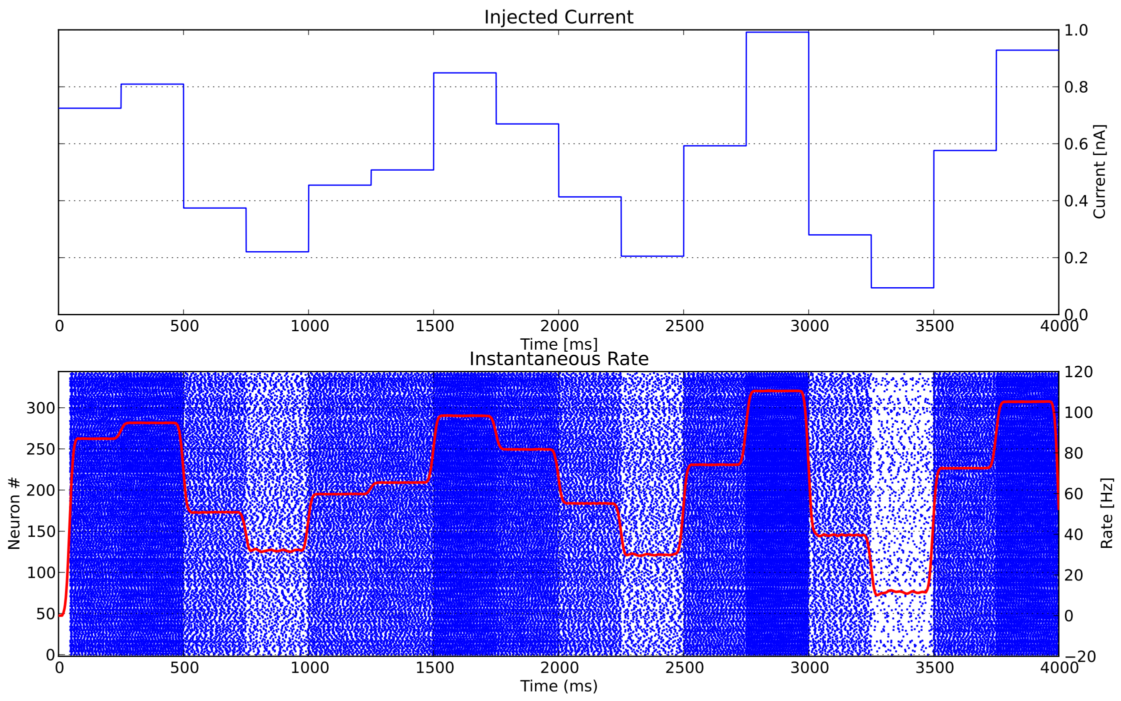


Figure 6.16.: *I*-*r* relationship of readout populations. **Top:** Measurement of the *I*-*r* dependency (black crosses) and polynomial fit (red curve). **Right:** Spike output of the readout for piecewise constant input current. Due to the randomly drawn threshold voltages, the output becomes asynchronous, whereas the Gaussian background causes irregularity, without changing the average output rate. Figures taken from Petkov (2012).

This still needs to be converted to an output rate. Although analytical approaches exist (see Section 6.5.3), we shall determine the relationship between I_{ij}^{syn} and r_{ij} empirically. To this end, each readout neuron receives a constant input current plus Gaussian white noise, as well as a random threshold, in order to avoid synchronous firing. The I-r curve is then measured by injecting the same input current I into all neurons of a readout population and then averaging their output rates, which are obtained by convolving their output spike trains with a Gaussian. The result of the measurement, along with a polynomial fit of the data, can be found in Figure 6.16.

I-r curve

We can now write down the full conversion equation from the liquid output to the encoded message:

output message of readout

$$\begin{aligned}\mu_{ij} &= \alpha^{-1} r_{ij}(I_{ij}^{\mathrm{syn}}) \\ &= \alpha^{-1} f(I_{ij}^{\mathrm{syn}}) \\ &= \alpha^{-1} f(\boldsymbol{w}_{ij}\boldsymbol{L}_i) \quad , \end{aligned} \tag{6.44}$$

where $f = r(I)$ represents the measured I-r dependency. We could now train the liquid-to-readout weight vectors $\boldsymbol{w}_{ij}$ by repeating the following simple procedure:

readout training

1. provide the liquid L_i with a training stimulus (which represents a set of input messages $\boldsymbol{\mu}_{ki}$);
2. compute the correct outgoing message μ_{ij}
3. measure the output message of the readout $\tilde{\mu_{ij}} = \alpha r_{ij}$;
4. modify the liquid-to-readout projection weights $\boldsymbol{w}_{ij}$ in order to reduce the distance $\|\tilde{\mu}_{ij} - \mu_{ij}\|$.

However, instead of a time-consuming iteration, we can do the above in a single step if we consider the inverse of Equation 6.44:

$$f^{-1}(\alpha\mu_{ij}) = \boldsymbol{w}_{ij}\boldsymbol{L}_i(\boldsymbol{\mu}_{ki}) \quad . \tag{6.45}$$

By providing a continuous input stream $\boldsymbol{\mu}_{ki}(t)$ which covers all relevant input message configurations and requiring that the above equation must hold at all times t, we obtain a system of linear equations. The least-squares-problem of minimizing $\|\tilde{\mu}_{ij} - \mu_{ij}\|$, is equivalent to performing the linear regression of $\mu_{ij}(t)$ to $\boldsymbol{L}_i(t)$ and is thereby reduced to finding the solution to the above linear system, which has a well-known analytical expression:

least-squares problem, linear regression

$$\boldsymbol{w} = \left(\tilde{\boldsymbol{L}}_i\tilde{\boldsymbol{L}}_i^T\right)^{-1}\tilde{\boldsymbol{L}}_i^T\boldsymbol{f}^{-1} \quad , \tag{6.46}$$

where we have used the vectors $\tilde{\boldsymbol{L}}_i$ and $\boldsymbol{f}^{-1}$ to represent the time series $\boldsymbol{L}_i(\boldsymbol{\mu}_{ki}(t))$ and $f^{-1}(\alpha\mu_{ij}(t))$, respectively.

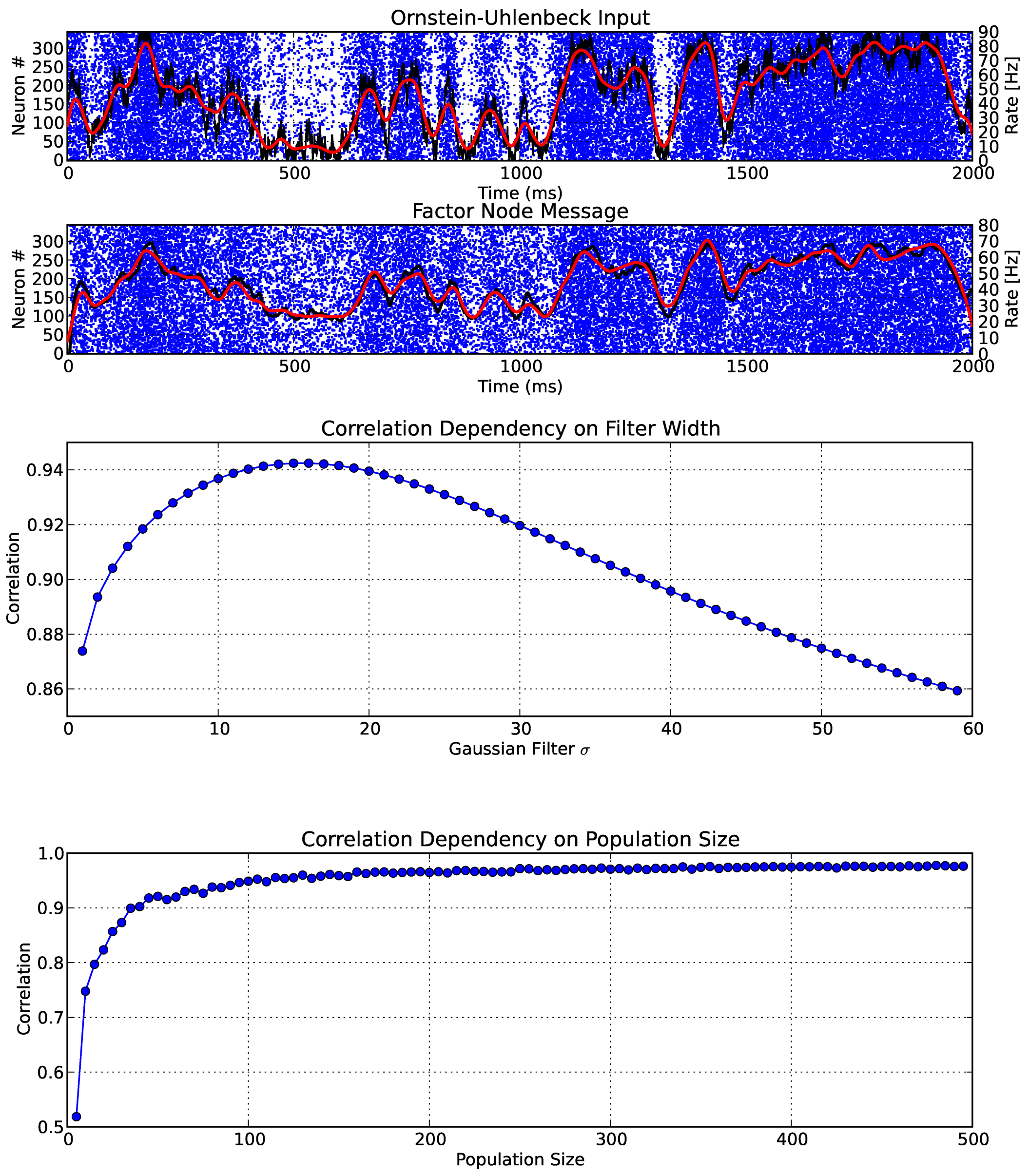


Figure 6.17.: Readout training for LFGs. **Top:** Example training input for a liquid and output of the readout after training (spike trains shown in blue). The underlying input rate process and correct output rate are shown in black. The filtered input rate and the measured output rate are shown in red. The post-training output agrees well with the analytically calculated correct output. **Middle:** Influence of the Gaussian kernel width (for converting spike trains to firing rates) on the quality of the training result, which was measured as the correlation between the correct and measured output rate of the readout. The optimum lies around 15 ms. **Bottom:** Influence of the readout population size on the quality of the training result. As expected, a larger population size helps smooth out errors produced by single neurons. Above 150 neurons, the correlation between correct and measured output rates does not increase significantly any more. Figure taken from Petkov (2012).

The only remaining issues are of practical nature and concern the definition of the training set $\boldsymbol{\mu}_{ki}(t)$ and the optimization of the readout size. Since we want to keep our training as generic as possible, we should keep the training messages non-specific, i.e., allow them to cover the entire possible range $[0, 1]$. In order to achieve that, we have generated spike trains as inhomogeneous Poisson processes with rates given by an Ornstein-Uhlenbeck process with reflective bounds (more on this process can be found in Section 6.5.2), which we have converted to rates and then messages by convolution with a Gaussian kernel (Figure 6.17, top panel). The optimization of its width was done experimentally, as can be seen in the middle panel of Figure 6.17.

training input

In order to optimize the readout population size, we have trained readouts of varying size and then measured the correlation between their output messages and the analytically calculated correct messages after training. As expected, the correctness of the output is a monotonically increasing function of the readout size. We have used a size of 343 neurons to match the study by Steimer et al. (2009), but a reduction by a factor of up to 2 would still be feasible (Figure 6.17, bottom panel) and could be interesting for limited-size neuromorphic systems.

readout size

6.3.3. Implementation of the Knill-Kersten Illusion

We will now apply the LFG framework to a well-known psychophysical inference problem – the Knill-Kersten illusion (Knill and Kersten, 1991). The illusion consists in the following cognitive phenomenon: the perceived contour of an object appears to alter an observer's judgement about its reflectance, despite the shading in the presented image remaining unchanged, as seen in the left panel of Figure 6.18. More specifically, the presented image shows two adjacent identical cylinders and two adjacent identical cubes with a sawtooth-like shading profile (the left and right objects are shaded identically, being light gray on their left side and dark gray on their right). Despite the shading profile of the cubes being identical to the one of the cylinders, most observers perceive the cylinders as being identical, but the left cube as being darker than the right one. The explanation of this illusion lies in the fact that no single source of light can explain the shading/luminance profile of the cubes (in particular, the jump in the perceived shading at their boundary), which causes the brain to (wrongly) infer that the reflectance of the cubes must be different. A round shape, however, explains away the shading jump under the assumption that the objects are illuminated by a light source on the left side of the objects.[15]

Knill-Kersten illusion

As already suggested by the above explanation of the Knill-Kersten illusion, the problem can be expressed as a BN with 4 binary RVs. Two RVs can not be directly observed, namely the 3D object shape O ($0 \equiv$cuboid or $1 \equiv$cylindric) and its surface reflectance R ($1 \equiv$uniform or $0 \equiv$not), but they causally determine the 2D contour C ($0 \equiv$straight or $1 \equiv$round) and the shading profile S ($0 \equiv$sawtooth-shaped or $1 \equiv$otherwise), which are observable RVS. The right panel of Figure 6.18 shows the corresponding BN.

Knill-Kersten BN

The joint distribution of the Knill-Kersten BN reads:

$$P(S, R, O, C) = P_1(S|R, O) P_2(C|O) P_3(R) P_4(O) \quad . \tag{6.47}$$

[15] The assumption of a single light source makes sense from an evolutionary point of view, since the brain has evolved over millions of years in a world where the single most dominant source of light was the sun.

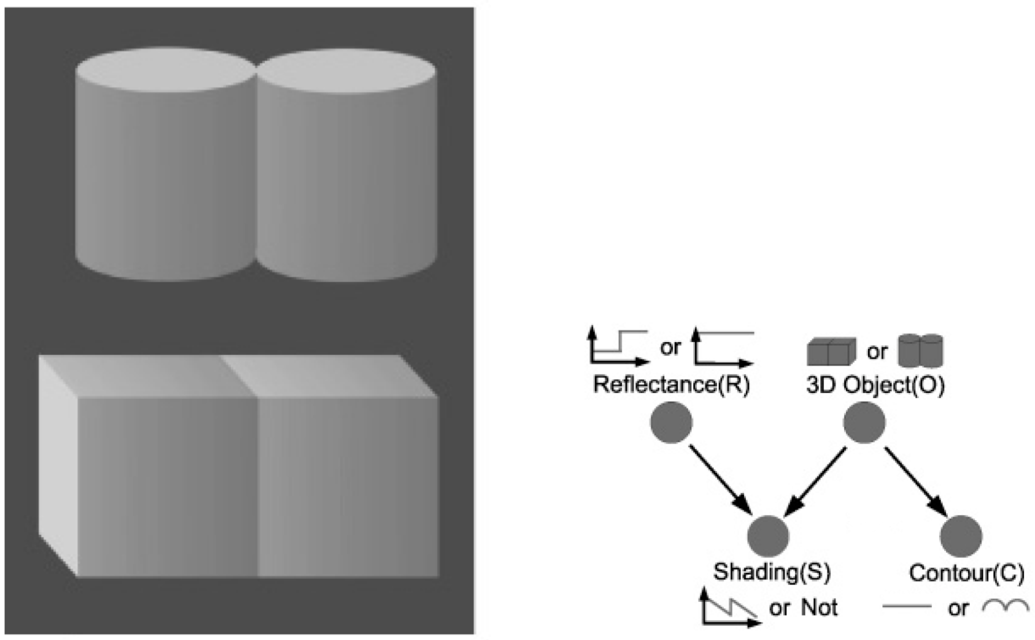


Figure 6.18.: Knill-Kersten inference problem. **Left:** Image producing the optical illusion. Although the shading profile of the two cylinders is identical to the one of the two cubes, observers perceive the cylinders as identical, but the left cube as darker than the right one. The reason for this illusion is that, under the assumption of a single light source, the inferred 3D shape of the cylinders explains away the observed shading profile, while the flat surface of the adjacent cubes does not, causing the brain to perceive them as different. **Right:** Representation as a Bayesian inference problem and associated binary BN. The perceived (observed) 2D contour C depends on the (unknown) 3D shape O of the object. The perceived shading profile S depends on both the 3D shape O and the reflectance R. Figures taken from Petkov (2012).

The factor P_1 encodes the fact that a cuboid shape with a reflectance jump and a cylindric shape with uniform reflectance both cause a sawtooth-shaped shading profile, i.e., $P_1(S = 0|O = 0, R = 0)$ and $P_1(S = 0|O = 1, R = 1)$ are both high. The factor P_2 simply encodes the correspondence between a cuboid/cylindric 3D shape and a straight/round 2D contour, i.e., $P_2(C = 0|O = 0)$ and $P_2(C = 1|O = 1)$ are both high. The exact numerical values of the above factors for all possible combinations of their arguments are given in Tables A.17 and A.18. The prior expectance of various reflectances and shapes, i.e., $P_3(R)$ and $P_4(O)$, was not fixed and will later be varied in order to evaluate the performance of our neural implementation.

The translation of the BN to an FFG is rather straightforward and can be seen in the left panel of Figure 6.19. The RV O appears in the three factors P_1, P_2 and P_4. We therefore require an equality node that creates three copies, O, O' and O'', and connects via corresponding edges to factor nodes P_1, P_2 and P_4, respectively. P_1 and P_2 further depend on the observed RVs S and C and therefore receive corresponding incoming half-edges.

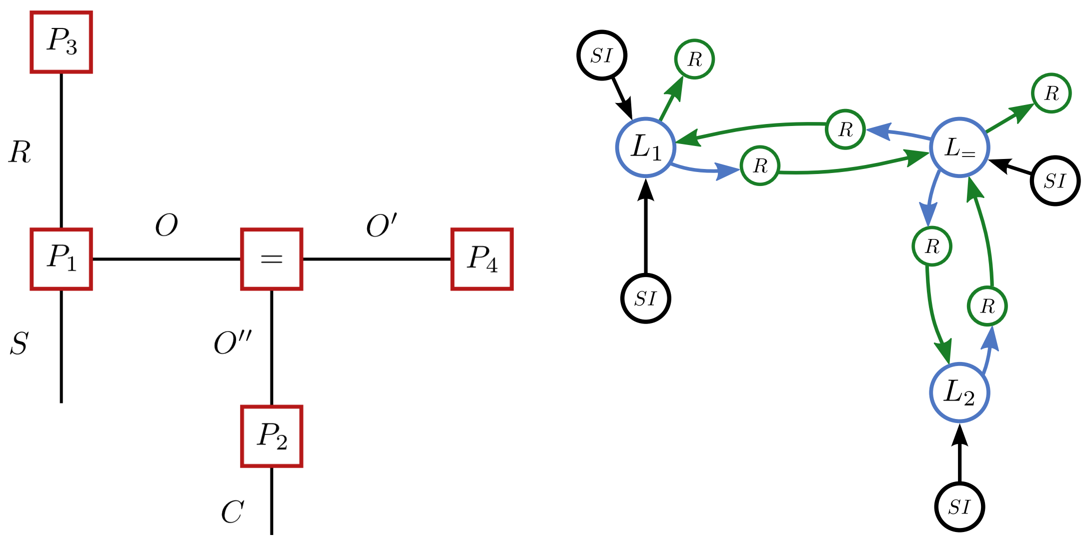


Figure 6.19.: Translation of the Knill-Kersten inference problem to a neural network. **Left:** FFG representation. Each factor in the joint distribution has its arguments represented as incident edges. Since the RV O is an argument of multiple factors, an equality factor is required to create an appropriate number of copies. **Right:** LFG representation. Only factors that depend on two or more variables need to be implemented by liquids, in this case: P_1, P_2 and $P_=$. Leaf factors and observed variables are implemented as Poisson inputs. Figures taken from Petkov (2012).

Finally, P_1 and P_3 both depend on R and are therefore connected by a corresponding edge. The translation of this FFG to an LFG follows the rules laid out in Section 6.3.1 and can be seen in the right panel of Figure 6.19.

After training (following the method described in 6.3.2, we have evaluated the performance of our neural implementation of the Knill-Kersten inference problem by providing the network with observations of S and C as well as various priors for R and O. For each trial, we have therefore randomly generated the messages $\mu_{S\to P_1}$, $\mu_{C\to P_2}$, $\mu_{P_3\to P_1}$ and $\mu_{P_4\to P_=}$ and compared all the messages computed by the network to the analytically calculated correct messages. In order to provide a reference, we have also generated random messages and compared them to the correct ones. Figure 6.20 shows the correctness of the messages computed by the LFG (colored histograms in the foreground) compared to that of random messages (black histograms in the background). The correctness of a message is given by the difference between its value and the value of the corresponding correct message, therefore a pronounced peak around 0 represents a good result.

Overall, we note that our neural implementation performs well, but not completely without drawbacks. In particular, the quality of the messages deteriorates as they pass through an increasing number of computational stages. The "most correct" message is $\mu_{2\to=}$ (yellow) since it is emitted by a node that receives only two inputs, one of which ($\mu_{C\to P_2}$) is precomputed analytically and therefore correct by definition. We find a similarly good performance for $\mu_{=\to 1}$ (red), which depends on a precomputed message ($\mu_{P_4\to P_=}$) and $\mu_{2\to=}$ (yellow), which, as we have already seen, has high accuracy as well. The other messages

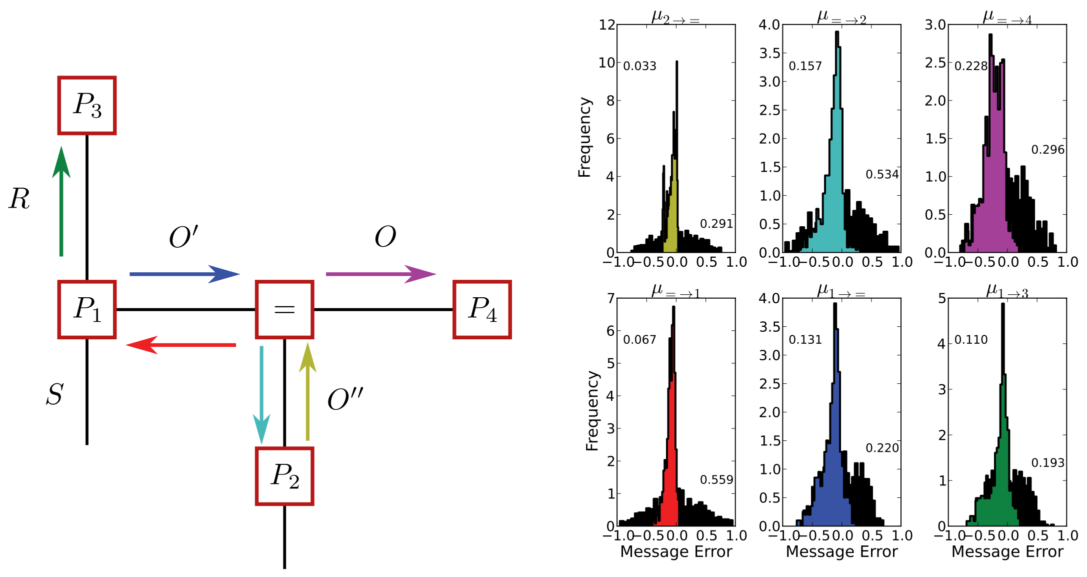


Figure 6.20.: Performance of the LFG implementing the Knill-Kersten inference problem. **Left:** Color coding of the transmitted messages computed by the network. **Right:** Correctness of the messages computed by the network. The six color-coded messages are measured for various inputs and compared to the corresponding, analytically computed, correct messages. The difference $\mu_{\text{correct}} - \mu_{\text{measured}}$ is plotted as a colored histogram. For comparison, random messages are generated for each trial, and the difference $\mu_{\text{correct}} - \mu_{\text{random}}$ is plotted as a black histogram in the background. The numbers attached to each curve are the trial-averaged Kullback-Leibler divergences (in bits) $D_{\text{KL}}(\mu_{\text{correct}} \parallel \mu_{\text{measured}})$ and $D_{\text{KL}}(\mu_{\text{correct}} \parallel \mu_{\text{random}})$. Figures taken from Petkov (2012).

require more complex computations, as well as depending on other, potentially erroneous messages, and therefore suffer from elevated error rates. We discuss the consequences of this observation in the following section.

6.3.4. Discussion: Pros, Cons and Ideas for a Neuromorphic Implementation

The quantitative experimental results of the previous section are, of course, not easily generalizable to other inference problems, but they do highlight some conceptual problems with the practicability of the LFG approach to more complicated scenarios.

liquid/read-out size As shown in Table A.19, the liquid populations we used were rather large. This was, indeed, necessary, as smaller liquids appear to be unable to perform the required computations. The readout populations could have been reduced in size, but not below ca. 100 neurons without significant performance losses (see Figure 6.17, bottom panel). In total, we end up with a neural network of over 6000 neurons for inference in a probability space over only four binary variables, with a total of 16 possible states. Even if there is room for improvement in the training methods, choice of parameters etc., it is unlikely to significantly reduce this gap of three orders of magnitude between the number of represented

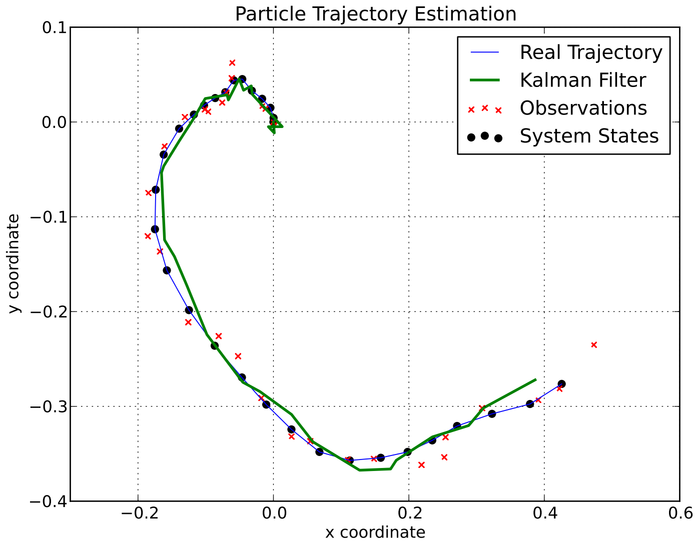


Figure 6.21.: Kalman filter example. A particle in a 2D space (blue trajectory) experiences a random acceleration at each time step. Measurements of the particle position at varoius points along its trajectory (red crosses) are noisy. With each update, the Kalman filter predicts a trajectory (green curve) based on its previous prediction and the new measurement. Figure taken from Petkov (2012).

RVs and the total number of neurons in the network.

The second issue concerns the accuracy of this approach and was already touched upon in our explanation of the experimental results in the previous section. Each additional computation stage (performed by a liquid and its corresponding readout) necessarily induces errors, since the liquid is not a Turing machine. These errors accumulate as the messages pass through an increasing number of factors; already after having passed through two liquids/readouts, the computed messages exhibit rather large deviations from the correct values. Given the described training method, the liquids themselves have no means of performing error correction, since they must react to the distorted input messages as if they were correct, as they cannot discern between correct and incorrect messages. The propagation of errors would be even further exacerbated by an imperfect physical substrate, as is, for example, the case for analog-VLSI neuromorphic devices.

error propagation

Altogether, we must conclude that although the LFG implementation proposed by Steimer et al. (2009) and reproduced by us is interesting and instructive, it appears, in general, unfeasible for more complicated inference problems and therefore both biologically unlikely as well as technically impractical. However, it might be applicable to situations where the resulting LFG is shallow, i.e., where at each stage of computation, the factors receive external observations, which effectively serve the purpose of error correction. Practical problems of this nature exist, and are represented by the general class of so-called hidden Markov models.

hidden Markov models

In particular, consider a chain of discrete-time observations $z(t)$ of an RV whose exact value $x(t)$ remains hidden. The evolution of the RV in discrete time has the Markov property, i.e., $p\,[x(t)|x(t-1),\ldots,x(1)] = p\,[x(t)|x(t-1)]$. A good prediction of $x(t)$ will take into account both the (noisy) observation $z(t)$, as well as the previous prediction of $x(t-1)$. A well-known algorithm that performs these calculations is the so-called Kalman filter. It is often used for tracking moving objects, from missiles detected by radar to particles in gas detectors (Figure 6.21). Figure 6.22 shows a possible implementation of the Kalman filter as an LFG. Since at each stage of the calculation the prediction is updated with a measurement, this algorithm could lend itself as a candidate for a practical LFG implementation.

Kalman filter

Figure 6.22.: Implementation of the discrete-time Kalman filter in a spiking neural network.
a) Representation as an FG. The position x_k of a particle in flight is observed through a noisy device, yielding the observation z_k, where the index k denotes the discrete time. The trajectory of the particle is Markovian, i.e., at any point in time, its position x_k depends only on its previous position x_{k-1}. The noisy observation is governed by so-called emission probability factors $g_k(x_k, z_k) := p(z_k|x_k)$ and the movement is determined by so-called transitional probability factors $f_k(x_k, x_{k-1}) := p(x_k|x_{k-1})$. The resulting factor graph represents the joint probability

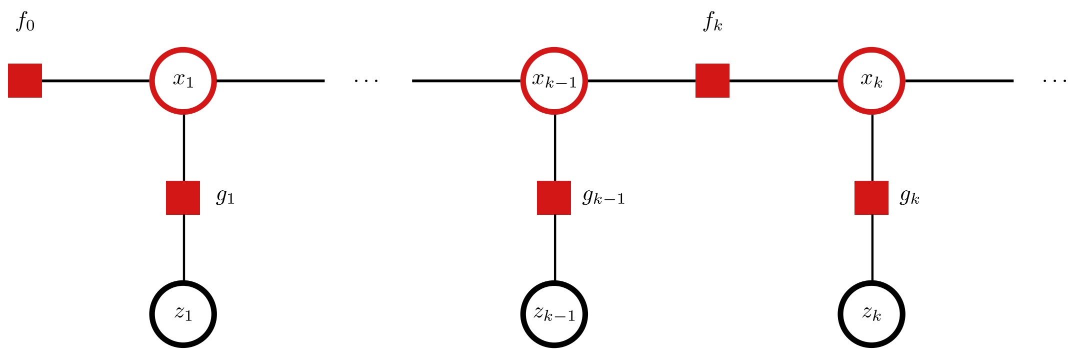
$$p(x_1,\ldots,x_n) = p(x_1)\prod_{k=2}^{n} p(x_k|x_{k-1})\prod_{k=1}^{n} p(z_k|x_k) \quad . \tag{6.48}$$

The Kalman filter algorithm is formally equivalent to the SPA in this FG.
b) Translation to an FFG-compatible graph. The factors f_k and g_k are absorbed into single factors over 3 RVs:

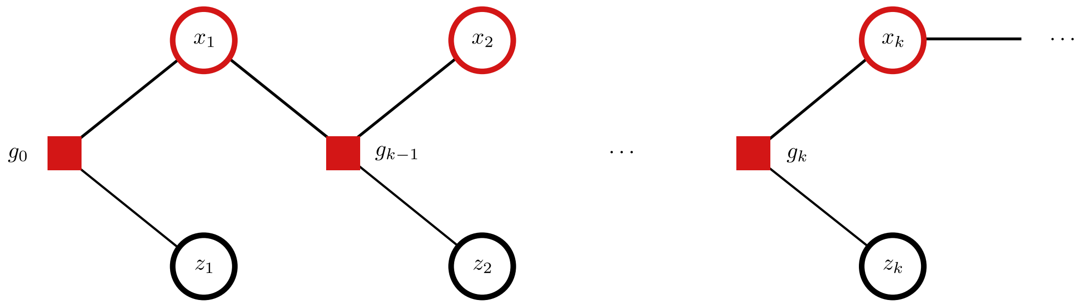
$$g_k^{\text{new}}(x_{k-1}, x_k, z_k) := f_k(x_k, x_{k-1})g_k(x_k, z_k) \quad . \tag{6.49}$$

c) Translation to an FFG. Due to the above redefinition of factors, all variable nodes are adjacent to at most two factor nodes. Therefore, they can be simply eliminated and replaced by corresponding edges.
d) Translation to an LFG following the rules described in Section 6.3.1.
Figure taken from Petkov (2012).

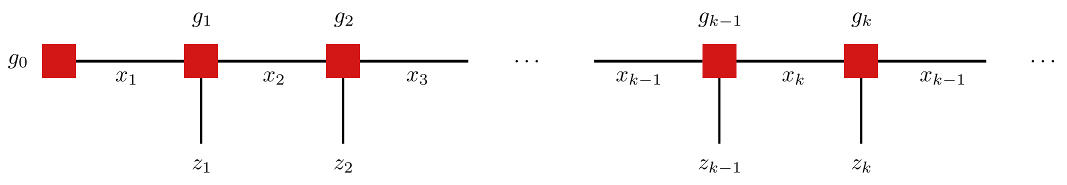
a)


f_0
x_1
$\cdots$
x_{k-1}
f_k
x_k
$\cdots$
g_1
g_{k-1}
g_k
z_1
z_{k-1}
z_k

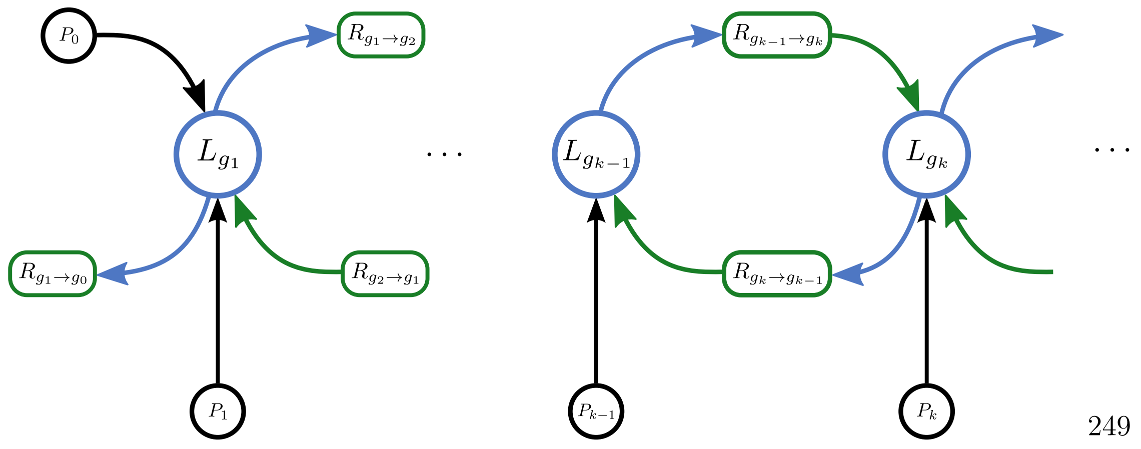
b)


x_1
x_2
x_k
$\cdots$
g_0
g_{k-1}
$\cdots$
g_k
z_1
z_2
z_k

c)


g_1
g_2
g_{k-1}
g_k
g_0
$\cdots$
$\cdots$
x_1
x_2
x_3
x_{k-1}
x_k
x_{k-1}
z_1
z_2
z_{k-1}
z_k

d)


P_0
$R_{g_1\to g_2}$
$R_{g_{k-1}\to g_k}$
L_{g_1}
$\cdots$
$L_{g_{k-1}}$
L_{g_k}
$\cdots$
$R_{g_1\to g_0}$
$R_{g_2\to g_1}$
$R_{g_k\to g_{k-1}}$
P_1
P_{k-1}
P_k

6.4. Spike-Based Stochastic Inference

In the previous section, we have presented a neural architecture that is in principle capable of performing analytical inference in arbitrary probability spaces over binary RVs. The main drawbacks of the approach were the comparatively large network size and the susceptibility of the information encoded by the network to local noise. In the following, we shall present a fundamentally different concept for stochastic sample-based inference in neural networks. Single binary RVs will be represented by single neurons, whereas temporal noise will be explicitly exploited rather than being regarded as a nuisance. This noise will drive the network state to sample from a target probability distribution, and, as we shall see, implicitly enable it to perform Bayesian inference in the represented probability spaces.

We will approach this topic gradually, starting with a very simple model, and incrementally extend it to finally obtain recurrent networks of LIF neurons which can be trained as both generative and discriminative models for complicated, state of the art pattern recognition problems. We shall start by describing an abstract neural network model with relatively simple and deterministic dynamics – the Hopfield network – in Section 6.4.1. This model can be, rather naturally, enhanced to form a stochastic network – the Boltzmann Machine – the dynamics of which we describe within the framework of Markov chain Monte Carlo sampling in Section 6.4.2. How these dynamics can be mapped to initially non-Markovian dynamics of refractory neurons is the subject of Section 6.4.3. This still quite abstract model is, finally, mapped to the biologically realistic microscopic dynamics of LIF neurons in Section 6.5.

The detailed understanding of noisy LIF dynamics in the context of neural sampling from Boltzmann distributions has enabled some intriguing experimental results. In Section 6.8, we describe the first implementation of neural sampling with LIF neurons (short: LIF sampling) on analog neuromorphic hardware. In Section 6.6.2.2, we show how these LIF-based Boltzmann machines can then be trained with well-known machine learning algorithms to represent and classify various benchmark datasets. Finally, in Section 6.7, we extend our LIF sampling framework from Boltzmann distributions to arbitrary distributions over spaces of binary RVs.

The novel concepts described in this section are the result of an ongoing collaboration and have already been the subject of several publications and theses. In particular, the architecture and theory of LIF sampling networks were developed together with Johannes Bill and Ilja Bytschok (Petrovici et al., 2013), the training for handwritten digit recognition was done together with Luziwei Leng (Leng, 2014) and Marco Roth (Roth, 2014), the experiments on neuromorphic hardware were performed together with David Stöckel (Stöckel, 2015) (and are the result of a long but highly instructive chain of trial and error, to which Johannes Bill, Marc-Olivier Schwartz, Alexander Kononov and Ilja Bytschok have contributed significantly), and the extension to arbitrary spaces was designed and implemented together with Dimitri Probst (Probst, 2014; Probst et al., 2015).

6.4.1. Hopfield Networks

Hopfield network

The Hopfield network, named after its inventor John Hopfield (see, in particular, Hopfield, 1982), is an abstract neural network model based on theoretical microscopic models of magnetic solids. By design, it shares many aspects with the Edwards-Anderson model of spin glasses, which itself borrows the fundamental structure of the Ising model of magnetism in statistical mechanics.

A Hopfield network is composed of simple binary units $z_k \in \{-1, 1\}$. The interaction strength between the units is given by a zero-diagonal symmetric real-valued matrix, i.e.,

$$w_{ij} = w_{ji} \in \mathbb{R} \quad , \tag{6.50}$$

$$w_{ii} = 0 \quad . \tag{6.51}$$

Each unit has an associated potential u_k which represents the sum of "forces" it experiences from interacting with all other units in the network:

$$u_k = \sum_{i=1}^{n} w_{ik} z_i \quad . \tag{6.52}$$

linear threshold perceptron

The units are, effectively, linear threshold perceptrons[16], i.e., they "activate" if their associated potential surpasses some fixed threshold θ_k:

$$z_k = \begin{cases} +1 & \text{if } u_k = \sum_i w_{ik} z_i \geq \theta_k \quad , \\ -1 & \text{otherwise} \quad . \end{cases} \tag{6.53}$$

The temporal dynamics of the network vary between implementations. The state of individual units may be updated sequentially (in a predefined or randomized oder) or simultaneously in discrete time. While the latter is a more biologically realistic scenario, the update schedule does not affect the ensemble behavior significantly.

For each state $\boldsymbol{z}$ of the network, we can define an energy function

energy function of a Hopfield network

$$E(\boldsymbol{z}) = -\frac{1}{2} \boldsymbol{z}^T \boldsymbol{W} \boldsymbol{z} + \boldsymbol{z}^T \boldsymbol{\theta} \quad , \tag{6.54}$$

where $\boldsymbol{W}$ represents the weight matrix and $\boldsymbol{\theta}$ the threshold vector. From a physics perspective, it is quite intuitive that the system prefers state with lower energy. Consider a pair of units (z_i, z_j) with a positive interaction $w_{ij} \geq 0$. If one of them is active (e.g., $z_i = 1$) at some point in time, it shifts the other's potential upwards (Equation 6.52), making it more likely to be in the active state as well. If one of them is inactive, (e.g., $z_i = -1$), the potential shift for the other one will be negative, increasing the likelihood of it being inactive as well. Therefore, a positive weight encodes a higher likelihood for symmetric states $(1, 1)$ and $(-1, -1)$. Following the same line of thought, we can easily see how a negative weight leads to a higher likelihood for the antisymmetric states $(1, -1)$ and $(-1, 1)$. If we now calculate the product $z_i w_{ij} z_j$ for all four preferred configurations of a pair of units given their interaction weight, we find that it is always positive – and therefore has a negative contribution to the energy function defined in Equation 6.54. As should indeed be the case with physical systems, the Hopfield network therefore prefers states with lower energies.

gradient descent

Since the dynamics of the ensemble follow a negative energy gradient, they effectively implement gradient descent on the energy function. Therefore – as is usually the case for gradient descent – a Hopfield network will almost surely converge towards a local (and not

[16] A.k.a. McCulloch-Pitts neurons with Heaviside transfer functions.

necessarily global!) energy minimum. Since the position of the energy minima in state space can be controlled by $\boldsymbol{W}$ and $\boldsymbol{\theta}$, and these parameters can be trained by various (Hebbian) algorithms, the Hopfield network lends itself as a simple model of memory storage.

We need to point out that the likelihood-of-particular-states-argument given above was only for intuitive support. The dynamics of the Hopfield network are entirely deterministic. In the following section, we will transform this deterministic model into a stochastic one and give the likelihood of states a formally sound, quantitative measure.

deterministic dynamics

Until now, we have used $z_k \in \Omega_k = \{-1, 1\}$ as possible values of our binary RVs. There is no particular reason behind this choice other than the historical origins of the Hopfield model. In the following, we shall switch to $\Omega_k^* = \{0, 1\}$, which, as we shall see later, is more intuitive in the context of spiking neurons. Nevertheless, the joint distribution should remain unchanged under the mapping $z_k = -1 \rightarrow z_k^* = 0$ and $z_k = 1 \rightarrow z_k^* = 1$. This can be ensured with appropriate transformations of $\boldsymbol{W}$ and $\boldsymbol{\theta}$ (for $\Omega_k = \{-1, 1\}$) to $\boldsymbol{W}^*$ and $\boldsymbol{\theta}^*$ (for $\Omega_k^* = \{0, 1\}$):

$z_k \in \{-1, 1\} \rightarrow \{0, 1\}$ mapping

$$\boldsymbol{\theta}^* = 2\boldsymbol{\theta} + 2\boldsymbol{W}\mathbf{1} \quad \text{and} \tag{6.55}$$

$$\boldsymbol{W}^* = 4 * \boldsymbol{W} \quad , \tag{6.56}$$

where $\mathbf{1}$ represents the vector $(1, \ldots, 1)^T$.

6.4.2. Boltzmann Machines and Markov Chain Monte-Carlo Sampling

In Section 6.3, we have already described a network model which performs inference in probability spaces over binary RVs. It is clear that, in order to be able to perform the necessary computations, the network structure must somehow store the information about the joint distribution. Indeed, the subdivision in liquid-state-machine submodules explicitly represents the factorization of the joint probability, whereas the connection matrix of each liquid and its surrounding readouts implicitly represents the functional form of the implemented factor. More importantly however, the network also encodes the computed (marginal) probabilities explicitly, as firing rates. This is, of course, but one of many ways to encode distributions, which incidentally appears somewhat at odds with the observations we have made regarding the ambiguous images from Figure 6.1. If the brain was indeed explicitly encoding probabilities, then there would be no perceptual jumps between the two perceived motifs; since the visual input remains unchanged and the prior can be safely assumed as constant, the posterior distribution would have to be constant as well.

An alternative hypothesis is that, at any time, the state of some relevant ensemble of neurons in the brain explicitly encodes a possible representation of the underlying physical object in the ambiguous image. In this scenario, the network activity would explicitly encode states in the sample space and not probabilities. Driven by some yet-to-be-defined dynamical process, the state of this neural ensemble could then jump across the associated probability landscape and sample from the posterior distribution (see the left panel of Figure 6.23 for an illustration). Assuming that the two possible percepts correspond to two high-probability modes of the posterior, jumping from one of these states to the other would be the direct dynamical correlate of the observed perceptual switches. Over the following sections, we shall incrementally build such "sampling neural networks".

Let us return to the deterministic Hopfield model we have described in the previous section. We shall start by removing the threshold θ_k and replace it with a bias b_k that serves a similar functional purpose:

unit potentials in a BM

$$u_k = \sum_{i=1}^{n} w_{ik} z_i + b_k \quad . \tag{6.57}$$

Assuming that u_k encodes the likelihood of a unit to have the value $z_k = 1$, then a positive b_k increases this likelihood in a similar fashion as a smaller (negative) θ_k would. We must therefore redefine the energy function as

energy function of a BM

$$E(\boldsymbol{z}) = -\frac{1}{2}\boldsymbol{z}^T \boldsymbol{W} \boldsymbol{z} - \boldsymbol{z}^T \boldsymbol{b} \quad . \tag{6.58}$$

Note the sign change in the second term, as a large bias has the opposite effect of a high threshold.

We now reverse our physics argument from the joint distribution of MRFs (see Equations 6.15-6.17). If we interpret the state $\boldsymbol{z}$ of our network as a possible microstate in a canonical ensemble, then we can assign it a probability that depends on its energy:

joint distribution and partition function of a BM

$$p(\boldsymbol{z}) = \frac{1}{Z}\exp[-E(\boldsymbol{z})] = \frac{1}{Z}\exp\left[\frac{1}{2}\boldsymbol{z}^T \boldsymbol{W} \boldsymbol{z} + \boldsymbol{z}^T \boldsymbol{b}\right] \quad , \tag{6.59}$$

where Z represents a normalization factor and is given by

$$Z = \sum_{\boldsymbol{z}} \exp[-E(\boldsymbol{z})] = \sum_{\boldsymbol{z}} \exp\left[\frac{1}{2}\boldsymbol{z}^T \boldsymbol{W} \boldsymbol{z} + \boldsymbol{z}^T \boldsymbol{b}\right] \quad . \tag{6.60}$$

We note that these equations are formally equivalent to the Ising model with unit inverse temperature $\beta = 1/k_B T = 1$.[17] As its target distribution $p(\boldsymbol{z})$ is a Boltzmann distribution, such a network is aptly called a Boltzmann machine (BM).

Boltzmann machine

We now need to define the dynamics of the system as a set of state update rules, such that the network will sample from its target distribution. In particular, this means that the distribution of the time series $\boldsymbol{z}^{(t)}$ should approach $p(\boldsymbol{z})$ as t approaches infinity. In other words, we require our dynamical system to be ergodic.

[17] The same equations can be obtained from a special subcase of the MRFs discussed in Section 6.1.2. We can define the factors not over maximal cliques, but over cliques of size 1 and 2 as

$$\Phi(z_k) = \exp[b_k z_k] \quad \text{and} \tag{6.61}$$

$$\Psi(z_i, z_j) = \exp\left[\frac{1}{2} w_{ij} z_i z_j\right] \quad . \tag{6.62}$$

The joint distribution then reads (Equation 6.17):

$$p(\boldsymbol{z}) = \frac{1}{Z}\prod_k \Phi(z_k) \prod_{ij} \Psi(z_i, z_j) = \frac{1}{Z}\exp\left[\frac{1}{2}\sum_{ij} w_{ij} z_i z_j + \sum_k b_k z_k\right] \quad , \tag{6.63}$$

which is identical to Equation 6.59.

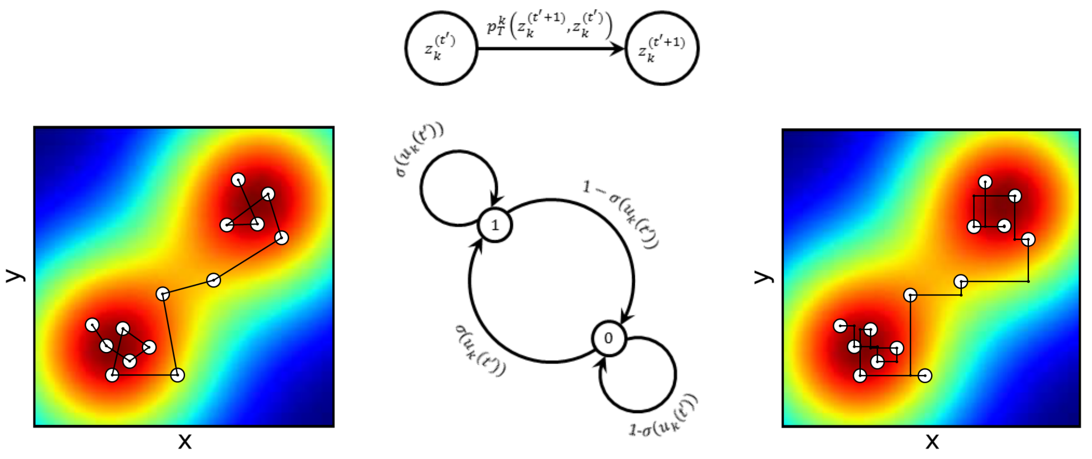


Figure 6.23.: Visualization of MCMC sampling. **Left:** Generic MCMC sampling from a joint distribution over two real-valued RVs. The jump from each state to the next one is defined by a transition operator that depends only on the current state. High-probability states (red) are more likely to be the target of a jump than low-probability states (blue). **Middle:** Transition probabilities for a single binary RV in Gibbs sampling. **Right:** Same as in left panel, but with Gibbs sampling. Since the RVs are updated sequentially, each jump happens in parallel to one of the axes. Individual samples are recorded only after all RVs have been updated.

We start by modifying the deterministic Hopfield update rule to a stochastic one. Instead of determining the state z_k by comparing the potential u_k to a fixed threshold θ_k, we consider u_k to encode the probability of $z_k = 1$. In particular, we (re)define

local computability condition

$$u_k := \ln \frac{p(z_k = 1|z_{\setminus k})}{p(z_k = 0|\boldsymbol{z}_{\setminus k})} \quad , \tag{6.64}$$

which we shall call the local computability condition. This functional relationship is, in principle, arbitrary, but this particular form turns out to be quite useful for our BMs. If we calculate $p(z_k = 1|\boldsymbol{z}_{\setminus k})$ and $p(z_k = 0|\boldsymbol{z}_{\setminus k})$ from Equation 6.59 and plug them into Equation 6.64, we obtain

$$u_k = \sum_{i=1}^{n} w_{ik} z_k + b_k \quad , \tag{6.65}$$

which is precisely the definition of u_k from Equation 6.57. In other words, the local computability condition ensures that, under the assumption that the network states follow a Boltzmann distribution, the potential has the functional form from Equation 6.57 – a linear sum of "forces" from other units plus a bias of its own.

Since unitarity requires that $p(z_k = 1|\boldsymbol{z}_{\setminus k}) = 1 - p(z_k = 0|\boldsymbol{z}_{\setminus k})$, we can now calculate the state probabilities for individual units from Equation 6.64:

activation function

$$p(z_k = 1|\boldsymbol{z}_{\setminus k}) = \frac{1}{1 + \exp(-u_k)} = \sigma(u_k) \quad , \tag{6.66}$$

where $\sigma(\cdot)$ represents the logistic function. The function $p(z_k = 1|\boldsymbol{z}_{\setminus k})$ is often called an activation function.

At this point, we introduce time as an explicit variable. In particular, we shall use t' as a finely granular time line, where each temporal increment corresponds to an update of a single unit $z_k^{(t')} \to z_k^{(t'+1)}$. This will become important later, when we will need to switch to a coarser time frame in which all units are updated between consecutive points in time. From a computational perspective, it would be advantageous for a unit to able to receive a new state every time it is updated, as well as being required to remember as little as possible of its past history. In particular, we can choose the local transition probability p_T^k of z_k at time t' as

- independent of $z_k^{(t')}$ itself and
- equal to the respective state probabilities given by the activation function:

local transition probability

$$\begin{aligned} p_T^k(z_k^{(t'+1)}, \boldsymbol{z}^{(t')}) &:= p(z_k^{(t'+1)}|z_k^{(t')}, \boldsymbol{z}_{\setminus k}^{(t')}) \\ &\overset{!}{=} p(z_k = 1|\boldsymbol{z}_{\setminus k}) \\ &= \begin{cases} \sigma(u_k(t')) & \text{if } z_k^{(t'+1)} = 1 \quad , \\ 1 - \sigma(u_k(t')) & \text{if } z_k^{(t'+1)} = 0 \quad . \end{cases} \end{aligned} \tag{6.67}$$

Note that all $u_k(t')$ are fully determined by $\boldsymbol{z}_{\setminus k}^{(t')}$. It should be quite obvious that this choice of p_T^k leaves $p(z_k|\boldsymbol{z}_{\setminus k})$ invariant and therefore samples correctly from the conditional distribution of z_k given $\boldsymbol{z}_{\setminus k}$. Having defined a transition probability, we have implicitly rendered all units stochastic, so we can now use "RV" and "unit" synonymously.

The local transition probability p_T^k is often described in literature as a local transition operator T^k. This is, however, rather misleading[18], since one would intuitively expect the transition operator T^k to actually produce a new state $\boldsymbol{z}^{(t'+1)} = (z_1^{(t')}, \ldots, z_{k-1}^{(t')}, z_k^{(t'+1)}, z_{k+1}^{(t')}, \ldots, z_n^{(t')})$ (where only z_k was updated) from the old one $\boldsymbol{z}^{(t')} = (z_1^{(t')}, \ldots, z_n^{(t')})$. In other words, for our state space $\boldsymbol{z} \in \Omega = \{0,1\}^n$ we should expect

$$p_T^k : \{0,1\}^{n+1} \mapsto [0,1] \quad \text{(see Equation 6.67)} \quad \text{and} \tag{6.68}$$

$$T^k : \{0,1\}^n \mapsto \{0,1\}^n \quad . \tag{6.69}$$

As we consider widespread use to be an insufficient argument for misleading denominations, we will follow the above intuition for the meaning of transition probabilities and transition operators. We therefore define our local transition operator T^k as

local transition operator

$$\boldsymbol{z}^{(t'+1)} = (z_1^{(t')}, \ldots, z_{k-1}^{(t')}, z_k^{(t'+1)}, z_{k+1}^{(t')}, \ldots, z_n^{(t')}) =: T^k(\boldsymbol{z}^{(t')}) \tag{6.70}$$

[18] The term "operator" does make sense when p_T is applied to a probability distribution over the entire state space, since it produces another probability distribution – similarly to to operators in quantum mechanics, which are homomorphisms on the Hilbert space of wavefunctions, which are, in turn, conceptually similar to probability distributions.

and set it to a Bernoulli process with success rate p_T^k that affects only z_k:

$$z_k^{(t'+1)} \sim B(1, p_T^k) \quad , \tag{6.71}$$

where $B(n,p)$ denotes a binomial distribution with parameters n and p. In practice, T^k will produce new states in conformity with the transition probability p_T^k following a simple algorithm:

1. draw a random number from a uniform distribution $r \sim \mathrm{unif}(0,1)$;
2. compare it to $p_T^k(z_k^{(t'+1)} = 1, \boldsymbol{z}^{(t')}) = \sigma(u_k(t'))$;
3. if $r < \sigma(u_k(t'))$ set $z_k^{(t'+1)} \to 1$, else $z_k^{(t'+1)} \to 0$.

A graphical representation of local updates is shown in the middle panel of Figure 6.23.

A full update of the system is realized in n local update steps. The global transition probability from $\boldsymbol{z}^{(t')}$ to $\boldsymbol{z}^{(t'+n)}$ is then given by a product of local transition probabilities:

global transition probability

$$p_T(\boldsymbol{z}^{(t'+n)}, \boldsymbol{z}^{(t')}) := \prod_{k=1}^{n} p_T^k(z_k^{(t'+k)}, \boldsymbol{z}^{(t'+k-1)}) \tag{6.72}$$

and the corresponding global transition operator T can be naturally defined as a chain over the local transition operators T_k:

global transition operator

$$\boldsymbol{z}^{(t'+n)} =: T(\boldsymbol{z}^{(t')}) := T_n \circ \cdots \circ T_1(\boldsymbol{z}^{(t')}) \quad . \tag{6.73}$$

For the global sampling process, we can now discard all intermediate updates and only keep the samples after a full update of all RVs. Our sample times will therefore be

"valid" samples

$$\boldsymbol{t} = (0, 1, 2, \ldots, T) \longleftrightarrow \boldsymbol{t}' = (0, n, 2n, \ldots, Tn) \tag{6.74}$$

and our full sampling process is defined by the global transition operator $T(\boldsymbol{z}^{(t)})$ and the associated transition probabilities $p_T(\boldsymbol{z}^{(t+1)}, \boldsymbol{z}^{(t)})$.

Since the update is stochastic (Monte Carlo) and the state of the system $\boldsymbol{z}^{(t)}$ at any time t depends only on its previous state $\boldsymbol{z}^{(t-1)}$ (Markov property), the chain induced by T represents a Markov chain Monte Carlo (MCMC) sampler. In particular, such a sequential update of individual units according to the local conditional $p(z_k = 1 | z_{\setminus k})$ is called Gibbs sampling and is a special case of the more general Metropolis-Hastings algorithm. A graphical representation of Gibbs sampling is shown in the right panel of Figure 6.23.

Markov chain Monte Carlo sampling, Gibbs sampling

As discussed above, in order to ensure that our algorithm does indeed sample from the target Boltzmann distribution $p(\boldsymbol{z})$, we need to prove that the joint distribution $p(\boldsymbol{z})$ is invariant under the global transition probability p_T and that the global transition operator T is ergodic. All of these follow from the fact that Gibbs sampling satisfies detailed balance, i.e.,

invariance

detailed balance

$$p(\boldsymbol{z}) p_T(\boldsymbol{z}', \boldsymbol{z}) = p(\boldsymbol{z}') p_T(\boldsymbol{z}, \boldsymbol{z}') \quad . \tag{6.75}$$

This equation describes an equal probability mass flow in both directions between $\boldsymbol{z}$ and $\boldsymbol{z}'$, and can be easily verified for our p_T.[19] In particular, this holds independently of the target distribution. If detailed balance holds, then a Markov chain is said to be reversible. *reversibility*

Although detailed balance already implies ergodicity, we shall discuss this property separately, since it may hold for a Markov chain even if it is not reversible. Ergodicity can be proven by demonstrating that the Markov chain induced by T is irreducible and aperiodic. *ergodicity*

A Markov chain is irreducible if any state $\boldsymbol{z}^{(t+1)}$ is accessible from any other state $\boldsymbol{z}^{(t)}$. For individual RVs z_k, this follows from the fact that the logistic function σ is strictly positive and strictly smaller than one, so arbitrary local jumps of z_k will always have nonzero probability ($\sigma(u_k)$ or $1 - \sigma(u_k)$, depending on the target state). Since the transition from any $\boldsymbol{z}(t)$ to any other $\boldsymbol{z}(t+1)$ can be represented as a chain of local jumps (Equation 6.73), it will always have nonzero probability as well. *irreducibility*

A Markov chain is aperiodic if, for any initial state $\boldsymbol{z}_0$, there exists a $\tilde{t}$ such that for every $t > \tilde{t}$ the system may return to its initial state: *aperiodicity*

$$p(\boldsymbol{z}^{(t)} = \boldsymbol{z}_0 | \boldsymbol{z}^{(t_0)} = \boldsymbol{z}_0) > 0 \ , \ \forall t > \tilde{t} \quad . \tag{6.76}$$

Since any state has a nonzero probability of remaining unchanged (due to $\sigma(u_k) \in (0,1)$, see above), aperiodicity is a trivial consequence.[20]

We end our brief description of Gibbs sampling and BMs with a discussion of the advantages of stochastic approaches to probabilistic inference (i.e., sampling) compared to analytical ones, such as the belief propagation algorithms from Section 6.3.

Let us first return to an earlier point we made in Section 6.1.2. As can be seen in the definition of the Boltzmann distribution (Equation 6.59), assigning a numerical value to the probability of some state $\boldsymbol{z}$ requires computing the partition function Z. This, in turn, requires computing a sum over all possible states $\boldsymbol{z}$ (Equation 6.60), which, for binary RVs z_k has a computational cost of $\mathcal{O}(2^n)$, where n represents the number of RVs in the BM. Clearly, this quickly becomes computationally unfeasible. Gibbs sampling, on the other hand, only requires the computation of local transition probabilities p_T^k. These are simple logistic functions over potentials u_k, which are, in turn, calculated as log-odds of Boltzmann probabilities (Equation 6.64), where the partition functions cancel out. This is one essential motivation for MCMC sampling: computing local updates as probability ratios gets rid of the troublesome partition function.

Another advantage of stochastic inference concerns the computation of marginal and conditional distributions. In general, computing the joint distribution of a subset of $\boldsymbol{z}'$ of the RVs in $\boldsymbol{z}$ requires marginalizing out all other RVs $\boldsymbol{z}_{\setminus \boldsymbol{z}'}$. Unless we have a special case

[19] The concept of detailed balance was initially developed in statistical mechanics for describing thermodynamic equilibrium systems. The reversal of the probability mass flow in Equation 6.75 is, for example, equivalent to PT-inversion for particle collisions in a gas. In this sense, detailed balance is therefore intimately linked to the concept of PT-invariance.

[20] Ergodicity is intimately connected to the mixing properties of a sampler, i.e., its ability to quickly travel through all high-probability regions of the state space. While in theory Gibbs sampling from Boltzmann distributions is guaranteed to converge to the target distribution for $t \to \infty$, any realistic scenario imposes a finite sampling time. This may critically impair the mixing ability of the Gibbs sampler, in particular in conjunction with high barriers separating deep troughs in the energy landscape.

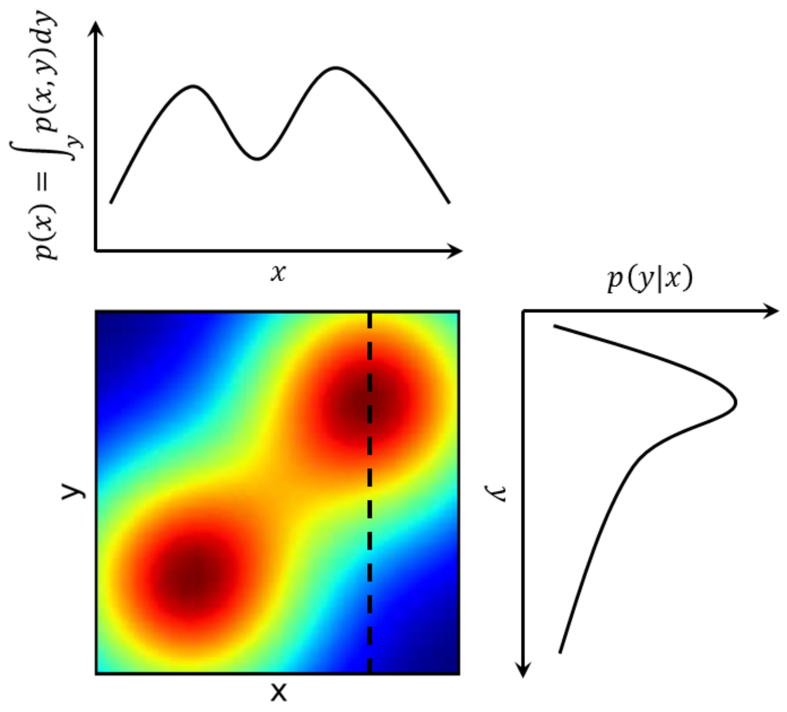


Figure 6.24.: Sample-based computation of marginals and conditionals. As in Figure 6.23, the probability space is spanned by a pair (x, y) of real-valued RVs, with red and blue denoting high- and low-probability states, respectively. Marginalization over y is performed simply by counting the x values of samples, while disregarding their y value. Visually, it can be interpreted as "squashing" the y axis, which can be easily seen to yield the bimodal distribution over x depicted on top. Conditioning on a fixed value of x is simple performed by fixing the value of x, i.e., not updating it during the sampling process. Samples are then gathered only along the dashed line, yielding the unimodal distribution depicted on the right.

for which an efficient algorithm can be found (as is the case for the sum-product algorithm in loop-free factor graphs), this operation requires summing up the joint distribution $p(\boldsymbol{z})$ over all possible states of $\boldsymbol{z}_{\setminus \boldsymbol{z}'}$. Just like in the case of the partition function – which can be viewed as the ultimate marginalization, since it marginalizes out all RVs – the computational cost of this operation is exponential in the number of RVs to be marginalized out. If, on the other hand, the distribution $p(\boldsymbol{z})$ is sampled, marginalization comes for free, since it simply requires neglecting the values of the RVs we want to marginalize out. Conditioning on observed variables is equally unproblematic, since they can be simply fixed in the stochastic update process, thus even saving computation time. The calculation of marginals and conditionals from sampled distributions is illustrated in Figure 6.24.

Evidently, sampling only offers an approximation $p^*(\boldsymbol{z})$ of the target distribution $p(\boldsymbol{z})$. However, this distribution becomes increasingly accurate as more samples are collected. Moreover, even after few steps, the Markov chain can offer a first approximation of the main modes of $p(\boldsymbol{z})$. This capability of providing (increasingly) useful results at any point in time is often referred to as anytime computing and is an essential argument for the biological plausibility of sample-based representations, as it is easy to imagine situations where an organism may not have enough time to wait until an analytical neural inference algorithm has converged to a solution.

anytime computing

6.4.3. MCMC Sampling with Spiking Neurons

In the previous sections, we have described the local variable u_k as being a “potential”, while arguing that it represents a “sum of forces” from other units in the network. The term “potential” was not used in the physical sense, but with a neural interpretation in mind: with the shape defined in Equation 6.57 (and derived from the Boltzmann distribution), it bears striking similarity to the membrane potential of LIF neurons (see Equations 4.38 and 4.58), albeit in a much simpler form. Indeed, the bias can be viewed as an analogue of the offset potential (either $E_\mathrm{l} + I^\mathrm{ext}/g_\mathrm{l}$ or ${u_\mathrm{eff}}^0$, depending on the model), whereas the sum is similar to the synaptic input, only with rectangular instead of DOE PSPs. The local computability condition defined in Equation 6.64 turns out to be a neural computability condition[21] (NCC) under the assumption of sampling from Boltzmann distributions over binary RVs.

unit potential as a membrane potential

neural computability condition

However, there is evidently much more to spiking neurons than just their passive membrane potential. In order to find a formal equivalence between the dynamics of neurons and stochastic sampling, we shall first introduce an abstract model of spiking neurons, for which we can formulate an exact theory of neural sampling. This model was first described in Buesing et al. (2011) and provides a series of fundamental concepts and relations. It shall later serve as an essential reference when we will formulate a sampling theory for the more complex dynamics of LIF neurons.

neural sampling

The fundamental assumption of the neural sampling theory is that the activity of a single neuron encodes the state of an associated binary RV z_k. Since biological neurons communicate with spikes, it is rather natural to interpret a spike as a switch from $z_k = 0$ to $z_k = 1$.[22] We can then define that the neuron remains in the state $z_k = 1$ for an arbitrary (for now) duration τ_on after the occurrence of a spike at time t_s:

spikes as state switches

$$z_k^{(t)} = \begin{cases} 1 & \text{if } t_s < t < t_s + \tau_\mathrm{on} \quad , \\ 0 & \text{otherwise} \quad . \end{cases} \tag{6.77}$$

In order to avoid the potential complication of two spikes occurring with a distance smaller than τ_on, we make use of another biological feature of neurons: the existence of an absolute refractory period τ_ref (see Section 2.1.2). By setting

$$\tau_\mathrm{on} = \tau_\mathrm{ref} \quad , \tag{6.78}$$

states and refractoriness

we define a clear mapping of spike trains to binary RVs: $z_k = 1$ if and only if the kth

[21] This term was initially coined in Buesing et al. (2011), but the functional form of the NCC is none other than the one used for Gibbs sampling in Boltzmann machines and Ising models.

[22] Strictly speaking, this is not entirely correct, since model neurons in discrete time may spike at maximum rate, with no 0-states between their refractory periods, as we also discuss further on. However, since any physical neuron requires nonzero time for the generation of an AP following absolute refractoriness, equating a spike to a $z_k = 0 \rightarrow 1$ switch remains, for all practical purposes, correct.

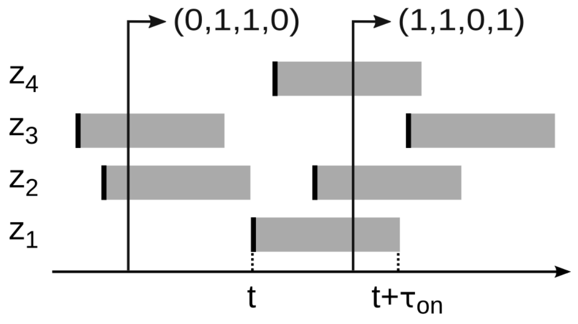


Figure 6.25.: Neural sampling with absolute refractory times: interpretation of spike trains. Each neuron is associated with a binary RV z_k, and its refractory periods are considered to represent 1-states of their associated RV. A spike can be interpreted as a switch from $z_k = 0$ to $z_k = 1$. A spike raster plot can therefore be read as a sequence of states of a binary vector $\boldsymbol{z}$. Figure taken from Petrovici et al. (2013).

neuron is refractory.[23] A spike raster plot can then be easily read as the temporal evolution of a binary vector $\boldsymbol{z}(t)$ representing the state of an ensemble of binary RVs (see Figure 6.25).

Let us first consider the neural dynamics of a single neuron with constant membrane potential u_k in discrete time, where τ_ref is simply an integer. As with Gibbs sampling, we shall use a more finely granular time frame $t' \to \cdots \to t' + n$, since the ζ_k are updated successively for all n neurons in the network within a global sampling step $t \to t + 1$. For $\tau_\text{ref} = 1$, we could simply represent the dynamics of z_k as Gibbs sampling. We would then define the neuron as inherently stochastic and have it spike with a probability given by the activation function 6.66: $p(\text{spike}) = \sigma(u_k)$. The dynamics of z_k would then be exactly those depicted in the middle panel of Figure 6.23 and, by definition of the update rule, Markovian – just as Gibbs sampling. However, if we allow a longer refractory period $\tau_\text{ref} > 1$, our spike probability may no longer be first-order Markovian, since the neuron would have to remember when it has last spiked. In particular, we would require

refractory period

$$p(\text{spike at time } t') = 0 \quad \text{if} \quad \sum_{\tau=1}^{\tau_\text{ref}} z_k^{(t'-\tau)} > 0 \quad . \tag{6.79}$$

In order to remain in the framework of MCMC sampling, we must therefore define an additional variable ζ_k that measures refractoriness. The dynamics of ζ_k are very simple: it jumps to τ_ref when the neuron has fired and then decays linearly until it reaches 0 (see Figure 6.26). The neuron may then only fire at time $t' + 1$ if $\zeta_k^{(t')} \leq 1$. Note how the refractory variable maps surjectively onto the state variable:

refractoriness variable

z-to-ζ mapping

$$\zeta_k^{(t')} \geq 1 \mapsto z_k^{(t')} = 1 \tag{6.80}$$

$$\zeta_k^{(t')} = 0 \mapsto z_k^{(t')} = 0 \tag{6.81}$$

[23] While we shall henceforth adhere to this one-to-one mapping between $z_k = 1$ and absolute refractoriness, we should note that Buesing et al. (2011) also provide an extension of the neural sampling theory to an abstract neuron model with relative refractoriness.

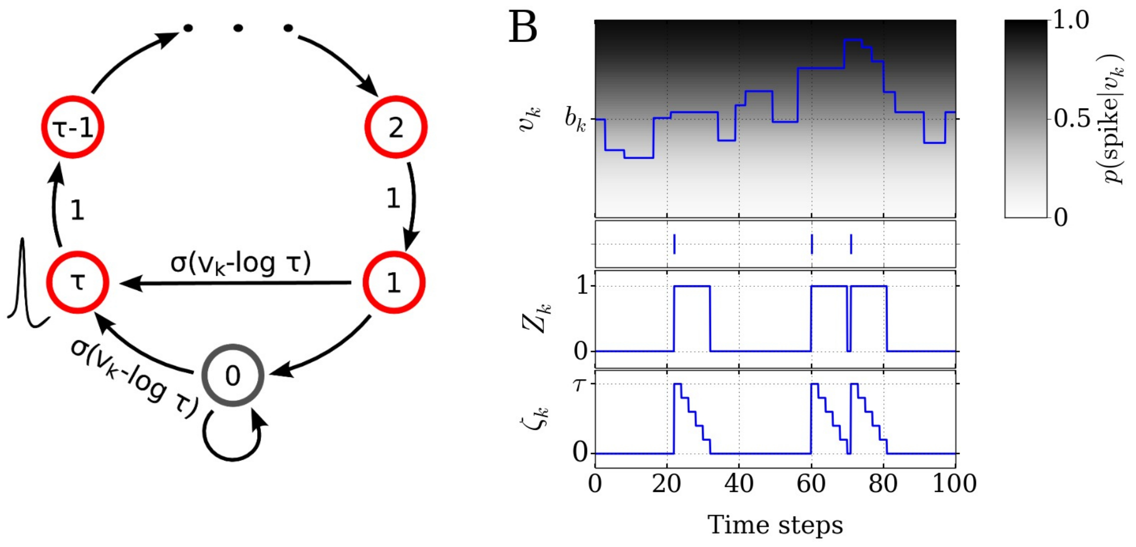


Figure 6.26.: Neural sampling with absolute refractory times: abstract model. **Left:** Transition probabilities p_T^k of the refractoriness variable ζ_k. An emitted spike causes ζ_k to jump to the value τ_{ref}, from where it decays by 1 in each time step. During this time, the neuron may emit no spike. Once it has reached the value 1 or 0, the neuron is allowed to spike again. **Right:** All dynamic variables associated with a single neuron. The membrane potential u_k represents a sum over rectangular PSPs elicited by incoming spikes. The probability of a neuron to emit a spike itself (in grayscale) depends on the membrane potential according to Equation 6.85. When a spike is emitted, the state variable z_k switches to a 1-state for the duration τ_{ref}. At the same time, the refractoriness variable ζ_k obeys the dynamics depicted in the left panel. Note how the entire state $(\boldsymbol{u}, \boldsymbol{z}, \boldsymbol{\zeta})$ of the network can be calculated directly from $\boldsymbol{\zeta}$, meaning that the dynamics of $\boldsymbol{\zeta}$ fully define the dynamics of the entire network. Figures taken from Probst et al. (2015).

This mapping also explains why spiking is allowed at time $t'+1$ if $\zeta_k^{(t')} = 1$: it allows $p(z_k) = 1$, or, in other words, a continuous 1-state.

Effectively, the dynamics of the state variable now follow from the dynamics of the refractory variable. In particular, the firing probability $p(z_k^{(t'+1)} = 1|\text{neuron is not refractory})$ now translates to $p(\zeta_k^{(t'+1)} = \tau_{\text{ref}}|\zeta_k^{(t'+1)} \leq 1)$. However, since with every spike the neuron is now forced into the 1-state for an extended period of time, we need to change the spiking probability in order to maintain the state probability defined by the activation function[24]:

$$p(z_k = 1) = \sigma(u_k) \quad . \tag{6.82}$$

If we denote the firing probability by p_{spike}, the unnormalized relative time spent by a

[24] Remember that this particular shape of the activation function is not arbitrary, but was derived from the chosen form of the NCC, which is, in turn, required for the membrane potential to assume a neurally compatible shape when sampling from Boltzmann distributions.

ζ_k	τ_{ref}	$\dots$	1	0
$p(\zeta_k^{(t)})$	$\sigma(u_k)/\tau_{\text{ref}}$	$\sigma(u_k)/\tau_{\text{ref}}$	$\sigma(u_k)/\tau_{\text{ref}}$	$1-\sigma(u_k)$
$p(\zeta_k^{(t+1)})$	$[\sigma(u_k)/\tau_{\text{ref}} + 1 - \sigma(u_k)] \cdot \sigma(u_k - \ln\tau_{\text{ref}})$	$\sigma(u_k)/\tau_{\text{ref}}$	$\sigma(u_k)/\tau_{\text{ref}}$	$[\sigma(u_k)/\tau_{\text{ref}} + 1 - \sigma(u_k)] \cdot [1 - \sigma(u_k - \ln\tau_{\text{ref}})]$

Table 6.1.: Stationarity of the refractory variable distribution $p(\zeta_k)$ under the transition probability p_T^k (Equation 6.85). The evolution of this distribution after a single transition is given by Equation 6.90.

neuron in the refractory ($z_k = 1$) state amounts to $p_{\text{spike}} \cdot \tau_{\text{ref}}$, whereas the unnormalized relative non-refractory time is simply $1 - p_{\text{spike}}$. By equating the refractory-time-to-total-time ratio to the activation function, we can immediately solve for p_{spike}:

firing probability

$$\frac{p_{\text{spike}} \cdot \tau_{\text{ref}}}{p_{\text{spike}} \cdot \tau_{\text{ref}} + 1 - p_{\text{spike}}} = p(z_k = 1) \overset{!}{=} \sigma(u_k) \tag{6.83}$$

$$\implies \quad p_{\text{spike}} = \sigma(u_k - \ln\tau_{\text{ref}}) \quad . \tag{6.84}$$

local transition probabilities

We can now fully define the MCMC dynamics of a single abstract spiking neuron by a transition probability over ζ_k:

$$p_T^k(\zeta_k^{(t'+1)}, \boldsymbol{\zeta}_k^{(t')}) := \begin{cases} p(\zeta_k^{(t'+1)} = \zeta_k^{(t')} - 1 | \zeta_k^{(t')} \geq 2) & = 1 \quad , \\ p(\zeta_k^{(t'+1)} = \tau_{\text{ref}} | \zeta_k^{(t')} \leq 1) & = \sigma(u_k - \ln\tau_{\text{ref}}) \quad \text{and} \\ p(\zeta_k^{(t'+1)} = 0 | \zeta_k^{(t')} \leq 1) & = 1 - \sigma(u_k - \ln\tau_{\text{ref}}) \quad . \end{cases} \tag{6.85}$$

local transition operator

Following our above definition of p_T^k, the definition of the local transition operator T_k follows the one we had for Gibbs sampling (compare Equation 6.70)

$$\boldsymbol{\zeta}^{(t'+1)} = (\zeta_1^{(t')}, \dots, \zeta_{k-1}^{(t')}, \zeta_k^{(t'+1)}, \zeta_{k+1}^{(t')}, \dots, \zeta_n^{(t')}) =: T^k(\boldsymbol{\zeta}^{(t')}) \tag{6.86}$$

and is straightforward for $\zeta_k^{(t')} \geq 2$, while otherwise resembling the functional form of the Gibbs local transition operator (Equation 6.71):

$$\zeta_k^{(t'+1)} \begin{cases} = \zeta_k^{(t')} - 1 & \text{if} \quad \zeta_k^{(t')} \geq 2 \quad , \\ \sim B(1, \sigma(u_k - \ln\tau_{\text{ref}})) & \text{if} \quad \zeta_k^{(t')} \in \{0, 1\} \quad . \end{cases} \tag{6.87}$$

The left panel of Figure 6.26 shows a graphical depiction of the above equations.

stationarity under local transitions

We can now check the correctness of the transition probability/operator defined above by verifying the stationarity of $p(\zeta_k)$ under p_T^k. The activation function requires $p(z_k = 0) = 1 - \sigma(u_k)$, so with the mapping from Equation 6.81 we obtain

$$p(\zeta_k = 0) = 1 - \sigma(u_k) \quad . \tag{6.88}$$

The state $z_k = 1$ must be equally shared by $\zeta_k \in \{1, \dots, \tau_{\text{ref}}\}$, since each of these states is reached exactly once after each spike (Equation 6.87), so we have

$$p(\zeta_k = \tau_{\text{ref}}) = \dots = p(\zeta_k = 1) = \frac{\sigma(u_k)}{\tau_{\text{ref}}} \quad . \tag{6.89}$$

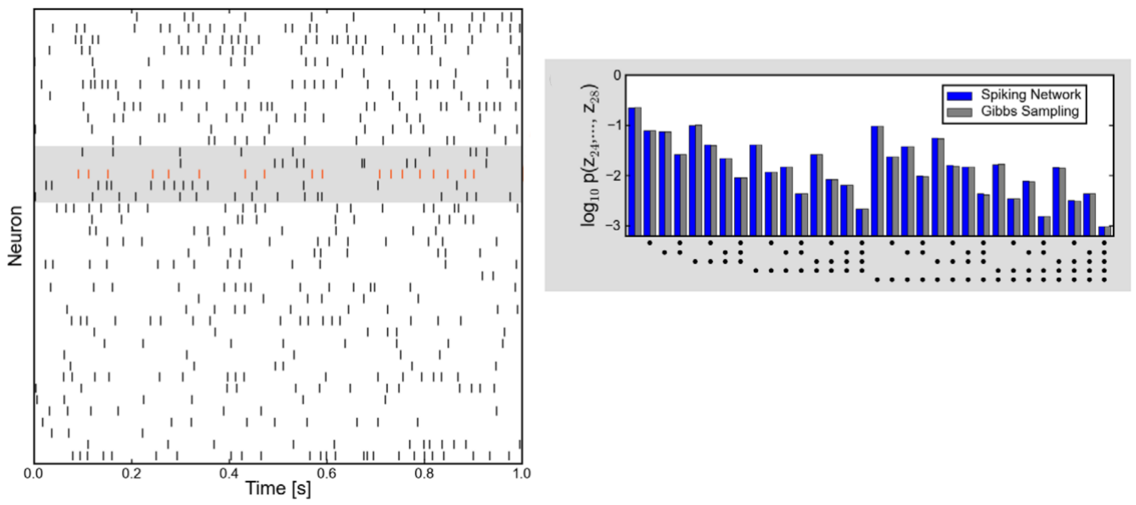


Figure 6.27.: Network of 40 abstract neurons with $\tau_{\mathrm{ref}} = 20\,\mathrm{ms}$ sampling from a target Boltzmann distribution with randomly chosen weighs and biases. **Left:** Raster plot showing spikes from the entire network. A subset of 5 neurons (gray box) is chosen for comparison with the target distribution. **Right:** Joint distribution over the 5 binary RVs associated with the selected neurons. Since already at this small size (40 RVs) an analytical computation of the distribution is completely unfeasible, Gibbs sampling was used as a reference. The distribution sampled by the network (blue) after $10^4\,\mathrm{s}$ is in excellent agreement with the one obtained from Gibbs sampling (gray). Figures taken from Buesing et al. (2011).

The evolution of these probabilities after one update step is given by the (discrete) Chapman-Kolmogorov equation:

Chapman-Kolmogorov equation

$$p(\zeta_k^{(t'+1)}) = \sum_{\zeta_k^{(t')}} p(\zeta_k^{(t')}) p_T^k(\zeta_k^{(t'+1)}, \zeta_k^{(t')}) \tag{6.90}$$

Table 6.1 represents these probabilities explicitly, which take on a rather sparse shape since only few transition probabilities are nonzero. It is then quite straightforward to check that with our above definition of p_T^k, all $p(\zeta_k)$ are left unchanged.

The extension from single neurons to networks of neurons can be done equivalently to the way we did it for Gibbs sampling. We first switch to the coarse time frame t where one update step for the entire system is equivalent to n local update steps, one for each neuron. We can now chain our local transition operators to form a global transition operator

global transition operator

$$T = T_n \circ \cdots \circ T_1 \quad , \tag{6.91}$$

with which we can sample from the joint distribution $p(\boldsymbol{\zeta})$ and thereby, as shown above, implicitly from $p(\boldsymbol{z})$. The global transition probability then reads

global transition probability

$$p_T(\boldsymbol{\zeta}^{(t+1)\equiv(t'+n)}, \boldsymbol{\zeta}^{(t)\equiv(t')}) := \prod_{k=1}^{n} p_T^k(\zeta_k^{(t'+k)}, \boldsymbol{\zeta}^{(t'+k-1)}) \quad . \tag{6.92}$$

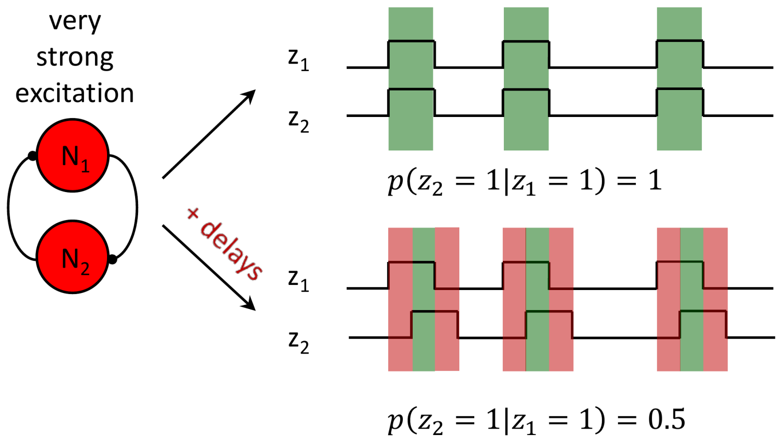


Figure 6.28.: Effect of spike transmission delays on the sampled distribution. The exemplary network has two neurons with strong excitatory weights, such that when one neuron spikes, the other one is forced to spike simultaneously. (Formally, we can achieve perfect correlation by setting $w_{12} = 2M$ and $b_1 = b_2 = -M$ while letting $M \to \infty$.) In these conditions, the 1-states are perfectly aligned, i.e., the sampled distribution does not contain mixed (01 or 10) states. Transmission delays destroy this alignment by inducing mixed states (correct joint states are depicted in green, incorrect ones in red). The only way to deal with transmission delays without changing the theory is to use refractory periods that are significantly longer than the delays, thus decreasing the relative time that the network spends in "wrong" states (the "red-to-green ratio").

irreversible Markov chain

As opposed to Gibbs sampling however, neural sampling with refractoriness is explicitly non-reversible (for example, the transitions from $\zeta_k + 1$ to ζ_k are all allowed, but the reverse transitions are forbidden). Therefore, detailed balance can not hold, so stationarity and ergodicity (irreducibility, aperiodicity) must be demonstrated separately. Indeed, the global transition probability and operator do have these properties, but since demonstrating this is more of a technical exercise, we refer to Buesing et al. (2011) for these formal proofs. An example of a network of stochastic spiking neurons sampling from a Boltzmann distribution over 40 RVs is shown in Figure 6.27.

spike-based communication

The membrane potentials $\boldsymbol{u}$ are not independent dynamic variables, since they can be computed directly from $\boldsymbol{z}$. In practice, this can be done analogously to Gibbs sampling, but that would require each neuron to gather the state of all the other neurons at every point in time in order to calculate u_k (Equation 6.57). This is where the efficiency of spike-based communication comes in. Exchange of information can happen only through spikes, with each spike emitted by a neuron manifesting itself as a PSP of appropriate weight on the membranes of all other neurons, as long as the PSPs are defined appropriately. In particular, we require the PSPs to be rectangular, with a duration ("time constant") equal to the refractory period

rectangular PSP duration

$$\tau^{\text{syn}} = \tau_{\text{ref}} \quad . \tag{6.93}$$

Then, we achieve a precise correspondence between $z_k = 1$ and $\Delta u_i = w_{ki}$, as required by Equation 6.57.

Note that Equation 6.57 says nothing about a reset upon emission of a spike. Indeed, in this model, u_k is not affected at all by emitted spikes or refractoriness, as would be the case in the most commonly used neuron models – or in biological neurons, for that matter. In the following sections, we shall discuss how this limitation can be overcome.

no membrane reset

We can now also recognize another essential requirement of this model: synaptic transmission must be instantaneous, otherwise the synchronicity of states and PSPs is violated. An intuitive example of how this affects the sampled distribution is shown in Figure 6.28. This issue can obviously become cumbersome for any physical implementation, be it biological or neuromorphic, since physical communication can never really be instantaneous. One possible solution is, however, straightforward: by increasing refractory times, the relative shift of states and PSPs can be made smaller, thereby restoring (approximately) correct sampling. We need to explicitly deal with this issue when aiming for a neuromorphic implementation of LIF sampling, which we discuss in Section 6.8.

no transmission delays

Since here we have, effectively, derived this model from Gibbs sampling, time was naturally discretized between update steps. The transition from discrete to continuous time is rather straightforward, by having the physical time corresponding to an update step converge to zero, while at the same time increasing the number of steps in the refractory period correspondingly. For the formal procedure, we point again to Buesing et al. (2011). For an intuitive understanding, it suffices to remember that the refractoriness variable ζ was only introduced artificially, in oder to allow us to represent the dynamics of spiking, refractory neurons as a first-order Markov chain and thereby as MCMC sampling. The nature (discrete or continuous time) and mechanism (countdown variable, checking for time of last spike, timed inactivation of ion channels etc.) of an actual implementation of refractoriness in a software model or physical system carries no relevance, as long as refractory times are fixed, absolute and equal with τ^{syn} for all neurons.

6.5. Stochastic Inference with Deterministic Spiking Neurons

While having elegance and clarity on its side, there are at least two reasons why the abstract, inherently stochastic neural sampling model from Buesing et al. (2011) needs to be extended to a more mechanistic neuron model. On one hand, experiments have demonstrated the largely deterministic nature of single neurons (Mainen and Sejnowski, 1995). On the other hand, the vast majority of software and hardware back-ends mainly use deterministic neuron models (Brette et al., 2007; Indiveri et al., 2011).

The "greatest common divisor" of these back-ends, as well as the arguably most widely used model in computational neuroscience, is the LIF model, which we have already extensively discussed in Section 2.2.1.1 and throughout Chapter 4. In the following, we shall describe how neural sampling can be implemented in networks of LIF neurons. The central ideas of LIF sampling have already been published in Petrovici et al. (2013). In the process of developing a rigorous theory of "LIF sampling", we shall also identify a new computational role of the HCS, as well as derive an analytical expression for the response function of LIF neurons in a regime where previously developed approaches (Brunel and Sergi, 1998; Moreno-Bote and Parga, 2004) do not hold. This section provides the foundation for our later discussion of interesting experiments in both software (Sections 6.5.5, 6.6 and 6.6.2.2) and neuromorphic hardware (6.8), as well as for the extension to arbitrary distributions over binary RVs (Section 6.7).

LIF sampling

6.5.1. From a Deterministic Model to a Stochastic One

As we have discussed in Section 6.4.3, the fundamental requirement of the neural sampling theory was the NCC, which describes how information about the local conditional spiking probability is encoded in the membrane potential. This expression is equivalent to the activation function (Equation 6.82), which defines the probabilistic nature of the abstract neuron model. When transferring the neural sampling framework to LIF neurons, we want to keep the interpretation of spikes unchanged, i.e., the refractory state corresponds to $z_k = 1$. The goal is therefore to get an LIF neuron to be refractory with a logistic probability as a function of its (linearly transformed) membrane potential. In particular, we require

target LIF activation function

$$p(z_k = 1|z_{\setminus k}) = \sigma\left(\frac{\bar{u}_k - \bar{u}_k^0}{\alpha}\right) \quad , \tag{6.94}$$

where we use the following notation: the input potential $\bar{u}_k$ represents the contributions of all other neurons in the network to the membrane, $\bar{u}_k^0$ represents an offset (which is influenced by, e.g., the resting potential of the neuron) and α is a conversion factor from the LIF domain (e.g., mV) to the dimensionless membrane potential from the abstract neural sampling model. The bars on the membrane potentials represent averages, which we shall explain shortly but can be neglected for now.

A rather straightforward way of adding stochasticity to an otherwise constant membrane potential $\bar{u}_k$ is Poisson noise, which is compatible with biology and widely used in network models. Mathematical support for the Poisson assumption is given by the Palm-Khintchine theorem, which states that the superposition of a large number of independent equilibrium renewal processes, each with a small intensity, behaves asymptotically like a

Palm-Khintchine theorem

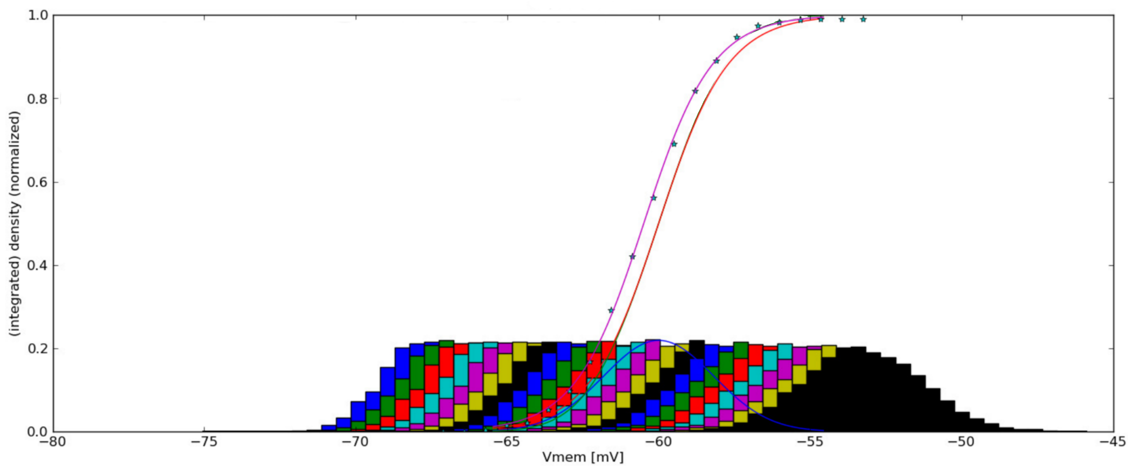


Figure 6.29.: An LIF neuron made stochastic by bombardment with Poissonian spike trains. As we have seen in Section 4.3, Poissonian stimulation leads to a Gaussian free membrane potential distribution, which can be shifted by modifying the resting potential of the neuron (colored histograms). These can be analytically predicted, which is depicted here for $E_l = \vartheta$ (blue Gaussian curve). The suprathreshold free membrane potential probability mass is therefore given by a linearly transformed error function (Equation 6.96 and red curve), which lies remarkably close to the actual activation function (cyan stars). The activation function appears to be well-fitted by a linearly transformed logistic function (purple curve), as would be required by classical neural sampling.

Poisson process. We can assume that apart from the functional connections to other neurons in the sampling network, each neuron receives diffuse "noise" input from many other, functionally unrelated neurons in the brain, which we can model as a high-frequency Poisson process. More precisely, we shall assume that each sampling neuron receives Poisson noise from two independent sources, one being excitatory and the other one inhibitory.

free membrane potential distribution under Poisson bombardment

As discussed in Section 4.3, this Poisson noise will cause a Gaussian distribution of the free membrane potential

$$p(u_k) = f_{\mathcal{N}}(u_k, \mu, \sigma) \quad , \tag{6.95}$$

with moments μ and σ that can be computed analytically. This allows the neuron to fire even when its input potential $\bar{u}_k$ lies below the threshold. For simplicity, let us assume that the excitatory and inhibitory contributions of the Poisson background are symmetric, so $\mu = \bar{u}_k$ (this is why we chose the notation $\bar{u}_k$, as it represents the "noise-free" membrane potential). Varying $\bar{u}_k$ shifts the free membrane potential distribution across the threshold, changing the suprathreshold area

suprathreshold free membrane potential probability mass

$$F(\bar{u}_k) = \int_{\vartheta}^{\infty} p(u_k) du_k = \frac{1}{2}\left[1 - \operatorname{erf}\left(\frac{\vartheta - \bar{u}_k}{\sqrt{2}\sigma}\right)\right] \quad . \tag{6.96}$$

Intuitively speaking, the probability of the neuron to fire (and then be refractory) should be proportional to this area, and since an appropriately shifted error function is very similar to the logistic function, the probability of a refractory state should be approximately logistic as well.

Indeed, it turns out that this intuitive reasoning is quite correct (Figure 6.29). However, this is clearly not a satisfactory description of our membrane dynamics, since the connection between firing probability and suprathreshold free membrane potential probability $F(\bar{u}_k)$ has no rigorous support. Indeed, if we actually measure the activation function, we can see that it deviates significantly from $F(\bar{u}_k)$. In order to provide a rigorous derivation of the LIF activation function under Poisson stimulation, we first require a correct description of the membrane potential dynamics in this regime.

6.5.2. Membrane Dynamics in the HCS and the Ornstein-Uhlenbeck Process

We start by restating the effective membrane potential formulation (Equation 4.31) of the COBA LIF equation:

$$\tau_\text{eff}\frac{du}{dt} = u_\text{eff} - u \quad , \tag{6.97}$$

where the effective membrane potential u_eff has the form (see Equation 4.30)

$$u_\text{eff}(t) = \frac{g_\text{l}E_\text{l} + \sum_{i=1}^{n} g_i^\text{syn}(t)E_i^\text{rev} + I^\text{ext}}{g^\text{tot}(t)} \quad . \tag{6.98}$$

In a first approximation, assuming a rapidly firing Poisson background ($\nu_\text{syn} \to \infty$), the total average conductance can become arbitrarily large ($\langle g^\text{tot}\rangle \to \infty$), causing the membrane potential to follow the effective potential nearly instantaneously ($\langle\tau_\text{eff}\rangle \to 0$). Here, we are evidently operating in the HCS, which we have already extensively discussed in Chapter 4. Equation (6.98) can then be rewritten as

$u \to u_\text{eff}$ in the HCS

$$u \approx u_\text{eff} = \frac{g_\text{l}E_\text{l} + \sum_{i=1}^{n} \langle g_i^\text{syn}\rangle E_i^\text{rev} + \sum_{i=1}^{n} \Delta g_i^\text{syn} E_i^\text{rev} + I^\text{ext}}{\langle g^\text{tot}\rangle + \sum_{i=1}^{n} \Delta g_i^\text{syn}} \quad , \tag{6.99}$$

where

$$\Delta g_i^\text{syn} = g_i^\text{syn} - \langle g_i^\text{syn}\rangle \tag{6.100}$$

denotes the fluctuations of the synaptic conductances.

For a single Poisson source with rate ν_i connected to the neuron by a synapse with weight w_i and time constant τ^syn, the conductance course can be seen as a sum of independent random variables, each of them representing the conductance change caused by a single spike. In Section 4.3.1, we have already derived the first two moments of the synaptic input (Equations 4.94 and 4.95) for exponential synaptic kernels (Equation 4.93):

$$E[g^\text{syn}] = \sum_{i=1}^{n} w_i \nu_i \tau_i^\text{syn} \quad , \tag{6.101}$$

$$\text{Var}[g^\text{syn}] = \sum_{i=1}^{n} \frac{1}{2} w_i^2 \nu_i \tau_i^\text{syn} \quad , \tag{6.102}$$

so the relative fluctuations of g^{syn} are of the order (see also Equation 4.96)

$$\frac{\sqrt{\mathrm{Var}\left[g^{\mathrm{syn}}\right]}}{E\left[g^{\mathrm{syn}}\right]} \approx \sqrt{\frac{1}{2\tau^{\mathrm{syn}} \sum_{k=1}^{n} \nu_k}} \tag{6.103}$$

and vanish in the limit of large firing rates.

This warrants an expansion of (6.99) in $\Delta g_i^{\mathrm{syn}}$, $\forall i$. Considering only the first-order term we obtain

u as a linear transformation of J^{syn}

$$u(t) = \frac{I^{\mathrm{ext}} + g_{\mathrm{l}} E_{\mathrm{l}} + \sum_{i=1}^{n} g_i^{\mathrm{syn}}(t) E_i^{\mathrm{rev}}}{\langle g^{\mathrm{tot}} \rangle} \quad , \tag{6.104}$$

which renders u simply a linear transformation of the synaptic input[25]

synaptic input

$$J^{\mathrm{syn}} = \sum_{i=1}^{n} g_i^{\mathrm{syn}} E_i^{\mathrm{rev}} \quad . \tag{6.105}$$

For exponential synapses, the synaptic input J^{syn} obeys the first-order inhomogeneous ODE

$$\frac{dJ^{\mathrm{syn}}}{dt} = -\frac{J^{\mathrm{syn}}}{\tau^{\mathrm{syn}}} + \sum_{\mathrm{syn}\, i} \sum_{\mathrm{spk}\, s} \Delta J_i^{\mathrm{syn}} \delta(t - t_s) \quad , \tag{6.106}$$

where $\Delta J_i^{\mathrm{syn}} = w_i E_i^{\mathrm{rev}}$. This equation is highly reminiscent of the ODE that defines the Ornstein-Uhlenbeck (OU) process

Ornstein-Uhlenbeck process

$$dx(t) = \theta \cdot (\mu - x(t))dt + \sigma dW(t) \quad . \tag{6.107}$$

A short discussion of the OU process is in order. The stochastic component of the OU process is given by the $dW(t)$ term, which represents the differential of the Wiener process. The Wiener process is the continuous-time equivalent of a random walk and is thus defined as follows:

Wiener process

- $W(0) = 0$;
- $W(t)$ is almost surely everywhere continuous;
- $W(t)$ has independent increments with $W(t) - W(t') \sim \mathcal{N}(\mu, \sigma^2)$.

In our above definition (Equation 6.107) of the OU process, we have externalized the parameters μ and σ, so our Wiener process is unbiased (zero-mean) and has unit variance. Formally, the Wiener process is self-similar (see Figure 6.30, top panel) since increments are differential, so in simulations the time must be appropriately discretized, on a time scale that is significantly smaller than the relevant time scale for the observed processes. This will become relevant shortly, when we establish the equivalence to Poisson-driven membrane potentials.

[25] Note that the synaptic input J^{syn} is not the same as the synaptic current I^{syn}. The synaptic current must depend on the membrane potential itself, i.e., $I^{\mathrm{syn}} = \sum_{i=1}^{n} g_i^{\mathrm{syn}}(E_i^{\mathrm{rev}} - u)$.

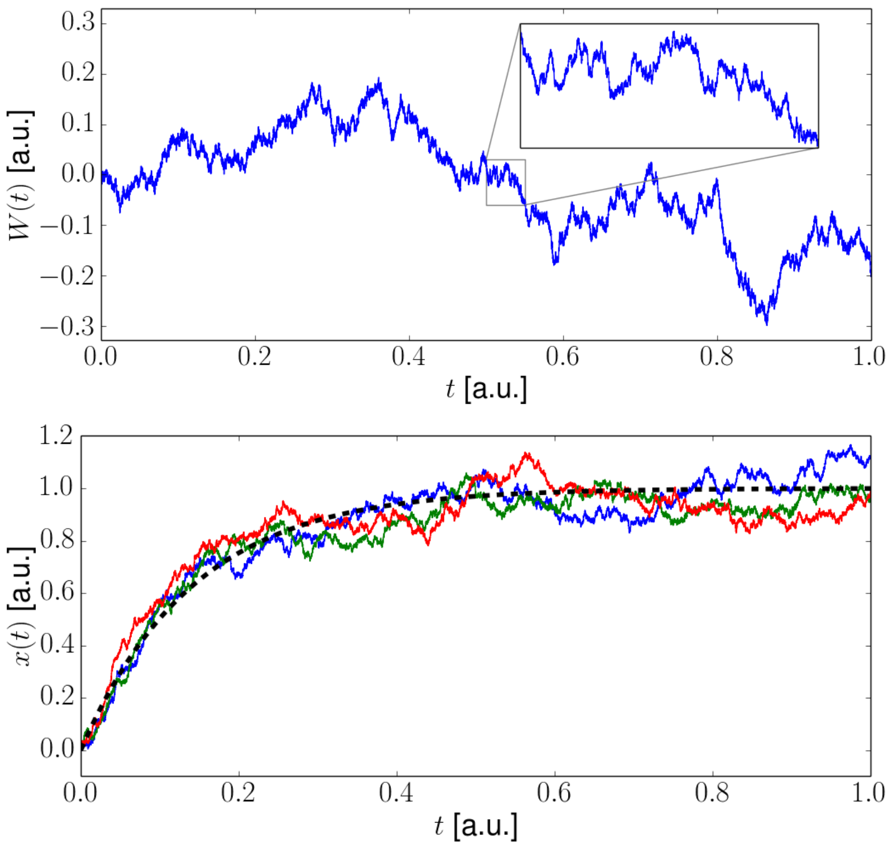


Figure 6.30.: Time-continuous stochastic processes. **Top:** The Wiener process is the time-continuous analogue of a 1-D random walk. Due to increments occurring on arbitrarily small time scales, it is self-similar (the inset is a zoom-in on the rectangularly enclosed area). The Wiener process diverges as $t \to \infty$. **Bottom:** The Ornstein-Uhlenbeck process adds an exponential decay (red curve) to the Wiener process. The colored curves represent OU processes with identical parameters and starting conditions, but different random seeds.

The OU process adds an attractor mechanism to the Wiener process by virtue of the $\theta \cdot (\mu - x(t))$ term. This induces an exponential decay towards μ with a time constant θ. Putting the noise and decay terms together (Figure 6.30, bottom panel), it now appears quite intuitive that the resulting trajectory of $x(t)$ could be a good model of noisy neural conductance (and membrane) dynamics.

PDF of the OU process

It is well-known that the PDF of the OU process

$$f(x, t|x_0) = \sqrt{\frac{\theta}{\pi\sigma^2(1 - e^{-2\theta t})}} \exp\left\{\frac{-\theta}{\sigma^2}\left[\frac{(x - \mu + (\mu - x_0)e^{-\theta t})^2}{1 - e^{-2\theta t}}\right]\right\} \tag{6.108}$$

is the unique solution of the Fokker-Planck equation

Fokker-Planck equation

$$\frac{1}{\theta}\frac{\partial f(x,t)}{\partial t} = \frac{\partial}{\partial x}\left[(x-\mu)f\right] + \frac{\sigma^2}{2\theta}\frac{\partial^2 f}{\partial x^2} \tag{6.109}$$

with starting condition $x_0 := x(t=0)$. We will now prove that, under certain assumptions, the distribution of the synaptic input J^{syn} obeys the same Fokker-Planck equation. To this end, we follow an approach similar to Ricciardi and Sacerdote (1979).

Consider the PDF of the synaptic input $f(J^{\mathrm{syn}},t)$. We can use the Chapman-Kolmogorov equation to describe its evolution after a short time interval Δt as an integral over all possible intermediate states J':

$$f(J^{\mathrm{syn}}, t+\Delta t) = \int_{-\infty}^{\infty} f(J^{\mathrm{syn}}, t+\Delta t | J', t) f(J', t) dJ' \quad . \tag{6.110}$$

For a small enough Δt, the probability of the occurrence of multiple spikes within Δt can be neglected. As incoming spikes are assumed to be generated by Poisson processes, the probability of a single spike occurring in Δt is $\Delta t \sum\limits_{i=1}^{n} \nu_i$. By summing over the two possible histories of J^{syn} within Δt (either a single incoming spike or no spike at all), we can use Equation 6.106 to find

$$\begin{aligned} f(J^{\mathrm{syn}}, t+\Delta t | J') = &\left[1 - \Delta t \sum_{i=1}^{n} \nu_i\right] \delta\left[J^{\mathrm{syn}} - J' \exp\left(-\frac{\Delta t}{\tau^{\mathrm{syn}}}\right)\right] \\ &+ \Delta t \sum_{i=1}^{n} \nu_i \delta\left[J^{\mathrm{syn}} - \left(J' + \Delta J_i^{\mathrm{syn}}\right)\exp\left(-\frac{\Delta t}{\tau^{\mathrm{syn}}}\right)\right] \quad . \end{aligned} \tag{6.111}$$

Plugging this into Equation 6.110 and integrating over J' yields

$$\begin{aligned} f(J^{\mathrm{syn}}, t+\Delta t) = &\left(1 - \Delta t \sum_{i=1}^{n} \nu_i\right) \exp\left(\frac{\Delta t}{\tau^{\mathrm{syn}}}\right) f\left[J^{\mathrm{syn}} \exp\left(\frac{\Delta t}{\tau^{\mathrm{syn}}}\right), t\right] \\ &+ \Delta t \sum_{i=1}^{n} \nu_i \exp\left(\frac{\Delta t}{\tau^{\mathrm{syn}}}\right) f\left[J^{\mathrm{syn}} \exp\left(\frac{\Delta t}{\tau^{\mathrm{syn}}}\right) - \Delta J_i^{\mathrm{syn}}, t\right] \quad . \end{aligned} \tag{6.112}$$

Here, we have implicitly used the relation $\delta(\alpha u) \equiv \frac{1}{|\alpha|}\delta(u)$. We can now expand $f(x, t+\Delta t)$ up to first order in Δt

Taylor expansion in Δt

$$f(J^{\mathrm{syn}}, t+\Delta t) \approx f(J^{\mathrm{syn}}, t) + \left.\frac{\partial f(J^{\mathrm{syn}}, t+\Delta t)}{\partial \Delta t}\right|_{\Delta t = 0} \Delta t \tag{6.113}$$

and rearrange the terms to obtain

$$
\begin{aligned}
\frac{f(J^{\mathrm{syn}}, t+\Delta t) - f(J^{\mathrm{syn}}, t)}{\Delta t} &= \left.\frac{\partial f(J^{\mathrm{syn}}, t+\Delta t)}{\partial \Delta t}\right|_{\Delta t=0} \\
&= \left\{ -\sum_{i=1}^{n} \nu_i \exp\left(\frac{\Delta t}{\tau^{\mathrm{syn}}}\right) f\left[J^{\mathrm{syn}} \exp\left(\frac{\Delta t}{\tau^{\mathrm{syn}}}\right), t\right] \right. \\
&+ \left(1 - \Delta t \sum_{i=1}^{n} \nu_i\right) \frac{1}{\tau^{\mathrm{syn}}} \exp\left(\frac{\Delta t}{\tau^{\mathrm{syn}}}\right) f\left[J^{\mathrm{syn}} \exp\left(\frac{\Delta t}{\tau^{\mathrm{syn}}}\right), t\right] \\
&+ \left(1 - \Delta t \sum_{i=1}^{n} \nu_i\right) \frac{1}{\tau^{\mathrm{syn}}} \exp\left(2\frac{\Delta t}{\tau^{\mathrm{syn}}}\right) J^{\mathrm{syn}} \frac{\partial f\left[J^{\mathrm{syn}} \exp\left(\frac{\Delta t}{\tau^{\mathrm{syn}}}\right), t\right]}{\partial J^{\mathrm{syn}} \exp\left(\frac{\Delta t}{\tau^{\mathrm{syn}}}\right)} \\
&+ \sum_{i=1}^{n} \nu_i \exp\left(\frac{\Delta t}{\tau^{\mathrm{syn}}}\right) f\left[J^{\mathrm{syn}} \exp\left(\frac{\Delta t}{\tau^{\mathrm{syn}}}\right) - \Delta J_i^{\mathrm{syn}}, t\right] \\
&+ \left.(\ldots)\Delta t \right\}_{\Delta t=0} \quad . \qquad (6.114)
\end{aligned}
$$

By taking the limit $\Delta t \to 0$, we obtain:

$$
\frac{\partial f(J^{\mathrm{syn}}, t)}{\partial t} = \frac{1}{\tau^{\mathrm{syn}}} \frac{\partial}{\partial J^{\mathrm{syn}}} \left[J^{\mathrm{syn}} f(J^{\mathrm{syn}}, t)\right] + \sum_{i=1}^{n} \nu_i \left[f(J^{\mathrm{syn}} - \Delta J_i^{\mathrm{syn}}, t) - f(J^{\mathrm{syn}}, t)\right] \quad . \qquad (6.115)
$$

Taylor expansion in w (Kramers-Moyal expansion)

In the limit of small synaptic weights (i.e., $\Delta J_i^{\mathrm{syn}} \to 0$), we can expand the second term on the RHS up to the second order in $\Delta J_i^{\mathrm{syn}}$. Such a Taylor expansion of a stochastic differential equation is often called a Kramers-Moyal expansion. This yields, after some rearrangement

$$
\frac{\partial f(J^{\mathrm{syn}}, t)}{\partial t} = \frac{1}{\tau^{\mathrm{syn}}} \frac{\partial}{\partial J^{\mathrm{syn}}} \left[\left(J^{\mathrm{syn}} - \sum_{i=1}^{n} \nu_i \Delta J_i^{\mathrm{syn}} \tau^{\mathrm{syn}}\right) f(J^{\mathrm{syn}}, t)\right] + \frac{\sum_{i=1}^{n} \nu_i (\Delta J_i^{\mathrm{syn}})^2}{2} \frac{\partial^2 f(J^{\mathrm{syn}}, t)}{\partial (J^{\mathrm{syn}})^2} \quad , \qquad (6.116)
$$

which is the exact equivalent of the Fokker-Planck equation of the OU process (Equation 6.109). Since $u(t)$ is only a linear transformation of $J^{\mathrm{syn}}(t)$, it can also be approximated by an OU process in the limit of large input frequencies and small synaptic weights, with

Equations 6.104 and 6.116 giving the specific time constant, mean value and variance

OU parameters of a Poisson-driven membrane in the HCS

$$\theta = \frac{1}{\tau^{\text{syn}}} \quad , \tag{6.117}$$

$$\mu = \frac{I^{\text{ext}} + g_\text{l} E_\text{l} + \sum\limits_{i=1}^{n} \nu_i w_i E_i^{\text{rev}} \tau^{\text{syn}}}{\langle g^{\text{tot}} \rangle} \quad , \tag{6.118}$$

$$\frac{\sigma^2}{2} = \frac{\sum\limits_{i=1}^{n} \nu_i \left[w_i \left(E_i^{\text{rev}} - \mu \right) \right]^2 \tau^{\text{syn}}}{2 \left\langle g^{\text{tot}} \right\rangle^2} \quad . \tag{6.119}$$

Note how the dominant time constant of the membrane dynamics in the HCS is explicitly no longer the membrane time constant, but the synaptic one. This is to be expected, since the membrane time constant vanishes in the limit of large synaptic conductances. PSPs then become nearly exponentially shaped, with a decay governed by τ^{syn}. This is precisely why we were able to transfer the OU dynamics of the total synaptic current, which is a sum of exponentials, to the membrane potential, which in the HCS is a (scaled) sum over the exact same exponential kernels.

τ^{syn} dominates membrane dynamics

It is instructive to point out how our derivation of the formal analogy between Poisson-driven membrane dynamics and the OU process has produced the same equilibrium distribution as we have already derived with a different formalism in Section 4.3 (see, in particular, the final results in Section 4.3.4, of which the above results for the HCS are a particular subcase). This is, of course, no surprise, since in both cases we were treating the same physical system, but such an agreement between different methodologies still offers a very satisfactory cross-confirmation. However, the equivalence to an OU process gives us insight not only into the (temporal evolution of) the distribution of Poisson-driven membranes, but also into their dynamics. Later on, this will allow us to answer questions such as "How much time does it take the membrane to get from some potential u_1 to some other potential u_2?". The problem of passage times from one state to another will turn out to be central to our derivation of the activation function of LIF neurons.

We conclude this section with two important comments. Firstly, for the above methodology to be generally applicable, we must be able to take the limit $\Delta J_i^{\text{syn}} \to 0$ for arbitrary first and second moments of $f(J^{\text{syn}}, t)$ without modifying them. This is possible if at least one excitatory and one inhibitory input is present, which then give us two degrees of freedom with a proper choice of $\nu^{\text{exc}} \to \infty$ and $\nu^{\text{inh}} \to \infty$. Secondly, all higher moments (3 and above) need to vanish in the abovementioned limit. This has been shown to also be the case under the above conditions (Lánský, 1997).

Note also how we have implicitly required high firing rates when taking the limit of small synaptic weights. This is not only a simple formality, but actually absolutely necessary to guarantee the possibility of random jumps on very small time scales. Only in this way can we obtain a formal equivalence to the Wiener process, which features jumps on arbitrarily small time scales (see our earlier notes on the self-similarity of the Wiener process).

validity of OU approximation only for $\nu_{\text{syn}} \to \infty$

Now that we have a comprehensive description of free membrane dynamics under Poisson bombardment, we can move on to analyzing what happens when a firing threshold is introduced.

6.5.3. Derivation of the Activation Function

Let us start with a simple visual inspection of the dynamics of an LIF neuron with absolute refractoriness in the HCS when the firing threshold lies well within its free membrane potential distribution. Figure 6.31A shows the time course of the membrane potential, as well as the firing times of the neuron.

bursting mode

We can clearly distinguish between two "behavioral modes" of the neuron. The first mode can be classified as "burst spiking", in which the neuron fires at nearly maximum frequency. In this bursting mode, multiple spikes occur in rapid succession with an expected ISI of approximately τ_ref. In order to gain a precise measure of the average ISI in the bursting mode, we need to also take into account the small but nonzero time τ_k^b that the membrane needs to jump from the reset ϱ to the threshold potential ϑ after the kth spike within the burst. The expected ISI after the kth spike in a burst therefore reads

expected ISI within a burst

$$\langle \Delta t_k \rangle := \langle t_{k+1} - t_k \rangle_\mathrm{b} = \tau_\mathrm{ref} + \overline{\tau_k^\mathrm{b}} \quad , \tag{6.120}$$

where the average is taken over many bursts and $\overline{\tau_k^\mathrm{b}}$ represents the average drift time from the reset to the threshold potential following the kth refractory period within a burst. Since spikes occur if and only if the free membrane potential lies above the threshold, we have

$$u_\mathrm{eff}(t_k) \geq \vartheta \tag{6.121}$$

and also, for all but the last spike,

$$u_\mathrm{eff}(t_k + \tau_\mathrm{ref}) \geq \vartheta \quad . \tag{6.122}$$

The burst ends precisely when the free membrane potential after the last refractory period lies below the threshold. The second mode appears between such bursts, where the membrane potential evolves freely in the subthreshold regime.

subthreshold mode

We can now define, just like in the abstract model, that the neuron is in the state $z = 1$ for a duration τ_ref following a spike. The average time spent by the neuron in the refractory state $\bar{t}(z = 1)$ can be taken as a weighted average over burst lengths:

average refractory time in bursts

$$\bar{t}_{(z=1)} = \sum_{n=1}^{\infty} P_n \cdot n \cdot \tau_\mathrm{ref} \quad , \tag{6.123}$$

where P_n represents the distribution of burst lengths, conditioned on the existence of the first spike. Since each burst is followed by a subthreshold period, the average duration of a 1-state followed by a 0-state is given by a similar weighted average:

average burst + subthreshold duration

$$\bar{t}_{(z\in\{0,1\})} = \sum_{n=1}^{\infty} P_n \cdot \left(n\tau_\mathrm{ref} + \sum_{k=1}^{n-1} \overline{\tau_k^\mathrm{b}} + T_n \right) \quad , \tag{6.124}$$

where the 1-states are given by $n\tau_\mathrm{ref}$ and the 0-states by the sum of all jump times $\overline{\tau_k^\mathrm{b}}$ within the burst and the mean time interval T_n between the end of a burst (i.e., the

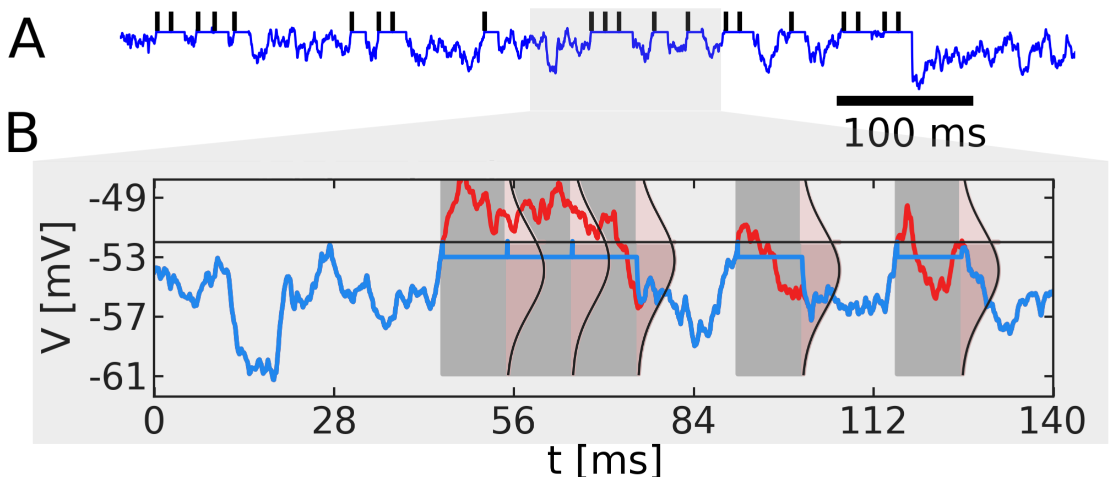


Figure 6.31.: Firing patterns and membrane dynamics in the HCS. **A:** Spike train and membrane potential. The neuron alternates between burst firing and free subthreshold evolution of the membrane potential. **B:** Zoom-in on the membrane dynamics around a burst. The blue and red curves depict the "true" and free membrane potential trajectories, respectively. The gray areas represent refractory periods. At the end of each refractory period, the predicted free membrane potential distribution $p(u_{i+1}|u_i)$ is plotted in pink. The light pink areas are used for computing P_n from P_{n-1} in the ACP formalism. The dark pink areas are used for computing T_n.

endpoint of its last refractory period) and the next spike. The activation function, which gives the relative probability of the neuron being in a 1-state, is therefore:

$$p(z=1) = \frac{\sum_{n=1}^{\infty} P_n \cdot n \cdot \tau_{\mathrm{ref}}}{\sum_{n=1}^{\infty} P_n \cdot \left(n\tau_{\mathrm{ref}} + \sum_{k=1}^{n-1} \overline{\tau_k^{\mathrm{b}}} + T_n \right)} \quad . \tag{6.125}$$

activation function

The variables P_n, T_n and $\overline{\tau_k^{\mathrm{b}}}$ depend on all neuron and noise parameters, but for calculating the activation function, we only need to vary $\bar{u}$.

We can now calculate both P_n and T_n iteratively. The idea behind this approach is to propagate the membrane potential PDF from spike to spike within a burst and cut off the irrelevant parts for a particular burst length n. Figure 6.31B shall serve as visual guidance for our autocorrelation propagation (ACP) formalism.

autocorrelation propagation

We denote the spike times within a burst of length n by $t_0, \ldots, t_{n-1}$ and the endpoint of such a burst by $t_n := t_{n-1} + \tau_{\mathrm{ref}}$. For brevity, we also use $u_i := u(t_i)$. Assuming a first spike at some time t_0 ($u_0 := u(t_0) = \vartheta$), a "burst" of length $n = 1$ requires a subthreshold free membrane potential after the first refractory period ($u_1 := u(t_0 + \tau_{\mathrm{ref}}) < \vartheta$). This occurs with the probability

P_1

$$\begin{aligned} P_1 &:= p(u_1 < \vartheta | u_0 = \vartheta) \\ &= \underbrace{\int_{-\infty}^{\vartheta} du_1 p(u_1 | u_0 = \vartheta)}_{\mathcal{I}_1}, \end{aligned} \tag{6.126}$$

where $p(u_{i+1}|u_i) := f(u, \tau_{\text{ref}}|u_i)$, which is given by Equation 6.108. As we shall see shortly, each probability P_n can be given as a product of a recursive factor and double integral (for P_1, it is a single one), which we denote by $\mathcal{I}_n$.

On average, the neuron then stays in the subthreshold regime for a period equal to the mean first passage time (FPT) from u_1 to ϑ, so the mean duration of the time interval until the onset of the next burst can be expressed as

T_1

$$T_1 = \int_{-\infty}^{\vartheta} du_1 p(u_1 | u_0 = \vartheta) \langle T(\vartheta, u_1) \rangle \quad . \tag{6.127}$$

first-passage time

The FPT problem of the OU process has often been discussed (see, e.g., Thomas, 1975). While no closed-form expression for the distribution of FPTs

$$T(b, a) := \inf t \geq 0 : x(t) = b | x(0) = a \tag{6.128}$$

is known, its moments can be computed analytically (see, e.g., Ricciardi and Sato, 1988). In particular, the mean FPT reads

$$\langle T(b, a) \rangle = \frac{\theta}{\sigma} \sqrt{\frac{\pi}{2}} \int_a^b dx \exp\left[\frac{(x-\mu)^2}{2\sigma^2}\right] \left[1 + \text{erf}\left(\frac{x-\mu}{\sqrt{2}\sigma}\right)\right] \quad . \tag{6.129}$$

A burst of $n = 2$ spikes can only occur when the effective membrane potential lies above the spiking threshold ($u_1 \geq \vartheta$) after the first refractory period and below after the second ($u_2 < \vartheta$). This makes P_2 and T_2 recursive functions of P_1:

P_2

$$\begin{aligned} P_2 &= p(u_2 < \vartheta, u_1 \geq \vartheta | u_0 = \vartheta) \\ &= p(u_1 \geq \vartheta | u_0 = \vartheta)\, p(u_2 < \vartheta | u_1 \geq \vartheta, u_0 = \vartheta) \\ &\overset{(6.126)}{=} \underbrace{(1 - \mathcal{I}_1)}_{=1-P_1} \underbrace{\int_{\vartheta}^{\infty} du_1 p(u_1 | u_1 \geq \vartheta) \left[\int_{-\infty}^{\vartheta} du_2 p(u_2 | u_1)\right]}_{\mathcal{I}_2} \end{aligned} \tag{6.130}$$

T_2

$$T_2 = \int_{\vartheta}^{\infty} du_1 p(u_1 | u_1 \geq \vartheta) \left[\int_{-\infty}^{\vartheta} du_2 p(u_2 | u_2 > \vartheta, u_1) \langle T(u_2, \vartheta) \rangle\right], \tag{6.131}$$

where $p(u_i | u_i \geq \vartheta)$ is a shorthand notation for $p(u_i | u_i \geq \vartheta, u_{i-1} \geq \vartheta, \ldots, u_1 \geq \vartheta, u_0 = \vartheta)$. In particular, this represents a renormalization of the PDF of the effective membrane potential to values above the spiking threshold after i refractory periods.

We can now continue this recursion up to an arbitrary burst length and write

$$\begin{aligned}
P_n &= p(u_n < \vartheta, u_{n-1} \geq \vartheta, \dots, u_1 \geq \vartheta | u_0 = \vartheta) \\
&= p(u_1 \geq \vartheta | u_0 = \vartheta)\, p(u_n < \vartheta, u_{n-1} \geq \vartheta, \dots, u_2 \geq \vartheta | u_1 \geq \vartheta, u_0 = \vartheta) \\
&\overset{(6.126)}{=} (1 - \mathcal{I}_1)\, p(u_2 \geq \vartheta | u_1 \geq \vartheta, u_0 = \vartheta) \\
&\qquad \cdot p(u_n < \vartheta, u_{n-1} \geq \vartheta, \dots, u_3 \geq \vartheta | u_2 \geq \vartheta, u_1 \geq \vartheta, u_0 = \vartheta) \\
&\overset{(6.130)}{=} (1 - \mathcal{I}_1)(1 - \mathcal{I}_2)\, p(u_3 \geq \vartheta | u_2 \geq \vartheta, u_1 \geq \vartheta, u_0 = \vartheta) \\
&\qquad \cdot p(u_n < \vartheta, u_{n-1} \geq \vartheta, \dots, u_4 \geq \vartheta | u_3 \geq \vartheta, \dots, u_1 \geq \vartheta, u_0 = \vartheta) \\
&= \prod_{i=1}^{n-1} (1 - \mathcal{I}_i)\, p(u_n < \vartheta | u_{n-1} \geq \vartheta, \dots, u_1 \geq \vartheta, u_0 = \vartheta) \qquad (6.132) \\
&= (1 - \sum_{i=1}^{n-1} P_i) \underbrace{\int_\vartheta^\infty du_{n-1} p(u_{n-1} | u_{n-1} \geq \vartheta) \left[\int_{-\infty}^\vartheta du_n p(u_n | u_{n-1}) \right]}_{\mathcal{I}_n} \qquad (6.133)
\end{aligned}$$

P_n

$$T_n = \int_\vartheta^\infty du_{n-1} p(u_{n-1} | u_{n-1} \geq \vartheta) \left[\int_{-\infty}^\vartheta du_n p(u_n | u_n < \vartheta, u_{n-1}) \langle T(u_n, \vartheta) \rangle \right] \qquad (6.134)$$

T_n

The transition from a product to a sum between (6.132) and (6.133) requires the identity

$$\prod_{i=1}^{n-1} (1 - \mathcal{I}_i) = 1 - \sum_{i=1}^{n-1} P_i \quad , \qquad (6.135)$$

which can be easily shown by induction from $P_n = \mathcal{I}_n \prod_{i=1}^{n-1} (1 - \mathcal{I}_i)$ (Equation 6.132) and $P_1 = \mathcal{I}_1$ (Equation 6.126). In practice, because $\lim_{n \to \infty} P_n = 0$, one can stop the recursion at some small enough P_n.

What remains to be calculated is the average time-to-threshold $\overline{\tau_k^\mathrm{b}}$ within a burst that follows the kth refractory period. Since we assume a HCS, we are looking at a regime in which $\tau_\mathrm{eff} \ll \tau^\mathrm{syn}$. Therefore, we can assume u_eff to be approximately unchanged during the short time interval $\overline{\tau_k^\mathrm{b}}$ (adiabatic approximation, see also Moreno-Bote and Parga, 2004). For a fixed u_k, the jump time can be easily calculated from Equation 6.97 (see also Equation 2.36):

adiabatic approximation

$$\tau_k^\mathrm{b}(u_k) = \tau_\mathrm{eff} \ln \left(\frac{\varrho - u_k}{\vartheta - u_k} \right) \quad . \qquad (6.136)$$

The average jump time can then be obtained by integrating over all suprathreshold values of u_k, which in turn have probabilities that follow from integrating over all suprathreshold values of u_{k-1} :

$$\overline{\tau_k^\mathrm{b}} = \tau_\mathrm{eff} \int_\vartheta^\infty du_k \ln \left(\frac{\varrho - u_k}{\vartheta - u_k} \right) \int_\vartheta^\infty du_{k-1} p(u_k | u_k > \vartheta, u_{k-1}) \quad . \qquad (6.137)$$

$\overline{\tau_k^\mathrm{b}}$

With Equations 6.125, 6.133, 6.134, 6.137 and 6.129, one could now predict the activation function of an LIF unit in an extreme high-noise regime ($\tau_\mathrm{eff} \to 0$). We can, however, generalize our approach by taking the finite nature of the effective time constant into account.

If we go back to Equation 6.97 and leave $\tau_{\text{eff}} = C/\left\langle g^{\text{tot}} \right\rangle$ small but finite, we can still perform all the remaining approximations, but are required to modify Equation 6.104:

$$\tau_{\text{eff}} \dot{u}(t) = \frac{I^{\text{ext}} + g_l E_l}{\langle g^{\text{tot}} \rangle} + \frac{J^{\text{syn}}(t)}{\langle g^{\text{tot}} \rangle} - u(t) \quad . \tag{6.138}$$

Together with Equation 6.106, we now have a system of first-order ODEs which can be solved analytically by standard techniques (variation of constants). We have already provided a detailed discussion of these systems in Section 4.2. The PSPs are no longer a linear transformation of the exponentially shaped PSCs, but rather alpha-shaped (see Equation 4.59):

$$u_s(t) = \Theta(t - t_s) A \frac{\left(e^{-\frac{t-t_s}{\tau_{\text{eff}}}} - e^{-\frac{t-t_s}{\tau^{\text{syn}}}} \right)}{\tau_{\text{eff}} - \tau^{\text{syn}}} \quad , \tag{6.139}$$

with

$$A = \frac{w_i (E_i^{\text{rev}} - \langle u_{\text{eff}} \rangle) \tau^{\text{syn}}{}_i}{\langle g^{\text{tot}} \rangle} \quad . \tag{6.140}$$

This shape causes a lower PSP peak than in the case of exponential PSPs, decreasing the overall width of the membrane potential distribution. Intuitively, this would result in a horizontal shift and compression of the activation function.

More recently, analytical treatments of these phenomena have been proposed (Burkitt, 2006). In these approaches, authors usually consider large membrane time constants (equivalent to a long τ_{eff}) and small synaptic time constants. However, Equation 6.139 is symmetric in τ_{eff} and τ^{syn}, so the same argument applies to our case as well, but the two time constants need to be switched. It is, for example, possible to correct the FPT from the reset to the threshold potential by using an expansion in $\sqrt{\tau'/\tau}$, with τ' and τ being the smaller and the larger of the two time constants, respectively (Brunel and Sergi, 1998):

$$\langle T(\vartheta, u) \rangle = \tau \sqrt{\pi} \int_{\frac{u-\mu}{\sigma}}^{\frac{\vartheta_{\text{eff}}-\mu}{\sigma}} dx \exp(x^2)[\text{erf}(x) + 1] \quad , \tag{6.141}$$

with μ and σ^2 the first two moments of the free membrane potential distribution and an effective threshold

FPT correction

$$\vartheta_{\text{eff}} \approx \vartheta - \zeta\left(\frac{1}{2}\right) \sqrt{\frac{\tau'}{2\tau}} \sigma \quad , \tag{6.142}$$

in which ζ denotes the Riemann zeta function. In our particular case, the expansion is done in $\sqrt{\tau_{\text{eff}}/\tau^{\text{syn}}}$, so $\tau' = \tau_{\text{eff}}$ and $\tau = \tau^{\text{syn}}$. Note how Equation 6.141 is equivalent to a change of the integration variable and limits in the original equation 6.129 for the FPT.

With this approximation, we assume that u converges from ρ to u_{eff} in negligible time after it is released from the refractory state. Afterwards, its convergence to u_{eff} is determined by Equation 6.97.

We now have at our disposal a comprehensive prediction of the activation function of LIF neurons with absolute refractoriness. Our derivations were based on COBA synapses, but the general formalism is identically applicable to CUBA synapses, as long as the absolute refractory time following a spike is nonzero.

Figure 6.32 shows our prediction of the activation function for several different parameter sets and compares it to simulation data. Apart from the good predictions produced by our approach, we point out subplots A and F in particular.

symmetry of the activation function

In Figure 6.32 A, our neuron receives strong synaptic background stimulus, which puts the neuron into a pronounced HCS with an extremely short effective time constant. It turns out that such accelerated membrane dynamics are the key ingredient to LIF sampling, as they allow the neuron to spike again almost immediately after being refractory, in some sense creating a symmetry between 0- and 1-states. In turn, this causes the activation function to become symmetric and thereby well-fitted by a logistic curve (see also Figure 6.32 G). This represents a novel and extremely useful computational feature of the HCS.

In Figure 6.32 F, we show what happens in an "imperfect" HCS, where the 0-state is favored, since the reset-to-threshold jump time becomes large. Not only does the activation function become asymmetric, but it also converges to 1 only very slowly, impeding any meaningful mapping to the abstract sampling model. However, the activation function can at least partly be fitted by a logistic function, in the region below $p(z = 1) = 1/2$. This would still allow sampling from certain Boltzmann distributions, but only under severe parameter restrictions, i.e., predominantly negative biases and weights.

The formalism of our prediction of the activation function was motivated and guided by the interpretation of spikes as samples from binary RVs. Of course, the calculation of activation functions is an interesting problem in itself and we are not the first to perform such analytical studies. The most relevant results obtained in this context can be found in Brunel and Sergi (1998) and Moreno-Bote and Parga (2004). However, both of these studies are based on particular assumptions, which impose tight restrictions on their predictive power, as also depicted in Figure 6.32.

Brunel and Sergi (1998)

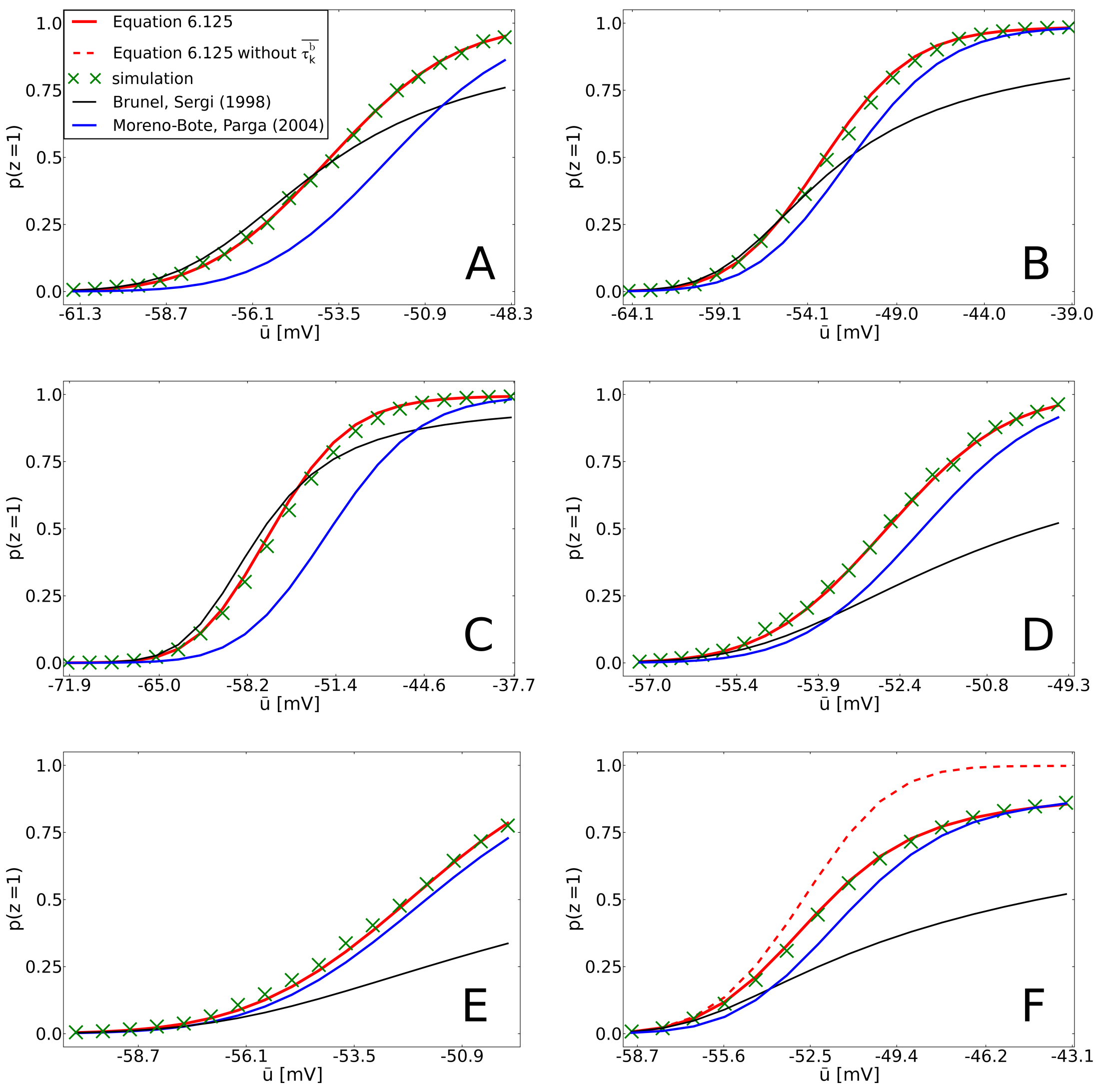
In Brunel and Sergi (1998) (from which we have borrowed the FPT correction in Equation 6.141), an essential assumption is that synaptic time constants are very small ($\tau^{\text{syn}} \to 0$). The reset then causes the membrane to forget all (or at least most of) its previous history, which eases the calculation of reset-to-threshold passage times, since they become largely independent of previous events. This is explicitly not the case in our regime, where the synaptic time constants are required to be comparable to refractory times (see Equation 6.93), and precisely the reason why our comparatively complicated ACP formalism was required.[26] Since Brunel and Sergi (1998) do not account for the fact that a neuron is more likely to spike if it has already spiked within the last τ_{ref} (which the ACP formalism accounts for by explicitly computing $p(u_{i+1}|u_i)$), their prediction systematically underestimates the output firing frequency (which is most pronounced for high frequencies at large $\bar{u}$). We can also see that when τ^{syn} becomes smaller, their prediction becomes more accurate (Figure 6.32 C).

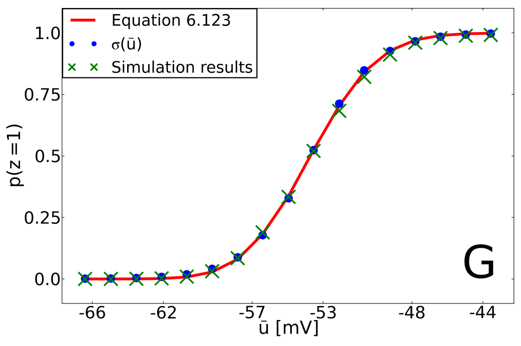
[26] Strictly speaking, it is only the synaptic time constant of the recurrent interaction between the sampling neurons that has to be equal (or at least similar) to the refractory time. The noise synapses could have, in principle, arbitrary time constants. However, we still require the neurons to operate in the HCS, so it is likely that even very short $\tau^{\text{syn}}_{\text{noise}}$ will dominate membrane dynamics. Another reason for using the same τ^{syn} for both noise and recurrent connections is purely practical: PyNN assigns every neuron a single excitatory and a single inhibitory time constant.

Moreno-Bote and Parga (2004)

Moreno-Bote and Parga (2004), on the other hand, analyze the opposite case of long τ^{syn} and short τ_{m}. This is why they are able to use the adiabatic (or quasistatic) approximation, since the membrane potential, which is governed by τ_{m}, reacts so quickly to u_{eff}, which is governed by τ^{syn}. The latter then effectively appears static from the point of view of the former. Indeed, we also make use of the adiabatic approximation for the calculation of our $\overline{\tau_k^{\mathrm{b}}}$. However, Moreno-Bote and Parga (2004) explicitly set $\tau_{\mathrm{ref}} = 0$, which, in a sense, is the exact opposite of the situation from Brunel and Sergi (1998). Without refractoriness, the neuron may spike with very high frequencies, allowing the membrane potential to "see" u_{eff} very often before it changes, as would be required by the quasistatic assumption. This is explicitly no longer the case when τ_{ref} is in the same order as τ^{syn}, since u_{eff} can then change significantly within a refractory period, effectively invalidating quasistationarity. The implications for the activation function are a bit more subtle, but the pure adiabatic approach turns out to overestimate the probability of long ISIs, since it has no immediate concept of bursts. This, in turn, causes a systematic underestimation of the activation function. As refractory times become shorter (or synaptic time constants longer), the quasistationary scenario is restored and the predictions improve (Figure 6.32 D and E).

Figure 6.32.: Comparison of our prediction of the activation function (Equation 6.125) to simulation data, as well as to the predictions given by Brunel and Sergi (1998) and Moreno-Bote and Parga (2004), for several parameter sets. **(A)** Standard parameter set as given in Table A.20. This panel uses the same simulation results as panel F. **(B)** Same as A, but with doubled membrane capacitance C_{m} and halved leak conductance g_{l}, input rates $\nu_{\mathrm{exc,inh}}$ and weights $w_{\mathrm{exc,inh}}$. This parameter set has the effect of slowing the membrane, i.e., increasing τ_{eff} by a factor of 16. **(C)** Same as A, but with a decreased synaptic time constant $\tau^{\mathrm{syn}} = 3\,\mathrm{ms}$. The prediction from Brunel and Sergi (1998) is improved, since the correlations in the pre- and post-refractory effective membrane potential are smaller in this scenario. **(D)** Same as A, but with an increased synaptic time constant $\tau^{\mathrm{syn}} = 30\,\mathrm{ms}$. The prediction from Brunel and Sergi (1998) deteriorates due to the longer-range membrane potential autocorrelation. Conversely, the prediction from Moreno-Bote and Parga (2004) improves somewhat, since the refractory time becomes shorter relative to the synaptic time constant. **(E)** Same as A, but with a very short refractory time $\tau_{\mathrm{ref}} = 1\,\mathrm{ms}$. Here, we enter the parameter range where Moreno-Bote and Parga (2004) provides good predictions, as argued above. **(F)** Same as A, but with the input rates $\nu_{\mathrm{exc,inh}}$ and weights $w_{\mathrm{exc,inh}}$ decreased by a factor of 10, thereby slowing the membrane considerably (imperfect HCS). Additionally, we have chosen a large reset-to-threshold distance of $u_{\mathrm{thr}} - \varrho = 10\,\mathrm{mV}$. In this scenario, the $\overline{\tau_k^{\mathrm{b}}}$-term in (6.125) becomes dominant and the activation function departs from the logistic shape that it has in the HCS. **(G)** Logistic fit of the activation function.


Equation 6.125
Equation 6.125 without $\overline{\tau_k^b}$
simulation
Brunel, Sergi (1998)
Moreno-Bote, Parga (2004)
p(z=1)
ū [mV]
A
B
C
D
E
F


Equation 6.123
σ(ū)
Simulation results
p(z=1)
ū [mV]
G

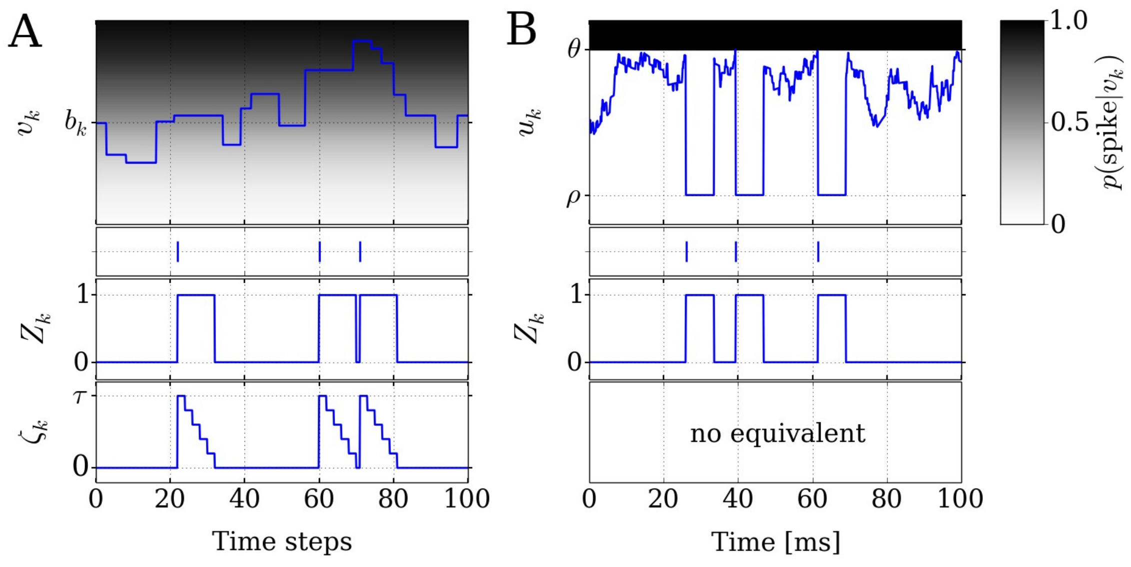


Figure 6.33.: Neural sampling: abstract model vs. implementation with LIF neurons. **(A)** Example dynamics of all the variables associated with an abstract model neuron. The neuron spikes stochastically as a function of its membrane potential, which is not reset by outgoing spikes. The refractoriness variable was introduced as a theoretical tool for representing the model's dynamics as a first-order Markov chain. **(B)** Example dynamics of the equivalent variables associated with an LIF neuron. The neuron fires deterministically when the membrane potential crosses the threshold. However, stochasticity is introduced by adding noise to the membrane. Nevertheless, the membrane potential is reset when the neuron spikes and clamped to some value ρ during refractoriness. However, the synaptic input J^{syn} is not reset by spikes and thereby retains the memory of past events. Figure taken from Probst et al. (2015).

6.5.4. Translation of Biases and Recurrent Synaptic Weights

translation of biases

Now that we have fully specified the dynamics of individual LIF sampling units, we can connect them to form sampling networks. The biases can be set directly with the parameter used to vary $\bar{u}$ in the activation function, usually E_l or I^{ext} (for an exact mapping rule for $b \rightarrow E_\text{l}$, see Section 6.7.3). The synaptic connections, however, require a more careful treatment.

In the abstract model, the synaptic connections were straightforward, as they followed directly from Equation 6.57: PSPs were rectangular and of duration $\tau^{\text{syn}} = \tau_{\text{ref}}$. Even more importantly however, the abstract membrane potential was not reset by outgoing spikes. Figure 6.33 shows a comparison between the dynamics of the abstract and LIF sampling neuron models.

In the LIF model, the reset and subsequent clamping of the membrane potential during refractoriness would seem to erase any information that arrives before the end of the refractory period. However, it is often neglected that LIF neurons have one dynamic variable which is not reset, namely the synaptic input J^{syn}. Therefore, in our model of

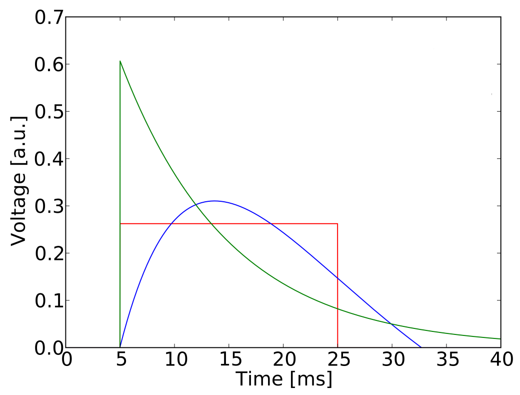


Figure 6.34.: Different PSP shapes. The abstract neuron model uses rectangular PSPs (red curve). For the arguably most commonly used model of exponential PSC kernels, LIF PSPs in the HCS are nearly exponential as well (green curve). Such PSPs are more difficult to deal with, due to their large initial overshoot and their long tails. Some authors therefore use more “benign” PSP shapes, such as cut-off alpha functions (blue curve, see, e.g., Pecevski et al., 2011).

LIF sampling, the effective membrane potential plays the role of the abstract membrane potential and the synapses store the memory of past events. The fast membrane reaction speed enabled by the HCS allows $\bar{u} \approx u_\mathrm{eff}$ at almost all nonrefractory times, such that the neuron effectively always samples from its correct local conditional.

synaptic memory role of the HCS

We therefore only need to establish a translation from the unitless abstract potential to $\bar{u}$. This translation follows straightforwardly from the activation function (which is equivalent to the NCC), which must be identical for both models (see Equations 6.82 and 6.94):

$$\sigma(u_k^\mathrm{abstract}) = p(z_k = 1|z_{\setminus k}) = \sigma\left(\frac{\bar{u}_k - \bar{u}_k^0}{\alpha}\right) \quad . \tag{6.143}$$

The parameters α and $\bar{u}_k^0$ can be directly obtained from the logistic fit to the predicted activation function (see Figure 6.32 G). As a consequence, also synaptic weights need to be rescaled by the factor α.

Additionally, the difference in PSP shapes needs to be taken into account. This difference will corrupt the precise equivalence between the abstract model and LIF sampling, but we can, at least, require that local sampling be correct on average during the refractory period of the corresponding afferent neuron:

average PSP approximation

$$E\left[u_k^\mathrm{abstract}\right]_\mathrm{PSP}\Big|_0^{\tau_\mathrm{ref}} = E\left[\bar{u}_k\right]_\mathrm{PSP}\Big|_0^{\tau_\mathrm{ref}} \quad . \tag{6.144}$$

This is equivalent to setting the area under a PSP (Equation 6.139) during the refractory

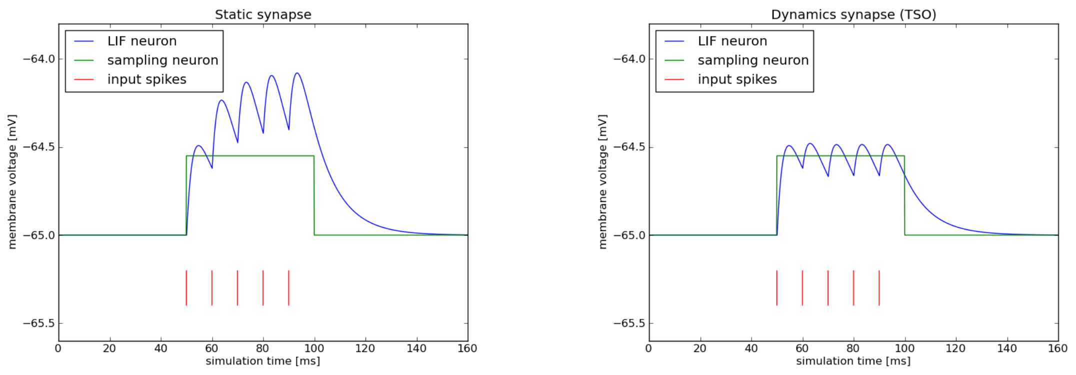


Figure 6.35.: Influence of short-term synaptic plasticity. **Left:** Due to the additive nature of PSPs, static synapses cause a cumulative effect when afferent neurons spike at high rates. This would cause deviations from the target distribution if marginals of individual RVs are too high (i.e., if their associated neurons tend to spike very often). **Right:** STD causes consecutive PSPs to shrink, attenuating additive effects. In particular, appropriate STD parameters can effectively emulate renewing PSPs, thereby allowing neurons to sample locally from correct conditionals (on average).

state of the corresponding afferent neuron (i.e., for a duration τ_{ref}) equal to $W_{kj}\,\tau_{\mathrm{ref}}\,\alpha$:

$$\begin{aligned} W_{kj}\tau_{\mathrm{ref}}\alpha &= \int_0^{\tau_{\mathrm{ref}}} \frac{w_{kj}(E^{\mathrm{rev}}_{kj} - \langle u_{\mathrm{eff}}\rangle)\tau^{\mathrm{syn}}}{\langle g^{\mathrm{tot}}\rangle} \frac{\left(e^{-\frac{t-t_s}{\tau_{\mathrm{eff}}}} - e^{-\frac{t-t_s}{\tau^{\mathrm{syn}}}}\right)}{\tau_{\mathrm{eff}} - \tau^{\mathrm{syn}}} dt \\ &= \frac{w_{kj}\tau^{\mathrm{syn}}}{\langle g^{\mathrm{tot}}\rangle} \frac{\left(E^{\mathrm{rev}}_{kj} - \mu\right)}{\tau_{\mathrm{eff}} - \tau^{\mathrm{syn}}} \left[\tau^{\mathrm{syn}}\left(e^{-\frac{\tau_{\mathrm{ref}}}{\tau^{\mathrm{syn}}}} - 1\right) - \tau_{\mathrm{eff}}\left(e^{-\frac{\tau_{\mathrm{ref}}}{\tau_{\mathrm{eff}}}} - 1\right)\right] \quad . \end{aligned} \tag{6.145}$$

translation of weights

By setting $\tau_{\mathrm{ref}} = \tau^{\mathrm{syn}}$, we obtain the mapping between the abstract and LIF synaptic weight domains:

$$W_{kj} = \frac{1}{\alpha C_{\mathrm{m}}} \frac{w_{kj}\left(E^{\mathrm{rev}}_{kj} - \mu\right)}{1 - \frac{\tau^{\mathrm{syn}}}{\tau_{\mathrm{eff}}}} \left[\tau^{\mathrm{syn}}\left(e^{-1} - 1\right) - \tau_{\mathrm{eff}}\left(e^{-\frac{\tau^{\mathrm{syn}}}{\tau_{\mathrm{eff}}}} - 1\right)\right] \quad . \tag{6.146}$$

A further problem that does not occur in the abstract model is the fact that PSPs from the same afferent neuron are additive. For rectangular PSPs that have the exact duration of the refractory period, superposition can not happen – this is not the case for the more realistic exponential synapse model, where PSPs have a DOE shape (which is almost exponential in the HCS). In order to maintain the average height of additive DOE PSPs, we can make use of short-term plasticity, in particular the TSO model of STD that we have discussed in Section 2.2.2.2. STD can been applied to attenuate the amplitudes of consecutively arriving PSPs from the same afferent neuron, thereby effectively emulating renewing synapses. In order to achieve this behavior, the synaptic efficacy parameter and recovery time constant must be chosen as $U_{\mathrm{SE}} = 1$ and $\tau_{\mathrm{rec}} = \tau^{\mathrm{syn}}$, respectively.

renewing PSPs

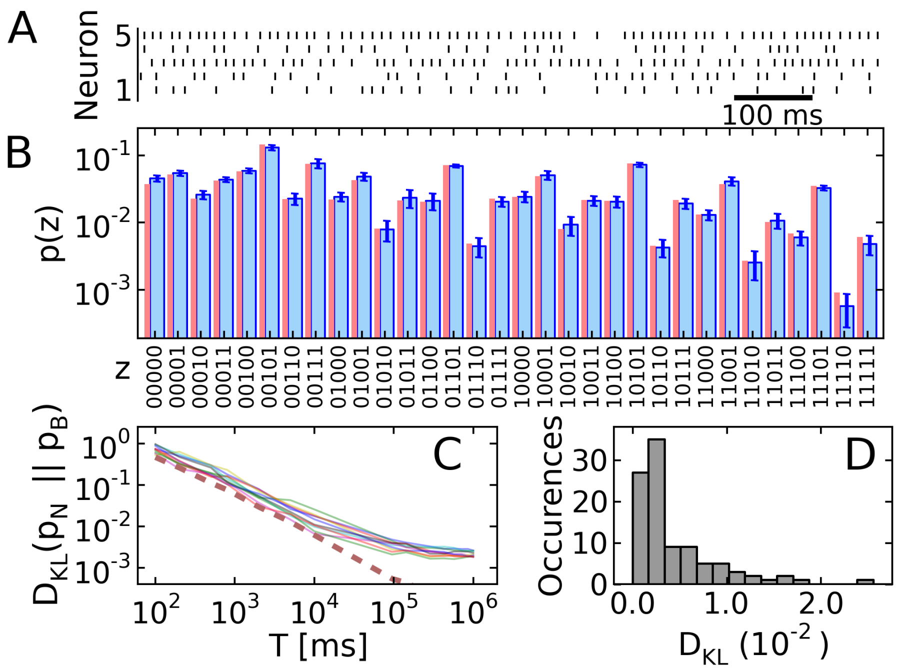


Figure 6.36.: LIF sampling demonstration for exemplary BMs with 5 RVs. **(A)** Spike trains representing the joint state $\boldsymbol{z}(t)$. **(B)** Sampled distribution $p(\boldsymbol{z})$ (blue) after 10 s of network evolution compared to the target distribution $p^*(\boldsymbol{z})$ (red). Error bars are calculated from 10 runs with different random seeds. **(C)** Convergence of the sampled distribution towards the target distribution. 10 runs of the LIF network (continuous lines) are compared to a single run of the abstract model (dashed line) with a comparable number of samples. **(D)** Distribution of $D_{\mathrm{KL}}\left(p(\boldsymbol{z}) \parallel p^*(\boldsymbol{z})\right)$ after sampling has converged (10^6 ms) from 100 different Boltzmann distributions (with randomly drawn parameters) over 5 RVs. Figure taken from Petrovici et al. (2013).

6.5.5. LIF-Based Boltzmann Machines

With our fully specified LIF sampling framework, we can now configure networks of LIF neurons to sample from arbitrary Boltzmann distributions. As a small-scale case study, we analyze the behavior of a 5-neuron network with randomly drawn weights and biases (Figure 6.36).

For the comparison of the sampled distribution to the target distribution, we have chosen an integration time of 10 ms, which is a conservative estimate of the maximum duration a neuronal ensemble will experience stable stimulus conditions in a behaving organism and can thus be expected to sample from a stable target distribution. Within this time frame, the sampled distribution $p(\boldsymbol{z})$ closely matches the set target $p^*(\boldsymbol{z})$ within the statistical errors, over more than two orders of magnitude.

LIF sampling performance

Over longer time scales, the sampled distribution improves, but not to an arbitrary degree of precision. As opposed to exact sampling algorithms such as Gibbs or neural sampling, the LIF-sampled distribution does not converge precisely towards the target distribution. However, the distance to the target distribution, given by the D_{KL} (see Equation 4.160), is small.

PSP shape Two factors exert critical influence on the sampling quality. The main reason for the imperfect convergence is the difference between our biologically realistic PSP shape and the theoretically optimal rectangular one. In particular, the large overshoot at the onset of the PSP and the nonzero tail following the end of the refractory period always induce small, but measurable deviations from the target distribution. These become more pronounced as the parameters of the Boltzmann distribution become more extreme (see also Figure 6.38). The second reason for the systematic deviations lies in the "imperfectness" of the HCS. *imperfect HCS* A slower effective time constant forces additional $z_k = 0$ states between spikes in a burst, which should however represent a continuous $z_k = 1$ state (see also Equation 6.120). The latter issue can be easily dealt with by increasing the synaptic weight of frequency of the noise sources or by reducing the membrane capacity (see also Figure 6.37 A).

An important question remains concerning the performance of LIF sampling over a wide range of parameters. As a default parameter set, we have already chosen biologically relevant values (see Table A.20), which we adopted from Naud et al. (2008), where they were explicitly fitted in order to reproduce experimentally observed neural firing patterns. In Figure 6.37 we demonstrate the robustness of our LIF sampling approach to large variations of critical parameters. Critical parameters are considered to be those which can not be trivially accounted for in the logistic fit of the activation function (Equation 6.143). The reversal potentials, the leak potential and the refractory period are examples of such unproblematic parameters. Parameters such as the membrane capacitance or the distance between reset and firing threshold can, on the other hand, be problematic, since *critical parameters* they can break the symmetry of the activation function.

As expected, the noise-to-capacitance ratio is an important factor, since it governs τ_{eff} the membrane reaction speed. However, a sufficiently pronounced HCS can always be achieved by increasing the background noise (Figure 6.37A).

τ^{syn} The relation between τ^{syn} and τ_{ref} is, by definition, important, since the duration of the PSP, which is governed by τ^{syn} should encode the simultaneous 1-state of the afferent neuron, which is given by τ_{ref}. In the abstract model, the equality of the two (Equation 6.93) therefore follows necessarily, but this is not the case for LIF sampling, where we required the PSP to only be correct on average (Equation 6.144). Indeed, it turns out that our initial choice of setting $\tau^{\mathrm{syn}} = \tau_{\mathrm{ref}}$ also in the LIF model minimizes the $D_{\mathrm{KL}}\left(p(\boldsymbol{z}) \parallel p^*(\boldsymbol{z})\right)$ (Figure 6.37B). However, even large variations in the synaptic time constant do not affect the sampling performance critically, as long as the synaptic weights are properly calibrated (Equation 6.146).

$\vartheta - \varrho$ Finally, the distance between the reset and threshold potentials is also important, for precisely the same reason as the intensity of the HCS. A large $\vartheta - \varrho$ has the same effect of increasing intra-burst ISIs as a low τ_{eff} (see our above discussion of systematic deviations between the sampled and target distributions). This is why a strong enough HCS renders the sampling performance virtually immune to changes of $\vartheta - \varrho$ in a biologically meaningful range (Figure 6.37C).

large fully-connected BMs As we shall discuss later (Section 6.6.2.2), large fully-connected BMs are impractical for training towards functional purposes (and also biologically unlikely). Nevertheless, it is still interesting to observe the firing behavior of our LIF-based BMs when their size is increased to a point where collective effects observably influence ensemble dynamics.

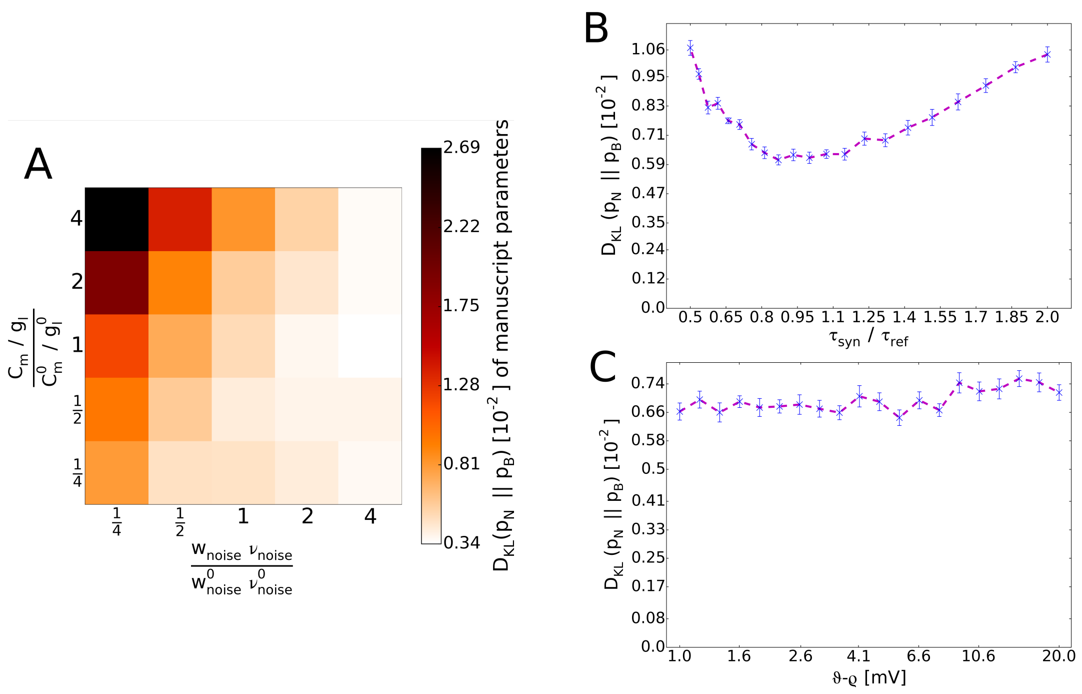


Figure 6.37.: Sampling with LIF neurons over a broad range of relevant model parameters. All plots depict the D_{KL} between the sampled distribution (with a certain set of parameters) and the target distribution. **(A)** Sweep over neuron size and leakage, as well as background input parameters. The axes represent multiplicative scaling values for four parameters: background (Poisson) synaptic weights $w_{\mathrm{exc,inh}}$ and firing rates $\nu_{\mathrm{exc,inh}}$ on the abscissa, neuron capacitance C_{m} and leak conductance g_{l} on the ordinate. The "default" parameter values (Table A.20) therefore have the coordinates $(1, 1)$. The network simulation runtimes were chosen as $T_{sim} = 10^6$ ms. As expected, the large neuron / weak noise scenario (top left square) does not permit accurate sampling, as the activation function is no longer logistic (see Figure 6.32). In general, the plot shows that good sampling quality can be achieved for any neuron capacitance and leak as long as the background noise is strong enough (HCS). **(B)** Sweep over the ratio of the synaptic and refractory time constants. The best sampling performance is, indeed, achieved for $\tau^{\mathrm{syn}} \approx \tau_{\mathrm{ref}}$, but the network still produces good approximations of the target distribution when the two time constants are not precisely identical. All D_{KL} data points result from 20 simulations with runtimes of $T_{sim} = 10^5$ ms each. The error bars represent the standard error of the mean. **(C)** Effect of an increased distance from reset to threshold potential. In general, an increase in $\vartheta - \varrho$ causes a gradual decay of the sampling quality, since the membrane requires additional time to reach a suprathreshold u_{eff} when the refractory period is over. This can, in principle, be accommodated by defining a larger time window τ_{on} during which the neuron is considered to encode the state $z = 1$. Nevertheless, in an HCS, the effective time constant can be low enough to render the threshold-to-reset distance irrelevant.

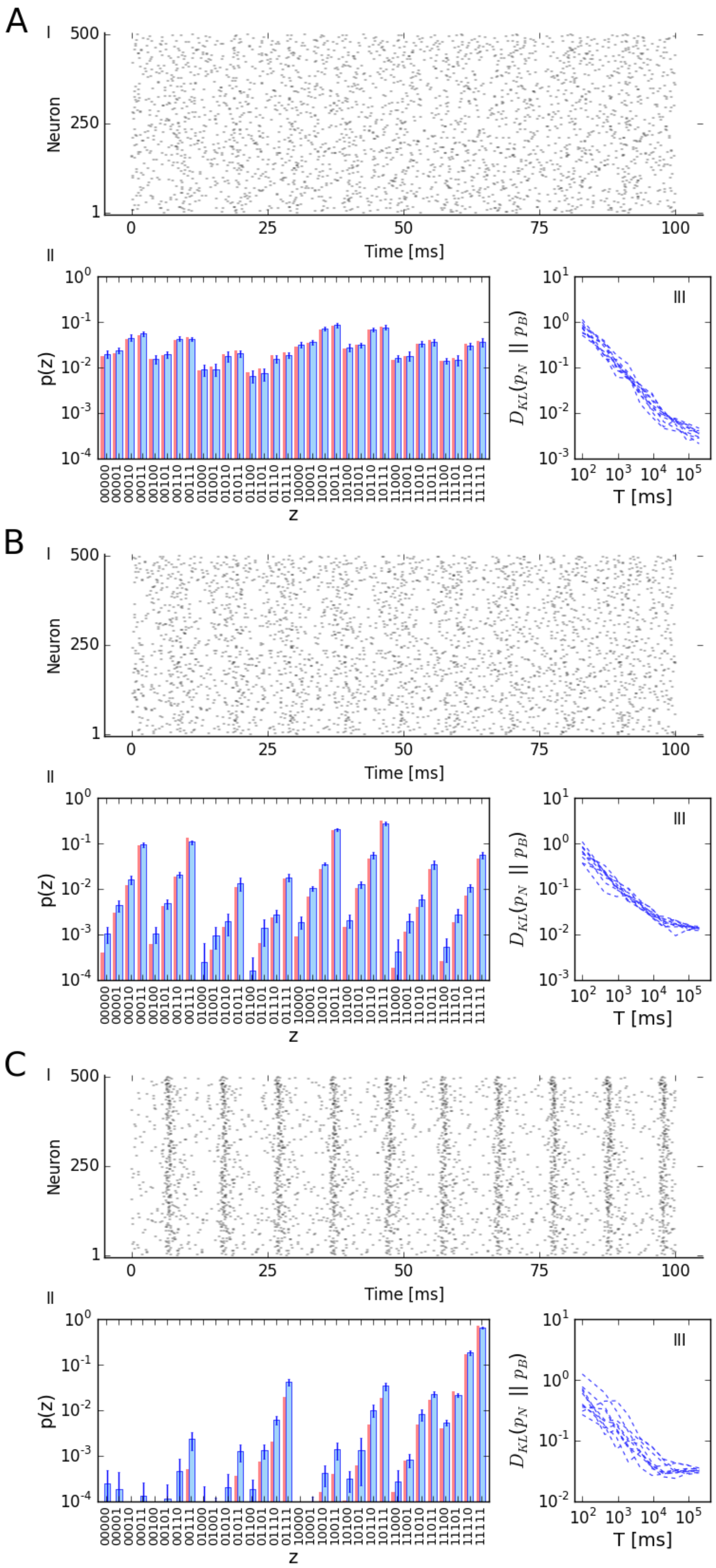
AI states

synchroniza-tion

When interaction is relatively weak, the sampling accuracy is excellent and the firing patterns are asynchronous and irregular (Figure 6.38A). As interaction strength increases, individual modes of the sampled probability distribution become more pronounced and slight synchronization effects appear (Figure 6.38B). The numerical accuracy of the sampled distribution remains very good over at least two orders of magnitude. As expected, for more extreme weights, firing becomes highly synchronous (Figure 6.38C). Due to the large absolute deviations of the LIF PSPs from the theoretically optimal rectangular PSPs, the distance between the sampled and the target distribution increases as well, but the network still provides a useful approximation of the highest-probability modes.

Apart from highlighting the quality of LIF sampling under challenging conditions, these observations offer an interesting complement to existing theories of synchronization in LIF networks (see, in particular, Brunel, 2000), especially since they happen in a biologically relevant, but theoretically not well-studied regime (see, e.g., Destexhe et al., 2003, for in-vivo measurements of the HCS). An analytical treatment comparable to Brunel (2000) should be feasible given the accurate predictions of our ACP formalism and constitutes an interesting venue for future theoretical research.

Figure 6.38.: Large BMs: 500 neurons, 10% connectivity. The distribution over 5 RVs (out of 500) obtained from the simulation of the LIF network ($T_{\text{sim}} = 10^4\,\text{ms}$) is plotted alongside the one obtained by Gibbs sampling from the target distribution. Error bars representing the standard error of the mean are obtained from 10 network simulations with different random seeds. Since the target distribution can not be computed analytically for such a large number of RVs, we define the Gibbs-sampled distribution after 10^6 sampling steps as the target distribution when computing the D_{KL}. Here, we study three different scenarios with increasing average synaptic weights. Biases were drawn from a beta distribution: $b_k \sim 1.2 \cdot [\mathcal{B}(0.5, 0.5) - 0.5]$. Weights were drawn from increasingly broad beta distributions of identical shape $\mathcal{B}(0.5, 0.5)$. (A) Small weights: $W_{kj} \sim 0.6 \cdot [\mathcal{B}(0.5, 0.5) - 0.5]$. Due to the rather small weights, the probabilities of the different states approximately span an order of magnitude. The network samples from the target distribution with a high degree of accuracy. (B) Intermediate weights: $W_{kj} \sim 1.2 \cdot [\mathcal{B}(0.5, 0.5) - 0.5]$. The still balanced synaptic connectivity allows the network to remain in an approximately asynchronous irregular state. The network is now predominantly interaction-driven, causing particular modes to become more prominent. The sampled distribution remains close to the target distribution. (C) Large weights: $W_{kj} \sim 2.4 \cdot [\mathcal{B}(0.5, 0.5) - 0.5]$. As expected, due to the strong synaptic interaction, spikes in the network tend to synchronize. In this extreme regime, the overall sampling quality decreases, since the absolute difference between the ideal, rectangular PSP shape, and the exponential PSPs of our model, increases proportionally to the synaptic weight. In particular, low-probability modes are sampled with less accuracy. Despite this disrupting effect, the network still samples from the main modes of the underlying distribution with relatively high fidelity.


A
I
Neuron
Time [ms]
II
p(z)
z
III
$D_{KL}(p_N \,||\, p_B)$
T [ms]
B
C

6.6. Applications of LIF Sampling

With LIF sampling now firmly in place, we can use this framework for various applications outside the field of neuroscience. In principle, anything that can be done using BMs can now be done with networks of LIF neurons. In the following, we shall use small networks of LIF neurons to perform some well-studied tasks such as image denoising, handwritten digit recognition and the emulation of magnetic solids. These experiments are not comprehensive studies, but rather serve to highlight the potential of our LIF sampling networks. These small demonstrations have provided an incentive for several lines of currently ongoing research in our theory department.

The applicability of LIF sampling to these problems highlights an intriguing convergence of what may superficially seem to be disparate disciplines – machine learning, solid state physics and neuroscience. However, we should point out that this is not just aesthetically pleasing, but also of both practical and theoretical value. The explicit compatibility to state-of-the-art neuromorphic devices, for which LIF models are a de-facto standard, paves a straightforward way for their application to problems where low power consumption and high computation speed are simultaneously crucial. A prime example would be the cognitive circuitry of autonomous intelligent agents (vulgo, robots). The analogy to magnetic solids, on the other hand, suggests further venues for theoretical research, as mathematical techniques from statistical physics now become applicable to spiking neural networks.

LIF sampling software

All of the following experiments were performed with a custom-designed software package which has been in active use for several years.[27] This software performs the translation of arbitrary Boltzmann machines to networks of LIF neurons following the rules laid out in Section 6.5.4. Written explicitly with neuromorphic applications in mind, it automatizes the characterization and configuration of each neuron on the simulation/emulation platform. These have, in general, diverse parameters due to hardware-specific variations, but can all achieve a symmetric activation function with sufficiently strong synaptic noise, which is, as we have argued, the only prerequisite for LIF sampling. Colloquially, we use the term "self-calibrating" to describe this software, since it automatically computes appropriate control parameters (i.e., synaptic weights and leak potentials) depending on the individual characteristics of the neurons in the substrate of choice. These control parameters must, of course, be precisely tunable and therefore require good prior calibration (see, e.g., Schwartz, 2013). With this software package, the instantiation and simulation of an LIF network that emulates a Boltzmann machine with arbitrary parameters literally happens at the push of a button.

6.6.1. Image Denoising

image denoising

Image denoising is a classical task in computer vision. Over the past decades, many effective denoising algorithms have been developed, using techniques such as anisotropic diffusion, local filtering and wavelet transforms. Our probabilistic approach can be compared to a smoothing process with a nonlinear filter. It is not intended to be quantitatively competitive with state-of-the-art methods, especially since it received no task-specific

[27] Recently, the software package was refactored by Oliver Breitwieser in order to allow fast parallel simulation, as well as a much faster instantiation of large networks.

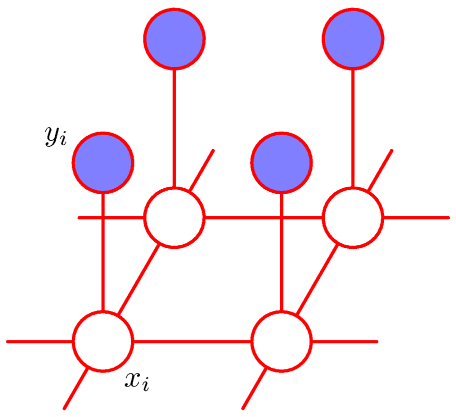


Figure 6.40.: Image denoising as an inference task on an Ising model. Observed pixels y_i represent noised information about the original, unknown pixels x_i. The edges connect each pixel to its nearest neighbors, as well as to its observed value, to which it is positively correlated. Figure taken from Bishop (2009).

tuning (see below), but rather a proof of concept and an example for the wide range of applications that are amenable to LIF sampling.

In its simplest version, the image denoising problem presents as follows. We start with an initial (unknown) black-and-white image, which can be represented as a state vector $\boldsymbol{x} \in (0,1)^n$, where n represents the number of pixels in the image. An uncorrelated noise process flips each pixel independently with a fixed probability $p_{\text{flip}} \ll 0.5$, yielding an observed image $\boldsymbol{y} \in (0,1)^n$. The task is to reconstruct $\boldsymbol{x}$ as well as possible from $\boldsymbol{y}$.

uncorrelated pixel noise

We can represent this problem as an inference problem which can be mapped to a Boltzmann machine (see also Figure 6.40). Since the pixels can only have two values (arbitrarily, we set black $\to 0$ and white $\to 1$), it is quite natural to represent each pixel with a binary RV x_k. As long as the noise level is small (which it must be for the image to be reconstructable at all), we can assume that there exists a positive correlation between the measured values y_k and the original ones x_k. We can therefore implement y_k as a bias for x_k:

$$b_k = \alpha(y_k - \frac{1}{2}) + \beta \quad , \tag{6.147}$$

where $\alpha > 0$ controls the (assumed) correlation between the measured and original pixel values and β represents a prior for the pixel state (we could, for example, choose $\beta > 0$ to encode the fact that the majority of pixels are likely to be white). Since the original image is only black-and-white, we can also assume that neighboring pixels are positively correlated. This defines our weight matrix

$$w_{ij} = \gamma \cdot \mathbf{1}_{\text{ne}(x_i)} x_j \quad , \tag{6.148}$$

where $\text{ne}(x_i)$ represents the set of neighbors of x_i, $\gamma > 0$ controls the (assumed) nearest-neighbor-correlation and $\mathbf{1}$ represents the indicator function

$$\mathbf{1}_A x := \begin{cases} 1 & \text{if} \quad x \in A \quad , \\ 0 & \text{if} \quad x \notin A \quad . \end{cases} \tag{6.149}$$

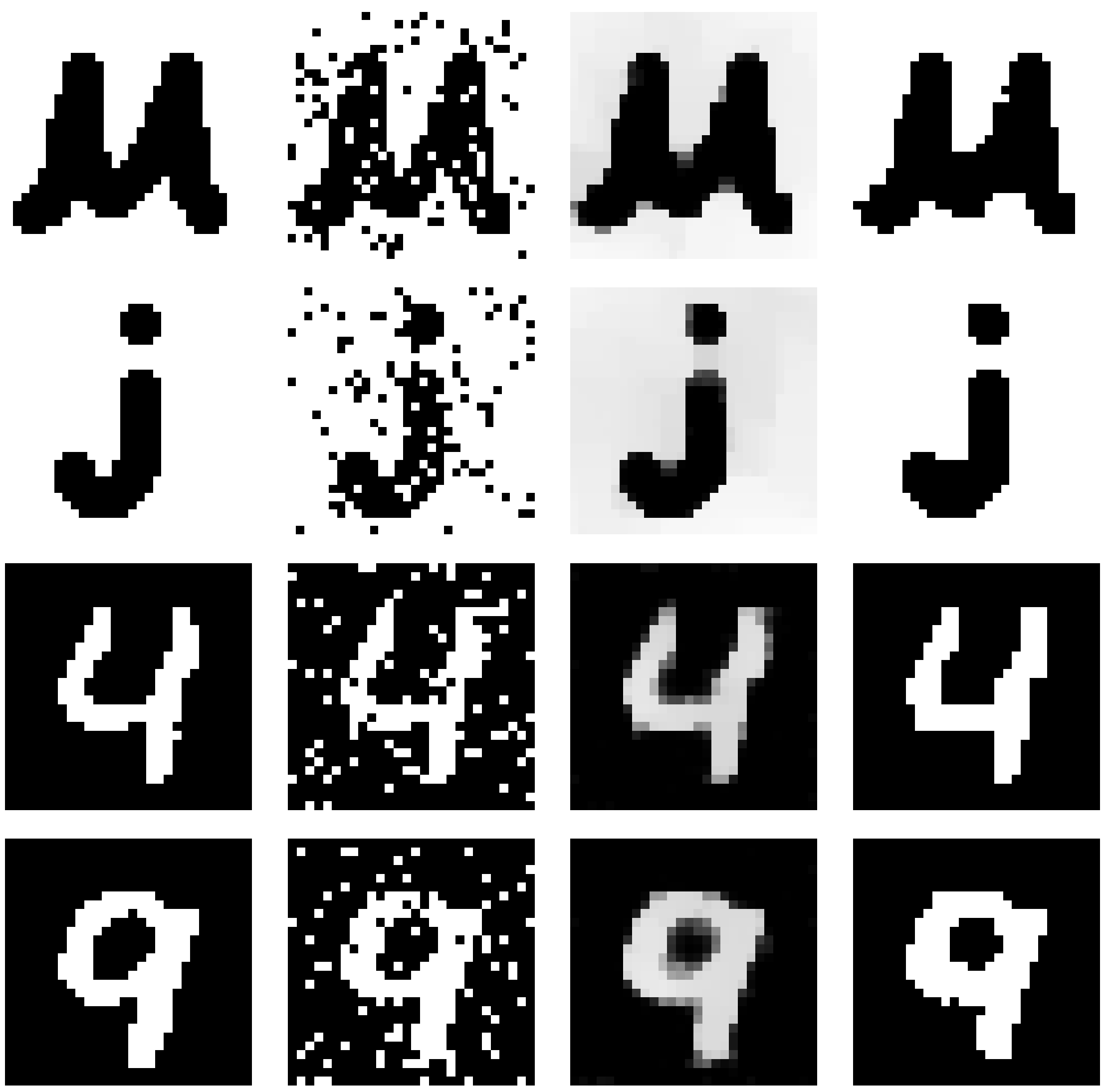


Figure 6.41.: Denoising of B/W images: examples. Each horizontal row is ordered as follows: original image, noisy image, inferred marginal probabilities for the original pixel values (in grayscale) and final denoised image after binarization of the marginals. Noise level (probability of pixel flip) set to 10%. The 'M' and 'j' are hand-drawn 30x30 pixel arrays. The 'j' was specifically chosen as a minuscule to show that small cohesive monochromatic areas (the dot on top of the 'j') can also be well reconstructed. The '4' and '9' are taken from the MNIST database. The switch from black-on-white to white-on-black serves to show that the reconstruction quality does not depend on the white-to-black pixel ratio.

The parameters $\boldsymbol{W}$ and $\boldsymbol{b}$ fully determine the target distribution for our LIF network. The resulting stochastic model is equivalent to the Ising model with positive nearest-neighbor interaction (more on this in Section 6.6.3). As such, it is quite intuitive to understand how its dynamics represent a nonlinear smoothing filter: each pixel has its state "pulled" towards the states of its neighbors x_i, as well as towards the actually measured value y_k: *Ising model*

$$u_k = b_k + \sum_{x_i \in \mathrm{ne}(x_k)} w_{ki} \tag{6.150}$$

$$= \alpha y_k + \sum_{x_i \in \mathrm{ne}(x_k)} \gamma x_i + \mathrm{const} \quad . \tag{6.151}$$

The nonlinearity is induced by the logistic activation function $p(z_k = 1) = \sigma(u_k)$.

Figure 6.41 shows exemplary results of denoising with LIF networks. For these experiments, we have chosen four 30×30 pixel images with a noise level of $p_{\mathrm{flip}} = 10\%$. The parameters were set at $\alpha = \gamma = 1$ and $\beta \in \{0.2, -0.2\}$, depending on the background pixel value. We point out that these are just guesstimates and explicitly not the result of careful tuning. Nevertheless, we can observe how the resulting LIF network already performs quite nicely on the chosen examples.

For denoising, we are only interested in the marginals $p(z_k = 1)$, which we can measure directly from the spike trains of the corresponding neurons. These can be represented as grayscale images and already clearly exhibit the desired denoising effect. Also, the blurriness of the edges serves as a nice visual cue for the analogy to a smoothing filter. The final results of denoising are obtained by defining an (arbitrary) cutoff θ for the marginals, above which the pixels are defined as white, otherwise black. We set the threshold at $\theta_{p(z_k=1)} = 0.5$, but the resulting images remain very similar for a broad range of θ. *binarization*

These results could be easily improved by choosing a more complex interaction that goes beyond the nearest neighbors, as well as by carefully tuning the parameters α, β and γ. In particular, these could be chosen in a task-specific manner, depending on the nature of the image. For example, β could be chosen non-uniformly, in accordance with the dominant pixel value within a local neighborhood. However, the main point we wish to illustrate here is how our LIF networks can effectively work "out of the box", without any tuning of neuron parameters.

6.6.2. Handwritten Digit Recognition

Pattern recognition and completion are, quite obviously, tasks that the brain is performing constantly, and at which it is exquisitely effective. Human-level performance at these tasks has therefore always been a holy grail for machine learning. While humans remain unsurpassed in many visual recognition tasks, algorithms are quickly catching up (see, e.g., Salakhutdinov and Hinton, 2009) – and some of them already boast superhuman performance (Ciresan et al., 2011; Cireşan et al., 2012). Not surprisingly, many state-of-the-art image classification algorithms are at least partly inspired by the architecture of the mammalian visual cortex.

In this section, we show how our LIF sampling networks can be trained for such tasks. Our experiments have been inspired by the highly successful application of Boltzmann machines to image classification by Hinton et al. (earliest results date back to the 80s, see, e.g., Hinton and Sejnowski, 1986), and represent the first implementation of these concepts in networks of LIF neurons. In particular, we discuss LIF-based Boltzmann machines both as generative (pattern completion) and discriminative (pattern recognition) models. The training and test data is taken from the probably most well-known benchmark database – the Mixed National Institute of Standards and Technology (MNIST) handwritten digit dataset (LeCun and Cortes, 1998).

generative and discriminative models
MNIST dataset

6.6.2.1. Fully Visible Boltzmann Machines

Our first experiment serves as a proof of concept and as a demonstration of probabilistic inference in LIF networks. We start with a straightforward implementation of a simple generative model for three easily recognizable digits - a 0, a 3 and a 4. The underlying assumption is that a Boltzmann distribution can be found which exhibits three distinct modes, each of which encodes one of the three digits. We further simplify the problem by having the associated BM be "fully visible", i.e., it only consists of units that explicitly represent the training data.

fully visible

As a first step, we have reduced the original 28×28 pixel arrays from MNIST to 12×12 arrays, in order to reduce simulation time. Each pixel is represented by a binary RV z_k and therefore assigned to one neuron, resulting in a fully connected BM of $K = 144$ neurons. The resulting joint distribution $p(\boldsymbol{z})$ should, after training, display three distinct high-probability modes with clearly recognizable shapes – the "prior knowledge" stored by the network. The probabilistic model was augmented by adding real-valued input channels for each pixel, associated with random variables $y_k \in \mathbb{R}$, $1 \leq k \leq K$. These RVs are used to represent observations and are therefore clamped during training or during inference tasks.

The resulting generative model $p(\boldsymbol{y}, \boldsymbol{z})$ has the structure shown in Figure 6.42A and connects the latent network variables z_k to observable inputs y_k by means of likelihood functions $p(y_k|z_k)$, which we have chosen as Gaussian with unit variance[28] (Figure 6.42B). The likelihoods $p(y_k|z_k)$ tend to align the network state with the observation, i.e. $z_k = 1$ for $y_k > 0$. The task for the network is to calculate and represent the posterior distribution according to Bayes' rule:

$$p(\boldsymbol{z}|\boldsymbol{y}) \propto p(\boldsymbol{z}) \cdot p(\boldsymbol{y}|\boldsymbol{z}) \quad . \tag{6.152}$$

A short derivation shows that the posterior $p(\boldsymbol{z}|\boldsymbol{y})$ yields the following abstract membrane potential:

$$v_k = b_k + y_k + \sum_i w_{ik} z_i \quad , \tag{6.153}$$

such that the sum $b_k + y_k$ is equivalent to an effective bias and corresponds to an external current ${I^{\text{ext}}}_k = I_k^b + I_k^y$ that shifts $\bar{u}_k$ appropriately (Equation 6.143). The input layer $\boldsymbol{y}$ must therefore not be represented explicitly, but can be translated to an additive term in the bias vector

$$\boldsymbol{b}^{\text{tot}} = \boldsymbol{b} + \boldsymbol{b}^y \quad . \tag{6.154}$$

[28] It is quite important to choose the variance large enough to allow a nonzero probability of pixels having the "wrong" value despite clamping. Otherwise, learning might lead to extreme values for weights and biases, creating a disconnected probability landscape with deep energy wells for particular states. If the high-energy states separating these modes can not be overcome with significantly large probabilities, the network will be stuck in one mode and lose its generative qualities.

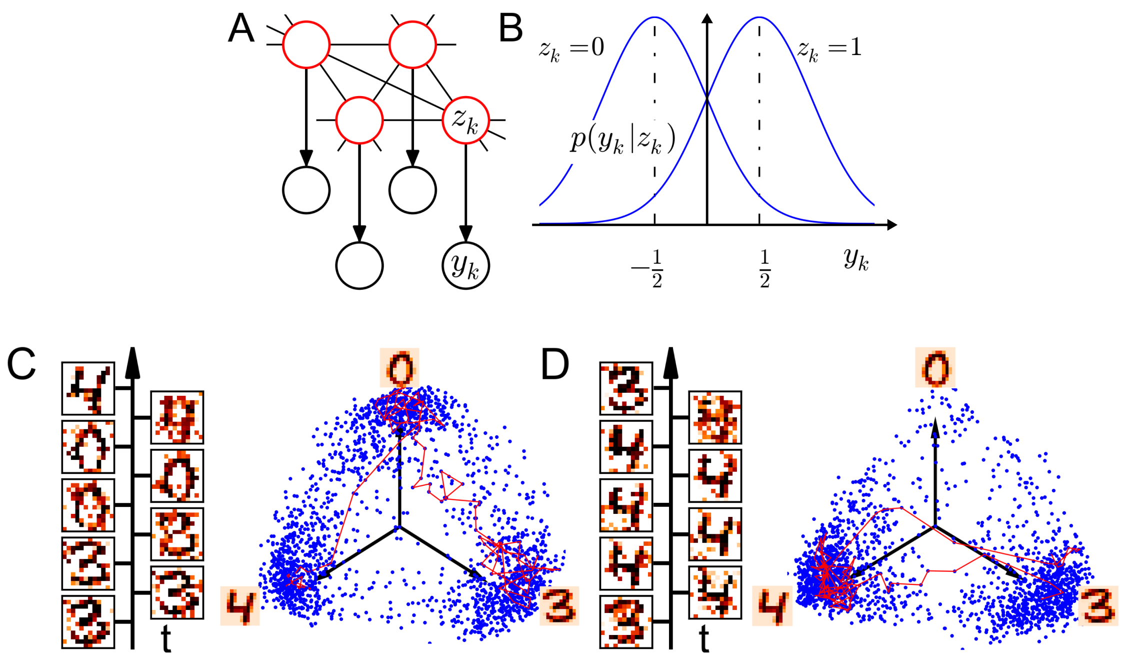


Figure 6.42.: A fully visible, LIF-based BM for handwritten digit recognition. **(A)** Probabilistic model. The (unknown) prior is implemented as a fully connected BM with RVs z_k. Observations are given by an input layer $\boldsymbol{y}$, but can be represented as biases for $\boldsymbol{z}$. **(B)** Likelihood functions $p(y_k|z_k)$. **(C)** Free evolution after learning (dreaming). The network spends most of the time in individual modes, with quick switching times between them. The modes are approximately uniformly distributed, correctly representing the uniform distribution of the training data (all images are clamped equally often). The states produced by the network at any point in time are clearly recognizable. **(D)** Stochastic inference. When presented with input that is incompatible with 0, but compatible with 3 and 4 (four pixels in the center of the image clamped to "black"), the network switches only between the two compatible states. The posterior distribution is still uniform (i.e., ca. 50% of time spent in each state), as required by the correct application of Bayes' rule to the prior. Figure taken from Petrovici et al. (2013).

The resulting network can now be trained by sequentially updating the bias vector $\boldsymbol{b}$ and weight matrix $\boldsymbol{W}$ in order for the images encoded by the sampled distribution $\boldsymbol{z}$ to become increasingly similar to the training images given by $\boldsymbol{y}$. This is achieved by first evaluating the target (posterior) expectation values $E\left[\tilde{z}_k\right]$ and correlations $E\left[\tilde{z}_i\tilde{z}_k\right]$ from the biases $\boldsymbol{b}^y$ corresponding to the pixel intensities in each of the three training images. Between two parameter updates, the network is allowed to evolve freely, in order to obtain the prior expectation values $E\left[z_k^*\right]$ and correlations $E\left[z_i^* z_k^*\right]$. Iteratively, following each freely evolving period, the biases and weights can be updated such that the priors are "pushed" towards the target posteriors:

posterior

prior

$$\Delta b_k \propto E\left[\tilde{z}_k\right] - E\left[z_k^*\right] \tag{6.155}$$

$$\Delta w_{ik} \propto E\left[\tilde{z}_i\tilde{z}_k\right] - E\left[z_i^* z_k^*\right] \quad . \tag{6.156}$$

wake-sleep algorithms, contrastive divergence

We note that this intuitive update rule is inherited from so-called wake-sleep algorithms and in particular contrastive divergence, which we discuss later on. The following suggestive nomenclature is routinely used in machine learning: During the "awake" state, the network receives observations and samples from the appropriate posterior (usually, the network is clamped to the training data, but for a fully visible BM, this step becomes redundant, since it is fully constrained when awake). During the "sleeping" state, the network is said to "dream" from the unconstrained prior distribution.

dreaming

state space projection

Since the joint distribution over all $\boldsymbol{z} \in \{0,1\}^{144}$ possible states can not be explicitly given, we use a 2D projection of the state space, similarly to Section A.2.3.5. The axes indicate the three basis vectors $\boldsymbol{B}$, representing pixel intensities $E\left[z_k\right]$ of the digits (0, 3, 4),

$$E\left[z_k\right]^{034} = (\boldsymbol{B}^0, \boldsymbol{B}^3, \boldsymbol{B}^4)^T \tag{6.157}$$

with a total intensity normalization $||\boldsymbol{B}^i|| = \sqrt{\sum_j |\boldsymbol{B}_j^i|^2} = 1$. The network states $\boldsymbol{z}^{(t)}$ acquired from the simulation are projected onto this basis:

$$\boldsymbol{z}^{034}(t) = (\boldsymbol{B}^0 \cdot \boldsymbol{z}(t), \boldsymbol{B}^3 \cdot \boldsymbol{z}(t), \boldsymbol{B}^4 \cdot \boldsymbol{z}(t))^T \quad . \tag{6.158}$$

This three-dimensional vector is then projected onto a two-dimensional plane with coordinates

$$\boldsymbol{z}^{\text{proj}}(t) = \begin{pmatrix} \sin(\phi_\text{B}^0) & \sin(\phi_\text{B}^3) & \sin(\phi_\text{B}^4) \\ \cos(\phi_\text{B}^0) & \cos(\phi_\text{B}^3) & \cos(\phi_\text{B}^4) \end{pmatrix} \boldsymbol{z}^{034}(t) \tag{6.159}$$

with $(\phi_\text{B}^0, \phi_\text{B}^3, \phi_\text{B}^4) = (0, \frac{2\pi}{3}, \frac{4\pi}{3})$ indicating the directions of the normalized basis vectors. These linear projections $\boldsymbol{z}^{\text{proj}}(t)$ of network states $\boldsymbol{z}^{(t)}$ therefore illustrate similarity of states as a distance in a 2D plane.

The temporal evolution of the unconstrained network states $\boldsymbol{z}(t)$ after training is shown in Figure 6.42C. Devoid of any observations, the network dreams from a distribution $p(\boldsymbol{z}) = p(\boldsymbol{z}|\boldsymbol{y} = \boldsymbol{0})$ with three distinct modes, clustered around the vectors representing the three training images. The network trajectory reveals that the system stays in one mode ("digit") for some duration, traverses the state space and then samples from a different mode of the distribution. Note the behavioral similarity to the perceptual jumps we experience when we see the ambiguous images from Figure 6.1. Snapshots taken at

25 ms intervals show how the network produces highly recognizable images that are very similar to the training set.

A typical inference scenario with incomplete observations is shown in Figure 6.42D. Four input channels at the center of the image were picked to inject positive currents $I_k^y > 0$ (corresponding to black pixels) to the network while all other inputs remained uninformative. The observation $\boldsymbol{y}$ therefore appeared incompatible with digit 0, and remained ambiguous with respect to digits 3 and 4. Accordingly, the resulting bimodal posterior distribution $p(\boldsymbol{z}|\boldsymbol{y})$ had a suppressed 0-mode, but preserved the 3 and 4 modes. Thus, the posterior reflects both the almost certain conclusion that "the input is not a zero" and the uncertainty that "the input could either be a three or a four".

6.6.2.2. Deep Learning Architectures

We shall start by generalizing the simple learning algorithm we used in the previous section. Consider, therefore, how the probability of a data sample given the momentary parameters of the model $p(\boldsymbol{z}|\boldsymbol{W}, \boldsymbol{b})$ changes when an individual weight w_{ij} is changed:

$$\begin{aligned}
\frac{\partial p(\boldsymbol{z})}{\partial w_{ij}} &= \frac{\partial}{\partial w_{ij}} \left[\frac{e^{-E(\mathbf{z})}}{\sum\limits_{\mathbf{z}'} e^{-E(\mathbf{z}')}} \right] \\
&= e^{-E(\mathbf{z})} \left[\sum_{\mathbf{z}'} e^{-E(\mathbf{z}')} \right]^{-1} z_i z_j - e^{-E(\mathbf{z})} \left[\sum_{\mathbf{z}'} e^{-E(\mathbf{z}')} \right]^{-2} \sum_{\mathbf{z}'} \left[e^{-E(\mathbf{z}')} z_i' z_j' \right] \\
&= p(\mathbf{z}) z_i z_j - p(\mathbf{z}) \left\{ \frac{\sum\limits_{\mathbf{z}'} \left[e^{-E(\mathbf{z}')} z_i' z_j' \right]}{\sum\limits_{\mathbf{z}'} e^{-E(\mathbf{z}')}} \right\} \quad , \qquad (6.160)
\end{aligned}$$

where we have omitted the conditioning on the model parameters $\boldsymbol{W}$ and $\boldsymbol{b}$ for clarity. By dividing both sides by $p(\boldsymbol{z})$, we obtain the gradient of the log-probability – or log-likelihood – of the training sample $\boldsymbol{z}$:

log-likelihood

$$\frac{\partial \ln p(\mathbf{z})}{\partial w_{ij}} = z_i z_j - \left\{ \frac{\sum\limits_{\mathbf{z}'} \left[e^{-E(\mathbf{z}')} z_i' z_j' \right]}{\sum\limits_{\mathbf{z}'} e^{-E(\mathbf{z}')}} \right\} \quad . \qquad (6.161)$$

The second term on the RHS represents the expectation value of $z_i z_j$ over all possible model states:

$$\frac{\partial \ln p(\mathbf{z})}{\partial w_{ij}} = z_i z_j - \langle z_i z_j \rangle_{\text{model}} \quad . \qquad (6.162)$$

By taking a final average over all training data,

$$\left\langle \frac{\partial \ln p(\mathbf{z})}{\partial w_{ij}} \right\rangle_{\text{data}} = \langle z_i z_j \rangle_{\text{data}} - \langle z_i z_j \rangle_{\text{model}} \quad , \qquad (6.163)$$

we can express the maximization of the log-likelihood of the data as a gradient ascentgradient ascent learning problem for each of its parameters:

$$\Delta w_{ij} = \eta(\langle z_i z_j \rangle_{\text{data}} - \langle z_i z_j \rangle_{\text{model}}) \quad , \qquad (6.164)$$

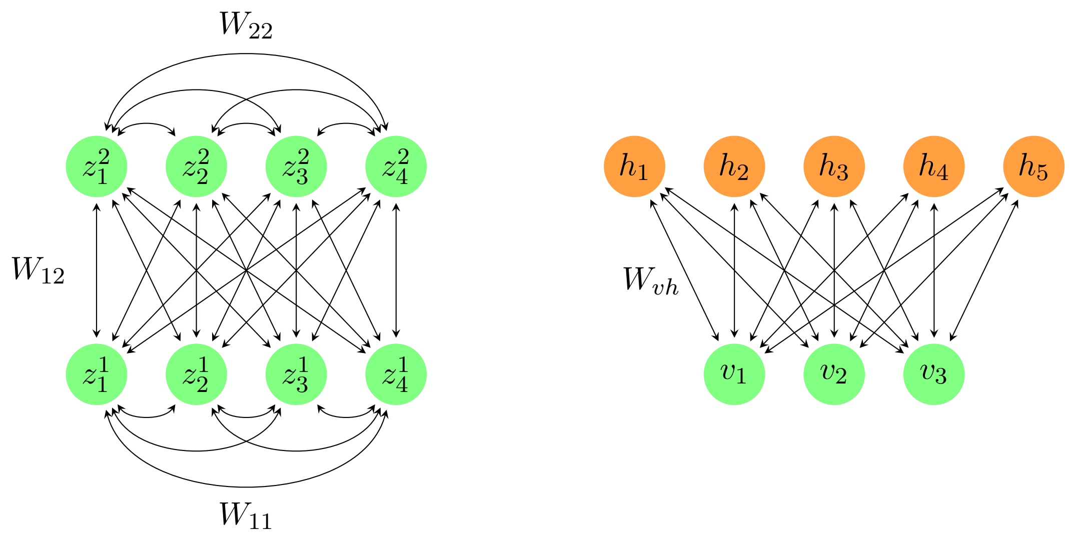


Figure 6.43.: Connectivity structure of two types of BMs. Visible (observable) units are marked green, hidden variables in orange. **Left:** Fully visible, fully connected BM. We have arbitrarily partitioned the constituent RVs into two sets, to highlight the existence of lateral connections, as opposed to RBMs. **Right:** Two-layer RBM with a visible and a hidden layer. Lateral connections within layers are omitted. This allows faster sampling, since all units in a layer can be updated in parallel. Experience also shows that such BMs are better-suited for training than their fully connected counterparts.

where η represents a learning rate. Appropriately, this method of parameter updating is dubbed maximum likelihood (ML) learning. The learning rule for the biases can be calculated analogously:

ML learning

$$\Delta b_i = \eta(\langle z_i \rangle_{\text{data}} - \langle z_i \rangle_{\text{model}}) \quad . \tag{6.165}$$

The remaining computational problem concerns the evaluation of the model average. Since this is evidently impractical due to the exponentially exploding number of possible states, it must be obtained from a large number of sampling steps n. However, there is no a priori way of knowing how long to sample until the sampled distribution $p_n(\boldsymbol{z})$ has converged closely enough to the true model distribution. To avoid this difficulty, Hinton (2002) has proposed to simply fix n to a small value, typically even $n = 1$. This so-called contrastive divergence (CD) method no longer maximizes the log-likelihood of the training data, but can rather be shown to minimize the difference between two DKLs:

contrastive divergence

$$CD_n = D_{\text{KL}}\left(p_{\text{data}} \parallel p_{\text{model}}\right) - D_{\text{KL}}\left(p_n \parallel p_{\text{model}}\right) \quad . \tag{6.166}$$

In practice, this method of updating weights and biases has become a de-facto standard, so we can adopt it for training our LIF networks as well.

We could now apply this algorithm to train fully visible BMs, as demonstrated in the previous section. However, it is quite clear that our previous example was just an extremely simple case of a generative model, where the training set was small and consisted of very distinct images. With each additional training image, a fully visible BM needs to

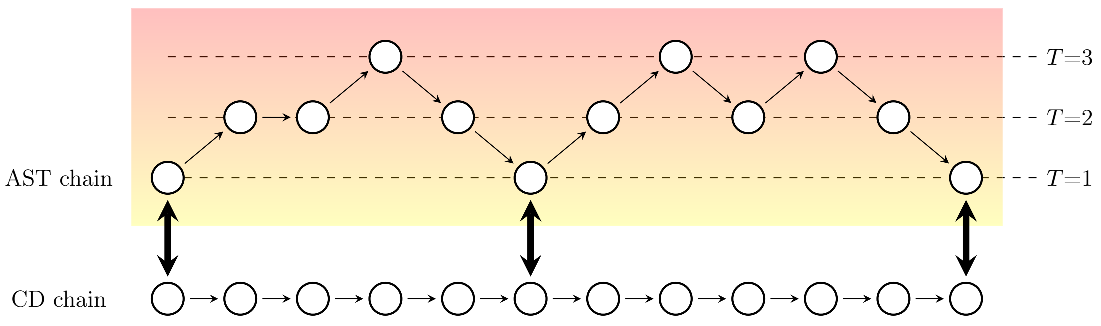


Figure 6.44.: Update chains in CAST training. The bottom chain represents a sequence of CD updates. The top AST chain samples with fluctuating temperatures T and swaps ($T = 1$)-states with the CD chain, thus improving mixing.

"overload" its weights and biases, for example when the same pixel is required to be black in one image and white in another, or when pairs of pixels are required to be positively correlated in one image and negatively correlated in another. As the number of training samples increases, the BM does not have enough parameters to store this information and gradually loses its generative properties by producing increasingly blurred images, which are essentially overlaps of multiple training samples.

The solution is quite straightforward and involves including additional RVs into the BM, which are, however, not directly observable. In probabilistic modeling such latent variables are also called unobserved or hidden causes. BMs with hidden units evidently can be given far more representational power than fully visible BMs, as the distribution of the visible RVs becomes a marginal over the joint distribution over all RVs. Thus, hidden variables also remove the constraint of the distribution over the visible RVs being necessarily Boltzmann. *latent/hidden variables*

However, this additional flexibility comes at an expensive price in terms of training. Since the relationships between the latent RVs themselves, as well as between latent and visible RVs are initially unknown, they represent parameters that need to be trained. Additionally, the training data only holds information about the visible RVs, so with each additional hidden RV, these relationships become increasingly less constrained. Indeed, the inefficiency of training large fully connected BMs has led to the practical necessity of imposing restrictions on the connectivity structure.

The most widely used BM model for both generative and discriminative tasks is the so-called restricted Boltzmann machine (RBM, see also Figure 6.43). In an RBM, the underlying graphical model is bipartite, with no lateral connections from visible to visible and from hidden to hidden units. This is particularly practical for software implementations, which most often use Gibbs sampling as a state update algorithm. Due to the lack of lateral connections, sampling alternates between updating all visible units v_k and updating all hidden units h_k, each of which can happen in parallel: *RBM* *parallel update*

$$p(h_k|\boldsymbol{h}_{\setminus k}, \boldsymbol{v}) = p(h_k|\boldsymbol{v}) \tag{6.167}$$

$$p(v_k|\boldsymbol{v}_{\setminus k}, \boldsymbol{h}) = p(v_k|\boldsymbol{h}) \quad . \tag{6.168}$$

More importantly however, CD has proven to be an efficient way of training RBMs as good generative models for complex datasets (see, e.g., Hinton, 2002), which we shall also find to be the case for LIF-based BMs.

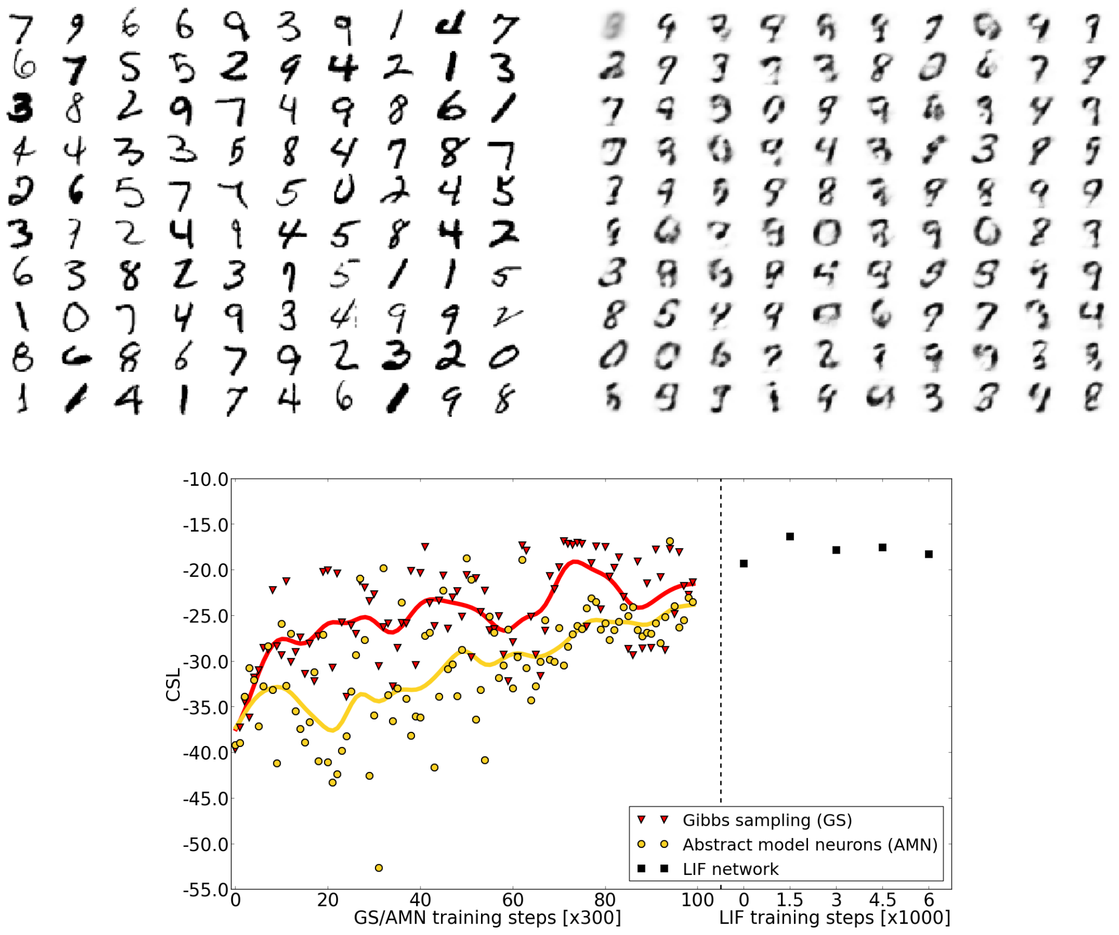


Figure 6.45.: LIF-based RBMs as generative models of MNIST handwritten digits. **Top left:** Subset of the MNIST training dataset (60000 images). **Top right:** Marginal probabilities of visible units after training represented as grayscale images. These can be easily calculated from the sampled states of the hidden units due to their conditional independence. **Bottom:** Evolution of the generative properties of the RBM during training measured by the log-likelihood (more precisely, CSL) of an MNIST test set under the momentary values of the network parameters. The left side of the plot represents the pretraining of abstract models (standard Gibbs sampling and abstract neural sampling). The solid curves were obtained from the data points by smoothing with a Gaussian. After pretraining, the parameters (weights and biases) were translated to the LIF domain and further trained for several thousand iterations. The generative performance of the LIF network improved quickly and consistently remained at a high level.

With these training tools, we can now investigate the performance of LIF-based RBMs on more relevant and difficult tasks, such as learning the full MNIST dataset. In the following, we briefly outline the training procedure – for a more detailed description, we refer to Leng (2014). Our approach is as follows: we first train a standard RBM with an augmented CD-based training algorithm and a decreasing learning rate ("pretraining"). *pretraining* After a certain number of steps, the trained network parameters are translated to an LIF network using the rules outlined in Section 6.5.4. The resulting LIF network already performs very well, but is further trained with CD for a small number of training steps as a final "fine tuning" step, which further improves its performance.[29] *fine tuning*

The extension of CD we use during pretraining is called coupled adaptive simulated tempering (CAST, see, in particular, Salakhutdinov, 2010) and serves for better mixing *CAST* within the update chain (see Figure 6.44). The idea of CAST is the following: In addition to the CD chain, another Gibbs sampler (AST chain) is run in parallel on an RBM with fluctuating temperature, which can be straightforwardly implemented as a multiplicative factor on all weights and biases, since a temperature-dependent Boltzmann distribution simply reads:

$$p(\boldsymbol{z}) \propto \exp\left[-\frac{1}{T}E(\boldsymbol{z})\right] \quad . \tag{6.169}$$

By varying the temperature T in the AST chain (tempering), we can improve mixing between high-probability modes that are separated by low-probability states. We can then periodically switch the tempered states from the AST chain (those states where $T = 1$) with the states in the CD chain to obtain a better estimate of $\langle z_i z_j \rangle_\text{model}$ $\langle z_i \rangle_\text{model}$ required by our update rules 6.164 and 6.165.

Some results of our training algorithm applied to LIF sampling networks can be seen in Figure 6.45. Apart from showing examples of network states during the dreaming phase of the trained network, we estimate its generative capabilities with the so-called conservative sampling-based likelihood (CSL) estimator (Bengio and Yao, 2013): *CSL*

$$CSL = \frac{\sum\limits_{j=1}^{N} \left\{ \ln \left[\frac{\sum_{i=1}^{M} p(\mathbf{v}_j|\mathbf{h}_i)}{M} \right] \right\}}{N} \quad , \tag{6.170}$$

where the first sum is taken over all test samples (images) $\boldsymbol{v}_j$ and the second sum is taken over M hidden states $\boldsymbol{h}_i$ sampled by the network. The CSL can be viewed as the average probability of the network to produce the set of N test images $\{\boldsymbol{v}_j | 1 \leq j \leq N\}$.[30] Note how the log-likelihood of the test images gradually increases as training progresses (Figure 6.45, bottom panel). After training, the network produces well-recognizable images with

[29] The search for good meta-parameters in machine learning is notoriously "voodoo" in nature, i.e., widely acknowledged to profit significantly from a long history of practical experience using the respective algorithm (Hinton, 2010). We have therefore explicitely avoided a discussion of training parameters such as the time course of the decreasing learning rate or the number of training steps in favor of a crisp and intuitive explanation. We point again to Leng (2014) for numerical details of the training algorithm.

[30] In principle, we could also use the sampled visible states $\boldsymbol{v}_j$ to compute something like a distance $||\boldsymbol{v}_i - \boldsymbol{v}_j||$ (as we did in Figure 6.42), but using the hidden states as a proxy is more efficient, since they allow the fast computation of $p(\boldsymbol{v}_j|\boldsymbol{h}_i)$ (due to the bipartite nature of the RBM), which is, of course, more precise than sampling.

a large variance of individual digits, thereby matching the diversity of the data it has learned from (Figure 6.45, top panels).

mixing

If we assign each of the produced images a particular mode based on its similarity to prototypical digits from 1 to 10, we can obtain a better visual representation of the state mixing (Figure 6.47, top panel). The LIF network appears to jump significantly more often between modes, allowing a faster coverage of the entire relevant state space. Even without additional tweaking, it mixes comparably to AST, which, however, requires the computation of many intermediate states with varying temperature between the accepted samples with $T = 1$. It turns out that, in particular cases, LIF networks can actually mix better than even AST (Figure 6.47, bottom panel). This phenomenon is extremely interesting, since it showcases how LIF sampling can even be superior to state-of-the-art sampling techniques from machine learning. The investigation of the mixing properties of LIF sampling networks is ongoing.

DBMs

With LIF-based RBMs now in place, we can add further layers to our networks, forming so-called deep Boltzmann machines (DBMs, see Figure 6.46, left panel). The training algorithm for these networks remains the same[31] and is simply applied repeatedly for each additional pair of neighboring layers: after the first hidden layer has been trained, it is used as a visible layer for training the second hidden layer, and so on. In order to use these networks as discriminative models, we can also add a so-called label layer consisting of K label RVs l_k (Figure 6.46, left panel), where K represents the number of image classes (for MNIST, $K = 10$). When the visible layer is clamped during training, the label layer is clamped simultaneously to $l_k = 1$ and $\boldsymbol{l}_{\setminus k} = 0$, where k represents the class of the currently clamped image.

label layer

The idea behind adding "depth" to the network is for each layer to learn an increasingly abstract representation of the training data. For example, when the test data is shown and the label neurons are not clamped, the label layer is simultaneously the deepest one. It can then be interpreted as featuring the highest level of abstraction, with each individual neuron responding to all written representation of a specific digit class.

classification performance

We can now compare the classification performance of LIF sampling networks to the performance of classical DBMs trained with the same algorithm. We specifically do not perform any additional training of the LIF networks, but rather use the pretrained parameters of classical DBMs and translate them to the LIF domain. The comparison of classification performances is shown in Table 6.2.

The translation of a classical RBM/DBM to an LIF network causes only a slight decline in classification performance. Coupled with the improved mixing features of LIF-based BMs and with the direct compatibility to existing accelerated neuromorphic devices, these results provide a strong incentive for using LIF sampling networks as substrate for complicated classification tasks. For completeness, we have also added results for networks of CUBA neurons, where the HCS was emulated artificially by choosing a very low membrane time constant. Note how the CUBA networks perform slightly better than COBA

[31] In machine learning, deep networks are usually trained with some form of backpropagation to obtain the highest classification performances. We omit such methods here, since our focus lies on the comparison between LIF-based DBMs and classical DBMs. However, the translation of DBM parameters is, of course, independent of the sophistication level of the training algorithm used in pretraining.

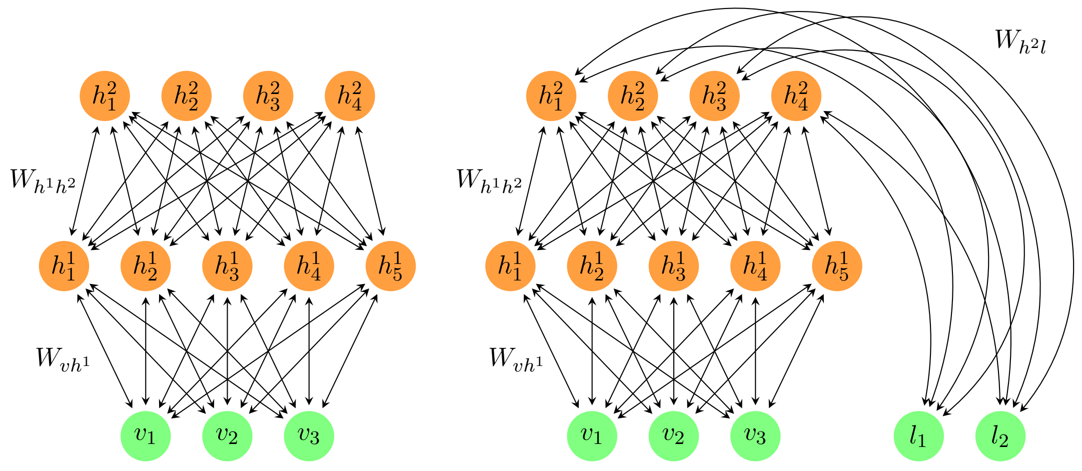
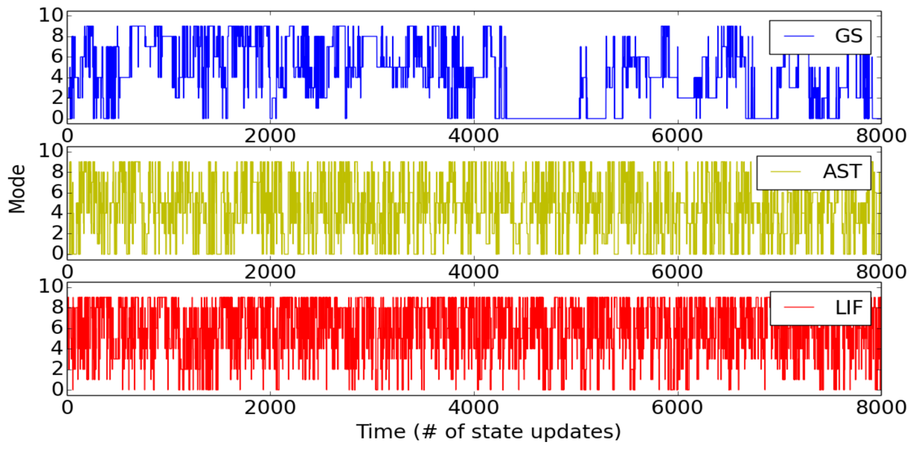


Figure 6.46.: RBMs with multiple layers (DBM). **Left:** Example DBM with two hidden layers. **Right:** Additional label layer connected to the last hidden layer. During training, the label units are clamped to the correct labels corresponding to the training sample clamped to the visible layer. After training, the label layer remains unconstrained, effectively serving as a final hidden layer.

networks. This is not unexpected, since in COBA networks, the interaction within the network slightly alters the HCS of the neurons, thereby modifying their activation functions. CUBA neurons do not suffer from such nonlinearities in the superposition of evoked PSPs.

implementation	RBM ($H_1 = 1300$)	DBM ($H_1 = 600, H_2 = 1100$)
traditional	96.7%	97.1%
CUBA LIF	96.5%	96.9%
COBA LIF	96.4%	96.6%

Table 6.2.: Classification performance of different DBM implementations on the full MNIST dataset.



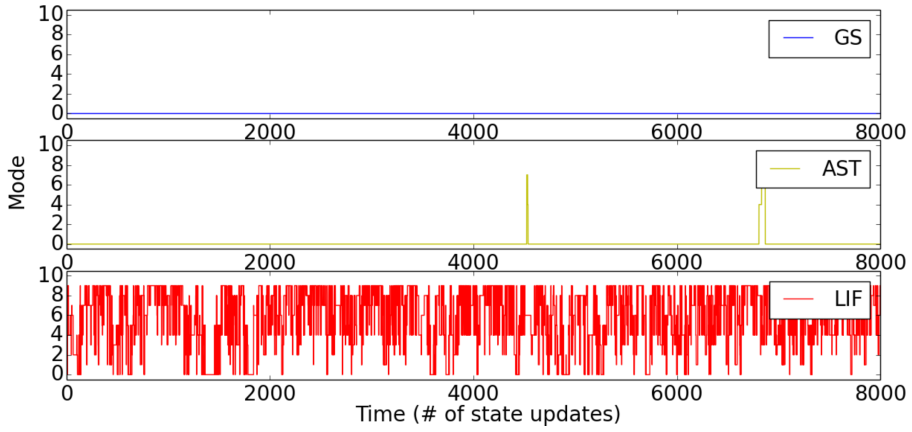


Figure 6.47.: Mixing in RBMs. **Top:** Mixing after pretraining. Gibbs sampling (blue) already spends extended periods in individual modes, requiring longer sampling time for good coverage of the entire relevant state space. Upon close inspection, LIF sampling (red) already switches more frequently than AST (yellow). **Bottom:** Mixing after fine-tuning. While LIF sampling only shows a slight decrease in switching frequency, Gibbs and AST almost exclusively sample from a single mode. This comparison is somewhat unfair, since AST was already mixing well before fine-tuning. However, it shows that parameter configurations exist for which LIF sampling is clearly superior to AST in terms of mixing.

6.6.3. Microscopic Magnetic Systems

At the end of our brief excursion into potential applications, we discuss a class of models that has already appeared several times throughout this chapter, both implicitly and explicitly. Indeed, BMs themselves were originally inspired not only by biological neural networks, but also by long-established ideas from statistical physics – in particular, the Ising model.

Ising model

The Ising model describes an ensemble of spin-endowed particles at fixed positions in a lattice.[32] Each particle has two possible configurations of its magnetic moment denoted by $\sigma_k \in \{-1, 1\}$, which correspond to the two possible spin projections of a spin-1/2 particle. The interaction between pairs of particles is governed by a bilinear term $J_{ij}\sigma_i\sigma_j$ whose sign favors either parallel or antiparallel spin configurations. The interaction with an external magnetic field is modeled by a linear term $h\mu_k\sigma_k$, where h gives the interaction amplitude and μ_k the magnetic moment of a particle. The Hamiltonian of the system therefor reads:

spin-1/2 particles

Ising Hamiltonian

$$H(\boldsymbol{\sigma}) = -\sum_{ij} J_{ij}\sigma_i\sigma_j - h\sum_k \mu_k\sigma_k \quad . \tag{6.171}$$

The temperature-dependent configuration probability (joint distribution) of $\boldsymbol{\sigma} \in \Omega = \{-1, 1\}^N$, where N represents the total number of particles, is given by the Boltzmann distribution

$$P_\beta(\boldsymbol{\sigma}) = \frac{e^{-\beta H(\boldsymbol{\sigma})}}{Z_\beta} \quad , \tag{6.172}$$

where β represents an inverse temperature

temperature

$$\beta = (k_B T)^{-1} \tag{6.173}$$

and Z_β is the partition function

partition function

$$Z_\beta = \sum_{\boldsymbol{\sigma}\in\Omega} e^{-\beta H(\boldsymbol{\sigma})} \quad . \tag{6.174}$$

It is quite obvious how the distribution of this ensemble of spins is formally identical to the distributions sampled by our BMs (Equation 6.59) by either Gibbs sampling, abstract neural sampling or LIF sampling.

In the simplest version of the Ising model, the interaction happens only between nearest neighbors (denoted by angular brackets $\langle ij\rangle$). Furthermore, all particles are identical ($\mu_i =: \mu$ and $J_{ij} =: J \; \forall i, j \in \{1, \dots, N\}$), so the Hamiltonian becomes

$$H(\boldsymbol{\sigma}) = -\sum_{\langle ij\rangle} J\sigma_i\sigma_j - h\mu\sum_k \sigma_k \quad . \tag{6.175}$$

[32] Originally, the Ising model was designed as a simplified model of ferromagnetism, with the individual units representing magnetic dipole moments of atomic spins.

lateral interaction

Depending on the sign of J, we can identify three types of lateral interaction:

- $J > 0$, ferromagnetic interaction, spins tend to align;
- $J = 0$, nonferromagnetic material, no spin-spin interaction;
- $J < 0$, antiferromagnetic interaction, neighboring spins tend to be antiparallel.

interaction with external field

The sign of h, on the other hand, governs the interaction with an external magnetic field:

- $h > 0$, paramagnetic interaction, the total magnetization of the ensemble tends to align with the external field;
- $h < 0$, diamagnetic interaction, the external field causes an opposing-sign magnetization of the ensemble.

Since Ising ensembles obey Boltzmann statistics, we can directly translate all interaction parameters to the LIF domain (see Section 6.5.4). We can therefore search for equivalent phenomena in LIF networks and solid-state magnetic materials.

We start with the simplest interesting system: a 2D lattice of a nonferromagnetic material ($J = 0$) with paramagnetic properties ($h > 0$). For such an ensemble, the dependence of the ensemble magnetization M on the external field B can be found analytically. Without lateral interaction, individual spin orientations are independent, so the joint distribution (and therefore also the partition function) factorizes:

$$p(\boldsymbol{\sigma}) = \prod_{i=1}^{N} p(\sigma_i) = [p(\sigma)]^N \quad , \tag{6.176}$$

where

$$p(\sigma) = \frac{e^{\beta\sigma\mu B}}{Z^*} \tag{6.177}$$

and

$$Z^* = \sum_{\sigma\in\{-1,1\}} p(\sigma) = e^{\beta\mu B} + e^{-\beta\mu B} = 2\cosh(\beta\mu B) \tag{6.178}$$

magnetization of a paramagnet

The total magnetization is proportional to the expectation value of the spin orientation, which can be easily derived from the above single-unit state distribution:

$$\begin{aligned} M = N\mu\langle\sigma\rangle = N\mu \sum_{\sigma\in\{-1,1\}} \sigma p(\sigma) &= N\mu \frac{\left(e^{\beta\mu B} - e^{-\beta\mu B}\right)}{Z^*} \\ = N\mu \frac{2\sinh(\beta\mu B)}{2\cosh(\beta\mu B)} &= N\mu \tanh(\beta\mu B) \quad . \end{aligned} \tag{6.179}$$

Curie law

For small magnetic fields and large temperatures, we have $\beta B \to 0$, so we can approximate $\tanh(\beta\mu B) \approx \beta\mu B$, leaving us with the well-known Curie law:

$$M = \frac{N\mu^2}{k_B}\frac{B}{T} \quad . \tag{6.180}$$

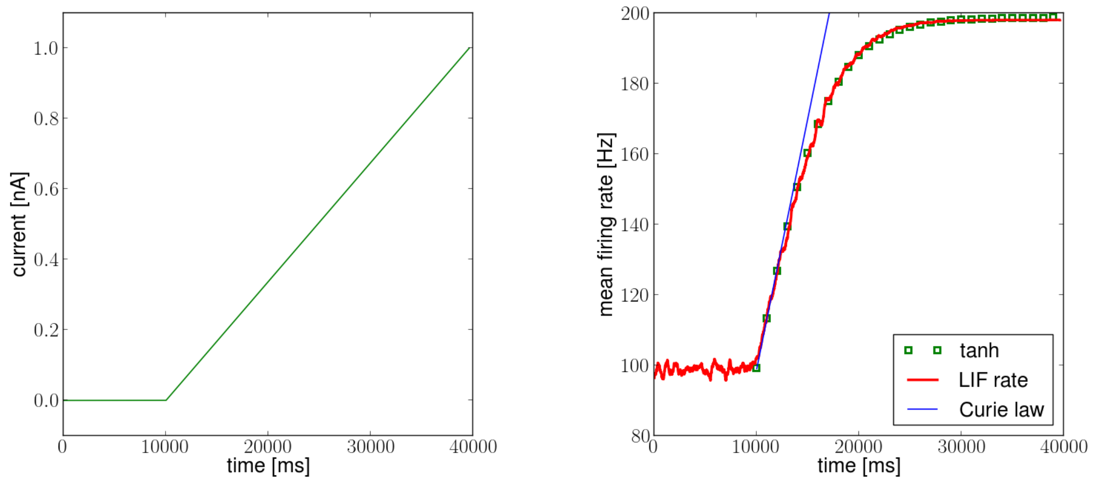


Figure 6.48.: Curie law of paramagnets observed in unconnected, externally driven LIF networks. **Left:** A linearly increasing external current emulates an external magnetic field. **Right:** The average refractoriness (equivalent to the mean firing rate) of the network is the equivalent of magnetization. For low external fields, the mean firing rate increases linearly. For higher values of the external drive, the mean firing rate saturates at $\nu_{\max} = 1/\tau_{\text{ref}}$ following a tanh curve.

We can now observe the exact same behavior in LIF networks without lateral connections (Figure 6.48). The LIF equivalent of magnetization is the fraction of neurons that are refractory at a given point in time. The linear increase of the external magnetic field is emulated by a linearly increasing background current I^b received by all neurons. For small values of I^b, we observe the linear increase of average refractoriness predicted by the Curie law. For larger values of I^b, we enter the saturation regime where the neurons are almost always refractory (the equivalent of all spins being oriented in the same direction) and the average refractoriness curve saturates along a tanh curve.

By adding positive lateral interactions to the system, we can now emulate the behavior of ferromagnets. One characteristic observable effect is the appearance of hysteresis: the total magnetization of the system no longer depends on the external field alone, but also on its history. Once the system has been fully magnetized and the external field is reduced again, the "demagnetization" follows a different path than the original magnetization curve. Due to the positive lateral interaction of the spins, an internal coercive force appears that acts to preserve the magnetization. The external field must switch direction before the magnetization switches direction as well. Note that this is not a latency-driven effect: if the lateral interaction J is strong enough, a hysteresis loop always appears, regardless of the speed at which the external field changes.

ferromagnets

hysteresis

coercive force

The same observations can now be made for LIF sampling networks with excitatory lateral connections (Figure 6.49). If we start with a network in the quiescent state (equivalent to a complete negative magnetization), a slowly linearly increasing background current $dI^{\text{ext}}/dt = \text{const} > 0$ causes the activity to increase slowly at first and then jump

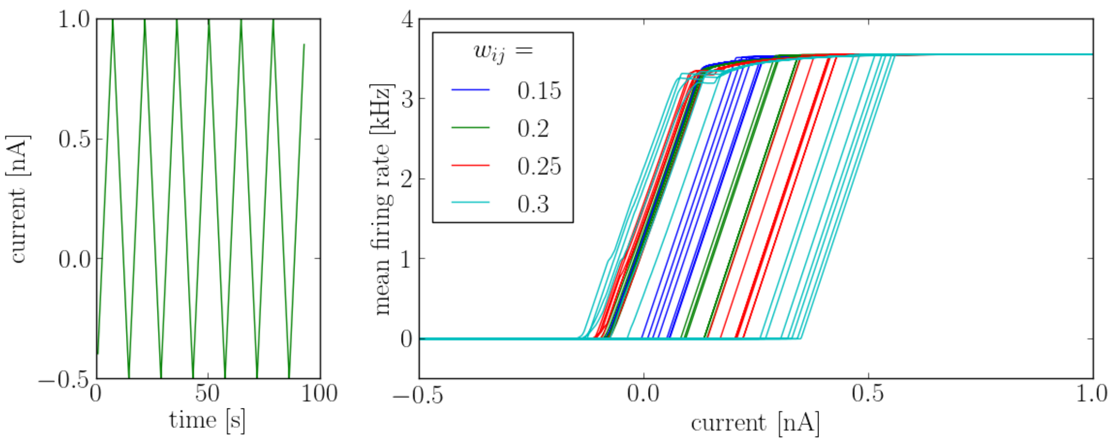


Figure 6.49.: Ferromagnetic hysteresis in excitatorily coupled recurrent LIF networks. **Left:** As in Figure 6.48, a varying current emulates an equivalent external magnetic field. **Right:** Depending on its previous history, the firing rate in the network follows a different path. The strength of the lateral connections determines the internal coercive force, with stronger synaptic weights causing a wider hysteresis cycle.

suddenly to a high firing rate, which then again increases slowly until saturation (the maximum firing rate of a single neuron being $\nu_{\max} = 1/\tau_{\text{ref}}$). The value of the external current around which the activity jumps is denoted by $I_\uparrow$. When, starting from this state, the current is lowered again with the same (but negative) rate, the lateral excitation causes the high firing activity to persist even for $I^{\text{ext}} < I_\uparrow$, until the external drive is strong enough to inverse the "magnetization" around $I_\downarrow$. Note that typical hysteresis curves of macroscopic ferromagnetic bulks are significantly more "smooth" – without discernable jumps around particular values of the external field. The reason is, of course, only one of scale: our ensemble, featuring only 200 neurons, is ca. 21 orders of magnitude smaller than $1\,\text{cm}^3$ of iron. It is therefore more appropriate to compare our networks to single-domain, microscopic magnets and the sudden switch in magnetization to a Barkhausen jump.

single-domain magnets, Barkhausen jumps

We can now introduce temperature into the picture. In ferromagnetic solids, a drop in temperature causes a phase transition at the Curie temperature T_C. Above the Curie temperature, the thermal noise is stronger than the lateral interaction and the material becomes paramagnetic – without an external field, the magnetization is zero. When the temperature drops below the Curie temperature, the spins spontaneously align themselves, and the material magnetizes spontaneously.

Curie temperature

spontaneous magnetization

In the Ising model, the (inverse) temperature has a multiplicative effect on the Hamiltonian, since

$$P_\beta(\boldsymbol{\sigma}) \propto e^{-H(\boldsymbol{\sigma})/k_B T} \quad , \tag{6.181}$$

so we can define an effective Hamiltonian $\tilde{H} = H/k_B T$. We have two options of translating this Hamiltonian to the LIF domain. On one hand, we can rescale all weights and biases by $\beta = 1/k_B T$, but this would be impractical, since a changing temperature would become difficult to emulate (standard neural network simulators are not built to

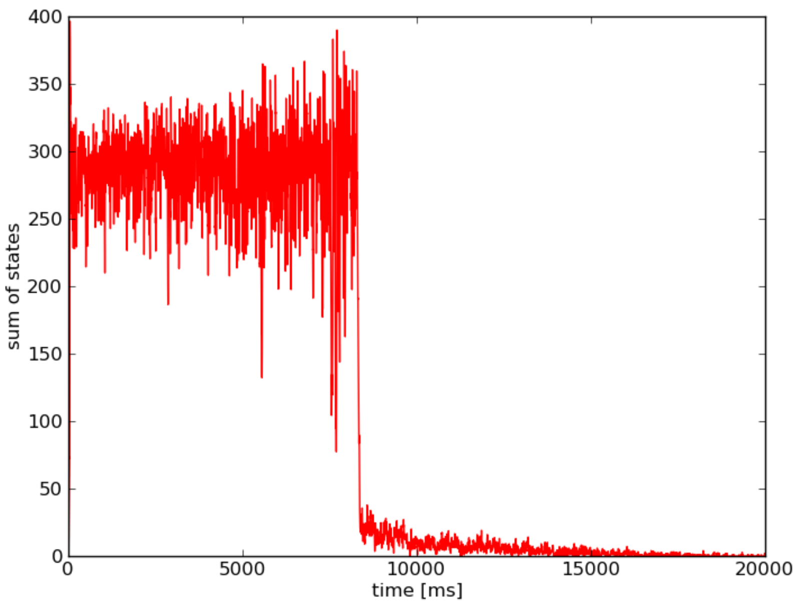


Figure 6.50.: Paramagnetic-to-ferromagnetic phase transition in single-domain magnets at critical temperature as observed in positively coupled recurrent LIF networks, where the rate of the background Poisson noise represents the equivalent of temperature. The noise rate decreases linearly in time. When approaching the critical rate (equivalent to the Curie temperature) from above, the spatial scale of the temperature-driven fluctuation increases, which is a universal feature of dynamical systems close to a critical point. When crossing the critical temperature, the network spontaneously jumps into an either completely silent or a maximum-frequency spiking state (here, we observe the former).

allow arbitrary, continuous changes of synaptic weights during simulation). On the other hand, temperature finds its much more natural equivalent in the characteristics of the background Poisson generators that are used in LIF sampling as a source of stochasticity (Section 6.5.1).

When all weights and biases are scaled by the same factor β, all membrane potentials will feature the same scaling: $\tilde{u} = \beta u$. Since the activation function depends only on $\tilde{u}$, changing the temperature results in a rescaling of the activation functions, with a lower temperature resulting in a steeper slope. We have already discussed how the activation function is closely related to the distribution of the free membrane potential (see Figure 6.29). The width of the free membrane potential distribution is therefore positively correlated (approximately proportionally) with the ensemble temperature. For the sake of simplicity[33], let us assume that our neural implementation consists of CUBA neurons, for

[33] For COBA neurons, the dependencies are more complicated, as we have discussed in Section 4.3.4. However, also for COBA neurons, a unique mapping of the background firing rate to the free membrane potential distribution width – and thereby to the ensemble temperature – can be found in the HCS, see Equation 4.108.

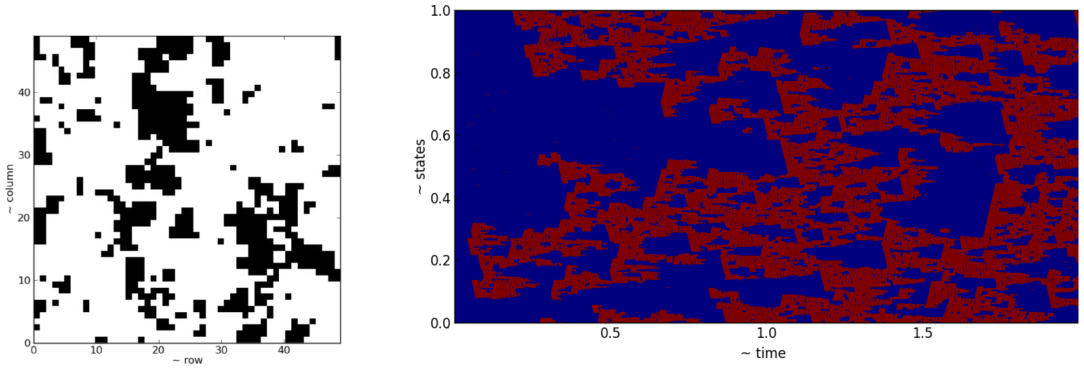


Figure 6.51.: Weiss domains in a ferromagnet above the critical temperature as observed in positively coupled recurrent LIF networks. **Left:** Weiss domains in a 2D lattice (snapshot) **Right:** Evolution of Weiss domains in a 1D Ising chain. The abscissa represents time, the ordinate represents the position along the Ising chain. In addition to the expected observation of moving Bloch walls, we can also observe an asymmetry between the ↑ and ↓ states.

which the width of the membrane potential distribution is proportional to the square root of the Poisson noise rates (Equation 4.98). We can thereby control the ensemble temperature by using inhomogeneous Poisson processes (i.e., Poisson processes with time-varying rates) as noise sources.

temperature as Poisson rate

If we now gradually lower the Poisson background rates in an excitatorily connected LIF network (the equivalent of a ferromagnet, see above), we observe similar phenomena as we would predict for ferromagnets around the Curie temperature 6.50. As the rates approach the critical point, the oscillations in the network activity (equivalent to the total magnetization) become larger. The increasing spatial scale of fluctuations as a phase transition is approached is a classical token of universality. Upon crossing the critical point, the network jumps into an almost perfectly quiescent state. Since the network is slightly biased towards $\boldsymbol{z} = \mathbf{0}$ (see also discussion below), the jump into the quiescent state (maximum negative magnetization) is more likely than the jump into a rapid-firing state (maximum positive magnetization).

critical point
universality

magnetic domains

As a final experiment, we study the appearance and evolution of magnetic domains. In macroscopic ferromagnetic solids, when the temperature drops below the Curie temperature, the material magnetizes spontaneously, but not uniformly. The resulting regions of uniform magnetization are called Weiss domains.

Weiss domains

On the scale of our networks, this phenomenon can not be replicated, since they are many orders of magnitude smaller than individual Weiss domains. However, similar patterns can appear above the Curie temperature as well (Figure 6.51, left panel). The domain walls (Bloch walls) are no longer stable, but evolve in time.

Bloch walls

In order to visualize this evolution, we relinquish one spatial dimension, leaving us with a 1D ferromagnet. A 1D Ising chain does not have a phase transition, but this does not matter since we study phenomena above the Curie temperature anyway. We can now represent the position-dependent magnetization of the chain on the ordinate and use the abscissa to study the temporal evolution of Bloch walls (Figure 6.51, right panel). As in the 2D case, we note the appearance of moving Weiss domains. However, we can also observe a manifest asymmetry in the behavior of "↑" (red) and "↓" (blue) domains.

↑-↓-asymmetry

For one, the ↓-domains are dominant, but that can be easily balanced out by appropriate biasing.

Secondly however, the evolution of the domains is clearly not T-symmetric (time-reversal invariant). When an ↑-domain forms, it tends to spread quickly, but when it shrinks, it does so more gradually. The reason for this asymmetry lies in the T-asymmetry of the interaction in LIF networks: PSPs are not symmetric, but start off by rising quickly and then gradually (exponentially) decline back to zero (see Figure 6.34). When a neuron surrounded by non-refractory neurons spikes, it initiates the spread of an ↑-domain. The membrane potentials of its neighbors jump nearly instantaneously, causing a high likelihood of them spiking themselves. The spread of activation thus happens quickly, explaining the fast expansion of ↑-domains. Analogously, the slow recession can be traced back to the long PSP tails.

T-asymmetry in LIF networks

We can also find a third discrepancy from the standard Ising model: upon closer inspection, we can observe that ↑-domains are interspersed with very narrow bands of blue color. These are not true ↓-domains, but represent the $\overline{\tau_k^{\mathrm{b}}}$, the short jump intervals between refractory states within bursts (see Equations 6.125 and 6.137). In principle, they can be reduced by increasing the membrane speed or reducing the threshold-to-reset distance, but they always contribute to the ↑-↓-asymmetry.

In hindsight we can now also explain the (somewhat less apparent) ↑-↓-asymmetry in the 2D lattice, where the ↑-domains, represented in black, are smaller and less interconnected than the ↓-domains, represented in white.

6.7. Sampling in Discrete Spaces with Spiking Neurons

In the previous sections, we have examined a very powerful formal equivalence between the dynamics of LIF networks and MCMC sampling from Boltzmann distributions. This has allowed the training of LIF networks for difficult inference tasks in high-dimensional probability spaces. Although the addition of hidden layers allowed the representation of non-Boltzmann distributions over the visible RVs, the final network structures were the result of a training algorithm; in particular, we gave no explicit representation (i.e., a mathematical expression) of the resulting distributions over the visible RVs. The question of sampling in arbitrary discrete spaces has not yet been answered.

In this section, we catch up on this remaining debt. In particular, we provide an explicit translation of arbitrary distributions of binary RVs to networks of LIF neurons.

In Section 6.3, we have already discussed an implementation of BNs in LIF networks, but have argued that the reliance on LSMs makes it inefficient in terms of absolute network size and difficult to scale to more complex distributions. Here, we make explicit use of our previously discussed implementation of LIF sampling, which uses a single neuron per binary RV. However, the extension of the neural sampling formalism to arbitrary distributions, represented as BNs, will require an additional overhead that is proportional to the size of individual factors. Still, as we shall see, this implementation is several orders of magnitude more efficient in terms of total number of neurons, while providing a better approximation of the required results in example inference problems.

After explaining the abstract model of sampling in BNs (Pecevski et al., 2011), we describe its LIF implementation, which is an extended version of the LIF sampling formalism described earlier. We then implement the same inference problem as in Section 6.3.3 and study the convergence behavior of the sampled distribution. In particular, we also discuss the robustness of the model to various types of noise, which is an important feature when aiming for a neuromorphic implementation. Finally, we provide a more general assessment of the performance of our LIF implementation of arbitrary non-Boltzmann distributions over several binary RVs. The results presented here are the outcome of a collaboration with Dimitri Probst and have already been published in (Probst et al., 2015).

6.7.1. Bayesian Networks as Boltzmann Machines

We start by recalling that BNs provide a graphical representation of arbitrary probability distributions, as discussed in detail in Section 6.1.1. The joint distribution defined by a Bayesian graph is the product of conditional distributions, one for each RV, with its value conditioned on the values of its parent RVs. For a graph with K binary RVs Z_k, the joint probability distribution is given by

$$p(\mathbf{Z}=\mathbf{z}) =: p(\mathbf{z}) = \prod_{k=1}^{K} \frac{1}{Z}\Phi_k(\mathbf{z}_k) := \prod_{k=1}^{K} p(z_k|\mathbf{pa}_k) \quad , \tag{6.182}$$

principal RVs

where $\mathbf{z}_k$ represents the state vector of the variables $\mathbf{Z}_k$ in Φ_k, which we henceforth call principal RVs, and $\mathbf{pa}_k$ represents the state vector of the parents of Z_k. Z is a normalizing constant; without loss of generality, we assume $\Phi_k > 1$. The factor $p(z_k|\mathbf{pa}_k)$ is called an n^{th}-order factor if it depends on n RVs or rather $|\mathbf{pa}_k| = n-1$.

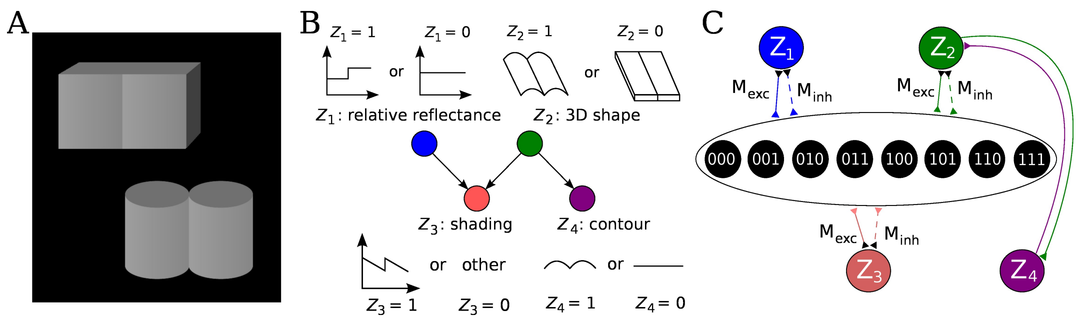


Figure 6.52.: Formulation of an example inference problem as a BN and translation to a BM. See also Section 6.3.3 for a more detailed description of the Knill-Kersten problem. (**A**) Knill-Kersten illusion from Knill and Kersten (1991). Although the four objects are identically shaded, the left cube is perceived as being darker than the right one. This illusion depends on the perceived shape of the objects and does not occur for, e.g., cylinders. (**B**) The setup can be translated to a BN with four binary RVs. The (latent) variables Z_1 and Z_2 encode the (unknown) reflectance profile and 3D shape of the objects, respectively. Conditioned on these variables, the (observed) shading and 2D contour are encoded by Z_3 and Z_4, respectively. (**C**) Representation of the Bayesian network from (B) as a BM. Factors of order higher than 2 are replaced by auxiliary variables as described in the main text. The individual connections with weights M_{exc}, $M_{\mathrm{inh}} \to \infty$ between each principal and auxiliary variable have been omitted for clarity. Figure taken from Probst et al. (2015).

BN as 2nd order MRF

Such a Bayesian network can now be transformed into a second-order MRF (i.e., an MRF with a maximum clique size of 2). First and second-order factors are easily replaceable by potential functions $\Psi_k(Z_k)$ and $\Psi_k(Z_{k1}, Z_{k2})$, respectively. For each n^{th}-order factor Φ_k with $n > 2$ principal RVs, we introduce 2^n auxiliary binary RVs $X_k^{\mathbf{z}_k \in \mathscr{Z}_k}$, where $\mathscr{Z}_k$ is the set of all possible assignments of the binary vector $\mathbf{Z}_k$ (Figure 6.52C). Each of these RVs encode the probability of a possible state $\mathbf{z}_k$ within the factor Φ_k by introducing the first-order potential functions $\Psi_k^{\mathbf{z}_k}(X_k^{\mathbf{z}_k} = 1) = \Phi_k(\mathbf{Z}_k = \mathbf{z}_k)$. The factor $\Phi_k(\mathbf{Z}_k)$ is then replaced by a product over potential functions

$$\Phi_k(\mathbf{Z}_k) = \prod_{\mathbf{z}_k} \Psi_k^{\mathbf{z}_k}(X_k^{\mathbf{z}_k}) \prod_{i=1}^{n} \chi_{ki}^{\mathbf{z}_k}(Z_{ki}, X_k^{\mathbf{z}_k}) \quad , \tag{6.183}$$

auxiliary RV

where an auxiliary RV $X_k^{\mathbf{z}_k}$ is active if and only if the principal RVs $\mathbf{Z}_k$ are active in the configuration $\mathbf{z}_k$. Formally, this corresponds to the assignment: $\chi_{ki}^{\mathbf{z}_k}(Z_{ki}, X_k^{\mathbf{z}_k}) = 1 - X_k^{\mathbf{z}_k}(1 - \delta_{Z_{ki}, z_{ki}})$. In the graphical representation, this amounts to removing all directed edges within the factors and replacing them by undirected edges from the principal to the auxiliary RVs. It can then be verified (Pecevski et al., 2011) that the target probability distribution can be represented as a marginal over the auxiliary variables.

As the resulting graph is a second-order MRF, its underlying distribution can be cast

Boltzmann form of BN

in Boltzmann form:

$$p(\mathbf{z},\mathbf{x}) = \frac{1}{Z}\exp\left(\frac{1}{2}\mathbf{z}^T\mathbf{W}\mathbf{z} + \frac{1}{2}\mathbf{z}^T\mathbf{V}\mathbf{x} + \mathbf{z}^T\mathbf{b} + \mathbf{x}^T\mathbf{a}\right) \quad , \tag{6.184}$$

weights and biases

where the (symmetric) weight matrices $\mathbf{W}, \mathbf{V}$ and bias vectors $\mathbf{b}, \mathbf{a}$ are defined as follows:

$$W_{Z_{ki},Z_{kj}} = \begin{cases} \log\frac{\Phi_k\left(Z_{ki}=0,Z_{kj}=0\right)\Phi_k\left(Z_{ki}=1,Z_{kj}=1\right)}{\Phi_k\left(Z_{ki}=0,Z_{kj}=1\right)\Phi_k\left(Z_{ki}=1,Z_{kj}=0\right)} & \text{within second-order factors } \Phi_k \\ 0 \quad \text{otherwise} \end{cases} \tag{6.185}$$

$$V_{Z_{ki},X_k^{\mathbf{z}_k}} = \begin{cases} M_{\text{exc}} & \text{if } \mathbf{z}_{ki} = 1 \\ M_{\text{inh}} & \text{if } \mathbf{z}_{ki} = 0 \end{cases} \tag{6.186}$$

$$b_{Z_{ki}} = \begin{cases} \log\frac{\Phi_k(Z_{ki}=1)}{\Phi_k(Z_{ki}=0)} & \text{within first-order factors} \\ \log\frac{\Phi_k\left(Z_{ki}=1,Z_{kj}=0\right)}{\Phi_k\left(Z_{ki}=0,Z_{kj}=0\right)} & \text{within second-order factors} \end{cases} \tag{6.187}$$

$$a_{X_k^{\mathbf{z}_k}} = \log\left(\Phi_k - 1\right) - L^1(\mathbf{z}_k)M_{\text{exc}} \quad , \tag{6.188}$$

all other matrix and vector elements being zero. $L^1(\cdot)$ represents the L^1 norm. In the theoretical model, $M_{\text{exc}} = \infty$ and $M_{\text{inh}} = -\infty$, but they receive finite values in the concrete implementation (Section 6.7.2). From here, it is straightforward to create a corresponding classical BM, which can then be translated to an LIF network following the rules laid out in Section 6.5.4. We therefore use a simplified notation from here on: we consider the vector $\boldsymbol{Z}$ to include both principal and auxiliary RVs and the Boltzmann distributions over $\boldsymbol{Z}$ are henceforth defined by the block diagonal weight matrix $\boldsymbol{W}$ and the bias vector $\boldsymbol{b}$.

6.7.2. Characterization of the Auxiliary Neurons

In the mathematical model in Section 6.7.1, the weights between principal and auxiliary RVs are $M_{\text{exc}} = \infty$ and $M_{\text{inh}} = -\infty$, to ensure a switching of the joint state whenever one of the auxiliary variables changes its assignment. In a concrete implementation, infinite weights are unfeasible. Here, we set the connection strengths $M_{\text{exc},k} = -M_{\text{inh},k} = \gamma \cdot \max\left[\Phi_k\left(\mathbf{z}_k\right)\right]$, where γ is a fixed number between 5 and 10. Neurons with a bias of $M_{\text{exc},k}$ will effectively spike at maximum rate and neurons with a bias of $M_{\text{inh},k}$ will remain silent, unless driven by afferent neurons with similarly high synaptic weights.

auxiliary neurons

The individual values of the factor $\Phi_k\left(\mathbf{z}_k\right)$ are introduced through the bias of the auxiliary neurons:

$$a_{X_k^{\mathbf{z}_k}} = \log\left(\mu\frac{\Phi_k\left(\mathbf{z}_k\right)}{\min_{\mathbf{z}_k}\left[\Phi_k\left(\mathbf{z}_k\right)\right]} - 1\right) - L^1(\mathbf{z}_k)\cdot M_{\text{exc},k} \tag{6.189}$$

where the factor $\mu/\min_{\mathbf{z}_k}\left[\Phi_k\left(\mathbf{z}_k\right)\right]$ ensures that the argument of the logarithm stays larger than 0 for all possible assignments $\mathbf{z}_k$.

observed RVs

Observed variables are clamped to fixed values 0 or 1 by setting the biases of the corresponding principal neurons to very large values (± 20), to ensure that they spike either at maximum rate or not at all. This could also be implemented by providing them with a strong excitatory or inhibitory external drive.

6.7.3. Lateral Interaction

In Section 6.5.4, we have already discussed how BM parameters must be translated to the LIF domain. In principle, these rules remain the same for the BM implementation of BNs.

For single neurons, if we denote by u_{eff} the effective membrane potential (i.e., the average membrane potential under constant external stimulus other than the synaptic noise), this yields a sigmoidal activation function $\tilde{\sigma}(u_{\mathrm{eff}})$ which can be linearly transformed to the logistic activation function $\sigma(v)$ to match the abstract model in Section 6.4.3:

$$v = \frac{u_{\mathrm{eff}} - \langle u \rangle_0}{\alpha} \quad , \tag{6.190}$$

where $\langle u \rangle_0$ represents the value of u_{eff} for which $p(Z = 1) = 1/2$. The factor α denotes a scaling factor between the two domains and is given by

$$\alpha = \left[4 \frac{\mathrm{d}}{\mathrm{d}u_{\mathrm{eff}}} \tilde{\sigma}(\langle u \rangle_0) \right]^{-1} \quad . \tag{6.191}$$

α

The bias b can then be set by changing the leak potential E_{l} such that the neuron is active with $\sigma(b)$ for $\boldsymbol{Z}_{\setminus k} = \mathbf{0}$:

$$E_{\mathrm{l}} = {u_{\mathrm{eff}}}^b \frac{\left\langle g^{\mathrm{tot}} \right\rangle}{g_{\mathrm{l}}} = (\alpha b + \langle u \rangle_0) \frac{\left\langle g^{\mathrm{tot}} \right\rangle}{g_{\mathrm{l}}} \quad , \tag{6.192}$$

where g^{tot} represents the total synaptic conductance and ${u_{\mathrm{eff}}}^b$ is the effective membrane potential that corresponds to the bias b: $\tilde{\sigma}({u_{\mathrm{eff}}}^b) = \sigma(b)$. For the translation of synaptic weights, we use Equation 6.145 for calculating the multiplicative scaling factor $\beta = w_{ki}/W_{ki}$:

$$\beta = \frac{\alpha C_{\mathrm{m}} \tau_{\mathrm{ref}} \left(\frac{1}{\tau_{\mathrm{syn}}} - \frac{1}{\tau_{\mathrm{eff}}} \right)}{E_k^{\mathrm{rev}} - \langle u \rangle} \cdot \left[\tau_{\mathrm{syn}} \left(\mathrm{e}^{-\frac{\tau_{\mathrm{ref}}}{\tau_{\mathrm{syn}}}} - 1 \right) - \tau_{\mathrm{eff}} \left(\mathrm{e}^{-\frac{\tau_{\mathrm{ref}}}{\tau_{\mathrm{eff}}}} - 1 \right) \right]^{-1} \quad . \tag{6.193}$$

β

In order to emulate renewing PSPs, we use the TSO STD mechanism with $U_{\mathrm{SE}} = 1$ and $\tau_{\mathrm{rec}} = \tau_{\mathrm{ref}}$ (see Figure 6.35).

TSO

Figure 6.53A shows the shape of such an LIF PSP with parameter values taken from Table A.21. The shape is practically exponential, due to the extremely short effective membrane time constant in the HCS. We will later compare the performance of the LIF implementation to two implementations of the abstract model from Section 6.4.3: neurons with theoretically optimal rectangular PSPs of duration τ_{ref}, the temporal evolution of which is defined as

$$u(t) = \begin{cases} 1 & \text{if } 0 < t_s - t < \tau_{\mathrm{ref}} \; , \\ 0 & \text{otherwise} \end{cases} \tag{6.194}$$

and neurons with alpha-shaped PSPs with the temporal evolution

$$u(t) = \begin{cases} q_1 \cdot \left[\mathrm{e} \cdot \left(\frac{t}{\tau_\alpha} + t_1 \right) \cdot \exp\left(-\frac{t}{\tau_\alpha} - t_1 \right) - 0.5 \right] & \text{if } 0 < t < (t_2 - t_1)\tau_\alpha \; , \\ 0 & \text{otherwise} \; . \end{cases} \tag{6.195}$$

Here, t_1 and t_2 are the points in time where the alpha kernel is $\mathrm{e} \cdot t \cdot \exp(-t) = 0.5$. The value $q_1 = 2.3$ is a scaling factor and $\tau_\alpha = 17\,\mathrm{ms} \cdot \frac{\tau_{\mathrm{ref}}}{30\,\mathrm{ms}}$ is the time constant of the kernel (Pecevski et al., 2011).

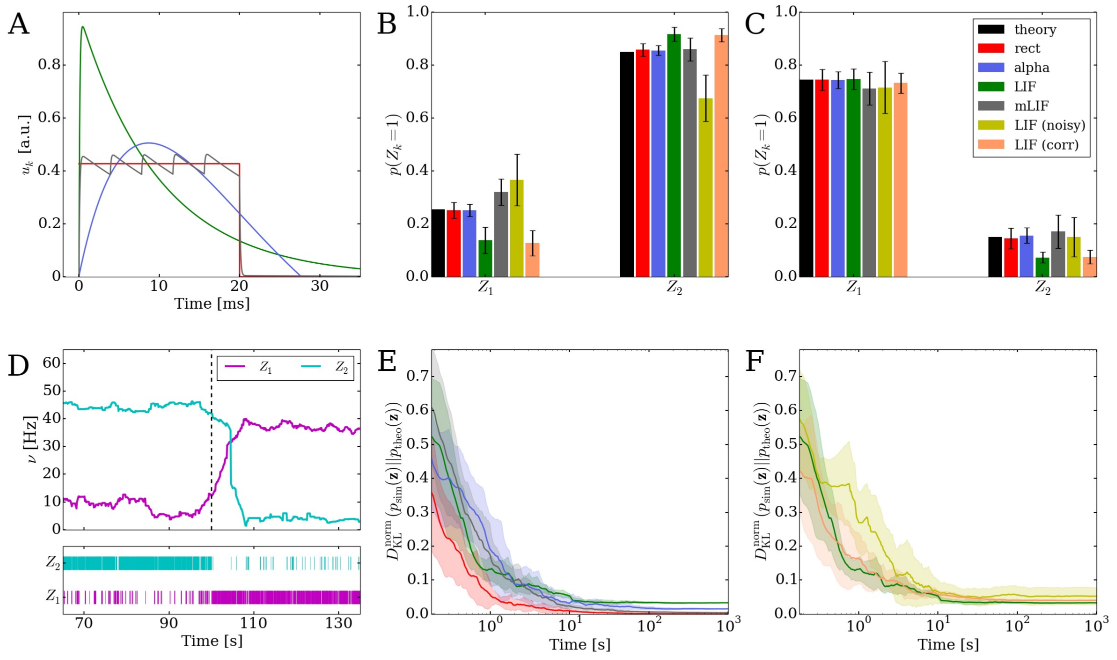


Figure 6.53.: Comparison of the different implementations of the Knill-Kersten graphical model (Fig. 6.52): LIF (green), LIF with noised parameters (yellow), LIF with small cross-correlations between noise channels (orange), mLIF PSPs mediated by a superposition of LIF PSP kernels (gray), abstract model with alpha-shaped PSPs (blue), abstract model with rectangular PSPs (red) and analytically calculated (black). The error bars for the noised LIF networks represent the standard error over 10 trials with different noised parameters. All other error bars represent the standard error over 10 trials with identical parameters. **(A)** Comparison of the four used PSP shapes. **(B, C)** Inferred marginals of the hidden variables Z_1 and Z_2 conditioned on the observed (clamped) states of Z_3 and Z_4. In B, $(Z_3, Z_4) = (1, 1)$. In C, $(Z_3, Z_4) = (1, 0)$. The duration of a single simulations is 10 s. (**D**) Marginal probabilities of the hidden variables reacting to a change in the evidence $Z_4 = 1 \rightarrow 0$. The change in firing rates (top) appears slower than the one in the raster plot (bottom) due to the smearing effect of the box filter used to translate spike times into firing rates. (**E, F**) Convergence towards the unconstrained equilibrium distributions compared to the target distribution. In D, the performance of the four different PSP shapes from A is shown. The abstract model with rectangular PSPs converges to $D_{\mathrm{KL}} = 0$, since it is guaranteed to sample from the correct distribution in the limit $t \rightarrow \infty$. In E, the performance of the three different LIF implementations is shown. Figure taken from Probst et al. (2015).

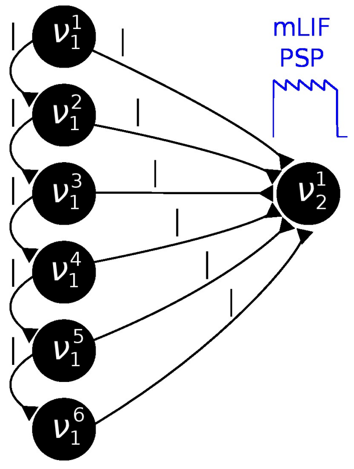


Figure 6.54.: In order to establish a coupling which is closer to the ideal one (rectangular PSP), the following network structure was set up. Instead of using one principal neuron ν per RV, each RV is represented by a neural chain. In addition to the network connections imposed by the translation of the modeled BN, feedforward connections between the neurons in this chain are also instantiated. Each of the chain neurons projects onto the first neuron of the chain representing the postsynaptic target RV (here: all connections from ν_1^i to ν_2^1). By choosing appropriate synaptic efficacies and delays, the chain generates a superposition of single PSP kernels that results in a sawtooth-like shape which is closer to the desired rectangular shape than a single PSP. Figure taken from Probst et al. (2015).

As we can clearly see in Figure 6.53A, there exists a pronounced difference in PSP shapes between the LIF domain and the theoretically optimal abstract model. This is the main reason why the direct translation to LIF networks causes non-negligible deviations from the target probability distribution. The sometimes strong interaction involved in the expansion of BNs into BMs (see Equation 6.186) leads to a large (in absolute terms) overshoot of the membrane potential at the arrival of a PSP, as well as a large PSP tail beyond $t = t_{\text{spike}} + \tau_{\text{ref}}$.

mLIF PSPs

In order to reduce this discrepancy, we replaced the single-PSP interaction between pairs of neurons by a superposition of LIF PSP kernels (mLIF PSPs). For this, we replaced the single neuron that coded for an RV by a chain of neurons (see Figure 6.54). In this setup, the first neuron in a chain is considered the “main” neuron, and only the spikes it emits are considered to encode the state $z_k = 1$. However, all neurons from a chain project onto the main neuron of the chain representing a related RV. This neuron then registers a superposition of PSPs, which can be adjusted (e.g., with the parameter values from Table A.22) to closely approximate a rectangular PSP. In particular, the long tail of the last PSP is cut off by setting the effect of the last neuron in the chain to oppose the effect of all the others (e.g., if the interaction between the RVs is to be positive, all neurons in the chain project with excitatory synapses onto their target, while the last one has an inhibitory outgoing connection). While this implementation only scales the number of network components (neurons and synapses) linearly with the chosen length of the chains, it improves the sampling results significantly (Figure 6.53 B, C, E, gray bars/traces).

6.7.4. Bayesian Model of the Knill-Kersten Illusion

Figure 6.52 illustrates the translation of the Bayesian graph describing the well-studied Knill-Kersten illusion (Knill and Kersten, 1991, see also Section 6.3.3) to the LIF domain. The underlying Bayesian model consists of four RVs: Z_1 (reflectance step versus uniform reflectance), Z_2 (cylindrical versus cuboid 3D shape), Z_3 (sawtooth-shaped versus some other shading profile) and Z_4 (round versus flat contour). The network structure defines the decomposition of the joint probability distribution:

$$p(Z_1, Z_2, Z_3, Z_4) = p(Z_1)\, p(Z_2)\, p(Z_3|Z_1, Z_2)\, p(Z_4|Z_2) \quad . \tag{6.196}$$

The inference problem consists in estimating the relative reflectance of the objects given the (observed) contour and shading. Analytically, this would require calculating $p(Z_1|Z_3 = 1, Z_4 = 0)$ for the cuboid shapes and $p(Z_1|Z_3 = 1, Z_4 = 1)$ for the cylindrical ones.

Figure 6.53 shows the behavior of the LIF network that represents this inference problem. When no variables are clamped, the network samples freely from the unconstrained joint distribution over the four RVs. The performance of the network, i.e., its ability to sample from the target distribution, is quantified by the Kullback-Leibler (KL) divergence between the target and the sampled distribution normalized by the entropy of the target distribution:

$$D_{\mathrm{KL}}^{\mathrm{norm}}\left(q \parallel p\right) = \frac{D_{\mathrm{KL}}\left(q \parallel p\right)}{H(p)} \quad , \tag{6.197}$$

with the KL divergence between the sampled distribution q and the target distribution p

$$D_{\mathrm{KL}}(q||p) = \sum_{\mathbf{z}} q(\mathbf{z}) \log\left(\frac{q(\mathbf{z})}{p(\mathbf{z})}\right) \tag{6.198}$$

and the entropy of the target distribution p

$$H(p) = -\sum_{\mathbf{z}} p(\mathbf{z}) \log\left[p(\mathbf{z})\right] \; . \tag{6.199}$$

When presented with the above inference problem the LIF network performs well at sampling from the conditional distributions $p(Z_1|Z_3, Z_4)$ (Figure 6.53B, C). When the stimulus is changed during the simulation, the optical illusion, i.e., the change in the inferred (perceived) 3D shape and reflectance profile, is clearly represented by a change in firing rates of the corresponding principal neurons (Figure 6.53D). For each point in time, the rate is determined by convolution of the spike train with a rectangular kernel

$$\kappa(t) = \begin{cases} 1/8\,\mathrm{Hz} & \text{for } -8\,\mathrm{s} < t < 0 \quad , \\ 0 & \text{otherwise} \quad . \end{cases} \tag{6.200}$$

At $t = 100\,\mathrm{s}$ (dotted line), the evidence is switched: $Z_4 = 1 \rightarrow 0$. The network reacts appropriately on the time scale of several seconds, as can be seen in the spike raster plot.

When not constrained by prior evidence, i.e., when sampling from the joint distribution over all RVs, the LIF network settles on an equilibrium distribution that lies close to the target distribution (Figure 6.53E, F, green traces). For this particular network, the convergence time is of the order of several tens of seconds.

6.7.5. Robustness to Parameter Distortions

We further investigated the robustness of our proposed implementation of Bayesian inference with LIF neurons to low levels of parameter noise (see Table A.21, noisy). Here, we focus on fixed-pattern noise, which is inherent to the production process of semiconductor integrated circuits and is particularly relevant for analog neuromorphic hardware (Mitra et al., 2009; Petrovici et al., 2014, see also Chapter 5). However, such robustness would naturally also benefit in-vivo computation.

fixed-pattern noise

Some of the noise (the one affecting the neuron parameters that are not changed when setting weights and biases) can be completely absorbed into the translation rules from Section 6.7.3. Once the neurons are configured, their activation curves can simply be measured, allowing a correct transformation from the abstract to the LIF domain. However, while the neurons remain the same between different simulation runs, the weights and biases may change depending on the implemented inference problem and are still subject to noise. Nevertheless, even with a noise level of 10% on the weights and biases, the LIF network still produces useful predictions (Figure 6.53B, C, F, yellow bars/traces).

6.7.6. Robustness to Noise Correlations

The investigated implementation of Bayesian networks ideally requires each neuron to receive independent noise as a Poisson spike train. When aiming for a hardware implementation of large Bayesian networks, this requirement may become prohibitive due to the bandwidth limitations of any physical back-end. We therefore examined the robustness of our LIF networks to small cross-correlations between the Poissonian noise channels of individual neurons.

cross-correlations

For both the excitatory and the inhibitory background pools, we induced pairwise noise correlations by allowing neurons within the network to share 10% of their background Poisson sources. The controlled cross-correlation of 10% between noise channels is achieved in the following way: each neuron receives Poisson background from three shared and seven private Poisson spike trains. The excitatory and inhibitory noise of each individual neuron remained uncorrelated in order to leave its activation function (Equation 6.125) unaltered. Each of the shared sources projects onto exactly two neurons in order to prevent higher-order correlations. The single Poissonian spike trains have a firing rate of $\nu/10$, such that their superposition is also Poisson, with the target firing rate of ν. With this setup, we were able to verify that small pairwise correlations in the background noise do not significantly reduce the ability of the LIF network to produce useful predictions (Figure 6.53B, C, F orange bars/traces).

6.7.7. General Bayesian Networks

In order to study the general applicability of the proposed approach, we quantified the convergence behavior of LIF networks generated from random BNs. Here, we used a method proposed in Ide and Cozman (2002) to generate random BNs with K binary RVs and random conditional probabilities. The algorithm starts with a chain graph $Z_1 \to Z_2 \to \cdots \to Z_K$ and runs for N iterations. In each iteration step, random RV pairs (Z_i, Z_j) with $i > j$ are created. If the connection $Z_i \to Z_j$ does not exist, it is added to the graph, otherwise it removed, with two constraints: any pair of nodes may not have more than 7 connections to other nodes and the procedure may not disconnect the graph.

random BNs

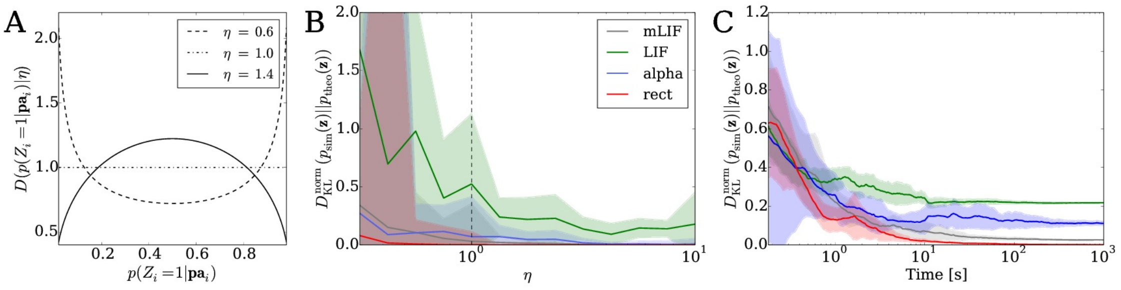


Figure 6.55.: Sampling from random distributions over 5 RVs with different networks: LIF (green), mLIF (gray), abstract model with alpha-shaped PSPs (blue) and abstract model with rectangular PSPs (red). **(A)** Distributions for different values of η from which conditionals are drawn. **(B)** D_{KL} between the equilibrium and target distributions as a function of η. The error bars denote the standard error over 30 different random graphs drawn from the same distribution. **(C)** Evolution of the D_{KL} over time for a sample network drawn from the distribution with $\eta = 1$. Error bars denote the standard error over 10 trials. Figure taken from Probst et al. (2015).

For every possible assignment of $\mathbf{pa}_i$, the conditional probabilities $p_i^{\mathbf{pa}_i} := p(Z_i = 1|\mathbf{pa}_i)$ are drawn from a second-order Dirichlet distribution

$$D(p_i^{\mathbf{pa}_i}, \eta_1, \eta_2) = \frac{1}{B(\eta_1, \eta_2)} (p_i^{\mathbf{pa}_i})^{\eta_1 - 1} (1 - p_i^{\mathbf{pa}_i})^{\eta_2 - 1} \quad , \tag{6.201}$$

with the multinomial Beta function

$$B(\eta_1, \eta_2) = \frac{\prod_{i=1}^{2} \Gamma(\eta_i)}{\Gamma\left(\sum_{i=1}^{2} \eta_i\right)} \quad , \tag{6.202}$$

where $\Gamma(\cdot)$ denotes the gamma function. We chose the parameters $\eta_1 = \eta_2 =: \eta$ in order to obtain a symmetrical distribution. Figure 6.55A shows three examples of a symmetrical two-dimensional Dirichlet distribution. A larger η favors conditional probabilities which are closer to 0.5 than to the boundaries 0 and 1.

We implemented Bayesian networks with $K = 5$ RVs running for $N = 50000$ iterations. The random graphs were then translated to sampling neural networks, both with abstract model neurons and LIF neurons. The performance was tested for sampling from the unconstrained joint distributions over the 5 RVs. In the simulations, we varied η between 0.3 and 10 and created 30 random Bayesian graphs for each η. Each network was then run for a total duration of 100 s.

Figure 6.55B illustrates the average sampling results for the different PSP shapes as a function of the "extremeness" of the randomized conditional probabilities, which is reflected by the parameter η. For larger η, conditionals cluster around 0.5 and the RVs become more independent, making the sampling task easier and therefore improving the sampling performance. The curves show the median of the D_{KL} between sampled and target distributions of the 30 random Bayesian graphs. The shaded regions denote the standard error. Overall, the LIF networks perform well, capturing the main modes of the target distributions.

Figure 6.55C shows the temporal evolution of the D_{KL} between sampled and target distributions for a sample Bayesian network drawn from the distribution with $\eta = 1$ that lied close to the D_{KL} median in Figure 6.55B. The curves illustrate the average results of 10 simulations, while the shaded regions denote the standard error.

As with the Bayesian model of the Knill-Kersten illusion, the main cause of the remaining discrepancy is the difference in PSP shapes between the LIF domain and the theoretically optimal abstract model. A modification of the RV coupling by means of the neuron chains described in Section 6.7.3 leads to a significant improvement of the sampling results for arbitrary Bayesian networks (Figure 6.55B, C gray traces).

6.8. Neuromorphic Neural Sampling

In the previous sections, we have given a brief overview of some interesting applications of LIF sampling. One of the key motivations behind the formulation of the LIF sampling framework was the possibility to use fast, inherently parallel hardware for sampling-based applications, which would obviously benefit both the convergence speed of individual experiments as well as the training of these networks. At first glance, it would seem that LIF sampling would not impose too harsh requirements on neuromorphic hardware that is able to handle experimental setups that are as varied and complex as those presented in Sections 5.3.9, 5.4.10 and 6.2.3. However, it turns out that the interplay between the non-negligible spike transmission delays (see also Table 3.3 for corresponding values on the wafer-scale system) and the implementation of refractoriness on the Spikey chip renders a neuromorphic implementation of LIF sampling nontrivial.

spike transmission delays

On the Spikey chip, spike transmission delays are approximately constant and, for all practical purposes, independent of the particular position of the neuron on the chip. They can therefore be measured easily in a single-neuron experiment (Figure 6.56). The neuron is connected to itself via a strong inhibitory synapse and then excited by a strong enough external stimulus (excitatory spike) to cause it to spike. The refractory period τ_{ref} is set to zero and the reset potential ϱ is chosen to be lower than the leak potential E_l, which causes the membrane potential to start rising again immediately following the spike reset. When its own inhibitory spike reaches the neuron again, it will manifest itself as a sudden drop (IPSP) in the membrane potential. The membrane reset and the sudden switch of the direction in which the membrane potential evolves are two easily measurable, well-defined points in time, the difference of which precisely yields the spike transmission delay Δt. As can be seen in Figure 6.56, spike transmission delays on the Spikey chip are approximately $\Delta t = 1.5\,\text{ms}$ long (in a biological frame of reference).

As already discussed in Section 6.4.3, spike transmission must be instantaneous in the ideal neural sampling model. The only way to minimize the distortive effects of delays (Figure 6.28) is to increase the absolute refractory times such that the neurons spend

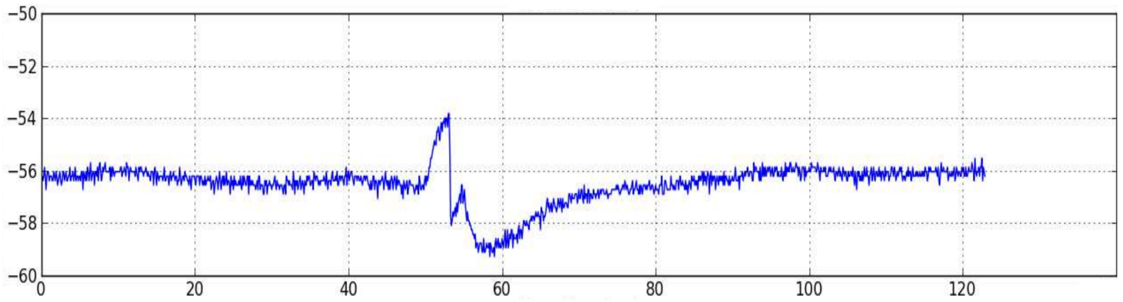


Figure 6.56.: Spike transmission delay on the Spikey chip. A neuron is connected to itself via an inhibitory synapse. Following a spike, the membrane potential relaxes towards its resting value, but is suddenly pulled down again by the incoming spike. The distance between the membrane potential reset and the following kink in its temporal evolution represents the spike transmission delay Δt. Figure taken from Petkov (2012).

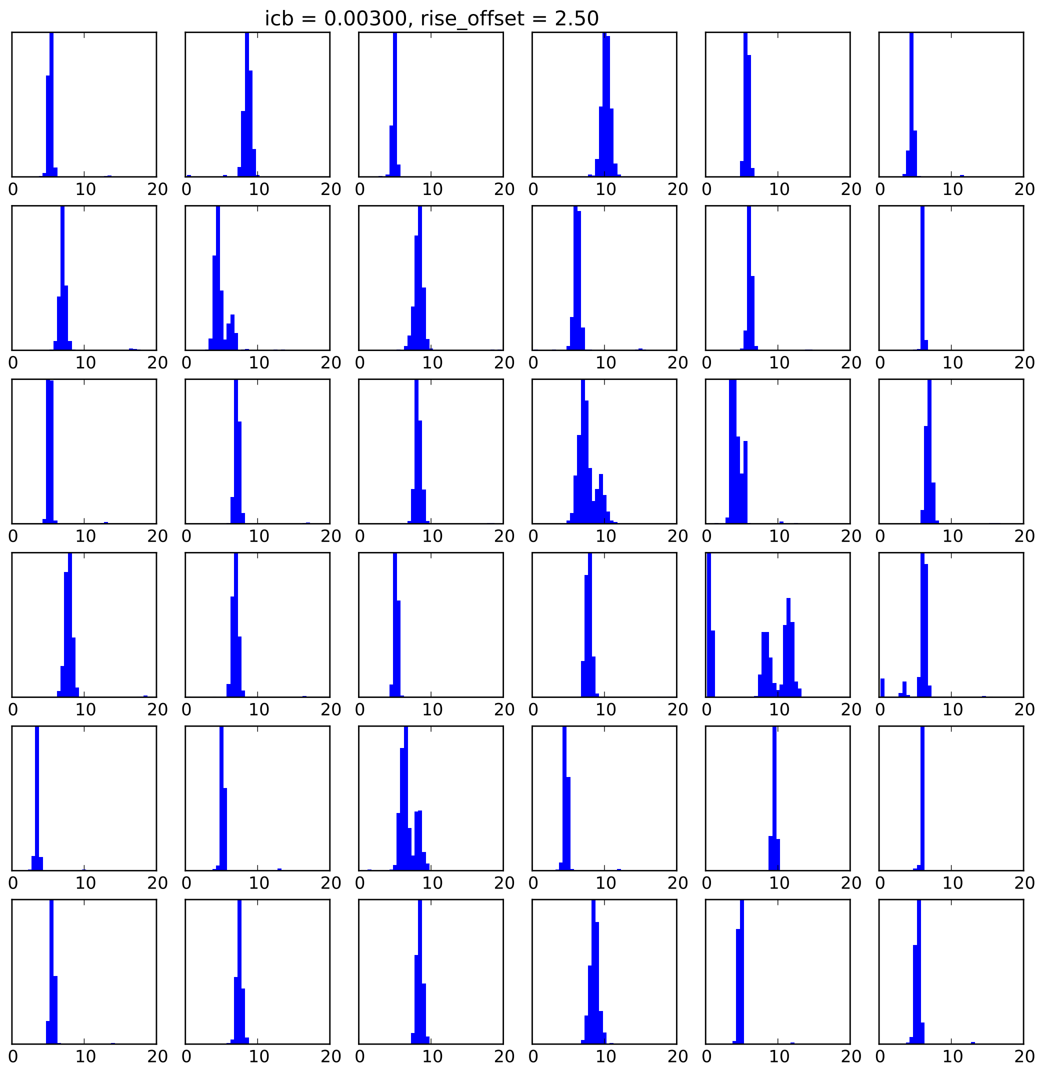


Figure 6.57.: Spike-to-spike variability of refractory times τ_{ref} on the Spikey chip. A random selection of 36 neurons is shown. Histograms are not normalized. A larger average τ_{ref} tends to correlate positively with a larger variability. Shorter τ_{ref} are more stable, but are not large enough compared to the spike transmission delay $\Delta T = 1.5\,\text{ms}$. This effectively prohibits a straightforward implementation of LIF sampling on the Spikey chip.

proportionately less time in “wrong” states. Fortunately, the Spikey chip features a global parameter that controls the refractory times of all neurons. Unfortunately, it turns out to be unfeasible for LIF sampling.

spike-to-spike variability of τ_{ref}

Figure 6.57 shows a random selection of 36 neurons from the Spikey chip for relatively high target values of τ_{ref}. The histograms show spike-to-spike variations of τ_{ref}. We note that, when the average τ_{ref} is large, the spike-to-spike variance tends to be large as well. This is, of course, a knock-out criterion for neural sampling, since the interpretation of spikes as samples is inherently based on a stable encoding of 1-states. When the variance-to-mean ratio of τ_{ref} is small enough, τ_{ref} itself tends to be too small (on the order of 5 ms) to effectively counter 1.5 ms delays.

The positive correlation between the mean and variance of τ_{ref} on the Spikey chip can be traced back to the physical implementation of refractoriness. The control parameter for τ_{ref} is a current variable `icb`. When a neuron spikes, this current is injected onto a capacitor, which releases the neuron from refractoriness once the voltage over the capacitor exceeds a certain value. Since the charge curve of the capacitor under constant-current stimulation is linear, `icb`is inversely proportional to τ_{ref}. The large τ_{ref} required for sampling must therefore be set with a small `icb`current. Assuming a fixed amplitude of electronic temporal noise, small currents experience larger relative variations, thus explaining the large spike-to-spike variability of long refractory times.

While the instability of long τ_{ref} effectively prohibits a straightforward implementation of LIF sampling on the Spikey chip, the availability of a sufficiently large number of neurons as well as the complete freedom in choosing a connectivity matrix can be put to use to create a more complex “sampling unit” that consists of more than a single neuron. This sampling unit is inspired by the previously discussed synfire chain model (Section 5.4) and the mLIF interaction mechanism used in the LIF-based BNs (Figure 6.54), but contains several important extensions.

sampling unit

A sampling unit is composed of a principal neuron and a synfire chain with feedback inhibition. The principal neuron is an LIF sampler with the same membrane dynamics as we have used in the previous sections, but with only a short refractory time τ_{ref}. In particular, its refractory state is no longer considered to encode the state $z_k = 1$. The principal neuron projects with an excitatory synapse onto all neurons of the first stage of the synfire chain, thus initiating a propagating pulse as soon as it spikes.

The synfire chain contains n stages, each of which comprises an excitatory and an inhibitory neuron population. Each excitatory population projects onto both populations of the following stage, thus enabling the forward propagation of the pulse. Each inhibitory population projects back onto both populations of the previous stage, ensuring that they become silent after they have spiked exactly once. Furthermore, each inhibitory population has strong projections towards the principal neuron, thus prohibiting from spiking while the synfire chain is active. This is reversed in the final synfire stage, where the excitatory population projects onto the principal neuron, thus causing its membrane potential to return quickly to its resting state.

pseudo-refractoriness

The synfire chain effectively replaces the refractory mechanism of the principal neuron. The period of pseudo-refractoriness τ_{on}, which now indeed encodes the $z_k = 1$ state, can be arbitrarily modulated by changing the length of the synfire chain. In principle, this setup would also work with one neuron per synfire population, but the advantage of using a larger number of neurons is that it averages out synaptic weight noise, as we have

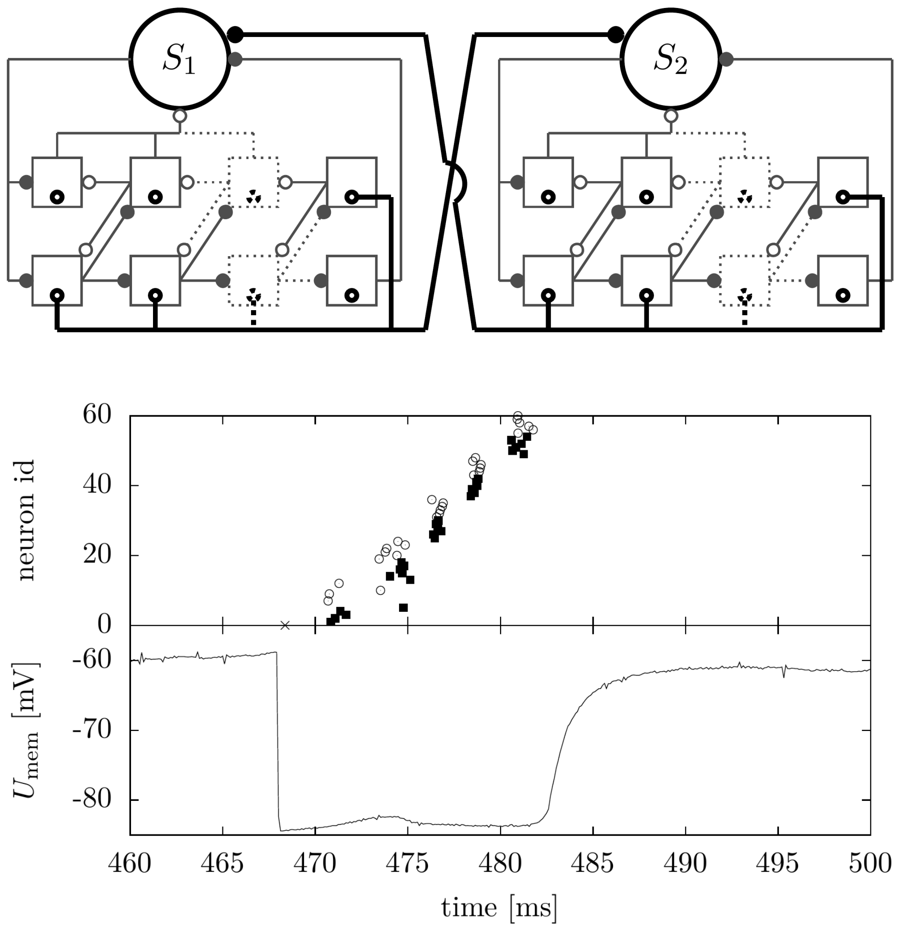


Figure 6.58.: Implementation of sampling units. **Top:** Principal neurons S_1 and S_2 obey the membrane dynamics of LIF samplers, but only have short refractory times. Each principal neuron is connected to a synfire chain, whose inhibitory neurons project back onto the principal neuron, thus causing a pseudo-refractory period τ_{on} as long as the spike pulse propagates along the chain. Lateral interaction is mediated by the same synfire chains, thus causing the equivalent of an mLIF PSP of duration τ_{on}. **Bottom:** Single pseudo-refractory period of the principal neuron S_1. Note the long refractory duration of approx. 15 ms. Figure taken from Stöckel (2015).

often observed in Chapter 5. The synfire chain would also work with a single population per stage which features both forward excitatory and backward inhibitory projections. However, the Spikey chip enforces Dale's law (see Section 2.1.3), which requires us to differentiate between purely excitatory and purely inhibitory neurons.[34]

lateral interaction

Figure 6.58 shows two such interacting sampling units (top panel), along with a depiction of the pseudo-refractory period of a single principal neuron (bottom). Note how the interaction between the principal neurons is mediated by the same synfire chains that cause the pseudo-refractoriness. It is thereby guaranteed that the duration of the effective interaction, which is equivalent to the mLIF PSPs from Figure 6.54, is equal to the pseudo-refractory period τ_{on}, as required by Equations 6.78 and 6.93.

In principle, these sampling units could be directly implemented on a calibrated Spikey chip and used to sample from Boltzmann distributions. However, neural sampling requires a relatively high precision of refractory times, interaction times and, most importantly, interaction strengths. In particular, the lateral interaction must be symmetric by definition. It turns out that the standard Spikey calibration routines are insufficiently precise for this task. In particular, they do not take into account effects such as capacitive crosstalk between synapse columns, which is, however, rather pronounced in synchronous firing scenarios, as is the case for the synfire chains in the sampling units.

sampling-specific calibration

Therefore, new calibration methods are needed, which are targeted specifically at the requirements of LIF sampling. Ideally, they should also be based on spike recordings (and not membrane potentials) in order to fully profit from the speedup factor of the hardware.[35] Recently, we have developed such methods, based on auto- and cross-correlograms of spike trains.

correlograms

The essential idea behind these routines is that the duration and symmetry properties of refractoriness and lateral interaction can be translated directly to identical properties of correlograms. In particular, the auto-correlogram of a principal neuron must be symmetric, with a pronounced peak at zero surrounded by two troughs of width τ_{on}. Similarly, the cross-correlogram of two principal neurons must also be symmetric, with two positive/negative peaks around zero for positive/negative w_{ij}, both of which must have the same approximate width τ_{on}.

Starting from an initial state of auto- and cross-correlograms, the calibration routine iteratively updates the network parameters in order to achieve the desired shape. Two exemplary correlograms, both before and after calibration, are shown in Figure 6.59. For a much more detailed technical description of the calibration procedure, we refer to Stöckel (2015).

After a successful calibration, the sampling performance of the network can be tested by calculating the $D_{\mathrm{KL}}\left(p_{\mathrm{sampled}} \parallel p_{\mathrm{target}}\right)$. Since the synfire populations require approximately 5 stages for a long enough pseudo-refractory period, and each population requires several neurons in order to compensate for synaptic weight noise (here: 6), a single sampling unit requires a total of 61 neurons, thus limiting the total number of implemented RVs to three (recall that the Spikey chip only has 192 neurons per block, see Section 3.2).

[34] This is not the case for the HICANN chip (Section 3.3.1), which would thereby allow a sparser implementation of such sampling units.

[35] Voltage recordings are more complicated, require additional readout hardware (ADCs or oscilloscopes) and can only be done for a limited number of neurons at the same time (see Section 3.2).

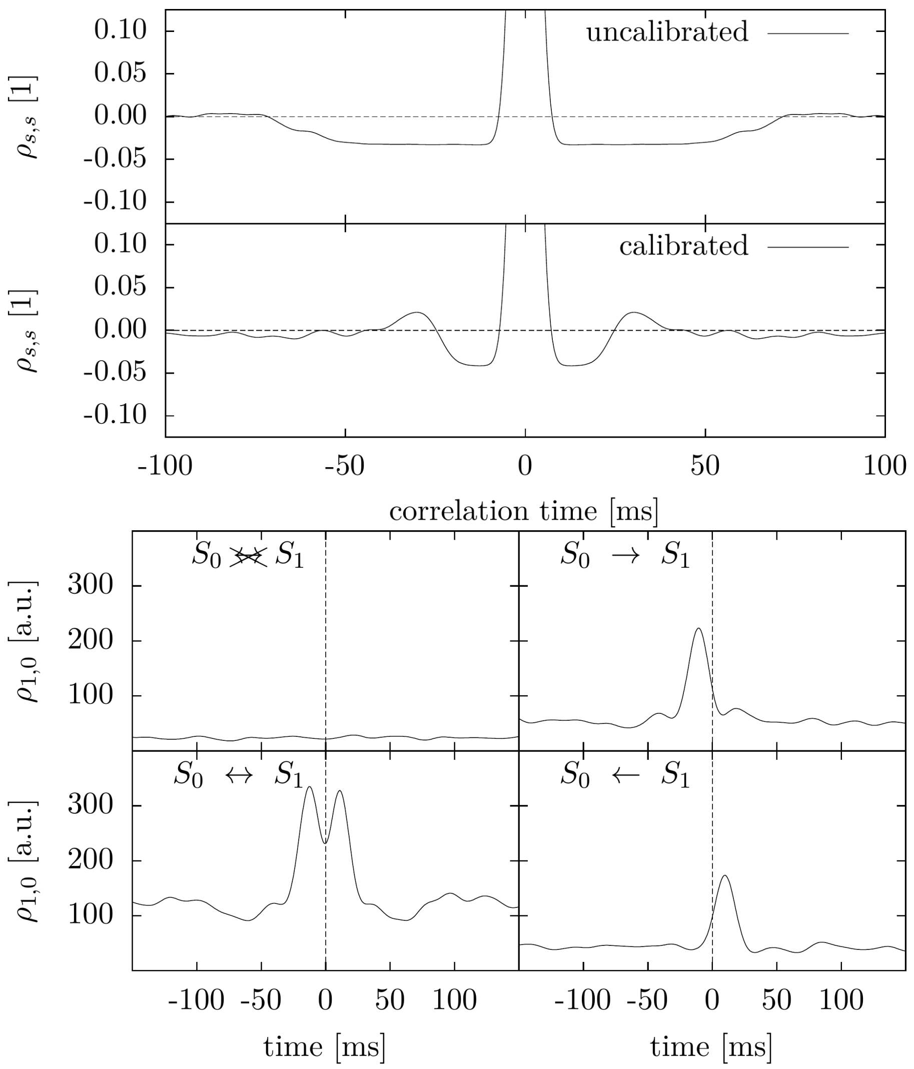


Figure 6.59.: Spike activity correlograms used for sampling-specific calibration. **Top:** Auto-correlogram of principal neuron spike trains. Before calibration, the duration of the pseudo-refractory period, represented by the symmetric troughs, is too long. It is also followed by extended periods of relative refractoriness (shallow recovery of the autocorrelation function), which is equivalent to an instable absolute refractory period and therefore disruptive for neural sampling. After calibration, the pseudo-refractory period is shorter and the recovery is faster (steeper rising flanks of the autocorrelation function). **Bottom:** Cross-correlogram of principal neuron spike trains. Without lateral projections, the cross-correlation function is flat (top left). When positive unidirectional connections are activated, peaks form (top and bottom right), which are asymmetric before calibration. After calibration (bottom left), the peaks are not only symmetric, but also have a width equal to the refractory period that was previously calibrated (left figure). Figure taken from Stöckel (2015).

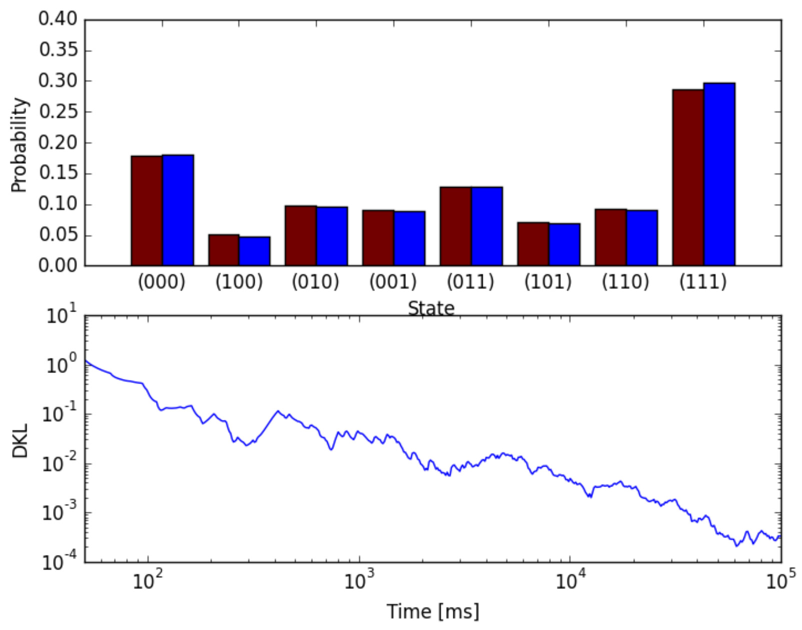


Figure 6.60.: Boltzmann distribution over 3 RVs: target distribution (red) vs. sampled distribution (blue) on the Spikey chip. **Top:** Direct comparison of the two distributions after 10^5 ms of sampling. Note that the wall-clock duration of this experiment was a mere 10 ms. **Right:** Evolution of $D_{\text{KL}}\left(p_{\text{sampled}} \parallel p_{\text{target}}\right)$ as a function of the simulation time.

Figure 6.60 shows the comparison between the sampled and the target distribution, along with the evolution of the D_{KL} as an increasing number of samples is acquired by the network. The exact value of $D_{\text{KL}}(t = 10^5\,\text{ms})$ varies slightly between runs, but remains sufficiently low to declare our modified implementation of LIF sampling successful. We need to stress that this experiment, with a duration of 100 biological seconds, only takes 10 ms on the Spikey chip. To our knowledge, this is the first implementation of neural sampling on accelerated analog neuromorphic hardware.

In a final batch of 50 experiments with randomly drawn, unbiased ($\boldsymbol{b} = \boldsymbol{0}$) distributions, we have evaluated the overall quality of our neuromorphic implementation of LIF sampling compared to software simulations of "standard" LIF sampling (one neuron per RV, as described in Section 6.5). Here, we have used reduced-size synfire chains in order to allow 4 sampling units to fit onto the Spikey chip. The bottom right panel of Figure 6.61 shows histograms over $D_{\text{KL}}\left(p_{\text{sampled}} \parallel p_{\text{target}}\right)$. While the histogram of the hardware results (top) has a few high-D_{KL} outliers, most runs have a D_{KL} below 10^{-2}, which is quite accurate for a Boltzmann distribution over 4 binary RVs, as seen in the surrounding panels. Overall, the hardware implementation performs only slightly worse than the ideally parametrized, noise-free software version. A further refinement of the calibration and training procedures is likely to improve these results even further.

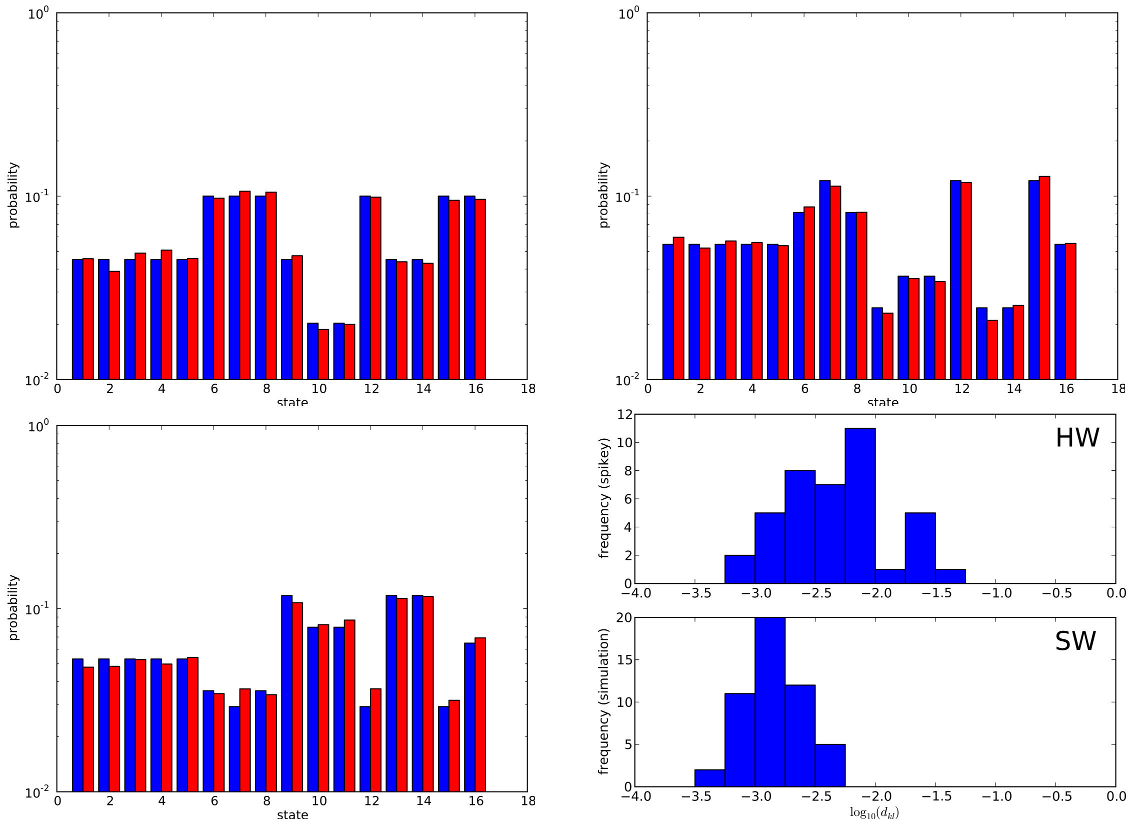


Figure 6.61.: Sampling from Boltzmann distributions over 4 binary RVs: Spikey implementation vs. ideally parametrized software simulation. **Top row and bottom left:** Exemplary sampled distributions (red) in direct comparison to the target distributions (blue). **Bottom right:** Histograms over $D_{\mathrm{KL}}\left(p_{\mathrm{sampled}} \parallel p_{\mathrm{target}}\right)$.

6.9. Conclusions and Outlook

Bayesian inference is a fundamental principle in the context of probabilitstic computation. In recent years, the hypothesis that the brain is operating, at some level, according to this principle, has been supported by increasing amounts of evidence. Long before that, Bayesian inference has been embedded in numerous machine learning models. The goal of this chapter was to study models of Bayesian inference that are compatible with biological neural structures and that are, at the same time, amenable for implementation on state-of-the-art accelerated neuromorphic devices. A marked difference to, for example, the networks studied in Chapter 5, is that the workflow here was not from biology to models, but rather the other way around. In other words, we took inspiration from powerful machine learning models and cast them onto a more biological scaffold.

We have started with a discussion of graphical models, in particular factor graphs, motivated by the efficient belief propagation algorithms that they enable. Since the nodes in generic factor graphs should be able to implement arbitrary computations (i.e., depending on the modeled inference problem, the output messages from a node could be, in principle, any function of the incoming messages), we turned to liquid state machines as a suitable implementation of generic spike-based computation. In particular, we have shown in Section 6.2.3 that liquid state machines are amenable to analog neuromorphic implementation. Following the ideas first proposed in Steimer et al. (2009), we have used such liquid architectures to implement belief propagation in Forney factor graphs. In software simulations of our liquid-based factor graphs, we have shown this approach to work in principle, but have also found several critical drawbacks, most importantly the requirement of a large number of neurons even for relatively simple graphs and the manifest lack of precision in messages computed by nodes that lie further away from static external inputs (observed variables).

We therefore went from these analytical inference models, where probabilities are represented explicitly by network observables (for the liquid-based graphs, messages were encoded by population firing rates) to sampling-based models, where the network activity at any point in time represents a state in the relevant probability space. Gathering these states over a certain time interval then leads to an increasingly correct representation of the target probability distribution. In this context of sample-based inference, we have described the abstract model of sampling from Boltzmann distributions proposed by Buesing et al. (2011) as a starting point for a more mechanistic implementation of neural sampling – in our case, an implementation with LIF neurons. The translation of this inherently stochastic model to networks of LIF neurons turns out to be non-trivial, since the necessary neural response (or activation) functions are not readily achieved under typical conditions (slow membranes and fast synapses). However, we were able to show that in the high-conductance state, LIF neurons can achieve a symmetric response function, thus enabling a correct representation of the so-called neural computability condition, which we assumed as the basis of local computations performed by single neurons.

In this regime, currently existing theories are not able to correctly predict the response function of LIF neurons. Based on a new method of membrane potential autocorrelation propagation, we were able to develop an analytical prediction of the LIF response function that not only covers the parameter ranges where other theories already perform well, but also the high-conductance state that is of particular interest for LIF sampling. Owing to this theory, we were able to formulate direct translation rules from the Boltzmann domain to the LIF domain of neuron and synapse parameters. We have demonstrated that such LIF networks exhibit good performance when sampling from arbitrary target (Boltzmann) distributions even for large network sizes and non-sparse connection matrices. Furthermore, we have shown that under the right circumstances, i.e., a pronounced high-conductance state, these networks are robust to parameter variations that are likely to appear in physical substrates, be they biological or neuromorphic.

Based on this framework of LIF sampling, we have crossed the boundary between brain science and biologically unconstrained machine learning to build LIF networks targeted at solving specific problems. In particular, we have discussed how relatively simple, untrained two-layer networks can perform well at denoising small black-and-white images. More importantly, we have shown how state-of-the-art generative and discriminative models (deep Boltzmann machines) for classical machine learning datasets (MNIST) can be translated to LIF networks with only minimal loss in classification performance. We have also discussed a formal analogy to solid-state systems, i.e., microscopic magnets. Since the high-level governing equations of our LIF networks are equivalent to those of the Ising model, we were able to observe activity patterns that are analogous to well-known magnetic phenomena such as the Curie law, hysteresis, Weiss domains and phase transitions.

Following this extended discussion of sampling from Boltzmann distributions, we have taken the natural step of extending our formalism to encompass arbitrary distributions over binary random variables. Building on ideas from Pecevski et al. (2011), we have shown how this can also be robustly achieved in LIF networks. Furthermore, we have discussed a more sophisticated interaction mechanism based on interneuron chains that drastically improves the sampling quality. With the help of an extended version of this chain-based interaction, we were finally able to overcome the sampling-disruptive interplay between transmission delays and unstable refractoriness on the Spikey chip in order to achieve, to the best of our knowledge, the first implementation of neural sampling on analog accelerated neuromorphic hardware.

Noise and Stochasticity

Our implementation of LIF sampling assumes that each neuron receives diffuse background spike noise that we have modeled as Poissonian, thus allowing our analytical derivation of the LIF activation function in the high-conductance state. For a scalable hardware implementation, providing such noise is crucial. As already discussed in Section 4.5, the requirement of having a large number of uncorrelated noise sources is very demanding for hardware devices, regardless of whether the noise sources are internal or external.

One idea that we are actively pursuing is the implementation of noise-producing spiking neural networks – so-called "sea of noise" networks – that serve as noise sources for other functional networks that are running on the same device. This solves the problem of having numerous noise sources but does not obviously address the issue of shared-input correlations. However, as was shown in recent literature (Tetzlaff et al., 2012), it turns out that, if properly tuned, the negative correlations induced by inhibition in such networks can precisely cancel out the positive correlations due to shared input pools. Figure 6.62 shows an exemplary functional network (an LIF sampling network trained to recognize handwritten digits) that suffers from a decreasing pool of noise sources and how the sea-of-noise implementation is able to maintain its functionality while a simple pool of Poisson

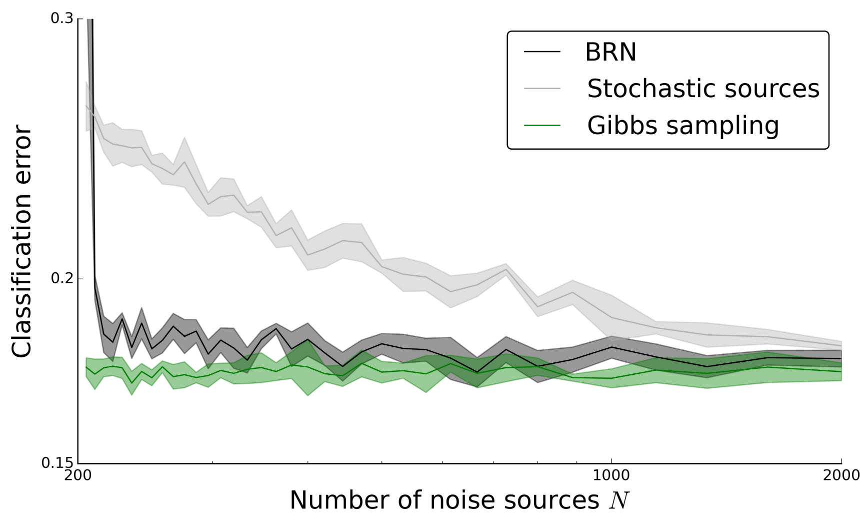


Figure 6.62.: Effect of a shrinking pool of noise sources on the performance of an LIF-based Boltzmann machine trained on the MNIST handwritten digit dataset. The green line represents the performance achieved by a classical Boltzmann machine with Gibbs sampling and serves as a reference. If a pool of Poisson sources is used, decreasing its size results in increasing shared-input correlations in the sampling network and leads to a deterioration of the classification performance. On the other hand, a "sea of noise" implemented as a balanced random network (BRN) actively decorrelates its output through its internal inhibition, canceling out shared-input correlations in the functional LIF network that it provides with noise. It is thereby able to sustain the performance of the LIF network even after being reduced in size by nearly an order of magnitude.

sources can not. While we do not address these experiments in detail here, we point to Jordan et al. (2014) for a more involved discussion. Active decorrelation in small versions of such inhibition-dominated networks has also been demonstrated in emulations on the Spikey chip (Pfeil et al., 2014).

The requirement of Poissonian noise is necessary for the theoretical derivations, but does not have to be strictly enforced in practice. This is particularly important for neuromorphic hardware, where internal noise sources are not truly random, but usually a digital implementation of a pseudo-random number generator (e.g., linear feedback shift registers of the wafer-scale hardware). However, if the statistics of the underlying noise process stray too far away from the Poisson assumption, the network performance will also suffer. The effects of non-Poissonian noise are a topic of active investigation.

An entirely different method of obtaining near-Poissonian noise might also be found by constructing random projections from neurons in other functional networks that run in parallel on the same device. Indeed, this is probably what happens in the brain, where it is rather unlikely to find a region dedicated to producing noise alone. This idea of interconnecting functional networks such that they supply each other with the necessary noise would do away with the requirement of specialized noise-generating components

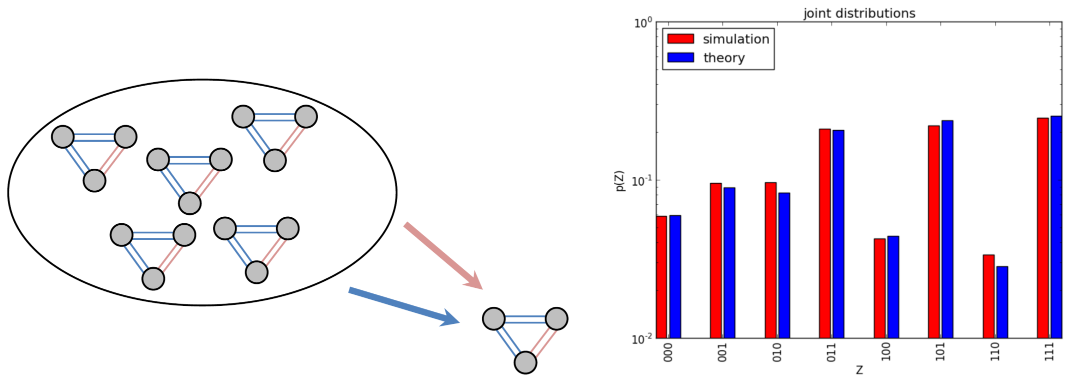


Figure 6.63.: An LIF network sampling from a randomly drawn Boltzmann distribution over 3 binary random variables that receives its noise input from several other LIF-based Boltzmann machines. **Left:** Sketch of the network architecture. **Right:** Sampled joint distribution after 10^4 ms (red) compared to the target distribution (blue).

altogether. First results obtained in this direction (Figure 6.63) look encouraging. For a further discussion of this topic, we also refer to Korcsak-Gorzo (2015).

Machine Learning Applications

The application of LIF sampling to machine learning problems becomes particularly intriguing when considering the advantages offered by neuromorphic hardware. Due to its typically low power consumption per synaptic interaction, neuromorphic devices represent ideal control units for autonomous agents (robots). With additional computational power coming from accelerated dynamics, as is the case for the hardware devices discussed in Chapter 3, the sampling approach can quickly produce useful predictions of a changing environment: while neural sampling explicitly deals only with stationary inputs, the external world would appear quasi-stationary to a sufficiently accelerated network. Such ideas for accelerated pattern completion have, for example, been studied in Roth (2014). Furthermore, neuromorphic implementations would also benefit from the observed superior mixing features of LIF networks as compared to classical sampling methods.

Regarding pattern completion, it is interesting to note not only a superficial, but also a functional similarity to attractor models such as the L2/3 network from Section 5.3. Essentially, both the L2/3 network and the sampling networks trained on MNIST data learn to recollect memories based on incomplete or noisy input. While here we have studied only orthogonal (i.e., non-overlapping) patterns, the L2/3 model is able to store a much larger quantity of memories when patterns are allowed to overlap (see, e.g., Rivkin, 2014, for a study of the memory capacity of the L2/3 network). A particularly relevant feature of the L2/3 network is the fact that it respects Dale's law, which LIF sampling explicitly violates. An interpretation of the activity of such cortically inspired attractor networks in terms of neural sampling is certainly an interesting venue for future theoretical research.

The training of such functional networks is one of the most important current problems. Here, we have used off-line training methods from machine learning, but on-line training using embedded plasticity mechanisms would offer obvious advantages in terms of convergence speed. Building on our LIF sampling approach, ideas of using STDP to implement contrastive divergence learning have been put forward (Neftci et al., 2015). We refer to Weilbach (2015) for a more detailed discussion of STDP-based contrastive divergence in LIF networks. Furthermore, we are currently working on implementing on-line tempering into our LIF-based deep Boltzmann machines. Together with the STDP-based contrastive divergence, this will give us an on-line version of the powerful CAST algorithm that we have used off-line in Section 6.6.2.2.

It will be also interesting to draw connections to other STDP-based unsupervised learning models, such as the ones from Habenschuss et al. (2012); Nessler et al. (2009, 2013), where spikes are interpreted as samples in precisely the same way as in the neural sampling framework. While these, too, are still rather abstract models, we point to Breitwieser (2015) for a comprehensive discussion of a hardware-compatible implementation. Furthermore, based on similar principles, unsupervised learning of temporal sequences has also recently been proposed Kappel et al. (2014). Once these methods are embedded in hardware-compatible networks, we will be able to take full advantage of the speedup offered by our accelerated devices.

Microscopic Magnetic Systems

The interesting analogies that we have observed between LIF networks and microscopic magnetic systems are not only intuitively intriguing, but also of practical concern. They open up the possibility of emulating semiclassical systems on accelerated neuromorphic hardware. The question whether true quantum systems can be embedded in the activity of LIF networks is a topic of active research. More immediately important is the fact that these analogies open up spiking neural networks to powerful methods from statistical physics. With these, it could become possible to predict interesting phenomena such as state transitions and the propagation of fluctuations.

Future Hardware Developments

We have discussed how the inevitable spike propagation delays can be countered with prolonged refractory periods, either implemented directly into the neural circuits, as will become available in future hardware revisions (HICANNv4), or by means of auxiliary units, as we have used on the Spikey chip. However, in bipartite structures such as restricted Boltzmann machines, such delays might play a less disruptive role, as the state of the visible layer is fixed during memory recall (i.e., after training, when the network is used for classification) and the hidden layer has no horizontal connections. It might therefore become possible to return to a single-neuron representation of a binary RV already on the Spikey chip and use a network structure that is previously trained off-line for classification. Only the hidden layer would then be implemented on the chip itself and the inputs from the visible layer would be fed in from the host computer. We are actively investigating this idea at the current time.

In any case, the fact that we were able to demonstrate accelerated neural sampling on the Spikey chip is a clear first step in the direction of large spiking Boltzmann machines on the wafer-scale hardware. We need to point out that it is not the correct retrieval of a Boltzmann distribution over 3-4 RVs that is particularly impressive. Indeed, this computation takes only milliseconds on standard desktop machines. The essential achievement is much rather the fact that we have shown accelerated neuromorphic LIF sampling to be realizable in principle, despite parameter variability and other hardware-related sources of distortion. Even more importantly, the achievable network size scales directly with the size of the hardware substrate, while the emulation time does not (in contrast to software implementations), as all hardware components operate asynchronously in parallel. If the design specifications of the HICANNv4 chip are met in practice, a HICANNv4 wafer will immediately allow the emulation of large, deep learning architectures that are competitive with state-of-the-art machine learning models.

7. Epilogue

Like organisms evolved in gentle tide pools, who migrate to freezing oceans or steaming jungles by developing metabolisms, mechanisms, and behaviors workable in those harsher and vaster environments, our descendants, able to change their representations at will, may develop means to venture far from the comfortable realms we consider reality into arbitrarily strange worlds. Their techniques will be as meaningless to us as bicycles are to fish, but perhaps we can stretch our common-sense-hobbled imaginations enough to peer a short distance into this odd territory.

Hans Moravec, Simulation, Consciousness, Existence, 1998

Early on in our work, we have established the fundamental form of the components that we were going to work with. This form was given as the differential equations that govern networks of pointlike LIF neurons with exponential synaptic kernels. This particular level of mathematical abstraction was chosen both for analytical tractability and for compatibility with the studied neuromorphic systems – which further restrict this otherwise well-defined form by burdening it with various distortions.

The function associated with this form was given by the computational properties of the studied network models. We have studied a rich repertoire of function in various network models, all of them based on the same mathematical and/or physical substrate given by the chosen form of their microscopic components. The computational properties of these networks are not necessarily related, but by sharing the same substrate, they take a small step in paving the way for true integration of function, as is the case in biological cognitive architectures. Essentially, the brain is nothing else than a network of networks, each consisting of roughly the same components but endowed with its own functionality, and it is out of this complex interaction that the still deeply mysterious phenomenon of consciousness arises.

Starting from the abstract mathematical description, we can choose a particular physical implementation of our microscopic quanta of thought. Quintessentially, we argue that the precise nature of the substrate does not matter, as long as, at some level of abstraction, it obeys the previously defined equations of motion, the chosen mathematical form. Currently, classical computing architectures dominate the landscape of computational neuroscience due to their versatility, but it is questionable whether this approach can scale to brain-size networks. However, even with their current limitations, accelerated neuromorphic devices have the potential to become indispensable scientific tools, as they allow us to verify and tune our models in much faster cycles than conventional computers do, whether we require long training periods for large networks or multiple instantiations of such networks for parameter sweeps or batch learning.

It remains an open question how similar to their biological archetypes our hardware substrates and network models must be. Computer science has a long and partly successful history of taking inspiration from biological neural networks, and for a long time the leading argument for taking inspiration from biology was the clear superiority of humans in virtually all machine learning tasks of sufficient complexity. In recent years, this has begun to change and an increasing number of machines are starting to break human records. These machines achieve their performance through a combination of engineered structure and brute-force algorithms and are, if at all, only remotely similar to what we currently know about the human cortex. The tasks at which machines achieve superhuman performance remain relatively simple, such as playing chess or recognizing traffic signs, but there is no reason to believe that this trend must stop here. While their performance is still very remote from true artificial intelligence, it does call into question the strict adherence to biological models.

The unreasonable ineffectiveness of mathematics in biology, as Gelfand has called it, may not come as a complete surprise when one considers the complexity of networks that exhibit complex functionality, be they biological or not. Theirs is a fundamentally different complexity from the one found in most ensembles studied with such remarkable success in mathematical physics. It is not simply the number of components – the number of molecules in a macroscopic volume of gas surpasses the number of neurons in the brain by many orders of magnitude – and neither is it the complexity of the individual units, as the success of simple neuron models in computational neuroscience and machine learning clearly shows.[1] It is much rather the fact that each of these components (or small ensembles thereof) plays a unique and important computational role that cannot be fully covered by macroscopic equations such as the law of ideal gases. Thus, while mathematics does provide us with powerful tools to understand various aspects of neural network dynamics, in the end, there is no way around the simulation or emulation of complex cognitive architectures.

We end this work by taking a bold look into the future. As all predictions of things that are to come, argued from the limited point of view of contemporary technology, they might soon be invalidated by our scientific impetus. However, from our current perspective, this argument provides a deep motivation for the development of artificial neural networks in general and neuromorphic implementations in particular.

One reason why it is so hard to correlate microscopic dynamics with macroscopic observables on the level of psychology and behavior is that we are severely restricted in the experiments that we can perform, both by ethical considerations and by experimental techniques. The most immediate way of gaining insight into neuroscientific questions would be gained by having full control of all the components inside the brain of human subjects and by studying their response to complex tasks while making infinitesimal changes to their brain structure. This would represent the direct translation to neurobiology of the overwhelmingly successful approach of experimental physics to understanding inanimate nature. Notwithstanding the fact that this is far from technically feasible with today's methods, this would infringe on the most elementary of ethical principles. With engineered

[1]This statement would not change significantly if, for example, we added several further equations per synapse that describe neuromodulation.

neural networks, the technical problems disappear, as they offer direct and non-invasive access to all the needed microscopic data. Also, within ethical boundaries that are yet to be established, they allow the possibility of manipulating network components with a virtually arbitrary degree of precision. Once an AI with sufficient – but compared to human adults, probably only rudimentary – cognitive capabilities can be created, it will, unlike us, have direct access to the means of not only *true* introspection on any level of detail it chooses, but also the possibility of taking direct, unmediated influence on its own neural structure. This shall become the moment where intelligent life, unshackled from the temporal and physical constraints of flesh and bone, can start writing a new – and, to us, unfathomable – chapter of the book of evolution.

A. Appendix

A.1. Acronyms and Abbreviations

ACP - AutoCorrelation Propagation
ADC - Analog-to-Digital Converter
AdEx - Adaptive Exponential (leaky integrate and fire neuron model)
AI - Asynchronous Irregular (firing)
AP - Action Potential
API - Application Programming Interface
ASIC - Application Specific Integrated Circuits
AST - Adaptive Simulated Tempering
BCM - Bienenstock-Cooper-Munro (synaptic plasticity rule)
BM - Boltzmann Machine
BN - Bayesian Network
CAST - Coupled Adaptive Simulated Tempering
CD - Contrastive Divergence
CDF - Cumulative Distribution Function
CI - Conditional Independence
CLT - Central Limit Theorem
CNS - Central Nervous System
COBA - COnductance-BAsed (synapse and/or neuron model)
CSL - Conservative Sampling-based Likelihood (estimator)
CV - Coefficient of Variation
CUBA - CUrrent-BAsed (synapse and/or neuron model)
DAC - Digital-to-Analog Converter
DBM - Deep Boltzmann Machine
DenMem - Dendritic Membrane (neuron circuit on the HICANN chip)
DNC - Digital Network Chips (BrainScaleS wafer-scale system component)
DOE - Difference Of Exponentials
EPSP - Excitatory PSP
ESS - Executable System Specification (software simulator of the waferscale neuromorphic hardware)
FACETS - Fast Analog Computing with Emergent Transient States (EU research project)
FFG - Forney Factor Graph
FG - Factor Graph
FPGA - Field Programmable Gate Array
FPT - First Passage Time
FS - Fast-Spiking (neurons)
GHK - Goldman-Hodgkin-Katz (equation)
HC - (cortical) HyperColumn
HH - Hodgkin-Huxley (neuron model)

HICANN - High Input Count Analog Neural Network (neuromorphic chip)
IID - Independent Identically Distributed (random variables)
IPSP - Inhibitory PSP
ISI - Inter-Spike Interval
KL - Kullback-Leibler (divergence, measure of similarity between probability distributions)
KTH - Kungliga Tekniska Högskolan (Royal Institute of Technology, Stockholm, Sweden)
LFG - Liquid Factor Graph
LIF - Leaky Integrate-and-Fire (neuron model)
LSM - Liquid State Machine
Lx - cortical Layer x
MC - (cortical) MiniColumn
ML - Maximum Likelihood (learning)
MNIST - Mixed National Institute of Standards and Technology (handwritten digit dataset)
MRF - Markov Random Field
NCC - Neural Computability Condition
ODE - Ordinary Differential Equation
OU - Ornstein-Uhlenbeck (stochastic process)
PCS - Pulse Communication Subgroup (component of the BrainScaleS wafer-scale system)
PDE - Partial Differential Equation
PDF - Probability Density Function
PSC - PostSynaptic Current / PostSynaptic Conductance (triggered by a single input spike)
PSP - PostSynaptic Potential (triggered by a single input spike)
RAM - Random Access Memory
RHS - Right Hand Side (of an equation)
RS - Regular-Spiking (neurons)
RV - Random Variable
SFA - Spike Frequency Adaptation
SPA - Sum-Product Algorithm
TUG - Technische Universität Graz (Graz University of Technology, Austria)
STDP - Spike Timing Dependent Plasticity
STD - Short-Term (synaptic) Depression
STP - Short-Term (synaptic) Potentiation
TSO - TSOdyks-Markram (STP model)
VLSI - Very-Large-Scale Integration

A.2. Supplementary Information

A.2.1. Hodgkin-Huxley Model

x	E_x	g_x
Na^+	115 mV	120 mS/cm^2
K^+	-12 mV	36 mS/cm^2
l	10.6 mV	0.3 mS/cm^2

Table A.1.: Parameters for the HH neuron simulations in Section 2.1.2: reversal potentials and conductances.

x	α_x(u[mV])	β_x(u[mV])
n	$(.1 - .01u)/(\exp(1. - .1u) - 1.)$	$0.125\exp(-u/80.)$
m	$(2.5 - .1u)/(\exp(2.5 - 0.1u) - 1.)$	$4.\exp(-u/18.)$
h	$.07\exp(-u/20.)$	$1./(\exp(3. - .1u) + 1.)$

Table A.2.: Parameters for the HH neuron simulations in Section 2.1.2: gating variables.

The values, as originally reported by Hodgkin and Huxley (1952), are based on a voltage scale where the resting potential is zero. To comply with electrophysiological measurements, the voltage scale had to be subsequently shifted down by 65 mV.

A.2.2. Wafer-Scale Neuromorphic Hardware

A.2.2.1. Short-Term Plasticity

Unlike the theoretical TSO model (Section 2.2.2.2), which allows the occurrence of both depression and facilitation at the same time, the hardware implementation does not allow their simultaneous activation. The ongoing pre-synaptic activity is tracked with a time-varying active partition I with $0 \leq I \leq 1$, which decays exponentially to zero with time constant τ_{stdf}. Following a pre-synaptic spike, I is increased by a fixed fraction $U_{\text{SE}}(1-I)$, resulting in the following dynamics for the active partition:

$$I_{n+1} = [I_n + U_{\text{SE}}(1 - I_n)] \exp\left(-\frac{\Delta t}{\tau_{\text{stdf}}}\right) \quad , \tag{A.1}$$

with Δt being the time interval between the nth and $(n+1)$st afferent spike.
This active partition can be used to model depressing or facilitating synapses as follows:

$$w_{\text{stdf}}^{\text{depression}} = 1 - \lambda \cdot I \tag{A.2}$$

$$w_{\text{stdf}}^{\text{facilitation}} = 1 + \lambda \cdot (I - \beta) \quad . \tag{A.3}$$

Here, w_{stdf}^{x} corresponds to a multiplicative factor to the static synaptic weight, with λ and β being configurable variables, and x denotes the mode being either depression or facilitation. The n-th effective synaptic weight is then given by

$$w_n^{\text{syn}} = w_{\text{static}} w_{\text{stdf}}^{x} \quad . \tag{A.4}$$

Due to a technical limitation, the change of synaptic weights by short-term plasticity can not be larger than the static weight, such that $0 \leq w_{\text{stdf}}^{x} \leq 2$. We refer to Schemmel et al. (2008) for details of the hardware implementation of the TSO model and to Bill et al. (2010) for neural network experiments on neuromorphic hardware using this plasticity model.

The original TSO model can be translated to the hardware model when one of the two time constants (τ_{rec} or τ_{facil}) is equal to zero. For depression only ($\tau_{\text{facil}} = 0$), the nth synaptic weight is given by

$$w_n^{\text{syn}} = w_{\max}^{\text{syn}} R_n U \quad . \tag{A.5}$$

The time course of R can be exactly represented by $(1 - I)$ if the scaling factor λ of the short-term plasticity mechanism is set to 1. Additionally, the static synaptic weight w_{static} has to be adapted such that the applied synaptic weights are equal, giving us the following transformation: $\tau_{\text{stdf}} = \tau_{\text{rec}}$, $U_{\text{SE}} = U$, $\lambda = 1$ and $w_{\text{static}} = w_{\max}^{\text{syn}} U$.
For facilitation only ($\tau_{\text{rec}} = 0$), the recovered partition remains fully available all the time ($R = 1 = \text{const}$) and only the utilization varies with time. Thus the nth synaptic weight is given by:

$$w_n^{\text{syn}} = w_{\max}^{\text{syn}} U_n \quad . \tag{A.6}$$

The time course of U now has to be emulated by the RHS of Equation A.3. Additionally we set $U_{\text{SE}} = U$ and $\tau_{\text{stdf}} = \tau_{\text{facil}}$, and level the limits for the synaptic weights. In the original model, U is always between U and 1, while for the hardware model the STP factor is limited to values between 0 and 2 due to technical reasons. By setting $\lambda = 1$ and considering that I is always within 0 and 1, the supplied range for $w_{\mathrm{STP}}^{\mathrm{facilitation}}$ is $[1-\beta, 2-\beta]$. In order to match the range of applied weights of both models, we need to solve the following system of equations:

$$(1-\beta) \cdot w_{\mathrm{static}} = U \cdot w_{\mathrm{max}}^{\mathrm{syn}} \tag{A.7}$$

$$(2-\beta) \cdot w_{\mathrm{static}} = 1 \cdot w_{\mathrm{max}}^{\mathrm{syn}} \quad . \tag{A.8}$$

Solving for w_{static} and β yields

$$w_{\mathrm{static}} = (1-U) \cdot w_{\mathrm{max}}^{\mathrm{syn}} \tag{A.9}$$

$$\beta = \frac{1-2U}{1-U} \quad . \tag{A.10}$$

A.2.2.2. Parameter Variation Measurements

Figure A.1 shows trial-to-trial parameter variation measurements on HICANN chips. The measurements are conducted on a single-chip prototype system (A-D) and on one chip on a prototype wafer system (E-F). Some neurons (on the right-hand-side of the plots) had been previously labeled non-functional and blacklisted, therefore showing no data points. Such neurons are also omitted during system operation. Additionally, neurons that exhibit a larger variation than a chosen threshold can be blacklisted as well, reducing the total number of available neurons, but also limiting the magnitude of parameter noise. This effect is not explicitly included in the ESS simulations in the main text, but it is conceptually covered by some of the experiments, where the network is restricted to only a small fraction of the wafer (e.g., in Section 5.3.8), or where additionally parts of the synapses are declared as not available (e.g., in Section 5.4.8).
From the measurements in Table A.1, we can estimate the variation of the voltages V_{spike}, V_{rest}, $E_{\mathrm{e}}^{\mathrm{rev}}$ and $E_{\mathrm{i}}^{\mathrm{rev}}$ in the biological domain. For all, the vast majority of neurons has a trial-to-trial variation below 10 mV on the hardware, which corresponds to 1 mV in the biological when using a voltage scaling factor $\alpha_V = 10$.

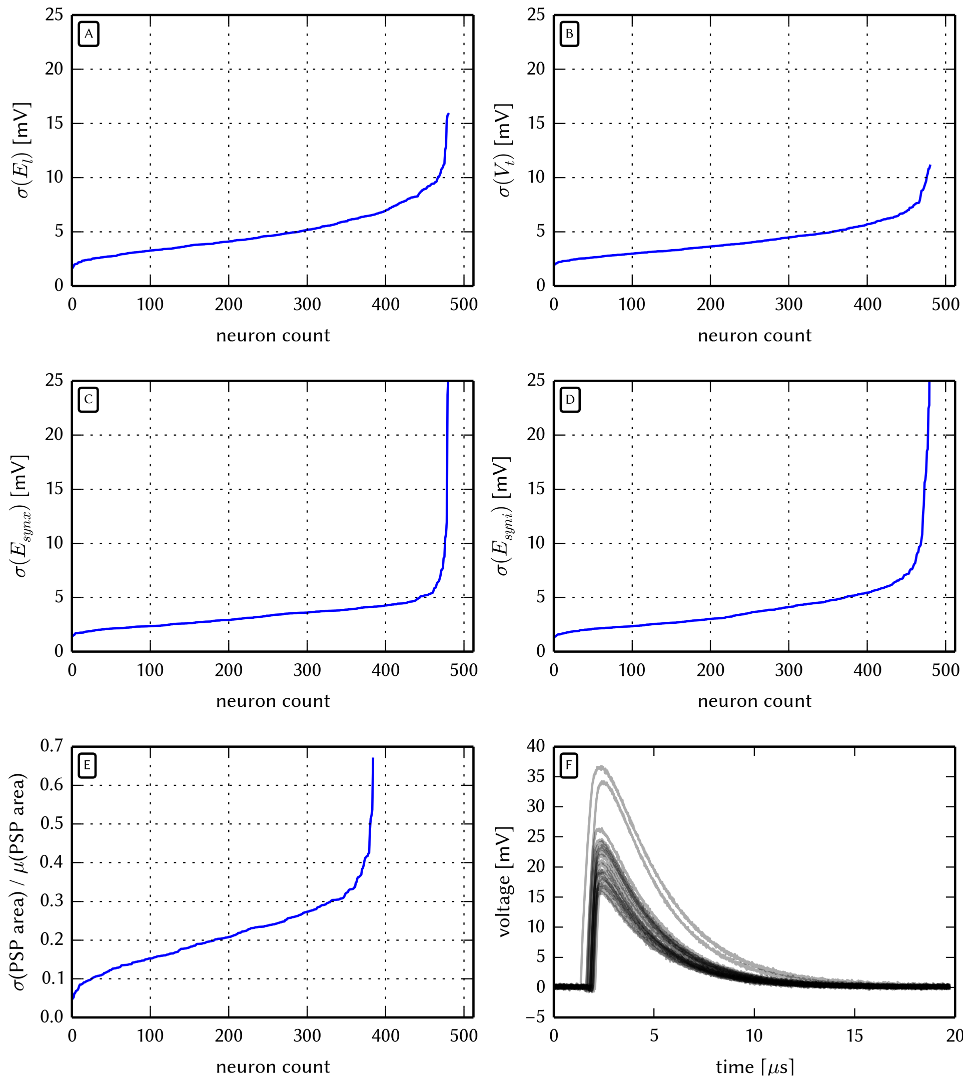


Figure A.1.: Trial-to-trial parameter variations on the HICANN chips. **(A) - (D)** Cumulative distributions for selected parameters. Each graph shows the number of neurons on one chip with a standard deviation of the measured value that is less than the value shown on the ordinate. All values are given in hardware units. In order to obtain values in the biological domain (see Section 3.3.3), the voltages must be divided by the conversion factor of $\alpha_V = 10$. The standard deviation was estimated from 30 measurements for each neuron. **(A)** Leakage potential. **(B)** Threshold potential. **(C)**, **(D)** Excitatory and inhibitory reversal potential. **(E)** Relative variation of the PSP integral. The standard deviation was estimated from 20 trials per neuron. Neurons were omitted from the measurements when an initial sweep over the available parameter range did not include the required PSP integral of 8×10^{-9} V s. **(F)** Example PSP traces for a randomly chosen neuron from the measurement in (E). In order to minimize readout noise, each trace is an average over 400 individual PSPs which were evoked in short succession without rewriting floating gate parameters. As the re-write variation is the main source of trial-to-trial variability (see Section 3.3.1), the variation within the 400 samples is much smaller than the trial-to-trial variation that is shown in (E) and (F).

A.2.3. Cortical Layer 2/3 Attractor Memory

A.2.3.1. Original Model Parameters

In Tables A.3, A.4, A.5 and A.6, we summarize the parameters and characteristics of the original model, as found in Lundqvist et al. (2006). These have served as the basis for the model fit, for which the parameters can be found in the next subsection.

Parameter	PYR	RSNP	BAS	Unit
g_{ext}	0.082	0.15	0.15	μS/mm^2
E_{l}	-75	0.15	-75	mV
E_{Na^+}	50	50	50	mV
$E_{\text{Ca}^{++}}$	150	150	150	mV
E_{K^+}	-80	-80	-80	mV
$E_{\text{Ca}^{++}}^{\text{NMDA}}$	20	20	20	mV
g_{l}	0.74	0.44	0.44	μF/mm^2
C_{m}	0.01	0.01	0.01	μF/mm^2
Soma diameter $\pm$ stdev	21 $\pm$ 2.1	7 $\pm$ 0.7	7 $\pm$ 0.7	μm
g_{Na^+} initial segment	2500	2500	2500	μS/mm^2
g_{K^+} initial segment	83	5010	5010	μS/mm^2
g_{Na^+} soma	150	150	150	μS/mm^2
g_{K^+} soma	250	1000	1000	μS/mm^2
g_{NMDA}	75.0	75.0	-	μS/mm^2
$\text{Ca}^{++}{}_V$ influx rate	1.00	1.00	1.00	mV^{-1}ms^{-1}mm^{-2}
$\text{Ca}^{++}{}_{\text{NMDA}}$ influx rate	2.96	0.0106	-	s^{-1}mV$^{-1}\mu$S^{-1}
$\text{Ca}^{++}{}_V$ decay rate	6.3	4	-	s^{-1}
$\text{Ca}^{++}{}_{\text{NMDA}}$ decay rate	1	1	-	s^{-1}
g_{K^+} ($\text{Ca}^{++}{}_V$)	29.4	105	0.368	nS
g_{K^+} ($\text{Ca}^{++}{}_{\text{NMDA}}$)	40	40	-	nS
# compartments	6	3	3	
Dendritic area (relative soma)	4	4	4	
Initial segment area (relative soma)	0.1	0.1	0.1	

Table A.3.: Original neuron parameters of the L2/3 model.

Pre $\rightarrow$ Post	Type	Duration [s]	τ_{rise} [s]	τ_{decay} [s]	E^{rev} [mV]	tso$_U$	τ_{rec} [s]	E_{slow} [mV]
PYR $\rightarrow$ PYR	Kainate/AMPA	0.0	0.0	0.006	0	0.25	0.575	-
PYR $\rightarrow$ PYR	NMDA	0.02	0.005	0.150	0	0.25	0.575	0.020
PYR $\rightarrow$ BAS	Kainate/AMPA	0.0	0.0	0.006	0	-	-	-
PYR $\rightarrow$ RSNP	Kainate/AMPA	0.0	0.0	0.006	0	-	-	-
PYR $\rightarrow$ RSNP	NMDA	0.02	0.005	0.150	0	-	-	0.020
BAS $\rightarrow$ PYR	GABA	0.0	0.0	0.006	-85	-	-	-
RSNP $\rightarrow$ PYR	GABA	0.0	0.0	0.006	-85	-	-	-

Table A.4.: Original synapse parameters of the L2/3 model.

	HCs	MCs	PYR	BAS	RSNP	total neurons
per MC	-	-	30	1	2	33
per HC	-	8	240	8	16	264
network total	9	72	2160	72	144	2376

Table A.5.: Original network structure of the L2/3 model: number of neurons per functional unit.

within an MC	
PYR → PYR	0.25
RSNP → PYR	0.70
between MCs inside the same HC	
PYR → BAS	0.70
BAS → PYR	0.70
between MCs in different HCs	
PYR → PYR	0.30
PYR → RSNP	0.17

Table A.6.: Original network structure of the L2/3 model: connection probabilities.

A.2.3.2. UP-State Detection

One crucial element of the analysis is the detection of UP-states from which various other properties such as dwell times, competition times or average spike-rates in UP- and DOWN-states are determined. The method of choice for detecting UP-states is based on the fact that the mean spike rate of an attractor during an UP-state is much higher than the spike rate in all remaining patterns in their corresponding DOWN-states, whereas in times of competition two or more attractors have elevated but rather similar spike rates. A measure which quantifies this relationship is the standard deviation σ of all mean spike rates per attractor at a given time t. The attractor with index i is then said to be in an UP-state at time t if the following relation holds:

$$r_i(t) > \sigma(t) > \max_{r \in \{1,\ldots,N_{\mathrm{MC}}\} \setminus i} r_k(t) \quad , \tag{A.11}$$

where $r_i(t)$ is the rate of attractor i at time t.
This method of detection has several advantages: it is based exclusively on spike trains (and not voltages or conductances, which are more difficult to read out and require much more storage space), it has a clear notion of there being at most one UP-state at any given time and it is completely local (in time), meaning that a very large value somewhere on the time axis cannot bias the detection at other times.
In small networks with randomly spiking neurons, it might happen by chance that all but one of the spike rates lie below the (approximately) constant standard deviation. These falsely detected UP-states are very short and can thus easily be filtered out by requiring

a minimal duration for UP states, which we chose at 100 ms. This value was chosen after investigating dwell time histograms, as it distinguishes reliably between random fluctuations and actual active attractors.

A.2.3.3. Pattern Completion

Pattern completion is a basic property of associative-memory networks. By only stimulating a subset of PYR cells within a pattern, the complete pattern is recalled. The activity first spreads within the stimulated MCs, turning them dominant in their corresponding HCs. After that, the activity spreads further to other HCs – while the already dominating MCs stabilize each other through mutual stimulation – activating the whole pattern while suppressing all others. All PYR cells in the corresponding attractor hence enter an UP-state.
To verify the pattern completion ability of the network, a series of simulations was performed. In order to reduce the occurrence of spontaneously activating attractors – which would interfere with the activation of the stimulated attractor – competition was investigated in larger networks of size 25HC×25MC, as they exhibit almost no spontaneous attractors (the competition time fractions are much higher, see Figure 5.3H).
For each network, all of the 25 patterns were stimulated one by one in random order. The time between consecutive stimuli was chosen to be 1000 ms to ensure minimal influence between patterns. The number of stimulated MCs (one per HC) was varied over the course of multiple simulations.
After simulation, each network was analyzed for successfully activated patterns. An activation attempt was said to be successful if the stimulated pattern was measured as active within 200 ms after the stimulus onset. If another pattern was active up to 75 ms or if the stimulated pattern had already been active between $20 - 500$ ms prior to the stimulus onset, the attempt was deemed invalid and ignored during the calculation of success ratios. This was done to take into account the fact that a pattern is more difficult to activate when another one is already active or while it is still recovering from a prior activation. From all valid attempts the success probability (assuming a binomial distribution of successful trials) was estimated using the Wilson interval

$$\tilde{p} = \frac{1}{1 + \frac{z^2}{n}} \left[\hat{p} + \frac{z^2}{2n} \pm z \sqrt{\frac{\hat{p}(1-\hat{p})}{n} + \frac{z^2}{4n^2}} \right] \tag{A.12}$$

where $\hat{p}$ represents the success ratio, n the number of valid attempts and $z = 1$ the desired quantile.
For most experiments (regular, synaptic weight noise and homogeneous synaptic loss) the number of invalid activations was always below 5 (out of 25). The only exception was the PYR population size scaling: starting at 15 PYR cells, the validity rate roughly halves for every reduction in size (by 5 PYR cells per step) due to the increased occurrence of spontaneous attractors. For simulations carried out on the ESS, only 10 patterns out of 25 were stimulated. Out of these 10 attempts, only 5 were valid, on average.

A.2.3.4. Pattern Rivalry / Attentional Blink

Another important feature of the L2/3 model is its ability to reproduce the attentional blink phenomenon, i.e., the inability of one pattern, stimulated by layer 4 input, to

terminate another already active pattern and become active itself. This phenomenon was investigated through a series of different networks of same size as in Section A.2.3.3 (25HC×25MC). For each network, 24 out of 25 patterns were randomly assigned to 12 non-overlapping pairs. Afterwards, pattern rivalry was tested on all of these pairs in intervals of 1000 ms.
Let the two patterns in each pair be denoted A and B. In order to guarantee an immediate activation of pattern A, 6 out of 25 HCs were stimulated (as then all completion attempts are successful, see Figure 5.3N). Then, after a certain delay ΔT, pattern B was stimulated with a varying amount of HCs. Both the number of stimulated HCs as well as the delay ΔT were varied for each network.
The same way as in Section A.2.3.3, each network was then analyzed as to whether pattern B was successfully activated or not. If the competing pattern B was activated within 200 ms after the stimulus onset and stayed active for at least 100 ms, the attempt was counted as successful, otherwise it was deemed unsuccessful. As before, attempts during which spontaneously activated patterns intervened were ignored. From all successful and unsuccessful attempts, the success probability was then estimated the same way as in pattern completion, using Equation A.12.
The validity ratios for pattern rivalry are not significantly different from those discussed in Section A.2.3.3. Most experiments (regular, synaptic weight noise and homogeneous synaptic loss) have 10 to 12 valid attempts (out of 12). As before, for the PYR population size scaling experiments, the number of valid attempts dropped progressively (8.2 ± 1.7, 4.8 ± 2.1 and 2.2 ± 1.5 valid attempts for 15, 10 and 5 PYR per MCs respectively). Simulations carried out on the ESS had an average of 4 (distorted case) and 6 (compensated case) valid attempts (out of 10).
Different network configurations have been compared in terms of attentional blink by estimating the 0.5 iso-probability contour in the following way. For every delay Δ_T, the transition point from below to above 0.5 probability for successful activation of the second pattern was estimated by linearly interpolating between the two nearest data points with a success ratio of above and below 0.5, respectively. In case there were several such transition points only the one with the highest stimulus was considered. If no transition point could be identified, the transition was fixed at at either 25 or 0 stimulated MCs, depending on whether all success ratios were above or below 0.5. When there were no valid attempts for a certain delay/stimulus pair, its success probability estimate was replaced by the median of all valid activation attempts for that particular time delay ΔT (this only occurred sporadically in ESS and PYR population size scaling with less than 15 PYR cells per MC). After identifying the transition point for every time delay Δ_T, intermediate values were interpolated linearly. Finally, the interpolated values were Gauss-filtered ($\mu = 0.25 \times$ step size for Δ_T in the dataset) to better approximate the true 0.5 iso-probability contour.

A.2.3.5. Star Plots

While the spiking activity of many cells can be visualized quite well in raster plots, illustrating the temporal evolution of their membrane potentials is less straightforward. Here, we have chosen to use so-called star plots for visualizing both average voltages and average firing rates of entire cell populations.
In a system evolving in an abstract space with 3 dimensions, a star plot represents the

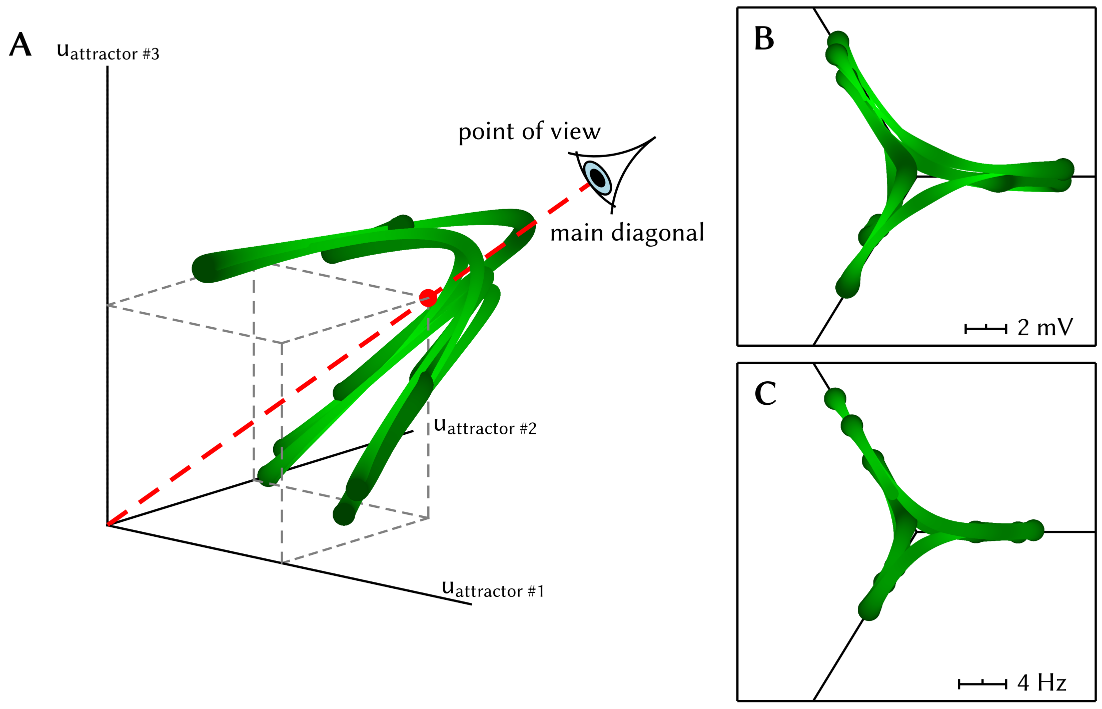


Figure A.2.: **Visualization of the star plot as a projection in the case of a three-dimensional state space.** (**A**) Illustration of the view point with average membrane voltage data plotted in three-dimensional Cartesian coordinates. The data was taken from a (9HC×3MC)-network and covers a 2.5 s period of network activity. (**B**) Resulting star plot from regular view point. (**C**) Star plot of the corresponding average attractor rate data.

orthogonal projection of the state space trajectory along the main diagonal of the corresponding Cartesian coordinate system onto a plane perpendicular to it. For n dimensions, points $\mathbf{x}$ in the star plot are no longer projections of the states $\mathbf{z}$, but are rather calculated as

$$\mathbf{x} = \sum_{i=1}^{n} z_i \left(\cos \frac{2\pi i}{n}, \sin \frac{2\pi i}{n} \right) \tag{A.13}$$

A visualization for $n = 3$ is illustrated in Figure A.2.
In case of the L2/3 network, the number of dimensions is given by the number of attractors, with each axis describing some particular feature of the corresponding attractor (such as the average voltage or spike rate of the constituent PYR cells).
In addition to the position in state space, the state space velocity is also encoded in a star plot by both the thickness and the color of the trajectory. Especially in the case of the L2/3 network, this can be very useful in visualizing e.g. attractor stability or competition times. Here, both line thickness and lightness were chosen proportional to $(\text{const} + e^{-|d\mathbf{x}|/dt})$, with $\mathbf{x}$ being the position in state space.
Figures 5.5B and C show two characteristic examples of star plots used for visualizing the dynamics of the L2/3 network.

A.2.3.6. Fitted Hardware-Compatible Parameters

Tables A.7, A.8 and A.9 contain all parameters required for the fits described in 5.3.4. All fits were performed by minimizing the L^2-norm of the distance between the simulated traces (Figure 5.4 A-C and G-L) or between spike timings (D-F).
The diffuse background stimulus was generated by Poisson spike sources at a total rate of 300 Hz per PYR cell.
Apart from random noise, the PYR cells further receive input from other PYR cells in cortical layer 4. The input intensity was calculated from the number of cells in layer 4 likely to project onto layer 2/3, which was estimated to be around 30 with a rate of approximately 10 Hz and a connection density of 25 % Lundqvist et al. (2006).
Therefore, a Poisson process with 75 Hz was used for each PYR cell input. Since we used static synapses for the Poisson input, the synaptic weights for source-PYR connections were chosen as 30% of PYR-PYR connections within the MCs. This was verified for compliance the original model from Lundqvist et al. (2006), which uses 7 to 8 sources per stimulated PYR cell with a rate of 10 Hz each and depressing synapses. For each stimulus event in the pattern completion and rivalry experiments (described below), layer 4 cells were set to fire for 60 ms. In each stimulated MC, 6 PYR cells were targeted from layer 4. Table A.10 shows the average firing rates for the different cell types in the network when only certain synapses are active.

Parameter	PYR	RSNP	BAS	Unit	Comment (see Figure 5.4)
C_m	0.179	0.0072	0.00688	nF	from the fits in A-C
E_e^rev	0.0	0.0	0.0	mV	difference to original model compensated by synaptic weights
E_i^rev	-80.0	-	-	mV	difference to original model compensated by synaptic weights
τ_m	16.89	15.32	15.64	ms	from the fits in A-C
τ_ref	0.16	0.16	0.16	ms	minimum available in hardware at the used speedup
τ_e^syn	17.5	66.6	6.0	ms	see paragraph "Synapses"
τ_i^syn	6.0	-	-	ms	see paragraph "Synapses"
ϱ	-60.7	-72.5	-72.5	mV	from the fits in D-F
V_rest	-61.71	-57.52	-56.0	mV	from the fits in D-F
a	0.0	0.28	0.0	nS	see fig from the fit in B
b	0.0132	0.00103	0.0	nA	from the fits in D and E
Δ_T	0.0	0.0	0.0	mV	from the fits in D-F
τ_w	196.0	250.0	0.0	ms	from the fits in D-F
V_spike	-53.0	-51.0	-52.5	mV	from the fits in D-F
ϑ	-	-	-	mV	not used since $\Delta_\mathrm{T} = 0$

Table A.7.: Fitted neuron parameters for the L2/3 model (see Figure 5.4).

A.2.3.7. Delays

Each connection within the same MC was set to have a constant synaptic delay of 0.5 ms. Additionally, axonal delays for connections between different MCs were realized by taking into account their spatial distance at an average axonal propagation speed of 200 µm/ms. Both the HCs in the whole network as well as the MCs within a single HC are laid out

Pre-Post	type	weight [μS]	τ^{syn} [ms]	U	τ_{rec} [ms]	τ_{facil} [ms]
PYR-PYR (local)	exc	0.004125	17.5	0.27	575.	0.
PYR-PYR (global)	exc	0.000615	17.5	0.27	575.	0.
PYR-BAS	exc	0.000092	6.0	-	-	-
PYR-RSNP	exc	0.000024	66.6	-	-	-
BAS-PYR	inh	0.0061	6.0	-	-	-
RSNP-PYR	inh	0.0032	6.0	-	-	-
background-PYR	exc	0.000224	17.5	-	-	-

Table A.8.: Fitted synapse parameters for the L2/3 model.

Background	
# of sources per PYR	1
rate	300 Hz
weight	0.000 224 µS
Shared background pool	
# of sources per PYR	100 out of 5000 total
rate	3 Hz
weight	0.000 224 µS
L4	
# of sources per MC	5
$p_{\text{L4}\rightarrow\text{PYR}}$	0.75
weight	0.001 237 5 µS
	(30% local PYR→PYR)

Table A.9.: Stimulus parameters for the L2/3 model.

setup no.	active synapses	ν_{PYR}	ν_{RSNP}	ν_{BAS}
1	background-PYR, PYR-BAS, PYR-RSNP	0.738 ± 0.096	57.946 ± 6.993	4.655 ± 1.081
2	same as 1 + BAS-PYR	0.174 ± 0.021	13.430 ± 1.910	1.119 ± 0.441
3	same as 1 + RSNP-PYR	0.257 ± 0.037	20.375 ± 2.536	1.783 ± 0.954
4	same as 2 + 3 + PYR-PYR (local)	0.200 ± 0.030	14.679 ± 2.261	1.258 ± 0.544
5	same as 2 + 3 + PYR-PYR (global)	0.204 ± 0.078	14.954 ± 5.680	1.337 ± 0.625

Table A.10.: Average firing rates (in Hz) of the different cell types of the L2/3 model with only certain synapses active.

on a hexagonal grid with a edge length of 500 µm (HC↔HC) / 60 µm (MC↔MC). In the default network (9HC×9MC) this leads to delays between 0.5 ms and 8 ms.

A.2.3.8. Scaling

Table A.11 shows the different instantiations of the L2/3 model that were used to produce the synapse loss values in Figure 5.11.

N_{HC}	N_{MC}	total neurons	N_{HC}	N_{MC}	total neurons
18	2	1188	27	9	8019
9	6	1782	18	18	10 692
27	3	2673	18	36	21 384
18	6	3564	36	24	28 512
36	4	4752	36	36	42 768
9	18	5346	27	54	48 114
18	12	7128	45	45	66 825

Table A.11.: Scaling table for the L2/3 model used for the synapse loss estimation in Figure 5.11.

A.2.4. Synfire Chain with Feedforward Inhibition

A.2.4.1. Model Parameters

Tables A.12 and A.13 list the parameters used for the synfire chain model in Section 5.4.

Parameter	Value	Unit
C_m	0.29	nF
τ_ref	2	ms
V_spike	-57	mV
E_r	-70	mV
V_rest	-70	mV
τ_m	10	ms
$E_\mathrm{e}^\mathrm{rev}$	0	mV
$E_\mathrm{i}^\mathrm{rev}$	-75	mV
$\tau_\mathrm{e}^\mathrm{syn}$	1.5	ms
$\tau_\mathrm{i}^\mathrm{syn}$	10	ms

Table A.12.: Neuron parameters used for the synfire chain model.

Projection	weight µS	incoming synapses	delay ms
$\mathrm{RS}_n \rightarrow \mathrm{RS}_{n+1}$	0.001	60	20
$\mathrm{RS}_n \rightarrow \mathrm{FS}_{n+1}$	0.0035	60	20
$\mathrm{FS}_n \rightarrow \mathrm{RS}_n$	0.002	25	4

Table A.13.: Projection properties for the feed-forward synfire chain.

A.2.4.2. Separatrix Fit

To compare different separatrices, the a-values of the last group are characterized as successful (+1) or extinguished (−1) and the resulting values interpolated and smoothed by a Gaussian kernel with a standard deviation $(1.5\,\mathrm{ms}, 1.5)$ in the (σ, a) space. The iso-contour line of the resulting surface at a value of 0 is used as an approximation of the separatrix location, as shown in Figure 5.14C together with the individual simulation results. Data points with $\sigma \leq 0.2\,\mathrm{ms}$ were not included in the fit to avoid distortions induced by bandwidth limitations in ESS simulations (Section 5.4.8) from affecting the fit quality. The data points are still shown individually as blue dots and regions, e.g., in Figure 5.19. This modification was also included in the software simulations for consistency. Cases in which the separatrix does not capture the relevant behavior, e.g., if the separation is not reliable in a large region of the state space, are shown separately.

A.2.4.3. Filtering of Spontaneous Activity

To prevent spontaneous background events from impeding the analysis, spikes are discarded as part of spontaneous activity if less then N spikes in the same excitatory group

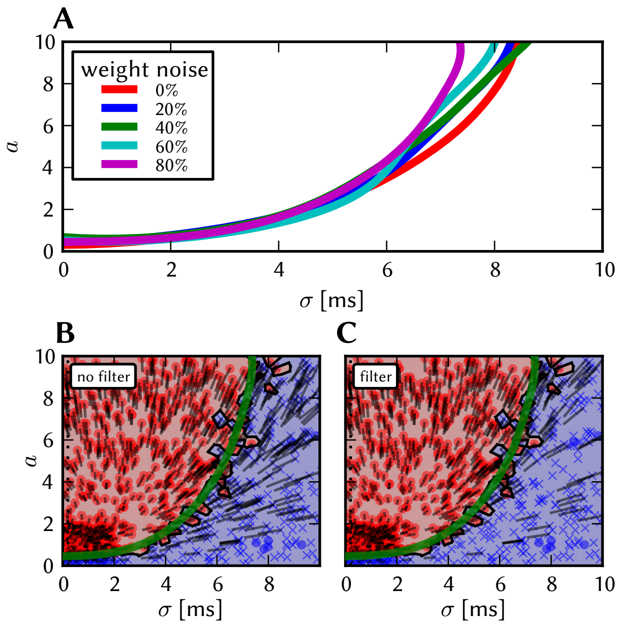


Figure A.3.: Demonstration of spontaneous event filter in the weight noise compensation (Section A.2.4.3). **(A)** The same experiment as in Figure 5.16C (weight noise with active compensation) but without the filter for background spikes. The separatrix locations are comparable as the filter does not influence the result significantly in the compensated case. **(B)**, **(C)** Complete state space response for weight noise of 80%, once with, once without filter. This demonstrates that the applied filter does not affect the result in the compensated case.

occur in a time window of $\pm T$. The utilized values for N and T are given at each point where the filter is applied. They are chosen such that authentic synchronous volleys with $a \geq 0.5$ (which would be counted as successful propagation, as defined above) are not removed. Figure A.3B and C show that the influence of the filter for spontaneous activity is minimal in the compensated case.

A.2.5. Self-Sustained Asynchronous Irregular Activity

A.2.5.1. Model Parameters

The neuron parameters used for this model are listed in Table A.14 and are largely the same as those in Muller and Destexhe (2012), with the only difference being that excitatory pyramidal cells have neuronal spike-triggered adaptation while inhibitory cells do not. Sweeps are performed over the two-dimensional ($g_\mathrm{e}^\mathrm{syn}$, $g_\mathrm{i}^\mathrm{syn}$) parameter space, with the

Parameter	Pyramidal	Inhibitory	Unit
C_m	0.25	0.25	nF
τ_ref	5	5	ms
V_spike	-40	-40	mV
E_r	-70	-70	mV
V_rest	-70	-70	mV
τ_m	15	15	ms
a	1	1	nS
b	0.005	0	nA
Δ_T	2.5	2.5	mV
τ_w	600	600	ms
E_T	-50	-50	mV
$E_\mathrm{e}^\mathrm{rev}$	0	0	mV
$E_\mathrm{i}^\mathrm{rev}$	-80	-80	mV
$\tau_\mathrm{e}^\mathrm{syn}$	5	5	ms
$\tau_\mathrm{i}^\mathrm{syn}$	5	5	ms

Table A.14.: AdEx Neuron parameters used in the AI network.

ranges being 3 nS to 11 nS for $g_\mathrm{e}^\mathrm{syn}$ and 50 nS to 130 nS for $g_\mathrm{i}^\mathrm{syn}$. The Poisson sources for the initial network stimulation have a mean rate of 100 Hz and project onto the network's neurons with a synaptic weight of 100 nS. The distance-dependent connection probability has a Gaussian profile with a spatial width of $\sigma = 0.2\,\mathrm{mm}$. Synaptic delays depend on the distance according to the following equation: $t_\mathrm{delay} = 0.3\,\mathrm{ms} + \frac{d}{v_\mathrm{prop}}$, with d being the distance between two cells and $v_\mathrm{prop} = 0.2\,\mathrm{mm\,ms^{-1}}$ the spike propagation velocity. The distribution of delays is shown in Figure 5.24, the average delay in the network amounts to 1.55 ms.

A.2.6. Liquid Factor Graphs

A.2.6.1. Liquid Parameters

Connection Type	μ_w [nA]	CV_w	μ_D [ms]	CV_D	τ_{syn} [ms]
Exc $\rightarrow$ Exc	30	0.7	1.5	0.1	3
Exc $\rightarrow$ Inh	60	0.7	0.8	0.1	3
Inh $\rightarrow$ Exc	-19	0.7	0.8	0.1	6
Inh $\rightarrow$ Inh	-19	0.7	0.8	0.1	6

Table A.15.: Liquid pool connection parameters. The synaptic weights are drawn from a Gamma distribution

$$p(x;k,\theta) = \frac{1}{\Gamma(k)\theta^k} x^{k-1} e^{-\frac{x}{\theta}} \text{ for } x \geq 0 \text{ and } k,\theta > 0 \tag{A.14}$$

with mean $\mu = k\theta$ and variance $\sigma^2 = k\theta^2$. The table lists the chosen mean μ_w and CV CV_w. The synaptic delays are drawn from a Gaussian distribution with mean μ_D and CV CV_D.

Population Type	V_rest [mV]	V_reset [mV]	V_thresh [mV]	τ_m [ms]	τ_refrac [ms]
Exc	0	$[13.8, 14.5]$	15	30	3
Inh	0	$[13.8, 14.5]$	15	30	2

Table A.16.: Liquid pool neuron parameters. The reset voltages are drawn from a uniform distribution with the given range. Additionally, each cell receives a constant background current that is also drawn from a uniform distribution with the range [13.5,14.5]nA.

A.2.6.2. Readout Training

Each readout population consists of 343 (unconnected) CUBA LIF neurons with $E_\mathrm{l} = 0\,\mathrm{mV}$, $\tau_\mathrm{m} = 30\,\mathrm{ms}$, $u_\mathrm{reset} = 10\,\mathrm{mV}$, $\tau_\mathrm{ref} = 2\,\mathrm{ms}$, $\tau^\mathrm{syn} = 6\,\mathrm{ms}$. Thresholds are drawn from a uniform distribution in $[15, 20\,]mV$. Each neuron receives a constant background current of 0.5 nA and an independent Gaussian noise with a standard deviation of 0.14 nA.
The $I - R$ fit is a polynomial of degree 9.

A.2.6.3. Implementation of the Knill-Kersten Illusion

	$S = 0$	
	$O = 0$	$O = 1$
$R = 0$	0.8	0.2
$R = 1$	0.2	0.9

	$S = 1$	
	$O = 0$	$O = 1$
$R = 0$	0.2	0.8
$R = 1$	0.8	0.1

Table A.17.: Definition of the factor $P_1(S|R, O)$.

	$O = 0$	$O = 1$
$C = 0$	0.8	0.2
$C = 1$	0.2	0.8

Table A.18.: Definition of the factor $P_2(C|O)$.

Liquid Name	Column Structure [width × length × height]	Total Number	Connectivity λ
L_1	$5 \times 5 \times 78$	1950	7.0
L_2	$5 \times 5 \times 42$	1050	4.5
$L_=$	$5 \times 5 \times 42$	1050	4.5

Table A.19.: Liquid parameters.

A.2.7. LIF Sampling

All simulations have been performed with the NEURON simulation package (Hines and Carnevale, 2006) and the PyNN API (Davison et al., 2008), with a time step of $dt = 0.01\,\mathrm{ms}$. For the LIF neuron, we have chosen the parameters listed in Table A.20 (see also Naud et al., 2008):

C_m	0.1 nF	membrane capacitance
g_l	5 nS	leak conductance
E_l	-65 mV	leak potential
ρ	-53 mV	reset potential
$E^\mathrm{rev}_\mathrm{exc}$	0 mV	excitatory reversal potential
$E^\mathrm{rev}_\mathrm{inh}$	-90 mV	inhibitory reversal potential
ϑ	-52 mV	threshold voltage
τ^syn	10 ms	synaptic time constant
τ_ref	10 ms	refractory time constant

Table A.20.: Standard parameter set used for all simulations.

Synaptic noise was implemented as bombardment by inhibitory and excitatory Poisson stimuli with rates $\nu_\mathrm{inh} = \nu_\mathrm{exc} = 5000\,\mathrm{Hz}$. The excitatory synaptic weight for the noise stimuli was set to $w^\mathrm{noise}_\mathrm{exc} = 0.0035\,\mu\mathrm{S}$. The inhibitory weight $w^\mathrm{noise}_\mathrm{inh}$ was adjusted as to yield $p(z_k = 1) \approx 0.5$ with no current stimulus present. For above parameters, this happens at an average free membrane potential of $V_g = -55\,\mathrm{mV}$. This determines $w^\mathrm{noise}_\mathrm{inh}$ according to

$$\left|\frac{E^\mathrm{rev}_\mathrm{inh} - V_g}{E^\mathrm{rev}_\mathrm{exc} - V_g}\right| = \frac{w^\mathrm{noise}_\mathrm{exc}}{w^\mathrm{noise}_\mathrm{inh}} \quad . \tag{A.15}$$

A.2.7.1. Figure 6.32: Activation Functions

In order to sweep through the activation function, the external current I^ext was varied. However, in order to facilitate a comparison with the sigmoidal activation function from the abstract model, we have represented $p(z = 1)$ as a function of $\bar{u}$ instead. The latter is equivalent to the mean μ of the corresponding Ornstein-Uhlenbeck process, with Equation 6.118 allowing a direct translation between I^ext and $\bar{u}$.
The abscissa values represent averages of the free membrane potential obtained from 10 simulation runs with a total duration of $T_\mathrm{sim} = 100\,\mathrm{s}$ and firing threshold θ set to $E^\mathrm{rev}_\mathrm{exc} = 0\,\mathrm{mV}$. The deviations from the theoretical prediction (Equation 6.118) are smaller than the size of the symbols, therefore no error bars are shown.
The ordinate values and standard errors were calculated from the simulated spike train data according to

$$p(z = 1) = \frac{1}{N}\sum_{i=1}^{N} p_i \quad , \tag{A.16}$$

$$s = \sqrt{\frac{1}{N-1} \cdot \sum_{i=1}^{N} \left[p_i - p(z = 1)\right]^2} \quad , \tag{A.17}$$

with $p_i = \frac{N_i^{\text{spk}} \tau_{\text{on}}}{T_{\text{sim}}}$ being the fraction of time spent in $z = 1$ and N_i^{spk} representing the total number of spikes in the ith out of $N = 10$ performed simulations. Since the respective standard errors of the mean are smaller than the size of the symbols, no error bars are shown.

A.2.7.2. Figure 6.5.5: Convergence to Target Distributions

The simulated network consists of $K = 5$ neurons with a synaptic weight matrix $\boldsymbol{W}$ and a bias vector $\boldsymbol{b}$ (both in the Boltzmann domain). All entries were drawn from a beta distribution $\mathcal{B}(0.5, 0.5)$ and mapped linearly to the interval $[-0.6, 0.6]$. More specifically, $b_k, W_{kj} \sim 1.2 \cdot [\mathcal{B}(0.5, 0.5) - 0.5]$. The parameters and mapping of the beta distribution were chosen with the intent of generating diverse distributions, spanning multiple orders of magnitude. The bias b_k, defined in the Boltzmann machine domain, determines the probability $p(z_k = 1|\boldsymbol{z}\backslash k = \boldsymbol{0})$ for neuron k. In the LIF domain, the probability $p(z_k = 1|\boldsymbol{z}\backslash k = \boldsymbol{0}) = 0.5$ corresponds to the mean free membrane potential $\bar{u}_k^0$. Then, a nonzero bias can be described in the LIF domain as a linear shift from $\bar{u}_k^0$ to a mean membrane potential $\bar{u}_k^b$. This yields the linear transformation

$$b_k = (\bar{u}_k^b - \bar{u}_k^0)/\alpha \quad , \tag{A.18}$$

where α represents the scaling factor between the two domains. Both quantities $\bar{u}_k^0$ and α can be determined from the predicted activation function of a single LIF unit. The first quantity constitutes the inflection point of the activation function (at $p(z_k = 1|\boldsymbol{z}\backslash k = \boldsymbol{0}) = 0.5$), the latter follows from the slope of the function.
By computing $\bar{u}_k^b$, we can map any bias b_k of a single unit of the Boltzmann machine onto a yet unconnected LIF neuron. In simulations, $\bar{u}_k^b$ was established by injecting a temporally constant external current I_{ext}^k according to

$$I^{\text{ext}}{}_k = (\alpha b_k + \bar{u}_k^0) \cdot \left\langle g^{\text{tot}} \right\rangle - g_l E_l - \sum_i \nu_i w_i^{noise} E_i^{\text{rev}} \tau^{\text{syn}} \quad . \tag{A.19}$$

In order to achieve sampling network dynamics in the LIF domain faithful to those displayed by the Boltzmann machine, the Boltzmann weight matrix $\boldsymbol{W}$ was translated into LIF network weights w_{ij} according to Equation 6.146. Thus, superposing PSPs saturate the membrane potential, approximating the constant amplitude of a PSP in the abstract neuron model.
For Figure 6.36B, this setup of a random Boltzmann machine was simulated $N = 10$ times with different random seeds for the Poisson background for a duration of $T_{\text{sim}} = 10\,\text{s}$. The red bars show the analytically computed target joint distribution $p_{\text{B}}(\boldsymbol{z})$. The blue bars depict the network distribution $p_{\text{N}}(\boldsymbol{z})$, calculated from the firing activity of the simulated LIF network set up to match $p_{\text{B}}(\boldsymbol{z})$. The means and error bars have been calculated as in Equations A.16 and A.17, respectively.

The above simulations were repeated with a significantly longer duration in order to study systematic deviations due to the LIF implementation. Figure 6.36C shows the distance between the target distribution $p_{\text{B}}(\boldsymbol{z})$ and its LIF network representation $p_{\text{N}}(\boldsymbol{z})$ in form of the Kullback-Leibler divergence

$$D_{\text{KL}}\left(p_{\text{N}} \parallel p_{\text{B}}\right) = \left\langle \log\left[p_{\text{N}}(\boldsymbol{z}) \,/\, p_{\text{B}}(\boldsymbol{z}) \right] \right\rangle_{p_{\text{N}}(\boldsymbol{z})} \quad . \tag{A.20}$$

This estimate has been taken for one set of parameters ($\boldsymbol{W}$, $\boldsymbol{b}$) for ten independent trials (thin lines) in an LIF network at integration times T: $0 \leq T \leq T_{\mathrm{sim}} = 10^6\,\mathrm{ms}$. The red dashed line displays the averaged $D_{\mathrm{KL}}\left(p_{\mathrm{N}} \parallel p_{\mathrm{B}}\right)$ for the abstract network model with identical parameters ($\boldsymbol{W}$, $\boldsymbol{b}$). The decrease of $D_{\mathrm{KL}}\left(p_{\mathrm{N}} \parallel p_{\mathrm{B}}\right)$ for longer integration times indicates the increasing precision of the sampling network over time. Eventually, the $D_{\mathrm{KL}}\left(p_{\mathrm{N}} \parallel p_{\mathrm{B}}\right)$ converges for the LIF network to a nonzero value, reflecting small systematic errors.
Figure 6.36D shows the distribution of $D_{\mathrm{KL}}\left(p_{\mathrm{N}} \parallel p_{\mathrm{B}}\right)$ values for 100 randomly drawn Boltzmann machines emulated by LIF networks, evaluated from a single run of $T_{\mathrm{sim}} = 10^6\,\mathrm{ms}$ each.

A.2.8. LIF-Based BNs

LIF parameter	standard	noisy
Resting membrane potential V_{rest}	$\overline{V_k^b}$	$\overline{V_k^b} \pm 2.0\,\mathrm{mV}$
Capacity of the membrane C_{m}	$0.2\,\mathrm{nF}$	$0.2\,\mathrm{nF}$
Membrane time constant τ_{m}	$0.1\,\mathrm{ms}$	$(1.0 \pm 0.1)\,\mathrm{ms}$
Duration of refractory period τ_{ref}	$20.0\,\mathrm{ms}$	$(20.0 \pm 1.0)\,\mathrm{ms}$
Excitatory synaptic time constant $\tau^{\mathrm{syn,exc}}$	$10.0\,\mathrm{ms}$	$(20.0 \pm 2.0)\,\mathrm{ms}$
Inhibitory synaptic time constant $\tau^{\mathrm{syn,inh}}$	$10.0\,\mathrm{ms}$	$(20.0 \pm 2.0)\,\mathrm{ms}$
Reversal potential for excitatory input $E^{\mathrm{rev,exc}}$	$0.0\,\mathrm{mV}$	$(0.0 \pm 2.0)\,\mathrm{mV}$
Reversal potential for inhibitory input $E^{\mathrm{rev,inh}}$	$-100.0\,\mathrm{mV}$	$(-100.0 \pm 2.0)\,\mathrm{mV}$
Spike threshold V_{th}	$-50.0\,\mathrm{mV}$	$(-50.0 \pm 0.5)\,\mathrm{mV}$
Reset potential after a spike V_{reset}	$-53.0\,\mathrm{mV}$	$(-53.0 \pm 0.5)\,\mathrm{mV}$
Utilization of synaptic efficacy U_0	1.0	1.0
Recovery time constant τ_{rec}	$0.99 \cdot \tau^{\mathrm{syn}}$	$0.99 \cdot \tau^{\mathrm{syn}}$
Facilitation time constant τ_{facil}	$0.0\,\mathrm{ms}$	$0.0\,\mathrm{ms}$
Excitatory/inhibitory Poisson input rate ν^{syn}	$400.0\,\mathrm{Hz}$	$5000.0\,\mathrm{Hz}$
Excitatory/inhibitory background weight w^{syn}	$0.002\,\mu\mathrm{S}$	$0.001\,\mu\mathrm{S}$
Synaptic delays	$0.1\,\mathrm{ms}$	$1.2\,\mathrm{ms}$
Boltzmann machines: Parameter	**standard**	**noisy**
W_{ij}	W_{ij}	$\epsilon \cdot W_{ij}$
b_i	b_i	$\epsilon \cdot b_i$
γ (Equation 16)	10	5
μ (Equation 17)	$1 + 10^{-4}$	$1 + 10^{-4}$

Table A.21.: Standard neuron and network parameters used in this paper. The network parameter ϵ denotes a sample from the uniform distribution $\mathrm{unif}(0.9, 1.1)$

Parameters of the first chain neuron	
Capacity of the membrane C_m	0.2 nF
Membrane time constant τ_m	0.1 ms
Duration of refractory period τ_ref	29.5 ms
Decay time of the excitatory synaptic conductance $\tau_\mathrm{syn,exc}$	30.0 ms
Decay time of the inhibitory synaptic conductance $\tau_\mathrm{syn,inh}$	30.0 ms
Reversal potential for excitatory input $E_\mathrm{exc}^\mathrm{rev}$	0.0 mV
Reversal potential for inhibitory input $E_\mathrm{inh}^\mathrm{rev}$	-100.0 mV
Spike threshold V_th	-50.0 mV
Reset potential after a spike V_reset	-50.01 mV
Parameters of the remaining chain neurons	
Capacity of the membrane C_m	0.2 nF
Membrane time constant τ_m	0.1 ms
Duration of refractory period τ_ref	29.3 ms
Decay time of the excitatory synaptic conductance $\tau_\mathrm{syn,exc}$	2.0 ms
Decay time of the inhibitory synaptic conductance $\tau_\mathrm{syn,inh}$	2.0 ms
Reversal potential for excitatory input $E_\mathrm{exc}^\mathrm{rev}$	0.0 mV
Reversal potential for inhibitory input $E_\mathrm{inh}^\mathrm{rev}$	-100.0 mV
Spike threshold V_th	-50.0 mV
Reset potential after a spike V_reset	-52.3 mV
Resting membrane potential V_rest	-52.3 mV
Parameters of the neural chain	
Number of chain neurons	6
Delay: sampling $\rightarrow$ sampling neuron	0.1 ms
Delay: sampling $\rightarrow$ forwarding neuron	5.8 ms
Delay: forwarding $\rightarrow$ sampling neuron	0.1 ms
Delay: forwarding $\rightarrow$ forwarding neuron	5.8 ms
Delay: forwarding $\rightarrow$ last forwarding neuron	5.9 ms
Weight: sampling $\rightarrow$ sampling neuron	w
Weight: sampling $\rightarrow$ forwarding neuron	0.16 μS
Weight: forwarding $\rightarrow$ sampling neuron	$0.180 \cdot w$
Weight: last forwarding $\rightarrow$ sampling neuron	$-0.815 \cdot w$
Weight: forwarding $\rightarrow$ forwarding neuron	0.16 μS

Table A.22.: Parameters of the interneuron chain of 6 neurons, which are used to generate the mLIF PSP.

A.3. Previously Published Material

The following material was taken largely unchanged from previously published work that was either co-authored or co-supervised by the author of this thesis.

The contents of Chapters 3 and 5 were taken from Petrovici et al. (2014) and Pfeil et al. (2013).

The contents of Sections 6.2.3 and 6.7 were taken from Pfeil et al. (2013) and Probst et al. (2015).

Figures 4.7, 4.8, 4.9, 4.10, 4.11, 4.13, 4.14, 4.15, 4.16, 4.18, 4.19 and 4.20 were taken from Bytschok (2011).

Figures 6.25, 6.36 and 6.42 were taken from Petrovici et al. (2013).

Figures 5.2, 5.10, 5.11, 5.14, 5.15, 5.16, 5.17, 5.18, 5.19, 5.20, 5.22, 5.24, 5.25, 5.28, 5.29 and 5.30 were taken from Petrovici et al. (2014).

Figures 3.2, 3.3, 5.13, 5.21 and 6.13 were taken from Pfeil et al. (2013).

Figures 6.26, 6.33, 6.52, 6.53 and 6.54 were taken from Probst et al. (2015).

Figures 6.58 and 6.59 were taken from Stöckel (2015).

Figures 6.10, 6.11, 6.15, 6.16, 6.17, 6.18, 6.19, 6.20, 6.21, 6.22, 6.56 were taken from Petkov (2012).

Bibliography

L. F. Abbott and S. B. Nelson. Synaptic plasticity: taming the beast. *Nature neuroscience*, 3:1178–1183, 2000.

M. Abeles, G. Hayon, and D. Lehmann. Modeling compositionality by dynamic binding of synfire chains. *Journal of computational neuroscience*, 17(2):179–201, 2004.

A. Aertsen, M. Diesmann, and M. O. Gewaltig. Propagation of synchronous spiking activity in feedforward neural networks. *J Physiol Paris*, 90(3-4):243–247, 1996. ISSN 0928-4257. URL `http://view.ncbi.nlm.nih.gov/pubmed/9116676`.

B. Alberts, D. Bray, J. Lewis, M. Raff, K. Roberts, and J. D. Watsonx. *Molecular Biology of the Cell, third edition.* Garland Publishing, Inc, 1994. ISBN 0815316208.

J. Ambros-Ingerson and W. R. Holmes. Analysis and comparison of morphological reconstructions of hippocampal field ca1 pyramidal cells. *Hippocampus*, 15:302–315, 2005.

D. J. Amit and N. Brunel. Model of global spontaneous activity and local structured activity during delay periods in the cerebral cortex. *Cereb Cortex*, 7(3):237–52, Jan 1997.

R. Azouz and C. M. Gray. Cellular mechanisms contributing to response variability of cortical neurons in vivo. *The Journal of neuroscience*, 19(6):2209–2223, 1999.

Y. Bengio and L. Yao. Bounding the test log-likelihood of generative models. *arXiv preprint arXiv:1311.6184*, 2013.

P. Berkes, G. Orbán, M. Lengyel, and J. Fiser. Spontaneous cortical activity reveals hallmarks of an optimal internal model of the environment. *Science*, 331(6013):83–87, 2011.

G. Q. Bi and M. M. Poo. Synaptic modifications in cultured hippocampal neurons: dependence on spike timing, synaptic strength, and postsynaptic cell type. *The Journal of neuroscience : the official journal of the Society for Neuroscience*, 18(24):10464–10472, Dec. 1998. ISSN 0270-6474. URL `http://www.jneurosci.org/content/18/24/10464.abstract`.

E. L. Bienenstock, L. N. Cooper, and P. W. Munro. Theory for the development of neuron selectivity: Orientation specificity and binocular interaction in visual cortex. *Journal of Neuroscience*, 2(2):32–48, 1982.

J. Bill, K. Schuch, D. Brüderle, J. Schemmel, W. Maass, and K. Meier. Compensating inhomogeneities of neuromorphic VLSI devices via short-term synaptic plasticity. *Front. Comp. Neurosci.*, 4(129), 2010.

C. M. Bishop. *Pattern recognition and machine learning*, volume 1. springer New York, 2009.

J. W. Brascamp, R. Van Ee, A. J. Noest, R. H. Jacobs, and A. V. van den Berg. The time course of binocular rivalry reveals a fundamental role of noise. *Journal of vision*, 6(11):8, 2006.

O. Breitwieser. Investigation of a cortical attractor-memory network. Bachelor thesis, Ruprecht-Karls-Universität Heidelberg, 2011. HD-KIP 11-173.

O. Breitwieser. Towards a neuromorphic implementation of spike-based expectation maximization. Master thesis, Ruprecht-Karls-Universität Heidelberg, 2015.

R. Brette and W. Gerstner. Adaptive exponential integrate-and-fire model as an effective description of neuronal activity. *J. Neurophysiol.*, 94:3637 – 3642, 2005. doi: NA.

R. Brette, M. Rudolph, T. Carnevale, M. Hines, D. Beeman, J. M. Bower, M. Diesmann, A. Morrison, P. H. Goodman, F. C. Harris Jr, M. Zirpe, T. Natschlager, D. Pecevski, B. Ermentrout, M. Djurfeldt, A. Lansner, O. Rochel, T. Vieville, E. Muller, A. P. Davison, S. E. Boustani, and A. Destexhe. Simulation of networks of spiking neurons: A review of tools and strategies. *Journal of Computational Neuroscience*, 23(3):349–398, December 2007.

V. Bringuier, F. Chavane, L. Glaeser, and Y. Frégnac. Horizontal propagation of visual activity in the synaptic integration field of area 17 neurons. *Science*, 283(5402):695–699, 1999. doi: 10.1126/science.283.5402.695. URL `http://www.sciencemag.org/content/283/5402/695.abstract`.

D. Brüderle, E. Müller, A. Davison, E. Muller, J. Schemmel, and K. Meier. Establishing a novel modeling tool: A python-based interface for a neuromorphic hardware system. *Front. Neuroinform.*, 3(17), 2009.

D. Brüderle, M. A. Petrovici, S. Jeltsch, B. Vogginger, S. Friedmann, M. Schmuker, J. Kremkow, T. Clayton, V. Petkov, J. Bill, M. Albert, A. Hartel, J. Partzsch, E. Müller, L. Muller, O. Bichler, J. Schemmel, and K. Meier. Simulator-like exploration of network architectures with the facets hardware systems and pynn, 2010. URL `http://www.kip.uni-heidelberg.de/cms/groups/vision/galleries_media/cne2010_experiments/`.

D. Brüderle, M. A. Petrovici, B. Vogginger, M. Ehrlich, T. Pfeil, S. Millner, A. Grübl, K. Wendt, E. Müller, M.-O. Schwartz, D. de Oliveira, S. Jeltsch, J. Fieres, M. Schilling, P. Müller, O. Breitwieser, V. Petkov, L. Muller, A. Davison, P. Krishnamurthy, J. Kremkow, M. Lundqvist, E. Muller, J. Partzsch, S. Scholze, L. Zühl, C. Mayr, A. Destexhe, M. Diesmann, T. Potjans, A. Lansner, R. Schüffny, J. Schemmel, and K. Meier. A comprehensive workflow for general-purpose neural modeling with highly configurable neuromorphic hardware systems. *Biological Cybernetics*, 104:263–296, 2011. ISSN 0340-1200. URL `http://dx.doi.org/10.1007/s00422-011-0435-9`.

N. Brunel. Dynamics of sparsely connected networks of excitatory and inhibitory spiking neurons. *Journal of Computational Neuroscience*, 8(3):183–208, 2000.

N. Brunel and S. Sergi. Firing frequency of leaky integrate-and-fire neurons with synaptic current dynamics. *Journal of Theoretical Biology*, 195:87–95, 1998.

N. Brunel and M. C. Van Rossum. Lapicque's 1907 paper: from frogs to integrate-and-fire. *Biological cybernetics*, 97(5-6):337–339, 2007.

L. Buesing, J. Bill, B. Nessler, and W. Maass. Neural dynamics as sampling: A model for stochastic computation in recurrent networks of spiking neurons. *PLoS Computational Biology*, 7(11):e1002211, 2011.

A. N. Burkitt. A review of the integrate-and-fire neuron model: Ii. inhomogeneous synaptic input and network properties. *Biological cybernetics*, 95(2):97–112, 2006.

D. Buxhoeveden and M. Casanova. The minicolumn and evolution of the brain. *Brain Behav Evol*, 60:125–151, 2002.

I. Bytschok. *From shared input to correlated neuron dynamics: Development of a predictive framework*. PhD thesis, Diploma thesis, University of Heidelberg, 2011.

P. Caroni, F. Donato, and D. Muller. Structural plasticity upon learning: regulation and functions. *Nature Reviews Neuroscience*, 13(7):478–490, 2012.

C. C. Chow and J. A. White. Spontaneous action potentials due to channel fluctuations. *Biophysical Journal*, 71(6):3013, 1996.

D. Ciresan, U. Meier, J. Masci, and J. Schmidhuber. A committee of neural networks for traffic sign classification. In *Neural Networks (IJCNN), The 2011 International Joint Conference on*, pages 1918–1921. IEEE, 2011.

D. Cireşan, U. Meier, J. Masci, and J. Schmidhuber. Multi-column deep neural network for traffic sign classification. *Neural Networks*, 32:333–338, 2012.

G. M. Clemence. The relativity effect in planetary motions. *Reviews of Modern Physics*, 19(4):361, 1947.

J. A. Connor, D. Walter, and R. McKowN. Neural repetitive firing: modifications of the hodgkin-huxley axon suggested by experimental results from crustacean axons. *Biophysical Journal*, 18(1):81, 1977.

B. Connors and M. Gutnick. Intrinsic firing patterns of diverse neocortical neurons. *Trends Neurosci.*, 13:99–104, 1990.

R. Cossart, D. Aronov, and R. Yuste. Attractor dynamics of network up states in the neocortex. *Nature*, 423:238–283, 2003.

A. P. Davison, D. Brüderle, J. Eppler, J. Kremkow, E. Muller, D. Pecevski, L. Perrinet, and P. Yger. PyNN: a common interface for neuronal network simulators. *Front. Neuroinform.*, 2(11), 2008.

P. Dayan and L. F. Abbott. *Theoretical Neuroscience: Computational and Mathematical Modeling of Neural Systems*. The MIT press, Cambride, Massachusetts, 2001. ISBN 0-262-04199-5.

R. C. Decharms and A. Zador. Neural representation and the cortical code. *Annual review of neuroscience*, 23(1):613–647, 2000.

S. Deneve. Bayesian spiking neurons i: inference. *Neural computation*, 20(1):91–117, 2008.

A. Destexhe. High-conductance state. *Scholarpedia*, 2(11):1341, 2007.

A. Destexhe. Self-sustained asynchronous irregular states and Up/Down states in thalamic, cortical and thalamocortical networks of nonlinear integrate-and-fire neurons. *Journal of Computational Neuroscience*, 3:493 – 506, 2009.

A. Destexhe and D. Contreras. Neuronal computations with stochastic network states. *Science*, 314(5796):85–90, 2006. doi: 10.1126/science.1127241. URL `http://www.sciencemag.org/content/314/5796/85.abstract`.

A. Destexhe and D. Pare. Impact of Network Activity on the Integrative Properties of Neocortical Pyramidal Neurons In Vivo. *J Neurophysiol*, 81(4):1531–1547, 1999.

A. Destexhe, M. Rudolph, and D. Pare. The high-conductance state of neocortical neurons in vivo. *Nature Reviews Neuroscience*, 4:739–751, 2003.

M. Diesmann. *Conditions for Stable Propagation of Synchronous Spiking in Cortical Neural Networks: Single Neuron Dynamics and Network Properties*. PhD thesis, Ruhr-Universität Bochum, 2002.

M. Diesmann and M.-O. Gewaltig. NEST: An environment for neural systems simulations. In T. Plesser and V. Macho, editors, *Forschung und wisschenschaftliches Rechnen, Beiträge zum Heinz-Billing-Preis 2001*, volume 58 of *GWDG-Bericht*, pages 43–70. Ges. für Wiss. Datenverarbeitung, Göttingen, 2002.

M. Diesmann, M.-O. Gewaltig, and A. Aertsen. Stable propagation of synchronous spiking in cortical neural networks. *Nature*, 402:529–533, 1999.

M. Diesmann, M.-O. Gewaltig, S. Rotter, and A. Aertsen. State space analysis of synchronous spiking in cortical neural networks. *Neurocomputing*, 38:565–571, 2001.

J. C. Eccles. From electrical to chemical transmission in the central nervous system. *Notes Rec. R. Soc. Lond. 1*, 30(2):219–230, 1976.

J. C. Eccles, P. Fatt, and K. Koketsu. Cholinergic and inhibitory synapses in a pathway from motor-axon collaterals to motoneurones. *J. Physiol.*, 126:524–562, 1954.

M. Ehrlich, C. Mayr, H. Eisenreich, S. Henker, A. Srowig, A. Grübl, J. Schemmel, and R. Schüffny. Wafer-scale VLSI implementations of pulse coupled neural networks. In *Proceedings of the International Conference on Sensors, Circuits and Instrumentation Systems (SSD-07)*, March 2007.

M. Ehrlich, K. Wendt, L. Zühl, R. Schüffny, D. Brüderle, E. Müller, and B. Vogginger. A software framework for mapping neural networks to a wafer-scale neuromorphic hardware system. In *Proceedings of the Artificial Neural Networks and Intelligent Information Processing Conference (ANNIIP) 2010*, pages 43–52, 2010.

S. El Boustani and A. Destexhe. A master equation formalism for macroscopic modeling of asynchronous irregular activity states. *Neural Computation*, 21(1):46–100, 2009. URL `http://dx.doi.org/10.1162/neco.2009.02-08-710`.

B. Ermentrout. Type i membranes, phase resetting curves, and synchrony. *Neural computation*, 8(5):979–1001, 1996.

R. P. Feynman, R. B. Leighton, and M. Sands. Feynman lectures on physics, vol. ii–the new millennium edition: Mainly electromagnetism and matter, 2011.

J. Fieres, J. Schemmel, and K. Meier. Realizing biological spiking network models in a configurable wafer-scale hardware system. In *Proceedings of the 2008 International Joint Conference on Neural Networks (IJCNN)*, 2008.

R. FitzHugh. Impulses and physiological states in theoretical models of nerve membrane. *Biophysical journal*, 1(6):445–466, 1961.

S. Friedmann, N. Frémaux, J. Schemmel, W. Gerstner, and K. Meier. Reward-based learning under hardware constraints - using a RISC processor in a neuromorphic system. *Frontiers in Neuromorphic Engineering*, 2013. URL `http://arxiv.org/abs/1303.6708`. submitted.

S. B. Furber, D. R. Lester, L. A. Plana, J. D. Garside, E. Painkras, S. Temple, and A. D. Brown. Overview of the SpiNNaker system architecture. *IEEE Transactions on Computers*, 99(PrePrints), 2012. ISSN 0018-9340. doi: http://doi.ieeecomputersociety.org/10.1109/TC.2012.142.

W. Gerstner and R. Brette. Adaptive exponential integrate-and-fire model. *Scholarpedia*, 4(6):8427, 2009. doi: 10.4249/scholarpedia.8427. URL `http://www.scholarpedia.org/article/Adaptive_exponential_integrate-and-fire_model`.

W. Gerstner and W. Kistler. *Spiking Neuron Models: Single Neurons, Populations, Plasticity.* Cambridge University Press, 2002.

M.-O. Gewaltig and M. Diesmann. NEST (NEural Simulation Tool). *Scholarpedia*, 2(4): 1430, 2007.

M. Giulioni, P. Camilleri, M. Mattia, V. Dante, J. Braun, and P. Del Giudice. Robust working memory in an asynchronously spiking neural network realized in neuromorphic vlsi. *Frontiers in Neuroscience*, 5(149), 2012. ISSN 1662-453X. doi: 10.3389/fnins.2011.00149. URL `http://www.frontiersin.org/neuromorphic_engineering/10.3389/fnins.2011.00149/abstract`.

G. González-Burgos, G. Barrionuevo, and D. A. Lewis. Horizontal synaptic connections in monkey prefrontal cortex: An in vitro electrophysiological study. *Cerebral Cortex*, 10 (1):82–92, 2000. doi: 10.1093/cercor/10.1.82. URL `http://cercor.oxfordjournals.org/content/10/1/82.abstract`.

C. G. Gross. Aristotle on the brain. *The Neuroscientist*, 1(4):245–250, 1995.

R. Gütig and H. Sompolinsky. The tempotron: a neuron that learns spike timing-based decisions. *Nat Neurosci*, 9(3):420–428, Mar. 2006. ISSN 1097-6256. URL `http://dx.doi.org/10.1038/nn1643`.

S. Habenschuss, J. Bill, and B. Nessler. Homeostatic plasticity in bayesian spiking networks as expectation maximization with posterior constraints. *Advances in Neural Information Processing Systems*, 25, 2012.

M. M. Halldórsson and J. Radhakrishnan. Greed is good: Approximating independent sets in sparse and bounded-degree graphs. *Algorithmica*, 18(1):145–163, 1997.

S. Hartmann, S. Schiefer, S. Scholze, J. Partzsch, C. Mayr, S. Henker, and R. Schuffny. Highly integrated packet-based aer communication infrastructure with 3gevent/s throughput. In *Electronics, Circuits, and Systems (ICECS), 2010 17th IEEE International Conference on*, pages 950–953, dec. 2010. doi: 10.1109/ICECS.2010.5724670.

D. Hebb. *The Organization of Behavior: A Neuropsychological Theory*. Taylor & Francis, 2002. ISBN 9780805843002.

B. Hellwig. A quantitative analysis of the local connectivity between pyramidal neurons in layers 2/3 of the rat visual cortex. *Biological Cybernetics*, 82:111–121, 2000. ISSN 0340-1200. URL `http://dx.doi.org/10.1007/PL00007964`. 10.1007/PL00007964.

M. Hines and N. Carnevale. *The NEURON simulation environment.*, pages 769–773. M.A. Arbib, 2003.

M. Hines, J. W. Moore, and T. Carnevale. Neuron, 2008. URL `http://neuron.duke.edu`.

M. L. Hines and N. T. Carnevale. *The NEURON Book*. Cambridge University Press, Cambridge, UK, 2006. ISBN 978-0521843218.

G. Hinton. A practical guide to training restricted boltzmann machines. *Momentum*, 9 (1), 2010.

G. E. Hinton. Training products of experts by minimizing contrastive divergence. *Neural computation*, 14(8):1771–1800, 2002.

G. E. Hinton and T. J. Sejnowski. Learning and relearning in boltzmann machines. *MIT Press, Cambridge, Mass*, 1:282–317, 1986.

J. Hirsch and C. Gilbert. Synaptic physiology of horizontal connections in the cat's visual cortex. *The Journal of Neuroscience*, 11(6):1800–1809, 1991. URL `http://www.jneurosci.org/content/11/6/1800.abstract`.

A. L. Hodgkin. The croonian lecture: Ionic movements and electrical activity in giant nerve fibres. *Proceedings of the Royal Society B*, 148(930):1–37, 1958.

A. L. Hodgkin and A. F. Huxley. A quantitative description of membrane current and its application to conduction and excitation in nerve. *J Physiol*, 117(4):500–544, August 1952. ISSN 0022-3751. URL `http://view.ncbi.nlm.nih.gov/pubmed/12991237`.

J. J. Hopfield. Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the National Academy of Sciences*, 79:2554–2558, 1982.

J. S. Ide and F. G. Cozman. Random Generation of Bayesian Networks. In *In Brazilian Symp. On Artificial Intelligence*, pages 366–375. Springer-Verlag, 2002.

G. Indiveri, E. Chicca, and R. Douglas. A VLSI array of low-power spiking neurons and bistable synapses with spike-timing dependent plasticity. *IEEE Transactions on Neural Networks*, 17(1):211–221, Jan 2006.

G. Indiveri, B. Linares-Barranco, T. J. Hamilton, A. van Schaik, R. Etienne-Cummings, T. Delbruck, S.-C. Liu, P. Dudek, P. Häfliger, S. Renaud, J. Schemmel, G. Cauwenberghs, J. Arthur, K. Hynna, F. Folowosele, S. Saighi, T. Serrano-Gotarredona, J. Wijekoon, Y. Wang, and K. Boahen. Neuromorphic silicon neuron circuits. *Frontiers in Neuroscience*, 5(0), 2011. ISSN 1662-453X. doi: 10.3389/fnins.2011.00073. URL `http://www.frontiersin.org/Journal/Abstract.aspx?s=755&name=neuromorphicengineering&ART_DOI=10.3389/fnins.2011.00073`.

E. M. Izhikevich. *Dynamical Systems in Neuroscience.* MIT Press, Cambridge, 2007.

H. Jaeger. The "echo state" approach to analysing and training recurrent neural networks. Technical Report GMD Report 148, German National Research Center for Information Technology, 2001. URL `ftp://borneo.gmd.de/pub/indy/publications_herbert/EchoStatesTechRep.pdf`.

S. Jeltsch. Computing with transient states on a neuromorphic multi-chip environment. Diploma thesis, Ruprecht-Karls-Universität Heidelberg, 2010. HD-KIP-10-54.

S. Jeltsch. *A Scalable Workflow for a Configurable Neuromorphic Platform.* PhD thesis, Universität Heidelberg, 2014.

J. Jordan, I. Bytschok, T. Tetzlaff, T. Pfeil, O. Breitwieser, J. Bill, M. Diesmann, A. Gruebl, J. Schemmel, M. Petrovici, et al. Neural networks as sources of uncorrelated noise for functional neural systems. Technical report, Computational and Systems Neuroscience, 2014.

B. M. Kampa, J. J. Letzkus, and G. J. Stuart. Cortical feed-forward networks for binding different streams of sensory information. *Nature Neuroscience*, 9(12):1472–1473, 2006.

B. Kaplan, D. Brüderle, J. Schemmel, and K. Meier. High-conductance states on a neuromorphic hardware system. In *Proceedings of the 2009 International Joint Conference on Neural Networks (IJCNN)*, 2009.

D. Kappel, B. Nessler, and W. Maass. Stdp installs in winner-take-all circuits an online approximation to hidden markov model learning. *PLoS computational biology*, 10(3): e1003511, 2014.

W. Kistler, W. Gerstner, and J. L. van Hemmen. Reduction of the Hodgkin-Huxley equations to a single-variable threshold model. *Neural Computation*, 9:1015–1045, 1997.

D. C. Knill and D. Kersten. Apparent surface curvature affects lightness perception. *Nature*, 351(6323):228–230, 1991.

H. Knodel. *Linder Biologie.* Schroeder Verlag, Hannover, 1998.

T. Korcsak-Gorzo. Firing states of recurrent leaky integrate-and-fire networks, January 2015. Bachelor thesis.

K. Körding and D. Wolpert. Bayesian integration in sensorimotor learning. *Nature*, 427 (6971):244–247, 2004.

J. Kremkow, A. Aertsen, and A. Kumar. Gating of signal propagation in spiking neural networks by balanced and correlated excitation and inhibition. *The Journal of neuroscience*, 30(47):15760–15768, Nov. 2010a. ISSN 1529-2401. doi: 10.1523/JNEUROSCI.3874-10.2010. URL `http://dx.doi.org/10.1523/JNEUROSCI.3874-10.2010`.

J. Kremkow, L. Perrinet, G. Masson, and A. Aertsen. Functional consequences of correlated excitatory and inhibitory conductances in cortical networks. *J Comput Neurosci*, 28:579–594, 2010b.

A. Kumar, S. Schrader, A. Aertsen, and S. Rotter. The high-conductance state of cortical networks. *Neural Computation*, 20(1):1–43, Jan 2008.

H. T. Kurata and D. Fedida. A structural interpretation of voltage-gated potassium channel inactivation. *Progress in Biophysics and Molecular Biology*, 92(2):185 – 208, 2006. ISSN 0079-6107. doi: http://dx.doi.org/10.1016/j.pbiomolbio.2005.10.001. URL `http://www.sciencedirect.com/science/article/pii/S0079610705000635`.

T. Lande, H. Ranjbar, M. Ismail, and Y. Berg. An analog floating-gate memory in a standard digital technology. In *Microelectronics for Neural Networks, 1996., Proceedings of Fifth International Conference on*, pages 271 –276, 12-14 1996. doi: 10.1109/MNNFS.1996.493802.

N. Lane and W. F. Martin. The energetics of genome complexity. *Nature*, 467:929–934, 2010.

N. Lane and W. F. Martin. The origin of membrane bioenergetics. *Cell*, 151(7):1406–1416, 2012.

P. Lánskỳ. Sources of periodical force in noisy integrate-and-fire models of neuronal dynamics. *Physical Review E*, 55:2040–2043, 1997.

L. Lapicque. Recherches quantitatives sur l'excitation electrique des nerfs traitee comme une polarization. *Journal de Physiologie et Pathologie General*, 9:620–635, 1907.

Y. LeCun and C. Cortes. The mnist database of handwritten digits, 1998.

L. Leng. Deep learning architectures for neuromorphic hardware. Master thesis, Ruprecht-Karls-Universität Heidelberg, 2014. HD-KIP 14-26.

S. B. Long, X. Tao, E. B. Campbell, and R. MacKinnon. Atomic structure of a voltage-dependent k+ channel in a lipid membrane-like environment. *Nature*, 450:276–382, 2007.

M. Lundqvist, M. Rehn, M. Djurfeldt, and A. Lansner. Attractor dynamics in a modular network of neocortex. *Network: Computation in Neural Systems*, 17:3:253–276, 2006.

M. Lundqvist, A. Compte, and A. Lansner. Bistable, irregular firing and population oscillations in a modular attractor memory network. *PLoS Comput Biol*, 6(6), 06 2010.

W. Maass, T. Natschläger, and H. Markram. Real-time computing without stable states: A new framework for neural computation based on perturbations. *Neural Computation*, 14(11):2531–2560, 2002.

Z. F. Mainen and T. J. Sejnowski. Reliability of spike timing in neocortical neurons. *Science*, 268(5216):1503–1506, 1995.

E. Marder. Variability, compensation, and modulation in neurons and circuits. *Proceedings of the National Academy of Sciences*, 108(Supplement 3):15542–15548, 2011.

E. Marder and J.-M. Goaillard. Variability, compensation and homeostasis in neuron and network function. *Nature Reviews Neuroscience*, 7(7):563–574, 2006.

E. Marder and A. L. Taylor. Multiple models to capture the variability in biological neurons and networks. *Nature neuroscience*, 14(2):133–138, 2011.

H. Markram. The blue brain project. *Nature Reviews Neuroscience*, 7(2):153–160, 2006.

H. Markram. The human brain project. *Scientific American*, 306(6):50–55, 2012.

H. Markram and B. Sakmann. Action potentials propagating back into dendrites trigger changes in efficacy of single-axon synapses between layer v pyramidal neurons. In *Soc. Neurosci. Abstr*, volume 21, page 2007, 1995.

H. Markram, J. Lübke, M. Frotscher, and B. Sakmann. Regulation of synaptic efficacy by coincidence of postsynaptic aps. *Science*, 275:213–215, 1997.

H. Markram, A. Gupta, A. Uziel, Y. Wang, and M. Tsodyks. Information processing with frequency-dependent synaptic connections. *Neurobiol Learn Mem*, 70(1-2):101–112, 1998a.

H. Markram, Y. Wang, and M. Tsodyks. Differential signaling via the same axon of neocortical pyramidal neurons. *Proceedings of the National Academy of Sciences of the United States of America*, 95(9):5323–5328, Apr. 1998b. ISSN 0027-8424. URL `http://view.ncbi.nlm.nih.gov/pubmed/9560274`.

H. Markram, M. Toledo-Rodriguez, Y. Wang, A. Gupta, G. Silberberg, and C. Wu. Interneurons of the neocortical inhibitory system. *Nat Rev Neurosci*, 5(10):793–807, Oct 2004a. ISSN 1471-003X. doi: 10.1038/nrn1519. URL `http://dx.doi.org/10.1038/nrn1519`.

H. Markram, M. Toledo-Rodriguez, Y. Wang, A. Gupta, G. Silberberg, and C. Wu. Interneurons of the neocortical inhibitory system. *Nature Reviews Neuroscience*, 5(10): 793–807, 2004b.

C. A. Mead. *Analog VLSI and Neural Systems.* Addison Wesley, Reading, MA, 1989.

C. A. Mead. Neuromorphic electronic systems. *Proceedings of the IEEE*, 78:1629–1636, 1990.

C. A. Mead and M. A. Mahowald. A silicon model of early visual processing. *Neural Networks*, 1(1):91–97, 1988.

S. Millner, A. Grübl, K. Meier, J. Schemmel, and M.-O. Schwartz. A VLSI implementation of the adaptive exponential integrate-and-fire neuron model. In J. Lafferty, C. K. I. Williams, J. Shawe-Taylor, R. Zemel, and A. Culotta, editors, *Advances in Neural Information Processing Systems 23*, pages 1642–1650, 2010.

S. Mitra, S. Fusi, and G. Indiveri. Real-time classification of complex patterns using spike-based learning in neuromorphic VLSI. *IEEE Transactions on Biomedical Circuits and Systems*, 3:(1):32–42, 2009.

C. Mohr, P. D. Roberts, and C. C. Bell. The mormyromast region of the mormyrid electrosensory lobe. i. responses to corollary discharge and electrosensory stimuli. *Journal of neurophysiology*, 90(2):1193–1210, 2003a.

C. Mohr, P. D. Roberts, and C. C. Bell. The mormyromast region of the mormyrid electrosensory lobe. ii. responses to input from central sources. *Journal of neurophysiology*, 90(2):1211–1223, 2003b.

R. Moreno-Bote and N. Parga. Role of synaptic filtering on the firing response of simple model neurons. *Physical Review Letters*, 92(2):028102, 2004.

C. Morris and H. Lecar. Voltage oscillations in the barnacle giant muscle fiber. *Biophysical journal*, 35(1):193–213, 1981.

A. Morrison, M. Diesmann, and W. Gerstner. Phenomenological models of synaptic plasticity based on spike timing. *Biological Cybernetics*, 98(6):459–478, June 2008. ISSN 0340-1200. doi: 10.1007/s00422-008-0233-1.

V. B. Mountcastle. The columnar organization of the neocortex. *Brain*, 120(4):701–722, 1997.

L. Muller and A. Destexhe. Propagating waves in thalamus, cortex and the thalamocortical system: Experiments and models. *Journal of Physiology-Paris*, 106(5–6):222 – 238, 2012. ISSN 0928-4257. doi: 10.1016/j.jphysparis.2012.06.005. URL `http://www.sciencedirect.com/science/article/pii/S0928425712000393`. <ce:title>New trends in neurogeometrical approaches to the brain and mind problem</ce:title>.

P. Müller. Distortions of neural network models induced by their emulation on neuromorphic hardware devices. Diploma thesis, Ruprecht-Karls-Universität Heidelberg, 2011. HD-KIP-11-172.

T. Murakoshi, J.-Z. Guo, and T. Ichinose. Electrophysiological identification of horizontal synaptic connections in rat visual cortex in vitro. *Neuroscience Letters*, 163(2):211 – 214, 1993. ISSN 0304-3940. doi: 10.1016/0304-3940(93)90385-X. URL `http://www.sciencedirect.com/science/article/pii/030439409390385X`.

R. Naud, N. Marcille, C. Clopath, and W. Gerstner. Firing patterns in the adaptive exponential integrate-and-fire model. *Biological Cybernetics*, 99(4):335–347, Nov 2008. doi: 10.1007/s00422-008-0264-7. URL `http://dx.doi.org/10.1007/s00422-008-0264-7`.

E. Neftci, S. Das, B. Pedroni, K. Kreutz-Delgado, and G. Cauwenberghs. Event-driven contrastive divergence: neural sampling foundations. *Frontiers in neuroscience*, 9, 2015.

B. Nessler, M. Pfeiffer, and W. Maass. Stdp enables spiking neurons to detect hidden causes of their inputs. In *NIPS*, pages 1357–1365, 2009.

B. Nessler, M. Pfeiffer, L. Buesing, and W. Maass. Bayesian computation emerges in generic cortical microcircuits through spike-timing-dependent plasticity. *PLoS Computational Biology*, 9(4):e1003037, 2013.

E. Oja. A simplified neuron model as a principal component analyzer. *Journal of Mathematical Biology*, 15:267–273, 1982.

D. Pecevski, L. Buesing, and W. Maass. Probabilistic inference in general graphical models through sampling in stochastic networks of spiking neurons. *PLoS computational biology*, 7(12):e1002294, 2011.

R. Perin, T. K. Berger, and H. Markram. A synaptic organizing principle for cortical neuronal groups. *PNAS*, pages 5419–5424, 2011.

A. Peters and C. Sethares. The organization of double bouquet cells in monkey striate cortex. *Journal of Neurocytology*, 26(12):779 – 797, 1997. doi: 10.1023/A:1018518515982.

V. Petkov. Toward belief propagation on neuromorphic hardware. Diploma thesis, Ruprecht-Karls-Universität Heidelberg, 2012. HD-KIP 12-23.

M. A. Petrovici and J. Bill. A new method for quantifying and predicting neural response correlations: Internal report., 2009.

M. A. Petrovici, T. Pfeil, S. Friedmann, J. Partzsch, S. Jeltsch, M.-O. Schwartz, B. Vogginger, A. Grübl, D. Husmann, J. Schemmel, and K. Meier. Exploring network architectures with the facets/brainscales hardware and pynn, 2011. URL `https://capocaccia.ethz.ch/capo/wiki/2011/bsshw11`.

M. A. Petrovici, S. Millner, M.-O. Schwartz, T. Pfeil, S. Jeltsch, A. Hartel, J. Schemmel, and K. Meier. Experiments on brainscales hardware systems, 2012. URL `https://capocaccia.ethz.ch/capo/wiki/2012/bssexperiments12`.

M. A. Petrovici, J. Bill, I. Bytschok, J. Schemmel, and K. Meier. Stochastic inference with deterministic spiking neurons. *arXiv preprint arXiv:1311.3211*, 2013.

M. A. Petrovici, B. Vogginger, P. Müller, O. Breitwieser, M. Lundqvist, L. Muller, M. Ehrlich, A. Destexhe, A. Lansner, R. Schüffny, et al. Characterization and compensation of network-level anomalies in mixed-signal neuromorphic modeling platforms. *PloS one*, 9(10):e108590, 2014.

T. Pfeil, T. C. Potjans, S. Schrader, W. Potjans, J. Schemmel, M. Diesmann, and K. Meier. Is a 4-bit synaptic weight resolution enough? - constraints on enabling spike-timing dependent plasticity in neuromorphic hardware. *Frontiers in Neuroscience*, 6(90), 2012a. ISSN 1662-453X. doi: 10.3389/fnins.2012.00090.

T. Pfeil, T. C. Potjans, S. Schrader, W. Potjans, J. Schemmel, M. Diesmann, and K. Meier. Is a 4-bit synaptic weight resolution enough? - constraints on enabling spike-timing dependent plasticity in neuromorphic hardware. *Frontiers in Neuroscience*, 6(90), 2012b. ISSN 1662-453X. doi: 10.3389/fnins.2012.00090.

T. Pfeil, A. Grübl, S. Jeltsch, E. Müller, P. Müller, M. A. Petrovici, M. Schmuker, D. Brüderle, J. Schemmel, and K. Meier. Six networks on a universal neuromorphic computing substrate. *Frontiers in Neuroscience*, 7:11, 2013. ISSN 1662-453X. doi: 10.3389/fnins.2013.00011. URL `http://www.frontiersin.org/neuromorphic_engineering/10.3389/fnins.2013.00011/abstract`.

T. Pfeil, J. Jordan, T. Tetzlaff, A. Grübl, J. Schemmel, M. Diesmann, and K. Meier. The effect of heterogeneity on decorrelation mechanisms in spiking neural networks: a neuromorphic-hardware study. *arXiv preprint arXiv:1411.7916*, 2014.

W. F. Pickard. Generalizations of the goldman-hodgkin-katz equation. *Mathematical Biosciences*, 30(1):99–111, 1976.

A. Pouget, J. M. Beck, W. J. Ma, and P. E. Latham. Probabilistic brains: knowns and unknowns. *Nature Neuroscience*, 16(9):1170–1178, 2013.

D. Probst. A neural implementation of probabilistic inference in binary probability spaces. Master thesis, Ruprecht-Karls-Universität Heidelberg, 2014.

D. Probst, M. A. Petrovici, I. Bytschok, J. Bill, D. Pecevski, J. Schemmel, and K. Meier. Probabilistic inference in discrete spaces can be implemented into networks of lif neurons. *Frontiers in computational neuroscience*, 9, 2015.

D. Purves, A. G. J., and F. D. *Neuroscience.* Sinauer Associates, 2001.

R. P. N. Rao. Hierarchical bayesian inference in networks of spiking neurons. In *Advances in Neural Information Processing Systems*, volume 17, pages 1113–1120, 2005. URL `http://papers.nips.cc/paper/2643-hierarchical-bayesian-inference-in-networks-of-spiking-neurons.pdf`.

L. M. Ricciardi and L. Sacerdote. The ornstein-uhlenbeck process as a model for neuronal activity. *Biological Cybernetics*, 35:1–9, 1979.

L. M. Ricciardi and S. Sato. First-passage-time density and moments of the ornstein-uhlenbeck process. *Journal of Applied Probability*, 25:43–57, 1988.

M. J. Richardson and W. Gerstner. Statistics of subthreshold neuronal voltage fluctuations due to conductance-based synaptic shot noise. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 16(2):026106, 2006.

C. Richet. An address on ancient humorism and modern humorism: Delivered at the international congress of physiology held in vienna, september 27th to 30th. *British medical journal*, 2(2596):921, 1910.

F. Rieke, D. Warland, R. de Ruyter van Steveninck, and W. Bialek. *Spikes - Exploring the neural code.* MIT Press, Cambridge, MA., 1997.

B. Rivkin. On the memory characteristic of a cortical atractor network. Bachelor thesis, Ruprecht-Karls-Universität Heidelberg, 2014.

P. Rocke, B. McGinley, J. Maher, F. Morgan, and J. Harkin. Investigating the suitability of fpaas for evolved hardware spiking neural networks. In G. Hornby, L. Sekanina, and P. Haddow, editors, *Evolvable Systems: From Biology to Hardware*, volume 5216 of *Lecture Notes in Computer Science*, pages 118–129. Springer Berlin / Heidelberg, 2008. ISBN 978-3-540-85856-0. URL `http://dx.doi.org/10.1007/978-3-540-85857-7_11`. 10.1007/978-3-540-85857-7_11.

E. T. Rolls and G. Deco. *The noisy brain: stochastic dynamics as a principle of brain function*, volume 34. Oxford university press Oxford, 2010.

M. Roth. Predictive stochastic inference - from abstract models to neuromorphic implementation. Bachelor thesis, Ruprecht-Karls-Universität Heidelberg, 2014.

A. Roxin, N. Brunel, and D. Hansel. Role of delays in shaping spatiotemporal dynamics of neuronal activity in large networks. *Phys. Rev. Lett.*, 94:238103, Jun 2005. doi: 10.1103/PhysRevLett.94.238103. URL `http://link.aps.org/doi/10.1103/PhysRevLett.94.238103`.

R. Salakhutdinov. Learning deep boltzmann machines using adaptive mcmc. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, pages 943–950, 2010.

R. Salakhutdinov and G. E. Hinton. Deep boltzmann machines. In *International Conference on Artificial Intelligence and Statistics*, pages 448–455, 2009.

J. Schemmel, A. Grübl, K. Meier, and E. Muller. Implementing synaptic plasticity in a VLSI spiking neural network model. In *Proceedings of the 2006 International Joint Conference on Neural Networks (IJCNN)*. IEEE Press, 2006.

J. Schemmel, D. Brüderle, K. Meier, and B. Ostendorf. Modeling synaptic plasticity within networks of highly accelerated I&F neurons. In *Proceedings of the 2007 IEEE International Symposium on Circuits and Systems (ISCAS)*, pages 3367–3370. IEEE Press, 2007.

J. Schemmel, J. Fieres, and K. Meier. Wafer-scale integration of analog neural networks. In *Proceedings of the 2008 International Joint Conference on Neural Networks (IJCNN)*, 2008.

J. Schemmel, D. Brüderle, A. Grübl, M. Hock, K. Meier, and S. Millner. A wafer-scale neuromorphic hardware system for large-scale neural modeling. In *Proceedings of the 2010 IEEE International Symposium on Circuits and Systems (ISCAS)*, pages 1947–1950, 2010.

D. Schmidt. Readout training for liquid factor graphs. Internship Report, University of Heidelberg, 2012.

E. Schneidman, B. Freedman, and I. Segev. Ion channel stochasticity may be critical in determining the reliability and precision of spike timing. *Neural computation*, 10(7): 1679–1703, 1998.

S. Scholze, S. Henker, J. Partzsch, C. Mayr, and R. Schuffny. Optimized queue based communication in vlsi using a weakly ordered binary heap. In *Mixed Design of Integrated Circuits and Systems (MIXDES), 2010 Proceedings of the 17th International Conference*, pages 316 –320, june 2010.

S. Scholze, H. Eisenreich, S. Höppner, G. Ellguth, S. Henker, M. Ander, S. Hänzsche, J. Partzsch, C. Mayr, and R. Schüffny. A 32 GBit/s communication SoC for a waferscale neuromorphic system. *Integration, the VLSI Journal*, 2011a. doi: 10.1016/j.vlsi.2011.05.003. in press.

S. Scholze, S. Schiefer, J. Partzsch, S. Hartmann, C. G. Mayr, S. Höppner, H. Eisenreich, S. Henker, B. Vogginger, and R. Schüffny. VLSI implementation of a 2.8GEvent/s packet based AER interface with routing and event sorting functionality. *Frontiers in Neuromorphic Engineering*, 5(117):1–13, 2011b.

S. Schrader, M. Diesmann, and A. Morrison. A compositionality machine realized by a hierarchic architecture of synfire chains. *Frontiers in Computational Neuroscience*, 4, 2010.

S. G. Schultz, thoomas E. Andreoli, A. M. Brown, D. M. Farmbrough, J. F. Hoffman, and M. G. Welsh. *Molecular Biology of Membrane Transport Disorders*. Plenum Press, 1996.

M.-O. Schwartz. *Reproducing Biologically Realistic Regimes on a Highly-Accelerated Neuromorphic Hardware System*. PhD thesis, Universität Heidelberg, 2013.

J. R. Searle. *Mind: a brief introduction*, volume 259. Oxford University Press Oxford, 2004.

T. J. Sejnowski. Storing covariance with nonlinearly interacting neurons. *Journal of Mathematical Biology*, 69:385–389, 1977.

H. Z. Shouval, S. S.-H. Wang, and G. M. Wittenberg. Spike timing dependent plasticity: a consequence of more fundamental learning rules. *Frontiers in computational neuroscience*, 4, 2010.

J. Sjöström and W. Gerstner. Spike-timing dependent plasticity. *Scholarpedia*, 5(2):1362, 2010. doi: 10.4249/scholarpedia.1362. URL `http://www.scholarpedia.org/article/Spike-timing_dependent_plasticity`.

S. Song, P. J. Sjöström, M. Reigl, S. Nelson, and D. B. Chklovskii. Highly nonrandom features of synaptic connectivity in cortical circuits. *PLOS Biology*, 3(3):517–519, 2005.

A. Steimer, W. Maass, and R. Douglas. Belief propagation in networks of spiking neurons. *Neural Computation*, 21(9):2502–2523, 2009.

D. Stöckel. Boltzmann sampling with neuromorphic hardware. Bachelor thesis, Ruprecht-Karls-Universität Heidelberg, 2015.

Y. Sugawara, K. Grant, V. Han, and C. C. Bell. Physiology of electrosensory lateral line lobe neurons in gnathonemus petersii. *Journal of experimental biology*, 202(10): 1301–1309, 1999.

D. Sussillo, T. Toyoizumi, and W. Maass. Self-Tuning of Neural Circuits Through Short-Term Synaptic Plasticity. *J Neurophysiol*, 97(6):4079–4095, 2007. doi: 10.1152/jn.01357.2006.

A. E. Telfeian and B. W. Connors. Widely integrative properties of layer 5 pyramidal cells support a role for processing of extralaminar synaptic inputs in rat neocortex. *Neuroscience Letters*, 343(2):121 – 124, 2003. ISSN 0304-3940. doi: 10.1016/S0304-3940(03)00379-3. URL `http://www.sciencedirect.com/science/article/pii/S0304394003003793`.

T. Tetzlaff, M. Helias, G. T. Einevoll, and M. Diesmann. Decorrelation of neural-network activity by inhibitory feedback. *PLoS computational biology*, 8(8):e1002596, 2012.

M. U. Thomas. Some mean first-passage time approximations for the ornstein-uhlenbeck process. *Journal of Applied Probability*, pages 600–604, 1975.

A. M. Thomson and J. Deuchars. Temporal and spatial properties of local circuits in neocortex. *Trends in neurosciences*, 17(3):119–126, 1994.

A. M. Thomson, D. C. West, Y. Wang, and A. P. Bannister. Synaptic connections and small circuits involving excitatory and inhibitory neurons in layers 2-5 of adult rat and cat neocortex: triple intracellular recordings and biocytin labelling in vitro. *Cerebral Cortex*, 12:936–953, 2002.

M. Tsodyks and H. Markram. The neural code between neocortical pyramidal neurons depends on neurotransmitter release probability. In *Proceedings of the national academy of science USA* Tsodyks and Markram (1997b), pages 719–723.

M. Tsodyks and H. Markram. The neural code between neocortical pyramidal neurons depends on neurotransmitter release probability. *Proceedings of the national academy of science USA*, 94:719–723, Jan. 1997b.

T. P. Vogels and L. F. Abbott. Signal propagation and logic gating in networks of integrate-and-fire neurons. *J Neurosci*, 25(46):10786–95, Nov 2005.

R. J. Vogelstein, U. Mallik, J. T. Vogelstein, and G. Cauwenberghs. Dynamically reconfigurable silicon array of spiking neuron with conductance-based synapses. *IEEE Transactions on Neural Networks*, 18:253–265, 2007.

B. Vogginger. Testing the operation workflow of a neuromorphic hardware system with a functionally accurate model. Diploma thesis, Ruprecht-Karls-Universität Heidelberg, 2010. HD-KIP-10-12.

J. Waters, A. Schaefer, and B. Sakmann. Backpropagating action potentials in neurones: measurement, mechanisms and potential functions. *Progress in Biophysics and Molecular Biology*, 87(1):145 – 170, 2005. ISSN 0079-6107. doi: http://dx.doi.org/10.1016/j.pbiomolbio.2004.06.009. URL `http://www.sciencedirect.com/science/article/pii/S0079610704000653`. <ce:title>Biophysics of Excitable Tissues</ce:title>.

T. Website. `http://www.nest-initiative.org`, 2009.

C. Weilbach. An online learning algorithm for lif-based boltzmann machines. Bachelor thesis, Ruprecht-Karls-Universität Heidelberg, 2015.

I. H. Witten and E. Frank. *Data Mining: Practical Machine Learning Tools and Techniques.* Morgan Kaufmann, 2005.

Acknowledgements

Until today, I keep struggling with remembering whether Batman is the hero this city needs but doesn't deserve, or rather the hero this city deserves, but doesn't need right now. Whichever they prefer to be, my deepest gratitude extends to quite a lot of superheroes.

Karlheinz Meier for being the personification of what Germans call a "Doktorvater". Several years ago, he took quite a risk with the whole TMA thing – I hope he finds it's paying off. For transitioning from particle physics to neuroscience. Too bad for particle physics. For being someone to undoubtedly rely on and for always finding time for counsel despite an impossibly overburdened schedule. Also, for ending all delicate discussions with "alles wird gut". And, in particular, for mercilessly mauling the wolf. For keeping all but the scientific burdens away from our minds. The efficiency with which required resources are made available sometimes borders on the miraculous. For supporting all our scientific inquiries with enthusiasm and wisdom. And for his impressive instinct in asking the right questions. For FACETS and BrainScaleS (not BrainScales!) and the communities that they have built. And for persistently working on fusing these heterogeneous ensembles into thriving scientific communities united by common goals. For the unique chance that is the HBP. For the big eagle and for the small eagle. And for providing the MacBook Air with appropriate packaging. Also, Eilif, if you haven't yet solved the puzzle, I have solved it for you: he does not.

Johannes Schemmel for fathering the hardware and thereby laying the groundwork for our entire group. None of us would be here without him. For being so enthusiastic about this whole neuromorphic thing. There is obviously a long way to go, but I am more and more convinced that we are on a promising path. And, of course, for all of the bugs being software bugs. Which is actually correct most of the time. For all of his support, especially in my early days. Just as for KM, I hope he finds it's paying off. For being a walking encyclopedia on the most varied of subjects. Especially neurobiology. Until today I often find myself in extended google scholar sessions as a follow-up to our discussions. And for setting up speed-eating records between 12:00 and 12:10. (Which I think is at least partly owed to Björn.)

Ulrich Schwartz for his kind, but most of all courageous offer of refereeing 400 pages of written text. I hope he is not too angry that I actually took him up on his selfless proposal.

Michael Hausmann for kindly agreeing to attack me during my defense. Also, for quite some memorable moments during our long exam sessions. Which also made me even more reluctant of letting doctors do things to me.

Daniel Durstewitz for kindly agreeing to also take part in the attacking procedure. Also, for a really nice project proposal which we haven't given up on. I'm confident we'll make it work in the near future.

The Golden Generation Softies, in particular:

Daniel "Dirty D" Brüderle for being my mentor during the early days and such an awesome one at that. For those paradigm-shifting weeks in Capo Caccia (FACETS rocks!) while putting up with my permanent crying about the hardware... imperfections. For PyNN – except for that ONE bug! For defining the term "softie" and for lots of boboaak.

For "letting" Mr. Bill and me win against him and Merd at foosball. For a legendary pool party with even more legendary pictures taken. Let's see if I can top that anytime soon... For that one night in the Baden-Württembergian Vegas. For probably running that all-you-can-eat Sushi restaurant out of business. For those tight-fitting cyclist outfits. Too bad I had to give them back. For never succeeding, despite innumerable attempts, to prove that bicycle racing (and especially watching other people do it) is anything but boring. For being one of the precious few Visionaries to enjoy MMA. How it is possible to have such good and bad taste in sports at the same time remains beyond me. For Eric's Hidden Movie Folders. The simultaneous definition of pride and shame. And, more than anything else, for his future return to Heidelberg as my (!) Postdoc student.

Johannes "Jigsaw J" Bill for all the persistence with the hardware in the early sampling days. For providing so much awesomeness during Capo Caccia 2011, 2012 and 2013. And of course, for all the awesome NeuroSempoling. For never ever being pushy about anything at all. But the end product always spoke for itself. And for always finding at least some time when there was no time to be found. For those early correlation studies which came out of nothing but were so much fun. For not becoming Romanian bear food. For Ascandahar. I told you I could win without backstabbing everyone else! And of course for all the handcheese with music.

Eric "Evil E" Müller for teh software. Without him everything that runs an operating system around here will shut down and die within days. (With him it takes slightly longer.) (But not much longer for UNIX systems.) For always helping when I'm whining about some software problem. And never bashing me for it afterwards. Also, for never bashing my beautiful tablet. For never outdated advice on fashion. Vogue writers wish they were him. For not having succeeded to convert me to using an editor from the time when they used punchcards for programming. For proving mind-body duality by typing faster than humans can think. Including himself. For earning a participation certificate. For strong opinions, weakly held. Especially during lunch break. Also, for never talking about bicycles during abovementioned break. For monosolosportive activities. For defensive warfare. Motherland is proud. For equating brown noise with music and beats per second with quality. For Eluveitie and Trance Metal, although he will never admit to enjoying the latter. And – speaking about music – for inspiring so many of us with his hidden music folders.

All of my students, some of which I can be grateful to still have today as friends and colleagues:

Ilja "Itchy I" Bytschok for the most beautiful galaxy plots ever. And for all the awesome NeuroSempoling. For taking such good care of our new students when I could not. And for always giving a helping hand when needed. For all our great blackboard discussions. Theory ftw! For chasing me around the badminton court. For having such good taste in sports and in particular for Trujilo the Killo. And for sneakily laughing at what he perceives to be our constant innuendo. Although we keep assuring him that our intentions are pure.

Paul "Pandemic P" Müller for being a modeler at heart and a coder in practice. Teach them how it's done! For transcending epicness in our analysis paper. For all the hardware simulations. Teach them how it's done – again! For his overpowering enthusiasm when answering the phone. For shirtweeks and persistent vegetarianism. Although I'm sure there's some backup roast beef in his closet. And of course for the ever challenging

quantum foosball.

Venelin "Venomous V" Petkov for the first usable implementation the L23 code. Also, for all the nice work on spiking factor graphs. And for a great collaboration in those early years.

Oliver "Omnivorous O" Breitwieser For at least 1e6 L23 simulations. And for an even larger number of plots. Never has the world witnessed a more inspiring depiction of science. For the 10^n speedup of the sampling code. Who needs analog hardware anyway? For hurr/durr and all of the afferent rage. For something always getting in the way of proving that he can beat 5 unfair bots at DotO. For attempting a facepiercing with a ski pole and, especially, for not succeeding. And, of course, for boldly tooling where no man has tooled before.

Dimitri Probst for the great work on the spiking Bayesian networks. And especially for literally winning a prize for awesomeness. Also, for pulling through some extended sessions of exhausting work. It did pay off, though. And for not bashing my (lack of) soccer skills too hard.

Luziwei "Lunatic L" Leng for all of the deep learning. And awesome tāigerū. Also, for the most awesome tea ever. It is really him (and Cocky) that actually got me hooked on tea. Especially when there are no energy drinks around.

David Stöckel for the most awesome Spikey experiments ever. He was the one to put the cherry on top of the cake we had been working on for years.

Roman Martel for his diligent work and permanent readiness to help. Also, for being a fan of martial arts not only in theory, but especially in practice!

Marco Roth for the beautiful pong bot. Imagine when we get this running on accelerated hardware...

Boris Rivkin for his memory capacity.

Agnes Korcsak-Gorzo for working on the ultimate goal of Boltzmann cannibalism.

Christian Weilbach for on-line learning. Also, for warning us of the NSA.

Dominik Schmidt for the early work on LSMs. And especially for returning to our group to calibrate the uncalibratable.

All other Visionaries, past, present and future (I'll omit the latter in this list). This is the most awesome workgroup. Ever.

Björn Kindler for always having a kind advice or feedback. And, of course, for the most impeccable organization skills the world has ever witnessed. For awesome websites and click-together pizzas. For non-alcoholic beverages only! For aggressive picture-taking and adding safety margins to safety margins. And for remaining a trekkie despite the last two installments being major fails (of course he would never admit to that).

Marc-O "Manly M" Schwartz for some crazy early attempts at neuromorphic neural sampling. That was quite some ∂_twtf - but at least we got those single neuron measurements. For getting neurons to work outside the specified parameter range. That was quite some cherrypicking in a world devoid of cherries. Refractoriness ftw. And of course for appreciating MMA.

Alex "AdEx Columnov" Kononov for even more crazy attempts at sampling on the HICANN. We dug up some serious bugs in Capo Caccia – and flooded our wtf-site accordingly. For temporarily being the one-man-army taking care of everything wrong with the wafer. For confirming every single cliché about Russians, especially with those stunts in slippery shoes on the cliffs above a raging Mediterranean. For organizing that epic tram ride I had the chance to take part in – and I regret until today not having done so. And, of course, for the still unsurpassed (!) Kononov chain.

Moritz Schilling for being himself in general. Also, in particular, for being himself during the filming of the Eric Movie. And, unforgettably, for being himself during those late hours in the office.

Tom "Tenacious T" Pfeil for all the Spikey action. Also, for helping out with Spikey-Sempoling. For taking the initiative on the Spikey paper I would have never found the time for. For watching the worst movie ever (is it by chance that it was German...?) and still having fun. And for taking care of difficult thesis situations despite knowing their history.

Sebastian "Jerky J" Jeltsch for the impressive MultiSpikey results that remain immortalized, to this day, on our office door. For the best mapping tool ever. Everybody keeps telling me how much they adore templates now. For being the only one to successfully apply the Jet – or was it the Snake? I actually still remember the precise defensive position. And, most importantly, for teaching me a sport where taking off your t-shirt correlates with your performance. And for not letting me break my back at the Riesenstein – unlike others...

Christoph "Cocky C" Koke for all the software bugfixing. Because all bugs are software bugs. And for teaching me how professional tea brewing actually works.

Sven "Swallow S" Schrader for the coolest Half-Life crowbar ever. Can't wait to see the suit. Eat your heart out, Fifty Shades... For the excellent local wine. For explaining the difference between neurons and neutrons. And for – the best comes last – "Fit mit Arnold Schwarzenegger". You still owe me a training session.

Andi "Gruesome G" Grübl for coordinating the FPGA activities. For the Spikey chip! Or at least half of it. But if you can do it with 384 neurons, then you can do it with 192 as well, right? For always being in a contagiously good mood. For truly German beer consumption skills. And for convincing me how awesome and manly e-bikes really are.

Andi "A0 A" Hartel for being a modeler at heart and a hardie in practice. For reviving our journal club. I can now finally start supporting you in practice. For bouldering – though not as reliably. Evidently, Pilates is a superior sport. For the sexiest pink pants ever. I remain jealous to this day. And for an awesome time in Fontainebleau. I promise to organize another trip soon.

Sebastian "Sexxy S" Millner for his cheerful attitude and the good times spent both within and outside the lab. For an epic karaoke night on a bed of roses. And, most memorably, for making NeuroSempoling a thing.

Simon "Seaman S" Friedmann for the Plasticity Processor that will solve all problems ever. Also, the layout pics look really cool. For getting NeuroSempoling into FPGAs. And Vitali. And, of course, for German proofreading.

Matthias Hock for being the only hardie to challenge our foosball supremacy from time to time. And for the hardie way of doing so. For late-night chats. For well-intended censorship. This is not a slippery slope at all. For the first WTF button. Should have made it Tom-robust. And, of course, for the second, nucular-launch-style WTF button.

Mitja Kleider for making software work. And for calibrating the uncalibratable.

Dan "Dandy D" Husmann for mad SolidWorks skillz. And for taking care that all those wafers don't explode. For fun on the Feldberg. For always helping when I needed a hand. And for having the most awesome hippie bus ever.

Maurice "Misbegotten M" Güttler for being the other guy who's keeping our wafers running. Because that's what true hardies do. And for uttering the memorable sentence "Umm... Is that your car that's being towed away over there?" approximately 10 seconds too late.

Sebastian Schmitt for getting things done. For somehow finding time for proofreading. And, of course, for the schmittfire chain.

Aamir Syed Ahmed for making neurons smaller and better at the same time. Also, for some really nice time outside the lab. And for changing my preconceptions about snooker.

The Stockholm Connection, especially

Anders Lansner for a great collaboration throughout all these years. And for a really marvelous dinner in Stockholm.

Mikael Lundqvist for all of the nice work on the L2/3 attractor model.

The Graz Connection, especially

Wolfgang Maass for the Fürberg workshops. This is really a tradition that is worth holding onto! For his interest and active support of the very successful Graz-Heidelberg collaboration. And for his experienced guidance throughout our many exchanges of ideas.

Stefan Habenschuss, **Zeno Jonke** and **Dejan Pecevski** for all those great Fürberg brainstorming sessions.

The Jülich Connection, especially

Markus Diesmann for all of his support throughout FACETS and BrainScaleS. And, of course, for NEST. I vow to never use Neuron again. Okay, I take that back. But he did convince me that if you know what you're doing, it's usually NEST that you want to use.

Jakob Jordan and **Tom Tetzlaff** for making noise. For some really fun and productive working sessions. For an awesome pingpong evening in Schladming. And for a great time in Capo Caccia.

The Gif Connection, especially

Kirsty Grant for a great ITN workshop in Gif. Also, for some fascinating insights into the sensory world of electric fish. I really must find some time for some Gnathonemus Neuromorphicus emulations.

Yves Fregnac for his élan in driving forward the collaboration between hardies, softies, modelers and biologists. For the BSS & ITN workshops in Gif. And for all of his kindness and support.

The Lausanne Connection, especially

Wulfram Gerstner for inviting me to give a talk in Lausanne and for some really intriguing follow-up discussions.

Tilo Schwalger for organizing a delightful visit in Lausanne.

The Zürich Connection, especially **Giacomo Indiveri** for all those awesome Capo Caccia workshops. Which have not only been a spawning pool for some really cool projects, but also the reason for some rather adventurous road trips.

All of my seminar students for giving me the occasion and motivation to learn a lot while striving to teach.

Rainer and Khukie for organizing the uniquely exquisite humanities journal club. Those were the days... For all those intriguing discussions on animal rights and the philosophy of mind. For inspiring me to read Searle and Nagel – and maybe I will also pick up that boring Singer book again, someday. And, especially, for surviving a motorbike accident under the most insane conditions.

Hosam "The Opossum" Esame for awesome times in Paradise. For expertly teaching me how to blow smoke rings. And for a most memorable birthday party on the Neckar.

Bi for not getting mad after me after having crashed his car. Also, this never happened. For bunnies!!! And for those awesome books by Searle and Beckermann – I will return them soon, I promise...

Manuel "Prince" Jütte for the best of times during our learning phase for our theory exams. Until today, this remains the best ∂physics/∂t I have ever achieved. For "Bullshit!". Penn&Teller are really awesome. And, of course, for teaching me how to cook onions properly. Since that day, I always prepare my tomato sauce this way.

Franzi for inviting me to the only gymnastics event I have ever witnessed live. For adding "coelom" and "gastrulation" to my vocabulary. For those insanely expensive cocktails. For showing me glowing neuron fibers on medaka fish, which are, of course, much more awesome than zebrafish. For asking me to examine her on phylogenetics and embryogenesis. Those were some awesome pieces of biology right there. And for showing me the labs where *real* experimental science is done. Who even cares for network models?!

Tobi for all the Arnie Training. For all the fun we had in Students' Paradise. And all the fun we had while bouldering. Which, of course, reminds me of those awesome days in Fontainebleau... For always triggering some really cool activities when visiting us in Heidelberg. For DotA rage. And all the Spartan action. Aouh, aouh, aouh!

Mircea for smoothing out my Bucharest/Heidelberg phase transition. And for finding me peacefully asleep at 5 AM.

Moro and **Radu** for a great time whenever I'm returning home. And for the fierce competition in all our laser tag encounters. Until today, I hold on to that one score sheet that documents one of my most cherished achievements.

Jes for having come to Heidelberg and having found the Students' Paradise. For being such a wonderful girl in general. Except when praying for hundreds of people to miss their appointments. For allowing me to write my Diploma thesis while watching Turtle Girl. And for watching Turtle Girl while I was writing my thesis. Which remains the most amazing TV show ever. For never letting me beg for anything. And for never pouting when I wouldn't play along. For never wanting to play and listen to sappy music. For helping Bernie prove the existence of God. And of course, for thanking Him, whose name and work are oh so great. Best. Acknowledgement. Ever.

Vitali "Vicarious V" Karasenko for allowing me to quote Searle before quoting him. For not unlike. And of course for what *she* said. For stoically enduring permabashing. And amplifiedly permabashing back. For Smirnoff Ice, Whips and Babybels. For Homo Sapiens and the Karasenko Work Law. For working FPGAs. At least I think they work... Oh, nevermind... For never flaming at DotA and not scaring (and scarring) Oli for years to come. For the only concert I have ever paid money for (rakkatakka m***********s!). For challenging my face during bouldering. Not to mention the reliable spotting... For so many minorities. For osum plots. For poking with a stick at every single one of my more metaphysical statements. For holding my hand during my final hours. And all of the Rrromanian love.

Bernhard "Vicious V" Vogginger for moving in with us and becoming family. And for letting me convince him to join the most awesome group ever. He should have stayed with me in Heidelberg, but I shall admit defeat when I must: the opponent was simply out of my league. For the original ESS. For debugging all those beautiful software layers. For an unforgettable all-nighter with a surprise visit in the morning. One of the coasters remains on the windowsill to this day! For being a humanist to the core, with only a touch of fundie on the surface. For trying to prove the existence of God while disproving the existence of Dawkins. And for bestowing me with the greatest of honors at his wonderful wedding.

Very for being everything that is worth fighting for.

And, most of all, **Mommy & Daddy**, to whom I owe everything. To you I dedicate this work.

Statement of Originality (Erklärung):

I certify that this thesis, and the research to which it refers, are the product of my own work. Any ideas or quotations from the work of other people, published or otherwise, are fully acknowledged in accordance with the standard referencing practices of the discipline.

Ich versichere, daß ich diese Arbeit selbständig verfaßt und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe.

Heidelberg, April 7, 2016

....
(signature)